Open Geospatial Consortium

Submission Date: 2014-03-11

Approval Date: 2016-02-23

Publication Date: 2016-06-10

Document uri: http://www.opengis.net/doc/IS/cat/csw/3.0

URL for this OGC® document: http://docs.opengeospatial.org/is/12-176r7/12-176r7.html

Reference number of this document: 12-176r7

Version 3.0

Category: OGC® Implementation Standard

Editors: Doug Nebert, Uwe Voges, Panagiotis Vretanos, Lorenzo Bigagli, Bruce Westcott

OGC® Catalogue Services 3.0 Specification - HTTP Protocol Binding

The OGC Catalog Service 3.0 SWG would like to dedicate this work to the memory of Douglas D. Nebert. Doug was the convener and chair of the group who coordinated the editing of this specification. He was a long time advocate for developing a catalogue standard within OGC and indeed was a strong, vocal and passionate supporter of the OGC and its ideals. He will be remembered and missed. You can read more about Doug on our website.

Warning

This formatted version of 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. The normative version is available at: http://docs.opengeospatial.org/is/12-176r7/12-176r7.html

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.

Document type:	OGC® Standard
Document subtype:	Part of Implementation Standard
Document stage:	Approved for public release
Document language:	English

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

Figures

Page

Tables

Page

i. Abstract

This document specifies the HTTP profile of the CSW General Model part (see OGC 12-168r6). The General Model specifies the abstract interfaces between clients and catalogue services. This standard specifies the mappingof the Catalogue abstract model interface into the HTTP protocol binding.

In this HTTP protocol binding, operation requests and responses are sent between clients and servers using the HTTP GET and/or HTTP POST methods. Two equivalent request encodings are defined in this standard. The first using keyword-value pairs (KVP) which is suitable for use with the HTTP GET method. The second using XML which is suitable for use with the HTTP POST method.

This standard defines operations that allow a client to get a service description document for the catalogue (i.e. GetCapabilities); operations that allow a client to, at runtime, interrogate the service about the kinds of data available (i.e. GetDomain); operations that allow a client to retrieve records from the catalogue (i.e. GetRecordById and GetRecords); operations that allow a client to add, modify and remove records from the catalogue service (i.e. Transaction, Harvest, UnHarvest).

ii. Keywords

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

ogcdoc, OGC document,asynchronous, ATOM, catalogue, CQL, client, CSW, csw:Record, distributed, Dublin Core, federated, filter, GetCapabilities, GetDomain, GetRecords, GetRecordById, Harvest, http, https, KVP, metadata, periodic, record, request, resource, response, search, server, schema, spatial, temporal, Transaction, UnHarvest, XML, XML-Schema

iii. Preface

This document is part of the Catalogue Services Implementation Standard. Unlike previous versions of this standard, Catalogue 3.0 is now divided into two parts: A general model and the specification of the HTTP protocol binding, building on the general model (see OGC 12-168r6).

This document, through its implementation profiles, references several external standards and specifications as dependencies. These documents are listed in Clause 3, Normative references.

Annex A refers to OGC document 14-014r3 – HTTP Protocol Binding – Abstract Test Suite. This Abstract Conformance Test Suite is normative to this standard and shall be implemented when the catalogue services HTTP protocol binding is implemented. Annex C is also normative. All other annexes are informative.

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.

iv. Document terms and definitions

This document uses the specification terms defined in Subclause 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 specification.

v. Submitting organizations

The organizations that submitted the Catalogue Services document to the Open Geospatial Consortium are listed in Clause iv of the general model (OGC 12-168).

vi. Submitters

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

Name	Organization
Doug Nebert	U.S. Federal Geographic Data Committee
Uwe Voges	con terra GmbH
Panagiotis (Peter) Vretanos	CubeWerx, Inc.
Lorenzo Bigagli	National Research Council of Italy (CNR)
Bruce A. Westcott	Intergraph Corporation

OGC® Catalogue Services 3.0 Specification - HTTP Protocol Binding

1 Scope

OGC Catalogue Services support the ability to publish and search collections of descriptive information (metadata records) for geospatial data, services, and related resources. Metadata in catalogues represent resource characteristics that can be queried and presented for evaluation and further processing by both humans and software. Catalogue services are required to support the discovery and binding to registered information resources within an information community.

