**Open Geospatial Consortium**

Submission Date: 2018-01-15

Approval Date:   <yyyy-mm-dd>

Publication Date:   <yyyy-mm-dd>

External identifier of this OGC® document: <[http://www.opengis.net/doc/[{doc-type/}]{standard}/{m.n}](http://www.opengis.net/def/%5b%7bdoc-type/%7d%5d%7bstandard%7d/%7bm.n%7d)>

Internal reference number of this OGC® document:    17-007

Version: 0.12

Category: OGC® Implementation Specification

Editor:   Andreas Matheus

OGC Web Services Security

**Copyright notice**

Copyright © 2018 Open Geospatial Consortium  
To obtain additional rights of use, visit <http://www.opengeospatial.org/legal/>.

**Warning**

This document is not an OGC Standard. This document is distributed for review and comment. This document is subject to change without notice and may not be referred to as an OGC Standard.

Document type:    OGC® Standard

Document subtype:    if applicable

Document stage:    Draft

Document language:  English

Recipients of this document are invited to submit, with their comments, notification of any relevant patent rights of which they are aware and to provide supporting documentation.

License Agreement

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

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

THIS LICENSE IS A COPYRIGHT LICENSE ONLY, AND DOES NOT CONVEY ANY RIGHTS UNDER ANY PATENTS THAT MAY BE IN FORCE ANYWHERE IN THE WORLD.

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

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

Except as contained in this notice, the name of LICENSOR or of any other holder of a copyright in all or part of the Intellectual Property shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Intellectual Property without prior written authorization of LICENSOR or such copyright holder. LICENSOR is and shall at all times be the sole entity that may authorize you or any third party to use certification marks, trademarks or other special designations to indicate compliance with any LICENSOR standards or specifications. This Agreement is governed by the laws of the Commonwealth of Massachusetts. The application to this Agreement of the United Nations Convention on Contracts for the International Sale of Goods is hereby expressly excluded. In the event any provision of this Agreement shall be deemed unenforceable, void or invalid, such provision shall be modified so as to make it valid and enforceable, and as so modified the entire Agreement shall remain in full force and effect. No decision, action or inaction by LICENSOR shall be construed to be a waiver of any rights or remedies available to it.

Contents

1. Scope (informative) 11

2. References 14

2.1 Normative references 14

2.2 Informative references 15

2.3 Bibliography 16

3. Terms and Definitions 17

4. Conventions 19

4.1 Identifiers for this Standard 19

4.2 Versioning 19

4.3 Backwards Compatibility 19

5. Use Cases (informative) 21

5.1 Use Case 0: Public Service / Public Data / Public Catalogue / Public Communication 21

5.2 Use Case I: Authenticated Public Service / Public Data / Public Catalogue / Secure Communication 21

5.3 Use Case II: Protected Service / Open Data / Public Catalogue / Secure Communication 22

5.4 Use Case III: Protected Service / Private Data / Public Catalogue 22

5.5 Use Case IV: Protected Service / Private Data / Protected Catalogue / Secure Communication 23

6. Conformance 24

6.1 Conformance class “OWS Common” 25

6.2 Conformance class “WMS 1.1.1” 26

6.3 Conformance class “WMS 1.3.0” 28

6.4 Conformance class “Common Security” 29

7. Conformance for a Service Implementation 31

7.1 Identifier URLs and URNs 31

7.2 Requirements Class “HTTPS” 31

7.3 Requirements Class “W3C CORS” 32

7.4 Requirements Class “HTTP Method” 32

7.5 Requirements Class “Exception Handling” 33

7.6 Requirements Class “HTTP POST Content-Type” 35

7.7 Requirements Class “Access Control” 35

7.8 Requirements Class “WS-Security” 36

7.9 Requirements Class “SAML2” 37

7.10 Requirements Class “OpenID Connect” 37

7.11 Requirements Class “OpenAPI 3.0” 38

7.12 Requirements Class “Authentication” 38

7.12.1 Authentication via SAML 39

7.12.2 Authentication via OpenID Connect 40

8. Conformance for a Client Implementation 41

8.1 General Client Functionality 41

8.2 Client Requirements Classes for “Parsing Annotated Capabilities” 42

8.2.1 Parse Security Annotations based on OWS Common Schema 42

8.2.2 Parse Security Annotations based on WMS 1.3.0 Schema 42

8.2.3 Parse Security Annotations based on WMS 1.1.1 DTD 42

8.2.4 Working with Complete Capabilities 43

8.2.5 Working with Partial Capabilities 43

8.3 Client Requirements Class “HTTPS” 44

8.4 Client Requirements Class “Exception Handling” 44

9. Authentication Codelist 46

9.1 Authentication Codelist URL 46

9.2 Authentication Codes 47

9.2.1 Authentication Codes defined by IETF 47

9.2.2 Authentication Codes defined by OASIS 47

9.2.3 Authentication Codes defined by OGC 48

9.3 Authentication Code Resolving 49

9.4 Initial Authentication Codelist 49

10. Security Considerations (informative) 50

10.1 Thread “Tampered Capabilities” 50

10.1.1 Mitigations to this threat: 50

10.1.2 Approaches to provide a digital signature to the Capabilities document 50

10.1.3 Recommendation 51

B.1 Conformance Test Activity Diagram 54

B.2 Conformance Tests for the Service-Side 56

B.2.1 Conformance Test “Capabilities Structure” 56

B.2.2 Conformance Test “HTTPS” 56

B.2.3 Conformance Test “HTTP 1/1” 57

B.2.4 Conformance Test “W3C CORS” 57

B.2.5 Conformance Test “Exception Handling” 57

B.2.6 Conformance Test “Authentication” 57

B.2.7 Conformance Test “Access Control” 58

B.2.8 Conformance Test “WS-Policy” 58

B.2.9 Conformance Test “SAML2” 58

B.2.10 Conformance Test “OpenAPI” 59

B.2.11 Conformance Test “HTTP POST XML Content-Type” 59

B.3 Conformance Tests for the Client-Side 59

B.3.1 Conformance Test “Parse Capabilities document” 59

B.3.2 Conformance Test “HTTPS” 60

B.4 Abstract Conformance Tests for the Client-Side 61

**Figures** Page

Figure 1 : Relationships between conformance classes and their requirements classes (simplified overview) 12

Figure 2 : Security Proxy to make Geoserver deployment compliant with this standard 13

Figure 3 : Conformance Classes of this standard 25

Figure 4 : OWS Common Exception Codes [OWS Common 1.2] 33

Figure 5 : Processing Metaphor how to achieve interoperability with this standard 34

Figure 6 : Conformance Classes defined for the client 41

Figure 7:Testing mandatory dependencies of Requirements Classes 55

Figure 8:Testing of independent Requirements Classes 55

Figure 9:Testing of independent Requirements Classes 56

Figure 10:Illustration of processing the Capabilities document 60

Figure 11: Illustration of the test for the mandatory requirements class: HTTPS 61

**Tables** Page

*Table 1: Expressing conformance towards the conformance class “OWS Common”* 26

Table 2:Expressing conformance towards the conformance class “WMS 1.1.1” 26

Table 3:WMS 1.1.1 definition of the SecurityExtendedCapabilities element in DTD 27

Table 4: Use of SecurityExtendedElement with WMS 1.1.1 Capabilities **and no other** vendor specific capabilities 28

Table 5: Definition of example vendor specific capabilities for WMS 1.1.1 Capabilities 28

Table 6: Use of SecurityExtendedElement with WMS 1.1.1 Capabilities **and** vendor specific capabilities 28

Table 7:Expressing conformance towards the conformance class “WMS 1.3.0” 29

Table 8: XML Schema Definition of SecurityExtendedCapabilities element for WMS 1.3.0 29

Table 9:Normative annotation for expressing compliance with HTTPS 32

Table 10:Normative annotation for expressing compliance with W3C CORS 32

Table 11:Informative example indicating support for the methods GET, POST and OPTIONS 32

Table 12:Normative annotation for expressing compliance with Exception Handling 33

Table 12:Normative annotation expressing compliance with HTTP POST Content-Type 35

Table 13:Normative annotation expressing compliance with Access Control 36

Table 14:Normative annotation expressing compliance with WS-Policy 36

Table 15:Normative annotation expressing compliance with SAML2 37

Table 15:Normative annotation expressing compliance with OpenID Connect 37

Table 16:Normative annotation expressing compliance with OpenAPI 38

Table 17:Example annotation expressing OAuth2 requirement with OpenAPI 38

Table 18: Example annotation expressing client authentication via certificate 39

Table 19: Section names and their content [OGC #06-121r9, p.25] 43

Table 20:IETF defined authentication methods 47

Table 21: List of Authentication methods defined for SAML2 48

Table 22: OGC identifier for IETF authentication method 49

Table 23:Example ExtendedCapabilities element including a Signature element 51

Table 24: Requirements Classes overview (1) Mandatory if CORS is advertised; 2) Mandatory if CORS or Authentication or Access Control is advertised) 54

Abstract

Implementations of OGC Web Services have implemented Information Assurance (IA) Controls for years. However, these implementations break interoperability, as they are not standardized by OGC Web Service standards. Interoperability between secured OGC Web Services and clients is limited to systems custom built to work with an IA implementation.

The goal of the OWS Common Security Standard is to allow the implementation of IA controls and to advertise its existence in an interoperable way with minimal impact to existing implementations using a backwards-compatible approach. That goal is being pursued in two ways:

1. Identify and document IA Controls for supporting authentication in a register maintained through the OGC.
2. Specify how a service can advertise their IA controls through the Service Capabilities Document.

This OGC Standard specifies how conformant OWS Services shall advertise their IA Controls. It also describes the governance process for the IA Control registers, examples of register contents, and descriptions on how this information should be used.

Next, this standard defines conformance classes and requirements classes to be used for reaching compliance and their validation via conformance tests.

Finally, this standard defines client behavior to ensure interoperable processing of advertised security controls.

Keywords

The following are keywords to be used by search engines and document catalogues.

Security, OWS Common, OWS Common Security, OGC Web Services Security, OAuth2, OpenID Connect, SAML2, HTTPS, WS-Security, WS-Policy, SOAP, WMS, WFS, WCS, WMTS, XACML, GeoXACML, Authentication, Access Control

Preface

This is version 1.0 of the OGC Web Services Security standard submitted to the Technical Committee by the OWS Common – Security Standards Working Group.

It standardizes an annotation mechanism for Capabilities documents or responses to the GetCapabilities request, ensuring interoperability between a secured OGC Web Service instance and a client application. It further overrides existing HTTP protocol limitations and exception handling from existing OGC Web Services standards for the purpose to achieve interoperability with main stream IT security standards and their implementations. To achieve this, no changes to existing OGC Abstract specifications and OGC Web Services standards are required.

This standard has no direct precursor document but can be seen as the result of previous OGC Testbeds, documented in different Testbed Engineering Reports ([34], [35], [36]).

The annotation approach and the service behavior regarding security is standardized in a backwards compatible way to ensure it can be applied to **existing** OGC Web Services with no change to the deployments.

Uptake of the standardized approach as a build-in to **new** OGC Web Services standards will ensure security interoperability. An applicable candidate standard is currently prepared by the WFS/FES Standards Working Group.

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.*

Submitting organizations

The following organizations submitted this Document to the Open Geospatial Consortium (OGC):

* University of the Bundeswehr
* NGA
* Geonovum
* WiSC
* DigitalGlobe

Submitters

All questions regarding this submission should be directed to the editor or the submitters:

|  |  |
| --- | --- |
| Name | Affiliation |
| Andreas Matheus | University of the Bundeswehr |
| Dave Wesloh | NGA |
| Frank Terpstra | Geonovum |
| Chuck Heazel | WiSC |
| Michael Leedahl | DigitalGlobe |

# Scope (informative)

This standard applies to a deployed OGC Web Service instance which:

* Exchanges requests and responses using HTTPS;
* Provides a security-annotated version of the Capabilities document (or short “annotated Capabilities”) as specified in section 8.

A security-annotated Capabilities document is one, which uses the <Constraint> element(s) to express the existence of security controls for operations of the service instance and compliance with security standards. Applying the tests as defined in the Annex “Conformance Tests” can validate compliance. Basically, a client loading the capabilities document and parsing for the <AccessConstraint> element can determine the compliance of the service instance. The string value of this element contains the identifier of the implemented conformance class.

*Note: How the client obtains the security-annotated capabilities is out of scope for this standard.*

This standard defines one common conformance class and three service specific conformance classes. The service specific conformance classes address how the requirements of the common conformance class are implemented for WMS 1.1, WMS 1.3 and OWS Common..

Conformance Class “**Common** **Security**”: This abstract conformance class ensures that the service instance is implementing HTTPS as specified by the IETF standards. This is the minimum capability required to be interoperable with mainstream IT security technology. It bundles mandatory requirements classes that address issues which inhibit operating an OGC compliant web service over HTTPS. This conformance class also provides a method for the client to use either the service exception handling compliant with OWS Common (for the OWS layer) or exception handling compliant with the HTTP specification for the security layer. This method ensures the elimination of unnecessary limitations regarding the HTTP protocol and exception handling from OWS Common and other OGC Web Service standards. It also defines other requirements classes that are optional to be implemented that address the description of further IAs to be able to convey as much information on existing security controls as possible.

The following conformance classes are concerned with how to apply the actual security annotations to the capabilities document that is associated with a service endpoint. There are three different conformance classes, because the way to insert security annotations into the capabilities document differs based on the underlying XML schema or DTD.

Conformance Class “**Service based on OWS Common**”: This conformance class defines how the security metadata is to be inserted into the OGC Web Service capabilities document for any service instance based on OWS Common XML schema.