This part of the OGC® Catalogue Services standard specifies the HTTP (or Catalogue Service for the Web (CSW)) protocol binding that builds on the general model as specified in OGC 12-168r6. The CSW General model abstractly specifies the interfaces between clients and catalogue services. This profile specifies the conversion of that abstract interface into the HTTP protocol binding. In the HTTP protocol binding, operation requests and responses are sent between clients and servers using the HTTP GET and/or POST methods.

This standard specified three classes of service operations: OGC_Service, Discovery and Manager.OGC_Service class operations are the standard set of operations that every OGC service must implemement in order to work with OGC Catalogue Services. The operations allow the service to provide metadata about itself via a standard service description document and also a means of retrieving catalogues records based on their unique identifier.

The Discovery class operations allow the service to be interrogated to retrieve run time information about the data offered by the service as well as providing means of retrieveing records from the catalogue using a general predicate lanauge to define constraints the define the subset of records to be retrieved.

The Manager class operations allow records to be added, modified and removed from the catalogue service.

This International Standard defines seven operations:

- GetCapabilities (OGC_Service)
- GetRecordById (OGC_Service)
- GetDomain (Discovery)
- GetRecords (Discovery)

- Transaction (Manager)
- Harvest (Manager)
- UnHarvest (Manager)

There are two equivalent encodings defined in this standard for most operations. A keyword-value (KVP) pair encoding suitable for use with the HTTP GET method and an XML encoding suitable for use with the HTTP POST method. Furthermore, the specification contains discussions about how to use the XML-encoded operations with SOAP (see SOAP) with this standard.

The target audience for this standard is the community of software developers who are implementers of OGC compliant Catalogue servers and clients using HTTP.

2 Conformance

Few usage scenarios require the full implementation of this standard. Therefore, service providers may want to specify requirements for only the subset needed to fulfil their service; or system developers may want to document which subset of this standard it is that that they have implemented and conform to. These named conformance classes defined in this clause help in specifying such subsets.

Tables 1 shows the conformance classes defined in this document. Table 2 shows the relative identifiers for each corresponding requirements class, requirement and abstract test. Table 3 shows how to form the complete URI for each of these components.

Table 1 defines conformance classes based on the operations and behaviour that a catalogue service claims to implement. The behaviours that shall be implemented for each of the conformance classes are described. All implementations of this standard shall implement the Basic-Catalogue conformance class which additionally requires the Filter-KVP conformance class, the CSW-response and the ATOM-response classes to be imlemented. All other conformance classes are optional.

Table 1 — Conformance classes

Conformance Class Name	Description

Basic-Catalogue	- Implements KVP-encoding of the GetCapabilities operation
	- Implements KVP-encoding of the GetRecordById operation
	- Implements KVP-encoding of the GetRecords operations
	  - Implements the KVP-encoding for the basic retrieval parameters for the GetRecords operation
	  - Does not implement the DistributedSearch parameter
	  - Does not implement the elapsedTime parameter
	  - Does not implement the FederatedSearchResult response
	  - Does not implement the ResponseHandler parameter
	  - Implements the Filter-FES-KVP conformance class
	  - Implements the CSW-response conformance class

Conformance Class Name	Description
	○ Implements the ATOM-response conformance class
OpenSearch	□ Provides link in the Capabilities document to an OpenSearch description document (mapping the CSW query parameters that the implementation supports to the OpenSearch parameters defined in 10-032r8) □ Implements the ATOM-Response conformance class □ Implements the Filter-FES-KVP conformance class
GetCapabilities-XML	□ Implements the XML-encoding of the GetCapabilities operation
GetRecordById-XML	□ Implements the XML-encoding of the GetRecordById operation
GetRecords-Basic-XML	□ Implements the XML-encoding of GetRecords operation ○ Implements the XML-encoding for the basic retrieval options ○ Does not implement the DistributedSearch parameter ○ Does not implement the elapsedTime parameter ○ Does not implement the FederatedSearchResult response ○ Does not implement the ResponseHandler parameter □ Implements the Filter-XML conformance class
GetRecords-Distributed-XML	□ Implements the GetRecords-Basic-XML conformance class □ Implements the XML-encoding for the DestributedSearch parameter □ Implements the elapsedTime parameter in the response □ Implements the FederatedSearchResult response
GetRecords-Distributed-KVP	□ Implements the Basic-Catalogue conformance class □ Implements KVP-encoding for the DistributedSearch parameter □ Implements the elapsedTime parameter in the response □ Implements the FederatedSearchResult response
GetRecords-Async-XML	□ Implements the GetRecords-Basic-XML conformance class □ Implements the XML-encoding for the ResponseHandler parameter for the GetRecords operation
GetRecords-Async-KVP	□ Implements the Basic-Catalogue conformance class □ Implements the KVP-encoding for the ResponseHandler parameter for the GetRecords operation
GetDomain-XML	□ Implements the XML-encoding for the GetDomain operation
GetDomain-KVP	□ Implements the KVP-encoding for the GetDomain operation
Transaction	□ Implements the Transaction operation □ Declares which information models that the server supports are transactional
Harvest-Basic-XML	□ Implements the XML-encoding for the Harvest operation ○ Does not implement the XML-encoding for the Response-Handler parameter

Conformance Class Name	Description
	○ Does not implement the XML-encoding for the HarvestInterval parameter □ Implements the XML-encoding for the UnHarvest operation ○ Does not implement the XML-encoding for the ResponseHandler parameter
Harvest-Basic-KVP	□ Implements the KVP-encoding for the Harvest operation ○ Does not implement the KVP-encoding for the Response-Handler parameter ○ Does not implement the KVP-encoding for the HarvestInterval parameter □ Implements the KVP-encoding for the UnHarvest operation ○ Does not implement the KVP-encoding for the Response-Handler parameter
Harvest-Async-XML	□ Implements the Harvest-Basic-XML conformance class □ Implements the XML-encoding for the ResponseHandler parameter
Harvest-Async-KVP	□ Implements the Harvest-Basic-KVP conformance class □ Implements the KVP-encoding for the ResponseHandler parameter
Harvest-Periodic-XML	□ Implements the Harvest-Basic-XML conformance class □ Implements the XML-encoding for the HarvestInterval parameters
Harvest-Periodic-KVP	□ Implements the Harvest-Basic-KVP conformance class □ Implements the KVP-encoding for the HarvestInterval parameters
Filter-CQL	□ Implements the text-encoding for CQL in all supported contexts
Filter-FES-XML	□ Implements the XML-encoding for Filter as per FES 2.0 □ Implements at least the following set of operators: ○ Logical operators: And, Or, Not ○ Comparison operations: PropertyIsEqualTo, PropertyIsNotEqualTo, PropertyIsLessThan, PropertyIsGreaterThan, PropertyIsLessThanOrEqualTo, PropertyIsGreaterThanOrEqualTo, PropertyIsLike, PropertyIsBetween ○ Spatial operators: BBOX ○ temporal operators: TVOverlaps ○ Implements SortBy
Filter-FES-KVP	□ Implements the Q KVP-parameter □ Implements the RECORDIDS KVP-parameter □ Implements the BBOX parameter ○ Additionally declares support for the BBOX operator in the Filter capabilities section
Filter-FES-KVP-Advanced	□ Implements the GEOMETRY, GEOMETRY_CRS, RELATION, DISTANCE, DISTANCE_UOM parameters

Conformance Class Name	Description
	○ Additionally declares the list of supported CRS' in the servers capabilities document ○ Additionally declares the list of supported spatial operators and operands in the Filter capabilities section □ Implements the LAT, LON, RADIUS KVP-parameters ○ Additionally declares support the Not Disjoint spatial operator in the Filter capabilities section ○ Declares the gml:CircleByCenterPoint geometry as a valid spatial operand in the Filter capabilities section □ Implements the TIME and TRELATION parameters □ Additionally declares the list of supported temporal operators in the Filter capabilities sectionImplements SORTBY parameter □ Implements the CONSTRAINT, CONSTRAINTLANGUAGE and CONSTRAINTVERSION parameters ○ Implements one of Filter-CQL or Filter-FES-XML
CSW-Response	□ Implements the csw:Record schema for XML response documents
ATOM-response	□ Implements the ATOM schema for XML response documents