Conformance Class “**WMS 1.1.1**”: This conformance class defines how the security metadata is to be inserted into the OGC Web Service capabilities document for a WMS 1.1.0 service instance based on the WMS 1.1.1 DTD.

Conformance Class “**WMS 1.3.0**”: This conformance class defines how the security metadata is to be inserted into the OGC Web Service capabilities document for a WMS 1.3.0 service instance based on the WMS 1.3.0 XML schema.

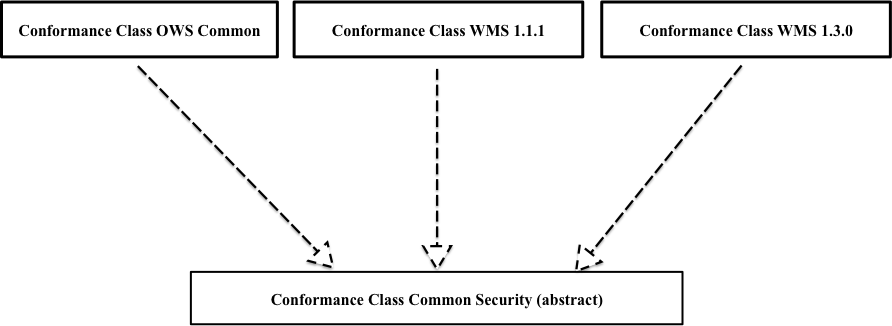


Figure 1 : Relationships between conformance classes and their requirements classes (simplified overview)

*Note: Implication to be compliant with this standard means that some requirements from existing OGC specifications can be superseded. Because this standard defines the compliance, it is NOT required to incorporate the requirements into the existing standards; so no change requests to the existing OGC standards are required!*

The following OGC standards are directly affected:

1. OWS Common 1.1.0, OGC 06-121r3

*OGC Web Services Common Specification*, *OGC® Implementation Standard*

1. OWS Common 2.0.0, OGC 06-121r9

*OGC Web Services Common Specification*, *OGC® Implementation Standard*

1. WMS 1.1.1, OGC 01-068r3

*Web Map Service* *Implementation Specification*

1. WMS 1.3.0, OGC 06-042

*OpenGIS Web Map Service (WMS) Implementation Specification*

The following OGC standards are impacted because they inherit from OWS Common.

1. WFS 1.1.0, OGC 04-094

*OpenGIS Web Feature Service (WFS) Implementation Specification*

1. WFS 2.0, OGC 09-025r1

*OpenGIS Web Feature Service 2.0 Interface Standard (also ISO 19142)*

1. WFS 2.0.2, OGC 09-025r2

*OGC® Web Feature Service 2.0 Interface Standard – With Corrigendum*

1. WCS 2.0, OGC 09-147r3

OGC® *WCS Interface Standard - KVP Protocol Binding Extension, version 1.0.1*

1. WCS 2.0, OGC 09-148r1

*OGC® WCS - XML/POST Protocol Binding Extension, version 1.0.0*

1. WCS 2.0, OGC 09-149r1

*OGC® Web Coverage Service 2.0 Interface Standard - XML/SOAP Protocol Binding Extension, version 1.0.0*

1. WMTS 1.0, OGC 07-057r7

*OpenGIS Web Map Tile Service Implementation Standard*

1. WPS 1.0.0, OGC 05-007r7

*Web Processing Service*

1. WPS 2.0, OGC 14-065

*OGC® WPS 2.0 Interface Standard*

1. SOS 2.0, OGC 12-006

*OGC® Sensor Observation Service Interface Standard*

1. SPS 2.0, OGC 09-000

*OGC® Sensor Planning Service Implementation Standard*

1. CSW 2.0.2, OGC 07-006r1

*OpenGIS Catalogue Service Implementation Specification*

1. CSW 3.0, OGC 12-176r7

*OGC® Catalogue Services 3.0 Specification - HTTP Protocol Binding*

One typical way to realize compliance without modifying the existing service implementations is via a security gateway or proxy. This proxy would have the duty to implement the compliance by injecting security annotations into the GetCapabilities response, operate the service endpoint on HTTPS but also support all HTTP methods and correct the OWS Common error code handling. Testbed 12 ER OGC16-048 describes a practical approach of a security proxy.



Figure 2 : Security Proxy to make Geoserver deployment compliant with this standard

Two different security proxies as proof of concept got deployed during Testbed 13:

* Security Proxy “OAuth2”: <https://rs.tb13.secure-dimensions.de/geoserver/web/>
* Security Proxy “SAML2”: <https://sp.tb13.secure-dimensions.de/geoserver/web/>

# References

The following normative documents contain provisions that, through reference in this text, constitute provisions of this document. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. For undated references, the latest edition of the normative document referred to applies.

## Normative references

1. HTTP Authentication: Basic and Digest Access Authentication

<https://tools.ietf.org/html/rfc2617>

1. Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing[[1]](#footnote-1)

<https://tools.ietf.org/html/rfc7230>

1. HTTP over TLS – RFC 2818

<https://tools.ietf.org/html/rfc2818>

1. OpenID Connect

<http://openid.net/specs/openid-connect-core-1_0.html>

1. OpenID Connect Discovery

<https://openid.net/specs/openid-connect-discovery-1_0.html>

1. Well Known URIs - IANA

<https://www.iana.org/assignments/well-known-uris/well-known-uris.xhtml>

1. The OAuth 2.0 Authorization Framework

<https://tools.ietf.org/html/rfc6749>

1. The OAuth 2.0 Authorization Framework: Bearer Token Usage

<https://tools.ietf.org/html/rfc6750>

1. Security Assertion Markup Language (SAML) v2.0

<https://www.oasis-open.org/standards#samlv2.0>

1. Authentication Context for the OASIS, Security Assertion Markup Language (SAML) V2.0, OASIS Standard, 15 March 2005

<https://www.oasis-open.org/standards#samlv2.0>

1. eXtensible Access Control Markup Language

<https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml>

1. Geospatial eXtensible Access Control Markup Language

<http://www.opengeospatial.org/standards/geoxacml>

1. W3C CORS (Common Object Resource Sharing)

<https://www.w3.org/TR/cors/>

1. OWS Common 1.0 Operations Metadata XSD <http://schemas.opengis.net/ows/1.0.0/owsOperationsMetadata.xsd>
2. OWS Common 1.1.0 Operations Metadata XSD

<http://schemas.opengis.net/ows/1.1.0/owsOperationsMetadata.xsd>

1. OWS Common 2.0 Operations Metadata XSD

<http://schemas.opengis.net/ows/2.0/owsOperationsMetadata.xsd>

1. WMS 1.1.1 DTD

<http://schemas.opengis.net/wms/1.1.1/capabilities_1_1_1.dtd>

1. WMS 1.3.0 XSD

<http://schemas.opengis.net/wms/1.3.0/capabilities_1_3_0.xsd>

1. OpenAPI 3.0

<https://github.com/OAI/OpenAPI-Specification/blob/master/versions/3.0.0.md>

## Informative references

1. WFS 1.1.0, OGC 04-094

*OpenGIS Web Feature Service (WFS) Implementation Specification*

1. WFS 2.0, OGC 09-025r1

*OpenGIS Web Feature Service 2.0 Interface Standard (also ISO 19142)*

1. WFS 2.0.2, OGC 09-025r2

*OGC® Web Feature Service 2.0 Interface Standard – With Corrigendum*

1. WCS 2.0, OGC 09-147r3

OGC® *WCS Interface Standard - KVP Protocol Binding Extension, version 1.0.1*

1. WCS 2.0, OGC 09-148r1

*OGC® WCS - XML/POST Protocol Binding Extension, version 1.0.0*

1. WCS 2.0, OGC 09-149r1

*OGC® Web Coverage Service 2.0 Interface Standard - XML/SOAP Protocol Binding Extension, version 1.0.0*

1. WMTS 1.0, OGC 07-057r7

*OpenGIS Web Map Tile Service Implementation Standard*

1. WPS 2.0, OGC 14-065

*OGC® WPS 2.0 Interface Standard*

1. SOS 2.0, OGC 12-006

*OGC® Sensor Observation Service Interface Standard*

1. SPS 2.0, OGC 09-000

*OGC® Sensor Planning Service Implementation Standard*

1. CSW 2.0.2, OGC 07-006r1

*OpenGIS Catalogue Service Implementation Specification*

1. CSW 3.0, OGC 12-176r7

*OGC® Catalogue Services 3.0 Specification - HTTP Protocol Binding*Bibliography

1. OWS Common 1.1.0, OGC 06-121r3

*OGC Web Services Common Specification*, *OGC® Implementation Standard*

1. OWS Common 2.0.0, OGC 06-121r9

*OGC Web Services Common Specification*, *OGC® Implementation Standard*

## Bibliography

1. OGC 15-022: OGC® Testbed 11 Engineering Report: Implementing Common Security Across the OGC Suite of Service Standards <https://portal.opengeospatial.org/files/?artifact_id=63312>
2. OGC 16-048r1: OGC® Testbed 12 Engineering Report: OWS Common Security Extension

<http://docs.opengeospatial.org/per/16-048r1.html>

1. OGC 17-021: OGC® Testbed 13 Engineering Report: Security

# Terms and Definitions

This document uses the terms defined in Sub-clause 5.3 of [OGC 06-121r9], which is based on the ISO/IEC Directives, Part 2, Rules for the structure and drafting of International Standards. In particular, the word “shall” (not “must”) is the verb form used to indicate a requirement to be strictly followed to conform to this standard.

For the purposes of this document, the following additional terms and definitions apply.

ADR XACML Authorization Decision Request

CSW Catalogue Service for the Web

DGIWG Defence Geospatial Information Working Group

DTD Document Type Definition

GMX ISO TC211 XML namespace http://www.isotc211.org/2005/gmx

GeoXACML Geospatial eXtensible Access Control Markup Language

HTTP Hypertext Transfer Protocol

HTTPS Hypertext Transfer Protocol Secure

IA Information Assurance

IANA Internet Assigned Numbers Authority

IETF Internet Engineering Task Force

ISO International Organization for Standardization

JSON JavaScript Object Notation

MIME Multipurpose Internet Mail Extensions

OASIS Organization for the Advancement of Structured Information

OAuth OAuth

OGC Open Geospatial Consortium

OpenID Connect OpenID Connect

OWS OGC Web Service

PAP XACML Policy Administration Point

RFC Request For Comments

SAML Security Assertion Markup Language

SDI Spatial Data Infrastructure

SOS Sensor Observation Service

SPS Sensor Planning Service

SWG Standards Working Group

URL Uniform Resource Locator

URN Uniform Resource Name

W3C World Wide Web Consortium

WCS Web Coverage Service

WFS Web Feature Service

WMS Web Map Service

WMTS Web Map Tile Service

WPS Web Processing Service

XACML eXtensible Access Control Markup Language

XML eXtensible Markup Language

# Conventions

This sections provides details and examples for any conventions used in the document. Examples of conventions are symbols, abbreviations, use of XML schema, or special notes regarding how to read the document.

All sections in this document are normative unless otherwise indicated.

## Identifiers for this Standard

The normative provision of this specification is available by this URL

https://www.opengis.net/spec/ogc-web-services-security/1.0

All Conformance Classes in this specification are identified by a resolvable URI with this base:

https://www.opengis.net/def/security/1.0/cc

All Requirements Classes in this specification are identified by a URN identifier with this base:

urn:ogc:def:security:1.0:rc

Requirement Class URN identifiers are used in the <ows:Constraint> element to identify the nature of each constraint. These URNs serve as identifiers only and are not expected to be resolvable..

Requirements Classes in this specification are also identified by a URL with this base:

https://www.opengis.net/def/security/1.0/rc

Requirement Class URL identifiers are used in the <ows:Meaning> element which is a child element of <ows:Constraint>. The purpose of this element is to provide a resolvable resource identifier for the definition of the Requirement Class.

## Versioning

The version of this standard can be maintained independent from the version of a Conformance or Requirements Class. Including the version in the standard URI as well as in the URIs of the Conformance and Requirements Classes ensures this.

## Backwards Compatibility

This standard leverages the existing OWS Constraint element to enable the annotation of service capabilities with IA controls present at operation(s) of the service instance. This solution ensures backwards compatibility, as a capabilities document that includes security annotations is valid against the existing OWS Common schema. For WMS 1.1.1 and WMS 1.3.0, which do not make use of OWS Common, a similar approach is standardized ensuring backwards compatibility.

All approaches ensure that a service endpoint can, independent from anything else, provide capabilities with security annotations. Client applications not capable of interpreting the annotation will simply ignore it but will not return expected results. Clients however, that properly interpret the security annotation can use that information to ensure interoperable functioning with secured OGC Web Services.

# Use Cases (informative)

The following use cases provide an overview on how to use annotated Capabilities. The term annotated Capabilities refers to the extension to a Capabilities document as defined by this standard. In the OGC world of services, the paradigm Publish-Find-Bind is based on a service / data provider describing the service with ISO metadata (ISO 19139) and registering that description in a catalog. From the catalog, the metadata for the data and for the service (providing access to the data) can be found. For this specification, it is out of scope to describe the security annotations in ISO 19139 metadata. But it is important to note that the service metadata, in the catalogue, contains a link to the service capabilities. This is usually the GetCapabilities operation of the actual service. For the use of annotated capabilities, this link must be freely accessible and therefore perhaps is a different URL (not the GetCapabilities operation of the actual secured service). Knowledge of the service “content” is restricted therefore public access to the annotated capabilities must be in compliance with the “need-to-know” principle. The following use cases illustrate the different constellations that might exist:

## Use Case 0: Public Service / Public Data / Public Catalogue / Public Communication

This is the current standardized use of OGC Web Services – no security. Therefore, there are no implications for this specification.