For each conformance class listed in Table 1 there is a corresponding requirement class with the same name. Table 2 lists each requirement class and the set of requirements that correspond to each class. Furthermore, each requirement in Table 2 has a corresponding test in the abstract test suite (see OGC 14014r3) with the same numerical sequence number (i.e. Request-013 has a corresponding test in the abstract test suite named Test-013).

Table 2 — Requirement classes

Requirement Class	Requirements
Basic-Catalogue	□ Requirements 001 thru 014, 035 thru 048, 063 thru 068 □ Requirements 073 thru 078, 081 thru 093, 095 thru 102 □ Requirements 123 thru 133, 136 thru 139, 133a, 141
OpenSearch	□ Requirements 008, 021 thru 023
GetCapabilities-XML	□ Requirements 044 thru 048, 174
GetRecordById-XML	□ Requirements 123 thru 133, 133a
GetRecords-Basic-XML	□ Requirements 073 thru 078, 081 thru 093, 095 thru 102
GetRecords-Distributed-XML	□ Requirements 069 thru 071
GetRecords-Distributed-KVP	□ Requirements 069 thru 071

Requirement Class	Requirements
GetRecords-Async-XML	Requirements 072, 111 thru 117
GetRecords-Async-KVP	Requirements 072, 111 thru 117
GetDomain-XML	Requirements 049 thru 062
GetDomain-KVP	Requirements 049, 050, 052 thru 062
Transaction	Requirements 142 thru 151
Harvest-Basic-XML	Requirements 152 thru 154, 159 thru 164, 167
Harvest-Basic-KVP	Requirements 152 thru 154, 167
Harvest-Async-XML	Requirements 155 thru 158, 165,166, 168 thru 172
Harvest-Async-KVP	Requirements 155 thru 158, 165, 166, 168 thru 173
Harvest-Periodic-XML	Requirements 175, 176
Harvest-Periodic-KVP	Requirement 175, 176
Filter-CQL	Requirement-104
Filter-FES-XML	Requirements 015, 016, 103, 105, 108 thru 110
Filter-FES-KVP	Requirement-017
Filter-FES-KVP-Advanced	Requirements 018, 106 thru 109
CSW-response	Requirements 024 thru 034, 079, 118, 120, 121, 134
ATOM-response	Requirements 019 thru 023, 080, 119, 122, 135, 140

Table 3 below shows how to form the complete URI that identifies each requirements class, requirement and abstract test. The values for the *"<req. class name>"*, "*<req. identifier>*" and "*<abstract test identifier>*" parameters corresponds to each of the colums from Table 2 respectively.

Table 3 — URIs for Requirements classes, requirements and abstract tests

Requirement Class URIs	`http://www.opengis.net/spec/csw/3.0/req/<req. class name>`
Requirement URIs	`http://www.opengis.net/spec/csw/3.0/req/<req. class name>/<req. identifier`[1]`>`

Abstract test URIs	`http://www.opengis.net/spec/csw/3.0/req/<req. class name>/<abstract test identifier[2]>`
1. The lexical pattern for requirement indentifiers is **Requirement-*NNN***. 2. The lexical pattern for abstract test identifiers is **Test-*NNN*.** where ***NNN*** is a 3-digit integer.	
Example URIs: Requirement class: http://www.opengis.net/spec/csw/3.0/req/Basic-Catalogue Requirement: http://www.opengis.net/spec/csw/3.0/req/Basic-Catalogue/Requirement-001 Abstract Test: http://www.opengis.net/spec/csw/3.0/req/Basic-Catalogue/Test-001	

Table 4 — HTTP Binding requirements classes that satisfy GM requirements

Requirement Class from GM	Satisfying HTTP Binding Requirement Class(es)
http://www.opengis.net/spec/cat/3.0/req/base/query-language	Filter-CQL, Filter-FES-XML, Filter-KVP, Filter-KVP-Advanced
http://www.opengis.net/spec/cat/3.0/req/common-query language	Filter-CQL, Filter-FES-XML, Filter-KVP, Filter-KVP-Advanced
http://www.opengis.net/spec/cat/3.0/req/classified-as	Filter-CQL, Filter-FES-XML, Filter-FES-KVP-Advanced
http://www.opengis.net/spec/cat/3.0/req/base/common-queryables	CSW-Response, ATOM-Response
http://www.opengis.net/spec/cat/3.0/req/base/common-returnables	CSW-Response, ATOM-Response
http://www.opengis.net/spec/cat/3.0/req/base/get-capabilities	Basic-Catalogue, GetCapabilities-XML
http://www.opengis.net/spec/cat/3.0/req/base/getresourcebyid	Basic-Catalogue, GetRecordById-XML
http://www.opengis.net/spec/cat/3.0/req/base/query	Basic-Catalogue, GetRecords-Basic-XML, GetRecords-Distributed-XML, GetRecords-Distributed-KVP, GetRecords-Async-XML, GetRecords-Async-KVP
http://www.opengis.net/spec/cat/3.0/req/base/describe-records	CSW-Response, ATOM-Response
http://www.opengis.net/spec/cat/3.0/req/getdomain	GetDomain-XML, GetDomain-KVP
http://www.opengis.net/spec/cat/3.0/req/transaction	Transaction

Requirement Class from GM	Satisfying HTTP Binding Requirement Class(es)
http://www.opengis.net/spec/cat/3.0/req/harvest	Harvest-Basic-XML, Harvest-Basic-KVP, Harvest-Async-XML, Harvest-Async-KVP, Harvest-Periodic-XML, Harvest-Periodic-KVP

3 Normative references

The following normative documents contain provisions that, through reference in this text, constitute provisions of this specification. 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.

IETF RFC 2045 (November 1996), *Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies*, Freed, N. and Borenstein N., eds., http://www.ietf.org/rfc/rfc2045.txt

IETF RFC 2141 (May 1997), *URN Syntax*, R. Moats, http://www.ietf.org/rfc/rfc2141.txt

IETF RFC 2616 (June 1999), *Hypertext Transfer Protocol – HTTP/1.1*, Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., Berners-Lee, T., http://www.ietf.org/rfc/rfc2616.txt

IETF RFC 2396 (August 1998), *Uniform Resource Identifiers (URI): Generic Syntax*, Berners-Lee, T., Fielding, N., and Masinter, L., eds., http://www.ietf.org/rfc/rfc2396.txt

IETF RFC 4287 (December 2005), The Atom Syndication Format, M. Nottingham, R. Sayre, eds., http://www.ietf.org/rfc/rfc4287.txt

IETF RFC 4646 (September 2006), *Tags for Identifying Languages*, Phillips, A., Davis, M., eds., http://www.ietf.org/rfc/rfc4646.txt

IANA, Internet Assigned Numbers Authority, *MIME Media Types*, available at http://www.iana.org/assignments/media-types/

OGC 10-032r8, OGC® OpenSearch GeoSpatial and Temporal Extensions

OGC 06-121r9 , *OGC Web Services Common Standard, 2010-04-07.*

OGC 09-026r1 (ISO/DIS 19143) , *OpenGIS Filter Encoding Standard, 2010-11-22.*

OGC 12-168, *OGC® Catalogue Services Specification Version 3.0 – Part 1 -- General model*

NOTE This abstract part of this Catalogue Services Implementation Specification contains a list of normative references that also apply to this part of this HTTP binding part.