Services can be discovered through a catalogue that has no security.

The client application can bind to the service instance via the Capabilities document. This reflects common practice for today’s SDIs.

## Use Case I: Authenticated Public Service / Public Data / Public Catalogue / Secure Communication

This is another common use of OGC Web Services – no client authentication or authorization is used – only server authentication and a protected communication channel. The server authentication assures the client it has the authentic source of the public data. The secure communication channel ensures privacy to the client, outsiders cannot determine what data the client is receiving.

Therefore, there are implications for this standard, as server authentication methods are not currently specified in OGC Web Services. In the most common case (HTTPS) server authentication and protected communications is not strictly compliant with existing OGC Web Services specifications. This specification corrects that oversight.

The client application can bind to the service instance via the Capabilities document. This reflects common practice for today’s SDIs. The client needs to support the server authentication method while accessing the public data on the server, e.g. support for HTTPS.

## Use Case II: Protected Service / Open Data / Public Catalogue / Secure Communication

For this use case, a public catalog holds data and service metadata for a protected service. The public access to the catalogue implies that authentication is not required. It is therefore not possible to provide user specific responses. Therefore, the user can discover service metadata that point to the public version of the annotated Capabilities.

Open data implies that the need-to-know principle does not apply. Therefore, the annotated Capabilities, accessible from a free and open URL, must contain all information relevant for the client application to bind to the service. In particular, this requires that the “content” section of the annotated capabilities list all accessible resources.

Any service that fits the description of use case II should include a “content” section in the annotated Capabilities that lists all accessible resources.

A client application that has implemented all the published security requirements is able to bind to the service and work with the Capabilities document in the usual fashion – likewise to use case 0.

The main difference with use case I is that both user and server are authenticated. Data is open but anonymous access is not allowed.

## Use Case III: Protected Service / Private Data / Public Catalogue

This use case is similar to use case I with the exception that the service provides access to private data. This must trigger the need-to-know constraint, which implies that the “content” section of the annotated Capabilities is empty or other indications exist from which the client application can conclude that the service is providing private resources. One example is the Defence Geospatial Information Working Group (DGIWG) profile for the WFS Web Service that uses the <AccessContraints> element of the Capabilities to indicate the highest level of classification.

In this case, the client application can only expect that the annotated Capabilities contain the <OperationsMetadata> element. It is therefore required for the client application to first determine its ability to comply with the security requirements outlined in the annotations. In cases where the application is interoperable with the security on the protected service instance, it must execute the advertised GetCapabilities operation and comply with the security constraints to fetch the full Capabilities that include the “Content” section.

If the annotated Capabilities document does not contain a “content” section (<Layer> element for WMS or a <FeatureTypeList> element for a WFS or a <Contents> element for WMTS, WCS or WPS) or the <AccessContraints> element does not contain the literal “None”, the application shall execute the GetCapabilities operation advertised (within the Capabilities document) to fetch the full service capabilities.

*Note: This assumes that the client is able to function on the advertised security requirements for the GetCapabilities operation.*

## Use Case IV: Protected Service / Private Data / Protected Catalogue / Secure Communication

This use case would assume that managed access to the catalogue is in place. It also assumes that somehow the user got access to the Catalogue – perhaps using a Web Browser – but for sure another client application, different from the one that is to bind to the protected service. The catalog would return information to enable the actual implementation to obtain the service capabilities document. One way could be to return capabilities with security annotations.

*Note: The credentials to access the catalogue might be different from the credentials to access the service.*

# Conformance

Conformance with this standard shall be checked using the applicable requirements of each conformance class. Compliance can be verified via the normative set of conformance tests, defined in Annex B.

The framework, concepts, and methodology for testing, and the criteria to be achieved to claim conformance are specified in the OGC Compliance Testing Policies and Procedures and the OGC Compliance Testing web site.

All requirements and conformance classes described in this document are owned by the standard(s) identified.

*Note: Special attention should be taken as this standard supersedes some requirements defined in OWS Common v1.1.0 and v2.0 as well as existing OGC Web Service Implementation Specifications where applicable to ensure establishing of interoperable secured service instances.*

The described approach to annotate capabilities documents for secured service instances is backwards compatible as only existing OWS Common elements from the Capabilities document structure are leveraged. This backwards compatibility enforces a particular kind of use of the element <ows:Constraint>, slightly different from the original intent.

In order to conform to this OGCstandard, a software implementation can be compliant to one of the three implementable conformance classes defined in the next sections.

1. A service SHALL advertise compliance with a conformance class from this standard by inserting the URI for the conformance class into the element <AccessContraints>, which commonly exists in any OGC Web Service capabilities document.

The following figure provides an overview of the requirements classes and the conformance classes defined in this standard.

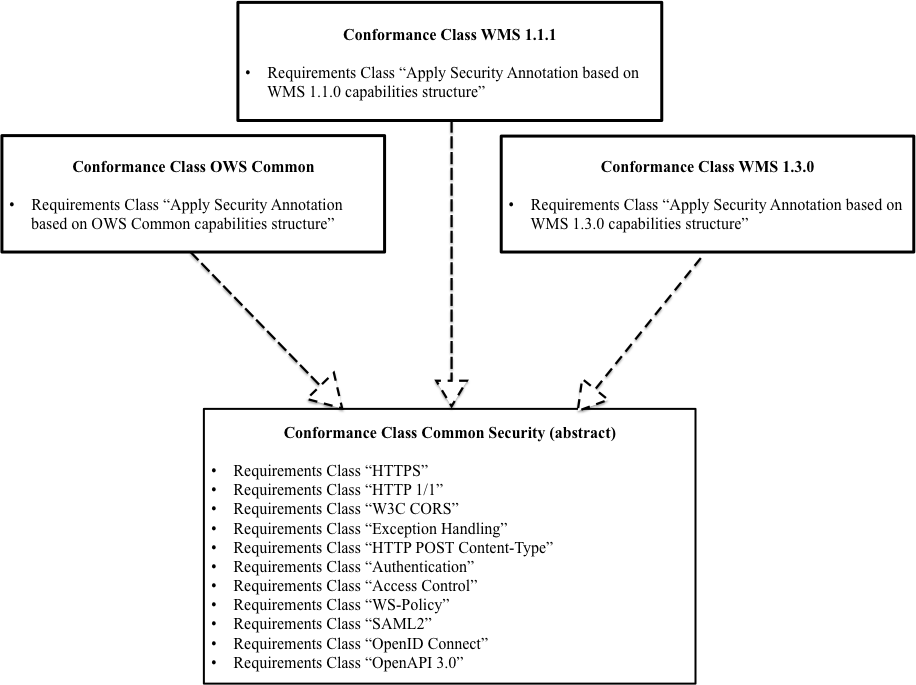


Figure 3 : Conformance Classes of this standard

## Conformance class “OWS Common”

This conformance class has this identifier: **https://www.opengis.net/def/security/1.0/cc/owsCommon**

1. This conformance class SHALL be applied if the service endpoint provides security anno­tations in the capabilities document and the capabilities are based on OWS Common Schema.
2. In order for the service to be compliant with this conformance class, the AccessConstraint element SHALL contain the identifier for this conformance class.

|  |
| --- |
| <AccessConstraints>  **https://www.opengis.net/def/security/1.0/cc/owsCommon**  </AccessConstraints> |

*Table 1: Expressing conformance towards the conformance class “OWS Common”*

1. In order for a service instance operation to be compliant, requirements from the conformance class **https://www.opengis.net/def/security/1.0/cc/commonSecurity** SHALL be implemented.
2. For Capabilities based on OWS Common Schema, the service instance inherits the Capabilities structure from OWS Common (any version). The security annotation SHALL use the <ows:Constraint> element as defined in the <ows:ServiceMetadataType> definition.

*Note: The ows:Constraint element is always identical independent from the OWS Common versions.*

1. The annotated Capabilities document SHALL be valid XML according to the underlying OWS Common schema.

## Conformance class “WMS 1.1.1”

This conformance class has this identifier: **https://www.opengis.net/def/security/1.0/cc/wms111**

1. This conformance class SHALL be applied if the service endpoint provides security anno­tations in the capabilities document and the capabilities are based on WMS 1.1.1 DTD.
2. In order for the service to be compliant with this conformance class, the AccessConstraint element SHALL contain the identifier for this conformance class.

|  |
| --- |
| <AccessConstraints>  **https://www.opengis.net/def/security/1.0/cc/wms111**  </AccessConstraints> |

Table 2:Expressing conformance towards the conformance class “WMS 1.1.1”

1. In order for a service instance operation to be compliant, requirements from the conformance class **https://www.opengis.net/def/security/1.0/cc/commonSecurity** SHALL be implemented.
2. For Capabilities based on WMS 1.1.1 DTD, the following DTD SHALL be used to provide the security annotation.

|  |
| --- |
| <!DOCTYPE WMT\_MS\_Capabilities SYSTEM "http://schemas.opengis.net/wms/1.1.1/WMS\_MS\_Capabilities.dtd"[ <!-- ============================================================== OWS Common Security Extension to annotate security  Definition of element ows:OperationsMetadata replicating the  definition from the OWS Common Schema to become available as DTD ============================================================== --> <!ELEMENT VendorSpecificCapabilities (ows\_security:SecurityExtendedCapabilities)>  <!ELEMENT ows\_security:SecurityExtendedCapabilities (ows:OperationsMetadata+)> <!ATTLIST ows\_security:SecurityExtendedCapabilities xmlns:ows\_security CDATA #FIXED "http://www.opengis.net/security/1.0">  <!ELEMENT ows:OperationsMetadata (ows:Operation\*)> <!ATTLIST ows:OperationsMetadata xmlns:ows CDATA #FIXED "http://www.opengis.net/ows/1.1">  <!ELEMENT ows:Operation (ows:DCP+) > <!ATTLIST ows:Operation name CDATA #REQUIRED>  <!ELEMENT ows:DCP (ows:HTTP) > <!ELEMENT ows:HTTP (ows:Get | ows:Post)+ >  <!ELEMENT ows:Get (ows:Constraint+)> <!ATTLIST ows:Get xmlns:xlink CDATA #FIXED "http://www.w3.org/1999/xlink" xlink:type CDATA #FIXED "simple" xlink:href CDATA #REQUIRED >  <!ELEMENT ows:Post (ows:Constraint+)> <!ATTLIST ows:Post xmlns:xlink CDATA #FIXED "http://www.w3.org/1999/xlink" xlink:type CDATA #FIXED "simple" xlink:href CDATA #REQUIRED >  <!ELEMENT ows:Constraint (ows:AllowedValues | ows:NoValues | ows:ValuesReference | ows:Meaning)+> <!ATTLIST ows:Constraint name CDATA #REQUIRED>  <!ELEMENT ows:AllowedValues (ows:Value+)>  <!ELEMENT ows:Value (#PCDATA)>  <!ELEMENT ows:NoValues EMPTY>  <!ELEMENT ows:ValuesReference (#PCDATA)> <!ATTLIST ows:ValuesReference reference CDATA #REQUIRED>  <!ELEMENT ows:Meaning (#PCDATA)> <!ATTLIST ows:Meaning reference CDATA #REQUIRED> ]> |

Table 3:WMS 1.1.1 definition of the SecurityExtendedCapabilities element in DTD

Installing the <ows:**SecurityExtendedCapabilities>** as a valid element into the Capabilities document, it must become a child to the element VendorSpecificCapabilities.

1. The <ows:SecurityExtendedCapabilities>element SHALL be added as a LAST child to the VendorSpecificCapabilities element.
2. If the actual capabilities document for the service instance does **not** include the element VendorSpecificCapabilities, then the capabilities document SHALL use the following DTD snippet to enable security annotation:

|  |
| --- |
| <!--  ============================================================== OWS Common Security Extension to annotate security requires adding the element ows\_security:**SecurityExtendedCapabilities** to your Vendor Specific Capabilities definition If you do not define your own VendorSpecificCapabiltiies then use this element <!ELEMENT VendorSpecificCapabilities (ows\_security:**SecurityExtendedCapabilities**) > ============================================================== -->  <!ELEMENT VendorSpecificCapabilities (ows\_security:**SecurityExtendedCapabilities**) > |

Table 4: Use of SecurityExtendedElement with WMS 1.1.1 Capabilities **and no other** vendor specific capabilities

1. For Capabilities where the service instance **does** include VendorSpecificCapabilities then the <ows\_security:SecurityExtendedCapabilities>element SHALL be included into the vendor specific capabilities definition.

Informative Example: Assuming the following existing VendorSpecificCapabilities

|  |
| --- |
| <!ELEMENT VendorSpecificCapabilities (Profiles) > <!ELEMENT Profiles (Profile\*) > <!ELEMENT Profile (#PCDATA) > |

Table 5: Definition of example vendor specific capabilities for WMS 1.1.1 Capabilities

then the security annotated version of the capabilities shall use the following DTD

|  |
| --- |
| <!--  ============================================================== OWS Common Security Extension to annotate security requires adding the element < ows\_security:**SecurityExtendedCapabilities>** to your Vendor Specific Capabilities definition If you do not define your own VendorSpecificCapabiltiies then use this element <!ELEMENT VendorSpecificCapabilities (<ows\_security:**SecurityExtendedCapabilities>**) > ============================================================== -->  <!ELEMENT VendorSpecificCapabilities (Profiles, ows\_security:**SecurityExtendedCapabilities**) > <!ELEMENT Profiles (Profile\*) > <!ELEMENT Profile (#PCDATA) > |

Table 6: Use of SecurityExtendedElement with WMS 1.1.1 Capabilities **and** vendor specific capabilities

1. The <ows:OperationsMetadata> elements of the <ows:SecurityExtendedCapabilities> SHALL contain all operations metadata for the secured service endpoint.

*Note: This might be a duplication of the operations metadata originally contained in the Capabilities document.*

1. The annotated Capabilities document SHALL be valid XML according to the underlying WMS 1.1.0 DTD.

## Conformance class “WMS 1.3.0”

This conformance class has this identifier: **https://www.opengis.net/def/security/1.0/cc/wms130**

1. This conformance class SHALL be applied if the service endpoint provides security anno­tations in the capabilities document and the capabilities are based on WMS 1.3.0 Capabilities Schema.
2. In order for the service to be compliant with this conformance class, the AccessConstraint element SHALL contain the identifier for this conformance class.

|  |
| --- |
| <AccessConstraints>  **https://www.opengis.net/def/security/1.0/cc/wms130**  </AccessConstraints> |

Table 7:Expressing conformance towards the conformance class “WMS 1.3.0”

1. In order for a service instance operation to be compliant, requirements from the conformance class **https://www.opengis.net/def/security/1.0/cc/commonSecurity** SHALL be implemented.
2. For Capabilities based on WMS 1.3.0 Schema, the service instance SHALL use the following schema definition as a substitution for the \_ExtendedCapabilities element for expressing the security annotation(s).

|  |
| --- |
| <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> |

Table 8: XML Schema Definition of SecurityExtendedCapabilities element for WMS 1.3.0

1. The <ows:OperationsMetadata> elements of the <ows:SecurityExtendedCapabilities> SHALL contain all operations metadata for the secured service endpoint.

*Note: This might be a duplication of the operations metadata originally contained in the Capabilities document.*

1. The annotated Capabilities document SHALL be valid XML according to the underlying WMS 1.3.0 Schema.

## Conformance class “Common Security”

This conformance class has this identifier: **https://www.opengis.net/def/security/1.0/cc/commonSecurity**

*Note: This conformance class is abstract. It cannot be used by itself, as it does not define how the security annotations are to be inserted into a particular Capabilities structure.*

1. A service instance operation SHALL implement the following mandatory Requirements Class urn:ogc:def:security:1.0:rc:https.
2. A service operation SHALL implement one or many of the following optional Requirements Class(es) to express the existence of a particular IA: urn:ogc:def:security:1.0:rc:authentication, urn:ogc:def:security:1.0:rc:authorization, or urn:ogc:def:security:1.0:rc:policy.
3. A service operation SHALL implement one or many of the following optional Requirements Class(es) to express support for a particular security feature: urn:ogc:def:security:1.0:rc:http-methods, urn:ogc:def:security:1.0:rc:exception-handling, urn:ogc:def:security:1:0:rc:content-type, or urn:ogc:def:security:1.0:rc:cors.
4. A service operation SHALL implement one or many of the following optional Requirements Class(es) to provide additional information for particular purpose: urn:ogc:def:security:1.0:rc:openapi, urn:ogc:def:security:1.0:rc:authentication:oidc, or urn:ogc:def:security:1.0:rc:authentication:saml2.
5. The Capabilities document returned by the GetCapabilities operation outlined in the annotated capabilities SHALL return all sections requested by the client, including the “content” section.

# Conformance for a Service Implementation

The following requirements classes define what a service endpoint must implement to be compliant. Usually, this is two-fold: (i) What is the actual functionality that the service must provide and (ii) How shall the annotation be done to indicate that the functionality is implemented.

## Identifier URLs and URNs

This standard uses URNs for the definition of requirement classes that are to be used in the name attribute of the <ows:Constraint> element. The use of a URN here is sufficient, because they are used for comparison only.

Each requirement class also has a URL that can be used optionally in the <ows:Meaning> element in those cases where a resolving of the definition is desired.

The URIs are to be registered with the OGC Naming Authority. The OWS Common – Security Standards Working Group (SWG) triggers this action following an approved submission.

1. Other URLs and URNs can be defined but SHALL be submitted via an OGC Change Request to the OWS Common – Security SWG for consideration.

In order to enable the service endpoint to specify the security annotations for its operations in the capabilities document, the following normative base URN shall be used

Base URL for Requirements Classes:

https://www.opengis.net/def/security/1.0/rc

Base URN for Requirement Classes:

urn:ogc:def:security:1.0:rc

1. In the case where the implementor wishes to provide a resolvable URL of the URN, the <ows:Meaning> element and the ows:reference attribute SHALL be used to provide that resolvable URL.

## Requirements Class “HTTPS”

URL Identifier: https://www.opengis.net/def/security/1.0/rc/https

URN Identifier: urn:ogc:def:security:1.0:rc:https

1. This requirements class SHALL be applied using the ows:Constraint identifier if the service endpoint is deployed on HTTP over TLS, or communicates via a trusted channel.
2. The URN identifier urn:ogc:def:security:1.0:rc:https SHALL be used to identify the <ows:Constraint> element. The only valid sub-element SHALL be <ows:NoValues>. The following is the normative definition.

|  |
| --- |
| <ows:Constraint name="urn:ogc:def:security:1.0:rc:https">  <ows:NoValues/>  </ows:Constraint> |

Table 9:Normative annotation for expressing compliance with HTTPS

## Requirements Class “W3C CORS”

URL Identifier: https://www.opengis.net/def/security/1.0/rc/cors

URN Idenitifier: urn:ogc:def:security:1.0:rc:cors

1. The URN identifier urn:ogc:def:security:cors SHALL be used for the <Constraint> element to signal that the service endpoint operation is compliant with the W3C recommendation “Cross Origin Resource Sharing” (see normative references). The sub-element SHALL be <ows:NoValues>.

|  |
| --- |
| <ows:Constraint name="urn:ogc:def:security:1.0:rc:http-methods">  <ows:NoValues/>  </ows:Constraint> |

Table 10:Normative annotation for expressing compliance with W3C CORS

## Requirements Class “HTTP Method”

URL Identifier: https://www.opengis.net/def/security/1.0/rc/http-methods

URN Identifier: urn:ogc:def:security:1.0:rc:http-methods

1. The service endpoint SHALL list all supported HTTP methods (likely a subset of the methods defined in HTTP 1/1 recommendation from the IETF - RFC 2616).
2. The URN identifier urn:ogc:def:security:1.0:rc:http-methods SHALL be used for the <ows:Constraint> element. The sub-selement <ows:AllowedValues> shall be used to list each supported HTTP method using a <Value> element. The value of the element SHALL be in all uppercase the name of the method as identified in RFC 2616.

|  |
| --- |
| <ows:Constraint name="urn:ogc:def:security:1.0:rc:http-methods">  <ows:AllowedValues>  <ows:Value>GET</ows:Value>  <ows:Value>POST</ows:Value>  <ows:Value>OPTIONS</ows:Value>  <ows:AllowedValues>  </ows:Constraint> |

Table 11:Informative example indicating support for the methods GET, POST and OPTIONS

## Requirements Class “Exception Handling”

URL Identifier: https://www.opengis.net/def/security/1.0/rc/exception-handling

URN Identifier: urn:ogc:def:security:1.0:rc:exception-handling

1. The URN identifier urn:ogc:def:security:1.0:rc:exception-handling SHALL be used to identify the <ows:Constraint> element. The only valid sub-element SHALL be <ows:NoValues>. The following is the normative definition.

|  |
| --- |
| <ows:Constraint name="urn:ogc:def:security:1.0:rc:exception-handling">  <ows:NoValues/>  </ows:Constraint> |

Table 12:Normative annotation for expressing compliance with Exception Handling

Different version of OWS Common and the OGC Web Services specifications itself define slight different exception handling. The following figure illustrates the exception handling via ExceptionReport as defined in table 28 (OWS Common 2.0 – OGC #06-121r9, section 8.6, table 28).

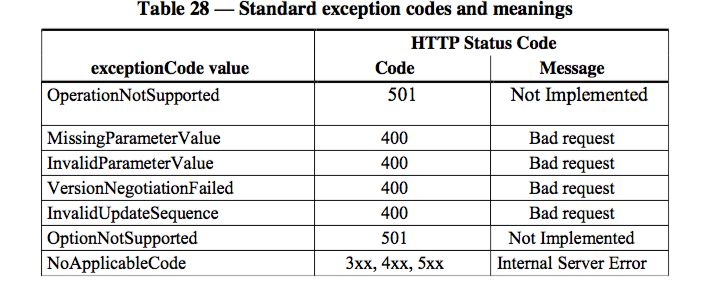


Figure 4 : OWS Common Exception Codes [OWS Common 1.2]

OWS Common 2.0 differentiates between an exception, which arises inside or outside the service implementation. In the case that the root cause is inside of the OGC service implementation, then HTTP status code and ExceptionReport according to Figure 4 above shall be used. In case that the root cause is outside the OGC service implementation, then HTTP status codes – with no ExceptionReport – is to be used. Unfortunately, OWS Common 1.1 – which is the mostly used version – does not differentiate the origin of the root cause. It is also unclear, which HTTP status code is to be used when delivering the ExceptionReport.

In addition, WMS introduces the EXCEPTION/EXCEPTIONS parameter that allows the client to control what the mime type of exception returned is going to be. This allows including the ExceptionReport into an image.

Regardless of these variations to exception handling, this standard defines a clear separation between the actual OGC Web Services, for which exception handling is defined, and the extra exception handling introduced by applying security to a service instance.

OGC web services do not exist in a vacuum. They are built on a set of services and standards, which define the underlying, distributed computing environment as illustrated in Figure 5 below.



Figure 5 : Processing Metaphor how to achieve interoperability with this standard

It is not enough for an OGC Web Service to correctly produce and handle exceptions defined in the OGC Standards. It must also correctly produce and handle exceptions in accordance with the standards for each service and protocol layer that the OGC Service is built upon. Therefore, an OGC Web Service which is compliant with this standards must comply with not just the exception handling requirements as defined in this standard, but also with the exception handling requirements of all of the supporting capabilities (protocols and services) which are advertised through the Capabilities document.

1. OWS services SHALL respect and implement exception handling for all supporting capabilities (protocols and services) identified through the Capabilities document in accordance with their governing standards.
2. A compliant implementation SHALL implement exception handling to use HTTP status codes as mandated by the relevant security standards..

## Requirements Class “HTTP POST Content-Type”

URL Identifier: https://www.opengis.net/def/security/1.0/rc/content-type

URN Identifier: urn:ogc:def:security:1:0:rc:content-type

1. The URN identifier urn:ogc:def:security:1:0:rc:content-type SHALL be used to signal that the service endpoint operation is compliant with the Conformance Class “HTTP POST Content-Type”. The value SHALL be “NoValues”.

|  |
| --- |
| <ows:Constraint name="urn:ogc:def:security:1:0:rc:content-type">  <ows:AllowedValues>  <ows:Value>text/xml</ows:Value>  <ows:Value>application/xml</ows:Value>  <ows:Value application/soap+xml</ows:Value>  <ows:Value>application/x-www-form-urlencoded</ows:Value>  </ows:AllowedValues>  </ows:Constraint> |

Table 12:Normative annotation expressing compliance with HTTP POST Content-Type

1. If the service instance supports HTTP POST requests (as declared in the capabilities document), the service instance SHALL support the mime-type “application/x-www-url-form-encoding” as registered with IANA (<https://www.iana.org/assignments/media-types/application/x-www-form-urlencoded>).

For additional information see the OGC CR #388 (<http://ogc.standardstracker.org/show_request.cgi?id=388>)

1. The name of the POST parameter for the application/x-www-form-urlencoded mime-type shall be OWSR carrying the text/xml formatted request.

## Requirements Class “Access Control”

URL Identifier: https://www.opengis.net/def/security/1.0/rc/authorization

URN Identifier: urn:ogc:def:security:1.0:rc:authorization

1. The URN identifier urn:ogc:def:security:1.0:rc:authorization SHALL be used as the name of the <ows:Constraint> element to signal that the service endpoint operation is on access control. The sub-element <ows:ValuesReference> SHALL use a URL for the reference attribute.

|  |
| --- |
| <ows:Constraint name="urn:ogc:def:security:1.0:rc:authorization">  <ows:ValuesReference ows:reference="{URL}"/> </ows:Constraint> |

Table 13:Normative annotation expressing compliance with Access Control

1. The URL provided with the urn:ogc:def:security:authorization constraint SHALL resolve to a XACML or GeoXACML policy.

*Note: The URL provided might be protected and returns a user specific instance of the general access control policy.*

1. For an XACML policy, the mime-type SHALL be used as registered with IANA and published informational by the IETF in RFC 7061 (<https://tools.ietf.org/html/rfc7061>): application/xacml+xml
2. For a GeoXACML policy, the mime-type SHALL be used as registered with IANA and published at <https://www.iana.org/assignments/media-types/application/geoxacml+xml>: application/geoxacml+xml

The implementation leveraging the URL to fetch the access policy must observe the Content-Type of the response to identify whether the policy is XACML or GeoXACML and which version.