OGC 14-014r3, *OpenGIS Catalogue Services Specification – HTTP Protocol Binding – Abtract Test Suite*

W3C Recommendation January 1999, *Namespaces In XML*, http://www.w3.org/TR/2000/REC-xml-names

W3C Recommendation 6 October 2000, *Extensible Markup Language (XML) 1.0* (Second Edition), http://www.w3.org/TR/REC-xml

W3C Recommendation 2 May 2001: *XML Schema Part 0: Primer,* http://www.w3.org/TR/2001/REC-xmlschema-0-20010502/

W3C Recommendation 2 May 2001: *XML Schema Part 1: Structures,* http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/

W3C Recommendation 2 May 2001: *XML Schema Part 2: Datatypes,* http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/

W3C Recommendation (24 June 2003): *SOAP Version 1.2 Part 1: Messaging Framework*, http://www.w3.org/TR/SOAP/

W3C Recommendation (16 November 1999): *XML Path Language (XPath)* Version 1.0, http://www.w3.org/TR/xpath.html

X.667 Information technology - Open Systems Interconnection - Procedures for the operation of OSI Registration Authorities: Generation and registration of Universally Unique Identifiers (UUIDs) and their use as ASN.1 object identifier components, http://www.itu.int/rec/T-REC-X.667-200808-I/en

In addition to this document, this standard includes several normative XML Schema files. These are posted online at http://schemas.opengis.net/csw/3.0. These XML Schema files are also bundled with this document. In the event of a discrepancy between the bundled and online versions of the XML Schema files, the online files shall be considered normative.

4 Terms and definitions

For the purposes of this specification, the definitions specified in Clauses 4 of the general model (see OGC 12-168) and the OWS Common Implementation Specification (see OGC 06-121r9, Subclause 5.3) shall apply. In addition, the following terms and definitions apply.

4.1 attribute

<XML>

name-value pair contained in an **element**

4.2 base URL

URL prefix to which parameters are appended to form a valid request

4.3 client

software component that can invoke an **operation** from a **server**

4.4 coordinate
one of a sequence of n numbers designating the position of a point in n-dimensional space

4.5 coordinate reference system
coordinate system that is related to an object by a datum

4.6 coordinate system
set of mathematical rules for specifying how **coordinates** are to be assigned to points

4.7 element
<XML>
basic information item of an XML document containing child elements, **attributes** and character data

4.8 filter expression
predicate expression encoded using XML

4.9 interface
named set of **operations** that characterize the behaviour of an entity

4.10 local resource
resource that is under the direct control of a system

4.11 locator attribute
attribute whose values is a reference to a **local** or **remote resource**

4.12 Multipurpose Internet Mail Extension (MIME) type
media type and subtype of data in the body of a message that designates the native representation (canonical form) of such data

4.13 namespace
<XML>
collection of names, identified by a URI reference which are used in XML documents as **element** names and **attribute** names

4.14 operation
specification of a transformation or query that an object may be called to execute

4.15 property
facet or attribute of an object, referenced by a name

4.16 resource
asset or means that fulfils a requirement

4.17 remote resource
a resource that is not under the direct control of a system

4.18 request
invocation of an **operation** by a **client**

4.19 resolve
retrieval of a referenced resource and its insertion into a server-generated response document

4.20 response
result of an **operation** returned from a **server** to a **client**

4.21 response model
schema defining the properties of each **feature** type that can appear in the **response** to a query **operation**

4.22 schema
formal description of a model

4.23 schema
<XML Schema>
collection of **schema** components within the same target **namespace**

4.24 server
particular instance of a **service**

4.25 service
distinct part of the functionality that is provided by an entity through **interfaces**

4.26 service metadata
metadata describing the **operations** and geographic information available at a **server**

4.27 traversal
<XML>
using or following an XLink link for any purpose

4.28 Uniform Resource Identifier
unique identifier for a resource, structured in conformance with IETF RFC 2396

5 Symbols (and abbreviated terms)

Some frequently used abbreviated terms:

CSW	Catalogue Services for the Web
DCP	Distributed Computing Platform
HTTP	Hypertext Transfer Protocol
ISO	International Organization for Standardization
KVP	Keyword Value Pair
MIME	Multipurpose Internet Mail Extensions
OGC	Open Geospatial Consortium, also referred to as OGC®
TBD	To Be Determined

TBR To Be Reviewed

XML Extensible Markup Language

6 Common service elements

6.1 Introduction

This HTTP protocol binding maps each of the abstract model operations, specified in the general model (see OGC 12-168), to a corresponding Catalogue Services for the Web (CSW) operation. This clause describes the request and response messages that are common to all web-based catalogue services. The basic message exchange pattern is illustrated in Figure 1.