*Note: It is not required that the access control layer at the service actually operates on a XACML or GeoXACML policy. However, to ensure interoperability and the ability of the client to determine the access denied case before executing a service, a standards compliant description is required. A proprietary language must not be used. An example where the client could leverage the obtained policy is before uploading tons of features to a WFS-T. In cases where the client has determined “deny” this would simply reserve bandwidth.*

## Requirements Class “WS-Security”

URL Identifier: https://www.opengis.net/def/security/1.0/rc/policy

URN Identifier: urn:ogc:def:security:1.0:rc:policy

1. The URN identifier urn:ogc:def:security:1.0:rc:policy SHALL be used as the name of the <ows:Constraint> element to signal that the service endpoint operation is on a WS-Security control. The sub-element <ows:ValuesReference> to signal that the service endpoint operation is on WS-Security. The URL provided SHALL resolve to the WS-Policy that defines the SOAP security conditions implemented.

|  |
| --- |
| <ows:Constraint name="urn:ogc:def:security:1.0:rc:policy">  <ows:ValuesReference ows:reference="{URL}"/> </ows:Constraint> |

Table 14:Normative annotation expressing compliance with WS-Policy

1. The service endpoint shall use the annotation to provide insight information about the SOAP + WS-Security based security. The URL SHALL resolve to the WS-Policy used to describe the WS-Security conditions.
2. For returning a WS-Policy, the official WS-Policy mime-type SHALL be used as registered with IANA and published by the W3C in the Web Services Policy 1.5 Framework (<https://www.w3.org/TR/2006/WD-ws-policy-20061117/>): application/wspolicy+xml

## Requirements Class “SAML2”

URL Identifier: https://www.opengis.net/def/security/1.0/rc/authentication/saml2

URN Identifier: urn:ogc:def:security:1.0:rc:authentication:saml2

1. The URN identifier urn:ogc:def:security:1.0:rc:authentication:saml2 SHALL be used to provide additional information if required by the identified authentication method. The provided URL SHALL resolve to the SAML2 metadata file for the federation of which this service endpoint is a member off.

|  |
| --- |
| <ows:Constraint name="urn:ogc:def:security:1.0:rc:authentication:saml2">  <ows:ValuesReference ows:reference="{URL}"/> </ows:Constraint> |

Table 15:Normative annotation expressing compliance with SAML2

## Requirements Class “OpenID Connect”

URL Identifier: https://www.opengis.net/def/security/1.0/rc/authentication/oidc

URN Identifier: urn:ogc:def:security:1.0:rc:authentication:oidc

1. The URN identifier urn:ogc:def:security:1.0:rc:authentication:oidc SHALL be used to provide additional information if required by the identified authentication method. The provided URL SHALL resolve to the well-known description of the relevant OAuth2 Authorization Server implementing the OpenID Connect extension.

|  |
| --- |
| <ows:Constraint name="urn:ogc:def:security:1.0:rc:authentication:oidc">  <ows:ValuesReference ows:reference="{URL}"/> </ows:Constraint> |

Table 15:Normative annotation expressing compliance with OpenID Connect

## Requirements Class “OpenAPI 3.0”

URL Identifier: https://www.opengis.net/def/security/1.0/rc/openapi

URN Identifier: urn:ogc:def:security:1.0:rc:openapi

1. The identifier urn:ogc:def:security:1.0:rc:openapi SHALL be used to inform about an OpenAPI compliant description of the service endpoint(s). The provided URL SHALL resolve to a valid OpenAPI instance document.

*Note: A referenced description may leverage OpenAPI extensions.*

|  |
| --- |
| <ows:Constraint name="urn:ogc:def:security:1.0:rc:openapi">  <ows:ValuesReference ows:reference="{URL}"/> </ows:Constraint> |

Table 16:Normative annotation expressing compliance with OpenAPI

The use of OpenAPI for describing an API can also be used to describe the communication protocols used by an interface of an OGC Web Service. We recognize that this description is **not** a replacement for the Capabilities document.

However, the main driver for using the OpenAPI format is to provide security constraints for the service instance using a format, well known in mainstream IT. In particular important is the ability to provide information about existing security controls. The example below illustrates how to provide additional (meta) information for an OAuth2 protected service instance.

|  |
| --- |
| type: oauth2  flows:  implicit:  authorizationUrl: https://example.com/api/oauth/dialog  scopes:  write:pets: modify pets in your account  read:pets: read your pets  authorizationCode:  authorizationUrl: https://example.com/api/oauth/dialog  tokenUrl: https://example.com/api/oauth/token  scopes:  write:pets: modify pets in your account  read:pets: read your pets |

Table 17:Example annotation expressing OAuth2 requirement with OpenAPI

This example from the OpenAPI specification indicates that the service endpoint requires OAuth2 Access Token released for particular scopes [https://github.com/OAI/OpenAPI-Specification/blob/master/versions/3.0.0.md#implicit-oauth2-sample].

## Requirements Class “Authentication”

URL Identifier: https://www.opengis.net/def/security/1.0/rc/authentication

URN Identifier: urn:ogc:def:security:1.0:rc:authentication

1. The URN identifier urn:ogc:def:security:1.0:rc:authentication SHALL be used for the name of the <ows:Constraint> element to signal that the service endpoint operation requires authentication. The sub-element SHALL be <ows:ValuesReference> where the reference attribute value contains the URN referencing the authentication code as idenfitied in the Authentication Codelist.

The following informative example illustrates the security annotation to indicate that the authentication method client side TLS certificate is in place.

|  |
| --- |
| <ows:Constraint name="urn:ogc:def:security:1.0:rc:authentication">\_   <ows:ValuesReference ows:reference="**urn:ogc:def:security:authentication:ietf:5246:client\_certificate**"/>  <ows:Meaning ows:reference="**https://www.opengis.net/def/security/1.0/rc/authentication/urn:ogc:def:security:authentication:ietf:5246:client\_certificate**"/> </ows:Constraint> |

Table 18: Example annotation expressing client authentication via certificate

1. In the case where the implementor wishes to provide a resolvable URL to the definition of the authentication method, the <ows:Meaning> element and the reference attribute SHALL be used to provide that resolvable URL. The URL SHALL fetch the definition from the Authentication CodeList that corresponds to the name attribute used with the <ows:Constraint> element. The URL has the form https://www.opengis.net/def/security/1.0/rc/authentication/<URN identifier>

### Authentication via SAML

In addition to annotating the authentication method as defined in above, additional information can be provided. For example, the service provider might want to let the client know to which SAML2 federation the service belong. This could be achieved by using the identifier urn:ogc:def:security:1.0:rc:authentication:saml2

1. The <ows:Contraint> element SHALL have the identifier urn:ogc:def:security:1.0:rc:authentication:saml2 to indicate additional SAML2 metadata information is available.
2. The <ows:ValuesReference> element and the reference attribute SHALL have the value of the URL which allows to fetch the SAML2 compliant metadata for the federation in which the service is participating in.

*Note: Before starting the authentication handshake, the client should check if the advertised SAML authentication method (see section above) is supported.*

### Authentication via OpenID Connect

Similar to SAML2, the client does need additional information to start the authentication handshake. For OpenID Connect, this is the metadata of the accepted Authorization Server linked with the secured service endpoint (OpenID Connect Discovery).

For an Authorization Server that is a compliant OpenID Connect implementation, a .well-known description exists as defined by IANA (URL ends with “.well-known/openid-configuration”).

1. The <ows:Contraint> element SHALL have the identifier urn:ogc:def:security:1.0:rc:authentication:oidcto indicate that additional OpenID Connect metadata information is available.
2. The <ows:ValuesReference> element and the reference attribute SHALL have the value of the URL which points to the .well-known configuration of the OpenID Connect Provider associated with the protected service endpoint.

# Conformance for a Client Implementation

In order for the concept of Capabilities with security annotations to work, these annotated capabilities must be available to the client application with no security challenges involved. The methods describing how to make the annotated Capabilities for a service instance available to the client application vary and depend on many factors. Methods describing how the annotated Capabilities for the service are actually made available to the client application are outside the scope of this standard.

A client can implement support for one or more conformance classes as defined by this standard:

* **https://www.opengis.net/def/security/1.0/cc/owsCommon**,
* **https://www.opengis.net/def/security/1.0/cc/wms130** or
* **https://www.opengis.net/def/security/1.0/cc/wms111**.

Each of these conformance classes defines the parsing of security annotations by obeying the different structures of the Capabilities document.

The following figure illustrates the relationship of the conformance classes for a client implementation.

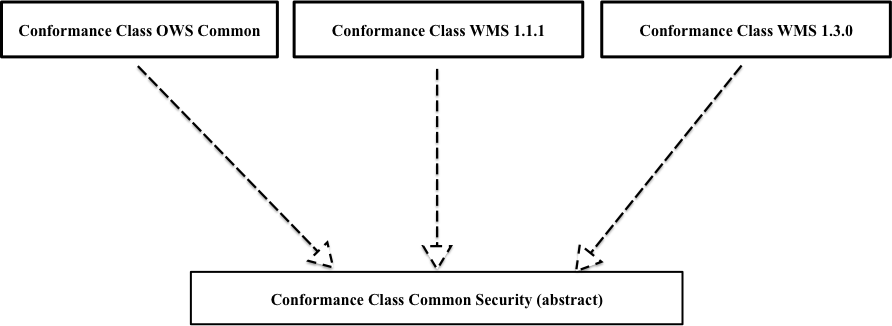


Figure 6 : Conformance Classes defined for the client

## General Client Functionality

The following steps outline the approach for what an OWS Common Security compliant client should do with the annotated capabilities document:

* Client should interpret the annotated capabilities by parsing the operations metadata to determine its compatibility with the outlined security conditions.
* If support exists for the security conditions and the “content” section is empty, the client should execute the GetCapabilities operation as published in the annotated Capabilities to get the full Capabilities document.
* If support exists for the security conditions outlined in the annotated capabilities and the “content” section is not empty, the client should proceed as usual by calling the service specific operations, e.g. GetMap, GetFeature, etc.

## Client Requirements Classes for “Parsing Annotated Capabilities”

According to the requirements for the service instance, the capabilities document advertises its compliance to this standard by inserting the Conformance Class identifier into the XML element <AccessContraint>. As this element is always present in the capabilities document, independent from the underlying schema, the client can detect service instance compliance by parsing the <AccessContraint> element.

*Note: Based on Requirement 1 a client can determine if the capabilities document contains security annotations which indicates that the service endpoint is complaint with this specification.*

1. The client SHALL parse the <AccessConstraint> element from the capabilities XML document to detect the service instance compliance.

The value from the <AccessContraint> element identifies the conformance class implemented by the service instance.

1. If the <AccessConstraint> indicate service compliance to this standard, the client SHALL parse the Capabilities document for security annotations, based on the conformance class as identified by the value of the <AccessConstraint> element.

### Parse Security Annotations based on OWS Common Schema

1. Parsing of the Capabilities document to extract descriptions of security controls, SHALL be based on the Conformance Class **https://www.opengis.net/def/security/1.0/cc/owsCommon**.

### Parse Security Annotations based on WMS 1.3.0 Schema

1. Parsing of the Capabilities document to extract descriptions of security controls, SHALL be based on the Conformance Class **https://www.opengis.net/def/security/1.0/cc/wms130**.

### Parse Security Annotations based on WMS 1.1.1 DTD

1. Parsing of the Capabilities document to extract descriptions of security controls, SHALL be based on the Conformance Class **https://www.opengis.net/def/security/1.0/cc/wms111**.

### Working with Complete Capabilities

According to Requirement 26 the Capabilities document returned by the service operation GetCapabilities, as outlined in the annotated capabilities document, returns a full capabilities document. Therefore, a client can be sure to work on a full capabilities document only is this case. In the case where security controls are implemented for the GetCapabilities operation their existence is indicated by the relevant security annotations.

*Note: In case that security controls are indicated for the GetCapabilities operation (thru <ows:Constraint> elements) the client must overcome the security controls to receive the full capabilities document.*

### Working with Partial Capabilities

As described in section 5.4 (Use Case III: Protected Service / Private Data / Public Catalogue) it is possible that publically accessible capabilities include none or a partial “content” section. In these cases, the client must execute the GetCapabilities operation as outlined in the publically accessible version of the capabilities to fetch the full capabilities document. The client can determine partial capabilities by parsing for the absence of the “content” section.

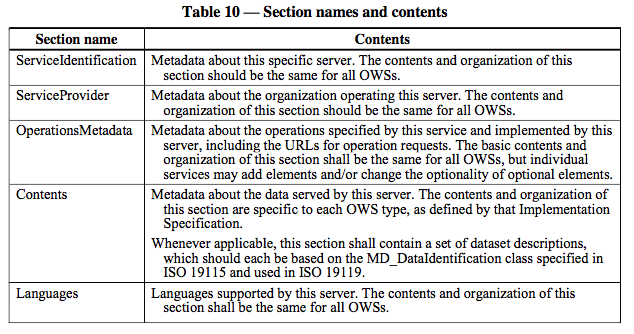


Table 19: Section names and their content [OGC #06-121r9, p.25]

The “content” section of the Capabilities is represented by different XML elements for different OGC Web Service types:

1. WMS: <Layer>
2. WMTS: <Contents>
3. WCS: <Contents>
4. WFS: <FeatureTypeList>

In any case, for the annotated capabilities to be present, the Capabilities instance document must at least contain the <ows:OperationsMetadata> element and the mandatory operation GetCapabilities. As illustrated in the use cases in section 2, the “content” part of the capabilities document might be omitted. But how could the client tell that a content section is just partial? Based on the current standardization, it is not possible for the client to determine whether the content section is just partial. This results in a server side requirement that either the “content” section is **complete** or **empty**.

1. The “content” section in the capabilities document SHALL either be empty or contain a complete content listing.
2. If the annotated Capabilities document does not contain a “content” section (<Layer> element for WMS or a <FeatureTypeList> element for a WFS or a <Contents> element for WMTS, WCS or WPS) and the <AccessContraints> element indicates compliance with this Standard, the application SHALL execute the GetCapabilities operation advertised (within the Capabilities document) to fetch the full service capabilities. *Note: This assumes that the client is able to function on the advertised security requirements for the GetCapabilities operation.*

## Client Requirements Class “HTTPS”

According to Requirement 2 any service instance must operate on HTTPS. Therefore, any client implementation claiming conformance has to support HTTPS.

1. Any compliant client implementation SHALL support HTTP over TLS as defined by RFC 2818. This includes certificate validation, verification and use of Revocation Lists.

## Client Requirements Class “Exception Handling”

According to Requirements Class “Exception Handling” a service instance may advertise its support for HTTP compliant exception handling. According to Requirement 36 the default exception handling for a service instance compliant to this standard is to use HTTP status codes. But a client can request a service to react with OWS Common based exception handling by submitting a query parameter as specified by the OWS Common standard applicable to the service instance.

1. The client SHALL expect exception handling compliant to HTTP of a service instance that returns the constraint with identifier urn:ogc:def:security:1.0:rc:exception-handling.
2. For the conformance class OWS Common, the client SHALL issue the request to the service to send error responses according to the OWS specification as defined by the underlying OWS Common specification.
3. For the conformance class WMS1.1.1, the client SHALL use the KVP **Exception** as standardized to request from the service to send error responses according to the WMS 1.1.1 specification.
4. For the conformance class WMS 1.3.0, the client SHALL use the KVP **Exceptions** as standardized to request from the service to send error responses according to the WMS 1.3.0 specification.

# Authentication Codelist

For ensuring interoperability with authentication methods implemented on a service instance, this standard defines an Authentication Codelist as a normative reference to identify authentication codes. The Authentication Codelist will be hosted by the OGC. The maintainer of the codes and values of the Authentication Codelist is the OWS Common – Security SWG.

Regarding the interoperability between secured OGC Web Services and client applications, the most important and critical topic is Authentication. The concept of annotated Capabilities allows authentication methods to be declared based on an Authentication Codelist, maintained by the OGC. The Authentication CodeList is an ISO 19119 service metadata compliant Authentication Codelist using the GMX namespace. Essentially, it contains identifiers in different name spaces that can be used in the security annotation for authentication, a human readable description and a link to the standard that defines it. The concept of name spaces is important as it enable the re-use of already defined authentication methods and protocols. For example, HTTP BASIC/DIGEST authentication is defined in the namespace IETF, as defined in RFC 2617. Likewise OAuth Bearer authentication is defined in the IETF namespace and SAML protocols are defined in the OASIS namespace. In case where vendor specific authentication is used, they should be included into the Authentication CodeList and the namespace would indicate that the owner is the 3rd party.

1. Authentication codes SHALL use the namespace that reflects the maintainer of the authentication code.

The maintainer of the Authentication CodeList is the OWS Common - Security SWG. The approval of new authentication codes must be submitted to this SWG via an OGC Change Request: <http://ogc.standardstracker.org/>.

1. New authentication codes for the Authication CodeList hosted by OGC shall be submitted via Change Request to the OWS Common Security SWG.
2. To ensure backwards compatibility of the Authentication CodeList, only new authentication codes SHALL be added to the Authentication CodeList. It is not possible to modify or remove existing codes.

## Authentication Codelist URL

A normative version of the Authentication Codelist is hosted and maintained by the OGC.

1. The OGC SHALL host the Authentication CodeList.
2. Hosting URL of the Authentication Codelist SHALL use scheme HTTPS

## Authentication Codes

For ensuring interoperability with authentication methods implemented on a service instance, this standard defines URNs in an Authentication Codelist. The Authentication Codelist uses the ISO GMX namespace to enable interoperable use within the security annotations of the service capabilities as well as service ISO metadata.

*Note: How to use the authentication codes to annotate ISO metadata is out of scope for this standard.*

### Authentication Codes defined by IETF

Based on the IANA Hypertext Transfer Protocol (HTTP) Authentication Scheme Registry (<http://www.iana.org/assignments/http-authschemes/http-authschemes.xhtml>) the following HTTP authentication methods are defined based on IETF RFCs:

|  |  |  |
| --- | --- | --- |
| **Identifier** | **Namespace** | **Reference** |
| Basic | IETF | <http://www.iana.org/go/rfc7617> |
| Bearer | IETF | <http://www.iana.org/go/rfc6750> |
| Digest | IETF | <http://www.iana.org/go/rfc7616> |

Table 20:IETF defined authentication methods

### Authentication Codes defined by OASIS

Based on OASIS SAML2 Authentication Context definitions, the following authentication URNs are defined in the OASIS namespace.

|  |  |
| --- | --- |
| **Identifier** | **Namespace** |
| urn:oasis:names:tc:SAML:2.0:ac:classes:InternetProtocol | OASIS |
| urn:oasis:names:tc:SAML:2.0:ac:classes:InternetProtocolPassword | OASIS |
| urn:oasis:names:tc:SAML:2.0:ac:classes:Kerberos | OASIS |
| urn:oasis:names:tc:SAML:2.0:ac:classes:MobileOneFactorUnregistered | OASIS |
| urn:oasis:names:tc:SAML:2.0:ac:classes:MobileTwoFactorUnregistered | OASIS |
| urn:oasis:names:tc:SAML:2.0:ac:classes:MobileOneFactorContract | OASIS |
| urn:oasis:names:tc:SAML:2.0:ac:classes:MobileTwoFactorContract | OASIS |
| urn:oasis:names:tc:SAML:2.0:ac:classes:Password | OASIS |
| urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport | OASIS |
| urn:oasis:names:tc:SAML:2.0:ac:classes:PreviousSession | OASIS |
| urn:oasis:names:tc:SAML:2.0:ac:classes:X509 | OASIS |
| urn:oasis:names:tc:SAML:2.0:ac:classes:PGP | OASIS |
| urn:oasis:names:tc:SAML:2.0:ac:classes:SPKI | OASIS |
| urn:oasis:names:tc:SAML:2.0:ac:classes:XMLDSig | OASIS |
| urn:oasis:names:tc:SAML:2.0:ac:classes:Smartcard | OASIS |
| urn:oasis:names:tc:SAML:2.0:ac:classes:SmartcardPKI | OASIS |
| urn:oasis:names:tc:SAML:2.0:ac:classes:SoftwarePKI | OASIS |
| urn:oasis:names:tc:SAML:2.0:ac:classes:Telephony | OASIS |
| urn:oasis:names:tc:SAML:2.0:ac:classes:NomadTelephony | OASIS |
| urn:oasis:names:tc:SAML:2.0:ac:classes:PersonalTelephony | OASIS |
| urn:oasis:names:tc:SAML:2.0:ac:classes:AuthenticatedTelephony | OASIS |
| urn:oasis:names:tc:SAML:2.0:ac:classes:SecureRemotePassword | OASIS |
| urn:oasis:names:tc:SAML:2.0:ac:classes:TLSClient | OASIS |
| urn:oasis:names:tc:SAML:2.0:ac:classes:TimeSyncToken | OASIS |
| urn:oasis:names:tc:SAML:2.0:ac:classes:unspecified | OASIS |

Table 21: List of Authentication methods defined for SAML2

### Authentication Codes defined by OGC

There are some authentication methods that do not have a well defined identifier. For these methods, the OGC specifies an identifier in their namespace.

Identifier **urn:ogc:def:security:authentication:ietf:5246:client\_certificate** has the meaning equivalent to the RFC 5246: “*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)."[RFC 5246] In case the client cannot provide a suitable and valid certificate, no TLS connection gets established*”[RFC 5246]

|  |  |
| --- | --- |
| **Identifier** | **Namespace** |
| urn:ogc:def:security:authentication:ietf:5246:client\_certificate | OGC |

Table 22: OGC identifier for IETF authentication method

## Authentication Code Resolving

The meaning of the authentication code can be resolved from the Authentication Codelist URL via the pattern defined in Requirement Class Requirement 51.

The resolver returns a human readable definition of the used authentication method in mime-type text/html.

Example: For evaluation during the RFC, the URI <https://www.opengis.net/def/security/1.0/authentication/SAML2_ECP> does not resolve!

Please use this URL to directly resolve the definition of authentication method urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp:

<https://www.tb13.secure-dimensions.de/authnCodeList#SAML2_ECP>

## Initial Authentication Codelist

Annex C contains the initial Authentication Codelist that is informative for ease of reading.

The official mandatory Authentication Codelist is hosted by OGC under this URL:

*Note: The official URL will be published here once the OGC Technical Committee has approved this document to become an OGC standard.*

# Security Considerations (informative)

Applying this standard to a service endpoint provides the opportunity to expose security metadata into the service Capabilities. The main purpose is to provide an interoperability mechanism such that the client can determine whether the security controls at the service are supported.

## Thread “Tampered Capabilities”

The mechanism of including security metadata into the Capabilities works well if the client could trust the Capabilities. For the purpose of the security considerations, it is best to differentiate if the Capabilities are used as an XML instance document or as the direct response from the service to the GetCapabilities request.

**Thread: Tampered Capabilities**

Would this thread lead to vulnerability? Yes, this threat could cause a client to wrongly submit user credentials to a malicious site!

Assuming that the attacker would be able to modify the URL of the service endpoint and assert that the authentication method were HTTP BASIC Authentication (as an example). This would cause the client to submit user credentials with the service request. This vulnerability must be considered high risk, as the client has no means to identify the attack.

### Mitigations to this threat:

For any service instance that is compliant to this standard, it is mandatory to have HTTPS in place. However, the Capabilities document being an XML instance document must not be trusted, as it has no means of protection applied.

### Approaches to provide a digital signature to the Capabilities document

The W3C XML Digital Signature is a method to provide integrity to an XML instance document. Applying a digital signature can take place using three methods:

1. Enveloping Signature
2. Enveloped Signature
3. Detached Signature

One of the main objectives to the OGC Web Services security standard was to ensure backwards compatibility which lead to the standardized approach: Insert security metadata into existing elements of the Capabilities structure; the <ows:Constraint> element published with the structure for the service metadata. In order to ensure backwards compatibility for the digital signature as well would only allow using the method enveloped signature. However, the Digital Signature element could not be in the usual place (either first or last element of document root) but would rather have to sit inside the ExtendedCapabilities element. Even though putting the digital signature element inside the ExtendedCapabilities element is compliant with the W3C Digital Signature recommendation, main stream IT tools would fail, as they look for the signature in the usual / recommended place.

|  |
| --- |
| <schema xmlns:xs="http://www.w3.org/2001/XMLSchema"  xmlns:ows\_security="http://www.opengis.net/security/1.0"   xmlns:ows="http://www.opengis.net/ows/1.1"  xmlns:wms="http://www.opengis.net/wms"   xmlns:ds="http://www.w3.org/2000/09/xmldsig#"  xmlns="http://www.w3.org/2001/XMLSchema"   xmlns:xlink="http://www.w3.org/1999/xlink"   targetNamespace="http://www.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"/>  <import namespace="http://www.w3.org/2000/09/xmldsig#" schemaLocation="http://www.w3.org/TR/xmldsig-core/xmldsig-core-schema.xsd"/>  <xs:complexType name="ExtendedXMLDSigCapabilitiesType">  <sequence>  **<element ref="ds:Signature" minOccurs="1" maxOccurs="1"/>**  <element ref="ows:OperationsMetadata" minOccurs="1" maxOccurs="1"/>  </sequence>  </xs:complexType>  <element name="ExtendedXMLDSigCapabilities" type="ows\_security:ExtendedXMLDSigCapabilitiesType" substitutionGroup="wms:\_ExtendedCapabilities"/> </schema> |

Table 23:Example ExtendedCapabilities element including a Signature element

Conclusion: Applying an enveloped signature as part of the ExtendedCapabilities document is possible but not feasible as a specific signature / validation library must be implemented to honor the non-typical position of the signature element.

### Recommendation

It is recommended that the client does only trust a GetCapabilities response of a service instance and not a Capabilities instance document obtained from another source. The client can trust the service response, as the communication via HTTPS must be considered secure so that no tampering could have occurred while the response to the GetCapabilities request was submitted to the client.

Perhaps there are other means to secure the Capabilities document, but are considered out of scope for this standard.

*Note: For the future, it seem to be reasonable to request that a digital signature can be applied to OGC encoding documents; e.g. inside a Capabilities instance document to enable enveloped signatures compliant with the main-stream IT approach (either have the Signature element first or last child of document root). But to secure any OGC instance document, like a FeatureCollection, a OWS Context instance document, etc. it would be necessary to provide an optional element to relevant OGC encoding schemas*.

Annex A: Revision history (informative)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Date | Release | Author | Paragraph modified | Description |
| 15.02.17 | 0.1 | AM[[2]](#footnote-2) | All | Creation |
| 20.02.17 | 0.2 | AM | All | Incorporation of comments from Frank Terpstra |
| 13.03.17 | 0.3 | AM | All | Andreas, Chuck, Dave, Frank, Michael telecom 13.03.17 |
| 27.03.17 | 0.4 | AM | All | Andreas, Chuck, Dave, Michael telecom 27.03.17 |
| 08.05.17 | 0.5 | AM | All | Incorporation of comments from Dave Wesloh, Don Sullivan and Frank Terpstra |
| 19.06.17 | 0.6 | AM | All | Incorporation of comments from Dave Wesloh |
| 18.07.17  -  01.08.17 | 0.7 | AM | All | Incorporating results from discussions in meetings and updating the structure of the document for improved readability |
| 24.08.17 | 0.8 | AM | All but mainly 7.5 | Revised Exception Handling |
| 18.09.17  -  26.09.17 | 0.9 | AM | Security Considerations | Creation |
| 06.11.17 | 0.10 | AM | All  Appendix B | Incorporation of comments from Frank Terpstra |
| 13.11.17 | 0.11 | AM | All  Annex C  All | Incorporation of comments from Chuck Heazel and Dave Wesloh  Including initial Authentication Codelist  Preparation for RFC |
| 15.01.18 | 0.12 | AM | Section 4, 6, 7 | Incorporation of NA guidance |

Annex B: Conformance Tests for the Service (normative)

The following table presents an informative overview of the different requirements class and whether mandatory, optional or conditional.

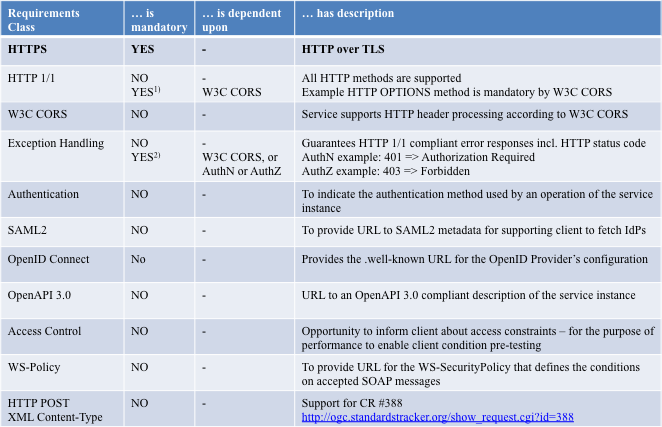


Table 24: Requirements Classes overview (1) Mandatory if CORS is advertised; 2) Mandatory if CORS or Authentication or Access Control is advertised)

1. The conformance tests defined in this Annex SHALL be used to determine compliance with this standard for the service side deployment.
   1. Conformance Test Activity Diagram

The following activity diagram illustrates the sequence of tests that must be applied to determine compliance with a particular requirements class. The activity diagram takes under consideration the dependency of some requirements classes.

The yellow boxes are labeled with the abstract that SHALL be carried out. The concrete instantiation of the test depends if it is to be applied to the service or client side.

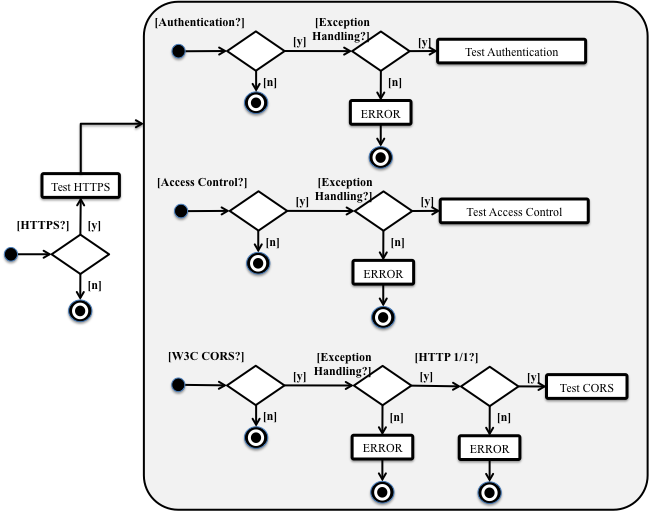


Figure 7:Testing mandatory dependencies of Requirements Classes

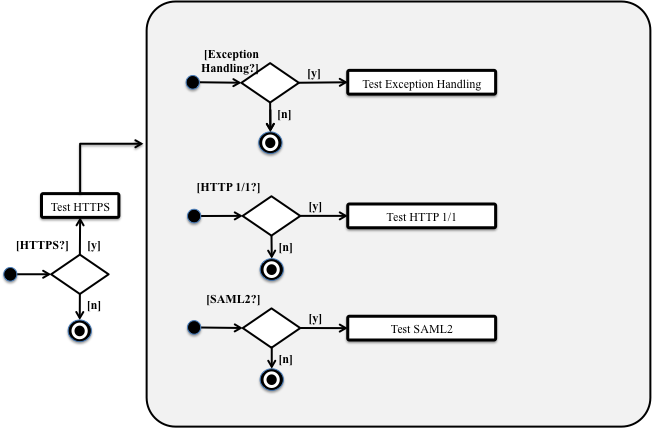


Figure 8:Testing of independent Requirements Classes

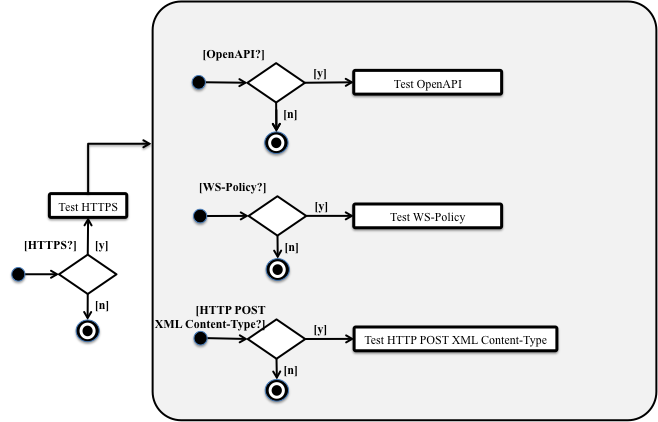


Figure 9:Testing of independent Requirements Classes

* 1. Conformance Tests for the Service-Side

For the service-side, it is the fact that a particular set of security controls is in place. The concrete tests for the service-side are designed to answer the question if the GetCapabilities response is compliant with this standard.

* + 1. Conformance Test “Capabilities Structure”

Condition: Applies always

Dependency: “HTTPS”

Purpose: This test SHALL determine whether the Capabilities document, including the security annotations, is valid.

Test: Use XML parser to validate XML instance document or GetCapabilities response.

* + 1. Conformance Test “HTTPS”

Condition: Applies always

Dependency: %

Purpose: Verify that requirements class urn:ogc:def:security:1.0:rc:https is supported

Test: Use Web Browser and establish connection with service

Pass: If HTTPS connection is established and lock in URL bar is green

* + 1. Conformance Test “HTTP 1/1”

Condition: Applies if service supports urn:ogc:def:security:1.0:rc:http-methods

Dependency: %

Purpose: Verify that service supports all HTTP methods.

Test: Execute separate HTTP requests using different HTTP method as defined in requirements class urn:ogc:def:security:1.0:rc:http-methods.

Pass: If service supports methods as defined in urn:ogc:def:security:1.0:rc:http-methods

* + 1. Conformance Test “W3C CORS”

Condition: Applies if service supports urn:ogc:def:security:1.0:rc:cors

Dependency: %

Purpose: Verify that service supports W3C CORS.

Test: Execute separate HTTP requests as defined in the W3C CORS recommendation and verify that HTTP response headers are provided as standardized.

Pass: Follow conditions as published in W3C CORS.

* + 1. Conformance Test “Exception Handling”

Condition: Applies if service supports urn:ogc:def:security:1.0:rc:exception-handling

Dependency: W3C CORS, Authentication, Access Control

Purpose: Verify that service supports Exception Handling as defined in this standard.

Test: Cannot be executed by itself. Must be executed as slave with CORS, Authentication or Access Control tests.

Pass: If master test passes.

* + 1. Conformance Test “Authentication”

Condition: Applies if service supports urn:ogc:def:security:1.0:rc:authentication

Dependency: Exception Handling

Purpose: Verify that service supports advertised authentication method.

Test: Execute separate HTTP requests that meet / violate the authentication method in place.

Pass: If valid authentication requests execute the service and invalid requests trigger appropriate error handling compliant with the authentication method in place.

* + 1. Conformance Test “Access Control”

Condition: Applies if service supports urn:ogc:def:security:1.0:rc:authorization

Dependency: Exception Handling

Purpose: Verify that service returns HTTP response compliant with HTTP 1/1.

Test: Execute separate HTTP requests that meet / violate the access control condition for Permit / Deny.

Pass: If service returns 403 on a request that is denied access.

* + 1. Conformance Test “WS-Policy”

Condition: Applies if service supports urn:ogc:def:security:1.0:rc:policy

Dependency: %

Purpose: Verify that URL from ows:Constraint element resolves to a WS-Policy document

Test: Parse Capabilities document provided by service, fetch the URL from constraint and obtain document.

Pass: If WS-Policy document is returned.

* + 1. Conformance Test “SAML2”

Condition: Applies if service supports urn:ogc:def:security:1.0:rc:authentication:saml2

Dependency: %

Purpose: Verify that URL from <ows:Constraint> element resolves to a SAML2 metadata document

Test: Parse Capabilities document provided by service, fetch the URL from constraint and obtain document.

Pass: If SAML2 metadata document is returned.

* + 1. Conformance Test “OpenAPI”

Condition: Applies if service supports urn:ogc:def:security:1.0:rc:openapi

Dependency: %

Purpose: Verify that URL from <ows:Constraint> element resolves to an OpenAPI document

Test: Parse Capabilities document provided by service, fetch the URL from constraint and obtain document.

Pass: If OpenAPI document is returned.

* + 1. Conformance Test “HTTP POST XML Content-Type”

Condition: Applies if service supports urn:ogc:def:security:1:0:rc:content-type

Dependency: %

Purpose: Verify that service can process the submitted XML via Content-Type application/x-www-form-urlencoded as defined in

Test: Create XML POST request compliant for service type and submit via HTTP POST using content-type “” and the parameters “OWSR =<XML document>”.

Pass: If request is accepted and processed by service.

* 1. Conformance Tests for the Client-Side

The target of the tests is to verify the structure of the capabilities and to process it correctly. Therefore, the conformance tests for the client-side have one objective: Parse the capabilities response to determine which security controls / features are supported by the service.

As there is only one mandatory requirements class – HTTPS – there are only two tests to be defined.

* + 1. Conformance Test “Parse Capabilities document”

Parsing the Capabilities requires knowing the document structure. This can be determined by the fetching the conformance class information from the <AccessConstraints> element as defined by Requirement XYZ.

Condition: Applies always

Dependency: %

Purpose: Verify that capabilities structure, including the security annotations, is compliant with this standard.

Test: Use XML parser and verify capabilities document structure with pre-defined or schema using hint from XML instance document.

Pass: If XML structure is valid.

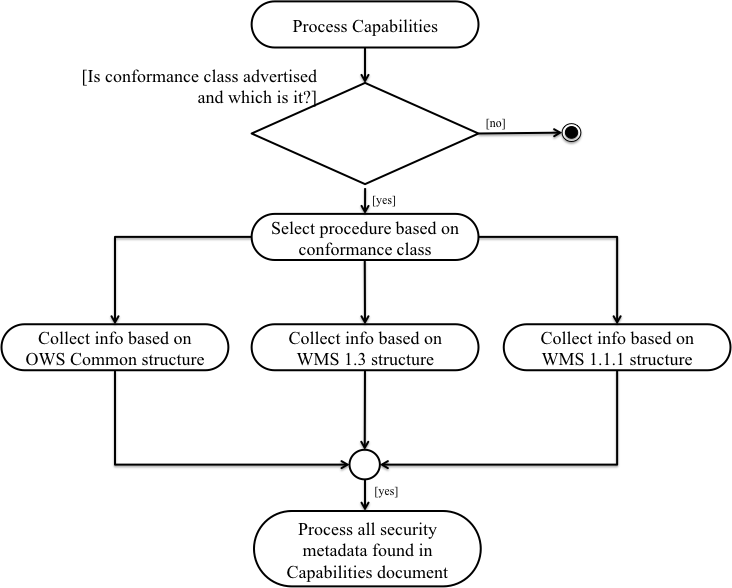


Figure 10:Illustration of processing the Capabilities document

* + 1. Conformance Test “HTTPS”

The only mandatory requirements class defined by this standard is urn:ogc:def:security:1.0:rc:https

This test SHALL be applied to determine that the service has advertised the use of HTTPS in the capabilities document.

Condition: Applies always

Dependency: %

Purpose: Verify that <ows:Constraint> with identifier urn:ogc:def:security:1.0:rc:https is present in the capabilities document.

Test: Use XPath expression to fetch the element and verify the value.

Pass: If constraint exists and is compliant with Requirements class urn:ogc:def:security:1.0:rc:https.

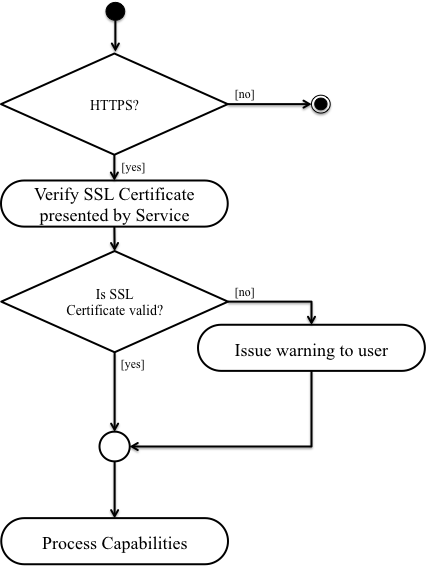


Figure 11: Illustration of the test for the mandatory requirements class: HTTPS

Note: The orange box “Issue warning to user” is not mandatory for this test as it is testing implementation details. However, it is recommended practice to warn the user about issues with the SSL certificate.

* 1. Abstract Conformance Tests for the Client-Side

The purpose of the abstract conformance test is to verify client implementations in a test harness, simulating the service. How such a service-side test harness does work and which concrete tests exist is dependent on many factors. Therefore, an extension to this standard shall define the concrete implementations of conformance tests with the target of client implementation.

1. For each security constraint found, the client SHALL construct valid and invalid requests. The service-side test harness will evaluate the requests and determine compliance.

Annex C: Initial Authentication Codelist (informative)

The official mandatory Authentication Codelist is hosted by OGC under the URL provided in section 9.4.

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?> <?xml-stylesheet type="text/xsl" href="authnCodeList.xsl"?> <gmx:CT\_CodelistCatalogue xmlns="http://www.opengis.net/security/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>authnCodelist</gco:CharacterString>  </gmx:name>  <gmx:scope>  <gco:CharacterString>OGC codelist for description security annotations regarding authentication</gco:CharacterString>  </gmx:scope>  <gmx:fieldOfApplication>  <gco:CharacterString>OGC</gco:CharacterString>  </gmx:fieldOfApplication>  <gmx:versionNumber>  <gco:CharacterString>1.0</gco:CharacterString>  </gmx:versionNumber>  <gmx:versionDate>  <gco:Date>2017-11-14</gco:Date>  </gmx:versionDate>  <!--============================================================================-->  <!--============================================================================-->  <!--============================= Codelists =======================================-->  <!--=== AuthenticationCode ===-->  <gmx:codelistItem>  <gmx:CodeListDictionary gml:id="AuthenticationCode">  <gml:description>identification of authentication methods</gml:description>  <gml:identifier codeSpace="OGC">urn:ogc:def:security: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="IETF">urn:ogc:def:security:authentication: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="IETF">urn:ogc:def:security:authentication: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="IETF">urn:ogc:def:security:authentication:ietf:6750:Bearer</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)."[RFC 5246]   In case the client cannot provide a suitable and valid certificate, no TLS  connection gets established</gml:description>  <gml:identifier codeSpace="IETF">urn:ogc:def:security:authentication: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="OASIS">urn:ogc:def:security:authentication:wsse:username\_token</gml:identifier>  </gmx:CodeDefinition>  </gmx:codeEntry>  <gmx:codeEntry>  <gmx:CodeDefinition gml:id="InternetProtocol">  <gml:description>As specified in https://docs.oasis-open.org/security/saml/v2.0/saml-authn-context-2.0-os.pdf</gml:description>  <gml:identifier codeSpace="OASIS">urn:oasis:names:tc:SAML:2.0:ac:classes:InternetProtocol</gml:identifier>  </gmx:CodeDefinition>  </gmx:codeEntry>  <gmx:codeEntry>  <gmx:CodeDefinition gml:id="InternetProtocolPassword">  <gml:description>As specified in https://docs.oasis-open.org/security/saml/v2.0/saml-authn-context-2.0-os.pdf</gml:description>  <gml:identifier codeSpace="OASIS">urn:oasis:names:tc:SAML:2.0:ac:classes:InternetProtocolPassword</gml:identifier>  </gmx:CodeDefinition>  </gmx:codeEntry>  <gmx:codeEntry>  <gmx:CodeDefinition gml:id="Kerberos">  <gml:description>As specified in https://docs.oasis-open.org/security/saml/v2.0/saml-authn-context-2.0-os.pdf</gml:description>  <gml:identifier codeSpace="OASIS">urn:oasis:names:tc:SAML:2.0:ac:classes:Kerberos</gml:identifier>  </gmx:CodeDefinition>  </gmx:codeEntry>  <gmx:codeEntry>  <gmx:CodeDefinition gml:id="MobileOneFactorUnregistered">  <gml:description>As specified in https://docs.oasis-open.org/security/saml/v2.0/saml-authn-context-2.0-os.pdf</gml:description>  <gml:identifier codeSpace="OASIS">urn:oasis:names:tc:SAML:2.0:ac:classes:MobileOneFactorUnregistered</gml:identifier>  </gmx:CodeDefinition>  </gmx:codeEntry>  <gmx:codeEntry>  <gmx:CodeDefinition gml:id="MobileTwoFactorUnregistered">  <gml:description>As specified in https://docs.oasis-open.org/security/saml/v2.0/saml-authn-context-2.0-os.pdf</gml:description>  <gml:identifier codeSpace="OASIS">urn:oasis:names:tc:SAML:2.0:ac:classes:MobileTwoFactorUnregistered</gml:identifier>  </gmx:CodeDefinition>  </gmx:codeEntry>  <gmx:codeEntry>  <gmx:CodeDefinition gml:id="MobileOneFactorContract">  <gml:description>As specified in https://docs.oasis-open.org/security/saml/v2.0/saml-authn-context-2.0-os.pdf</gml:description>  <gml:identifier codeSpace="OASIS">urn:oasis:names:tc:SAML:2.0:ac:classes:MobileOneFactorContract</gml:identifier>  </gmx:CodeDefinition>  </gmx:codeEntry>  <gmx:codeEntry>  <gmx:CodeDefinition gml:id="MobileTwoFactorContract">  <gml:description>As specified in https://docs.oasis-open.org/security/saml/v2.0/saml-authn-context-2.0-os.pdf</gml:description>  <gml:identifier codeSpace="OASIS">urn:oasis:names:tc:SAML:2.0:ac:classes:MobileTwoFactorContract</gml:identifier>  </gmx:CodeDefinition>  </gmx:codeEntry>  <gmx:codeEntry>  <gmx:CodeDefinition gml:id="Password">  <gml:description>As specified in https://docs.oasis-open.org/security/saml/v2.0/saml-authn-context-2.0-os.pdf</gml:description>  <gml:identifier codeSpace="OASIS">urn:oasis:names:tc:SAML:2.0:ac:classes:Password</gml:identifier>  </gmx:CodeDefinition>  </gmx:codeEntry>  <gmx:codeEntry>  <gmx:CodeDefinition gml:id="PasswordProtectedTransport">  <gml:description>As specified in https://docs.oasis-open.org/security/saml/v2.0/saml-authn-context-2.0-os.pdf</gml:description>  <gml:identifier codeSpace="OASIS">urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport</gml:identifier>  </gmx:CodeDefinition>  </gmx:codeEntry>  <gmx:codeEntry>  <gmx:CodeDefinition gml:id="PreviousSession">  <gml:description>As specified in https://docs.oasis-open.org/security/saml/v2.0/saml-authn-context-2.0-os.pdf</gml:description>  <gml:identifier codeSpace="OASIS">urn:oasis:names:tc:SAML:2.0:ac:classes:PreviousSession</gml:identifier>  </gmx:CodeDefinition>  </gmx:codeEntry>  <gmx:codeEntry>  <gmx:CodeDefinition gml:id="PublicKeyX509">  <gml:description>As specified in https://docs.oasis-open.org/security/saml/v2.0/saml-authn-context-2.0-os.pdf</gml:description>  <gml:identifier codeSpace="OASIS">urn:oasis:names:tc:SAML:2.0:ac:classes:X509</gml:identifier>  </gmx:CodeDefinition>  </gmx:codeEntry>  <gmx:codeEntry>  <gmx:CodeDefinition gml:id="PublicKeyPGP">  <gml:description>As specified in https://docs.oasis-open.org/security/saml/v2.0/saml-authn-context-2.0-os.pdf</gml:description>  <gml:identifier codeSpace="OASIS">urn:oasis:names:tc:SAML:2.0:ac:classes:PGP</gml:identifier>  </gmx:CodeDefinition>  </gmx:codeEntry>  <gmx:codeEntry>  <gmx:CodeDefinition gml:id="PublicKeySPKI">  <gml:description>As specified in https://docs.oasis-open.org/security/saml/v2.0/saml-authn-context-2.0-os.pdf</gml:description>  <gml:identifier codeSpace="OASIS">urn:oasis:names:tc:SAML:2.0:ac:classes:SPKI</gml:identifier>  </gmx:CodeDefinition>  </gmx:codeEntry>  <gmx:codeEntry>  <gmx:CodeDefinition gml:id="PublicKeyXMLDigitalSignature">  <gml:description>As specified in https://docs.oasis-open.org/security/saml/v2.0/saml-authn-context-2.0-os.pdf</gml:description>  <gml:identifier codeSpace="OASIS">urn:oasis:names:tc:SAML:2.0:ac:classes:XMLDSig</gml:identifier>  </gmx:CodeDefinition>  </gmx:codeEntry>  <gmx:codeEntry>  <gmx:CodeDefinition gml:id="Smartcard">  <gml:description>As specified in https://docs.oasis-open.org/security/saml/v2.0/saml-authn-context-2.0-os.pdf</gml:description>  <gml:identifier codeSpace="OASIS">urn:oasis:names:tc:SAML:2.0:ac:classes:Smartcard</gml:identifier>  </gmx:CodeDefinition>  </gmx:codeEntry>  <gmx:codeEntry>  <gmx:CodeDefinition gml:id="SmartcardPKI">  <gml:description>As specified in https://docs.oasis-open.org/security/saml/v2.0/saml-authn-context-2.0-os.pdf</gml:description>  <gml:identifier codeSpace="OASIS">urn:oasis:names:tc:SAML:2.0:ac:classes:SmartcardPKI</gml:identifier>  </gmx:CodeDefinition>  </gmx:codeEntry>  <gmx:codeEntry>  <gmx:CodeDefinition gml:id="SoftwarePKI">  <gml:description>As specified in https://docs.oasis-open.org/security/saml/v2.0/saml-authn-context-2.0-os.pdf</gml:description>  <gml:identifier codeSpace="OASIS">urn:oasis:names:tc:SAML:2.0:ac:classes:SoftwarePKI</gml:identifier>  </gmx:CodeDefinition>  </gmx:codeEntry>  <gmx:codeEntry>  <gmx:CodeDefinition gml:id="Telephony">  <gml:description>As specified in https://docs.oasis-open.org/security/saml/v2.0/saml-authn-context-2.0-os.pdf</gml:description>  <gml:identifier codeSpace="OASIS">urn:oasis:names:tc:SAML:2.0:ac:classes:Telephony</gml:identifier>  </gmx:CodeDefinition>  </gmx:codeEntry>  <gmx:codeEntry>  <gmx:CodeDefinition gml:id="TelephonyNomadic">  <gml:description>As specified in https://docs.oasis-open.org/security/saml/v2.0/saml-authn-context-2.0-os.pdf</gml:description>  <gml:identifier codeSpace="OASIS">urn:oasis:names:tc:SAML:2.0:ac:classes:NomadTelephony</gml:identifier>  </gmx:CodeDefinition>  </gmx:codeEntry>  <gmx:codeEntry>  <gmx:CodeDefinition gml:id="PersonalTelephony">  <gml:description>As specified in https://docs.oasis-open.org/security/saml/v2.0/saml-authn-context-2.0-os.pdf</gml:description>  <gml:identifier codeSpace="OASIS">urn:oasis:names:tc:SAML:2.0:ac:classes:PersonalTelephony</gml:identifier>  </gmx:CodeDefinition>  </gmx:codeEntry>  <gmx:codeEntry>  <gmx:CodeDefinition gml:id="AuthenticatedTelephony">  <gml:description>As specified in https://docs.oasis-open.org/security/saml/v2.0/saml-authn-context-2.0-os.pdf</gml:description>  <gml:identifier codeSpace="OASIS">urn:oasis:names:tc:SAML:2.0:ac:classes:AuthenticatedTelephony</gml:identifier>  </gmx:CodeDefinition>  </gmx:codeEntry>  <gmx:codeEntry>  <gmx:CodeDefinition gml:id="SecureRemotePassword">  <gml:description>As specified in https://docs.oasis-open.org/security/saml/v2.0/saml-authn-context-2.0-os.pdf</gml:description>  <gml:identifier codeSpace="OASIS">urn:oasis:names:tc:SAML:2.0:ac:classes:SecureRemotePassword</gml:identifier>  </gmx:CodeDefinition>  </gmx:codeEntry>  <gmx:codeEntry>  <gmx:CodeDefinition gml:id="TLSClient">  <gml:description>As specified in https://docs.oasis-open.org/security/saml/v2.0/saml-authn-context-2.0-os.pdf</gml:description>  <gml:identifier codeSpace="OASIS">urn:oasis:names:tc:SAML:2.0:ac:classes:TLSClient</gml:identifier>  </gmx:CodeDefinition>  </gmx:codeEntry>  <gmx:codeEntry>  <gmx:CodeDefinition gml:id="TimeSyncToken">  <gml:description>As specified in https://docs.oasis-open.org/security/saml/v2.0/saml-authn-context-2.0-os.pdf</gml:description>  <gml:identifier codeSpace="OASIS">urn:oasis:names:tc:SAML:2.0:ac:classes:TimeSyncToken</gml:identifier>  </gmx:CodeDefinition>  </gmx:codeEntry>  <gmx:codeEntry>  <gmx:CodeDefinition gml:id="Unspecified">  <gml:description>As specified in https://docs.oasis-open.org/security/saml/v2.0/saml-authn-context-2.0-os.pdf</gml:description>  <gml:identifier codeSpace="OASIS">urn:oasis:names:tc:SAML:2.0:ac:classes:unspecified</gml:identifier>  </gmx:CodeDefinition>  </gmx:codeEntry>  </gmx:CodeListDictionary>  </gmx:codelistItem>  <!--=== EOF ===--> </gmx:CT\_CodelistCatalogue> |

Annex D: Examples of Annotated Capabilities (informative)

For Testbed 13, the following two secured Geoserver 2.11.1 deployments are available as proof of concept for this standard.

Please request the Capabilities via the main user interface to observe the security annotations expressed for the different OGC Web Services and versions.

The initial Authentication Codelist is hosted here:

<https://www.tb13.secure-dimensions.de/authnCodeList>

Prototype implementation of this standard as a security proxy for a default Geoserver 2.11.1 with SAML2 based authentication:

<https://sp.tb13.secure-dimensions.de/geoserver/web/>

Prototype implementation of this standard as a security proxy for a default Geoserver 2.11.1 with OAuth2 Bearer Access Token:

<https://rs.tb13.secure-dimensions.de/geoserver/web/>

1. Obsoletes RFC 2616 – HTTP 1.1 [↑](#footnote-ref-1)
2. Andreas Matheus [↑](#footnote-ref-2)