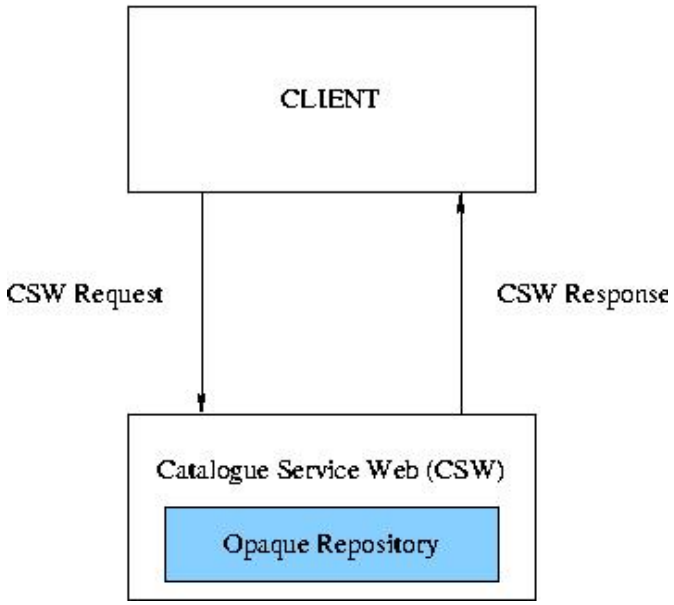

Figure 1 — CSW basic message exchange pattern

The interaction between a client and a server is accomplished using a standard HTTP request-response model. Specifically, a client sends a request to a server using HTTP, and expects to receive a response to the request or an exception message.

Request and response messages are encoded as keyword-value pairs either within a request URI or using an XML entity-body. Requests may also be embedded in a messaging frame-work such as SOAP.

6.2 HTTP methods

The Hypertext Transfer Protocol (HTTP) is a generic, stateless, application-level protocol that is widely used to exchange information on the web. The general rules for Hypertext Transfer Protocol (HTTP) based request and response encoding which have to be considered by all implementations of OGC Web Services (and so for all CSW implementations) are described in detail in chapter 11 of OWS Common (OGC 06-121r9).

The OGC Catalogue standard uses the following HTTP/1.1 methods.:

Table 5— Selected HTTP Request Methods

Method name	Semantics
GET	Used to retrieve whatever information (in the form of an entity) is identified by the Request-URI
POST	Used to request that the origin server accept the entity enclosed in the request as data to be processed by the resource identified by the Request-URI in the Request-Line

Requirement-001
CSW HTTP/1.1 messages containing an entity-body shall include a Content-Type header field defining the media type of that body (RFC 2616, 7.2.1)

Implementers should be mindful that the function of some request parameters (e.g. outputFormat) overlap the functionality of some of the HTTP message headers (e.g., Accept).

Requirement-002

If a CSW request includes a parameter (e.g. outputFormat) that performs a function that is similar to an HTTP message header (e.g. Accept), the server shall use the value of the request parameter to process the request.

Requirement-003

If a CSW request includes an HTTP messages header (e.g. Accept) that performs a function that is similar to a CSW request parameter (e.g. outputFormat), the server shall use the value of HTTP header if the corresponding CSW request parameter is not included in the request.

Requirement-004

If a CSW request includes a parameter (e.g. outputFormat) that performs a function that is similar to an HTTP message header (e.g. Accept) and both are included in a request then their values must agree otherwise an InvalidParameterValue exception shall be raised as specified in Subclause 6.7.

Requirement-005

If a CSW request defines an optional parameter (e.g. outputFormat) that performs a function that is similar to an HTTP message header (e.g. Accept) and neither is specified,

then the default value of the CSW request parameter shall apply if one is defined for that parameter.

6.3 Namespaces

Namespaces (see W3C Recommendation January 1999) are used to discriminate XML vocabularies from one another. For this standard, the following normative namespaces are used:

(http://www.opengis.net/cat/csw/3.0) – for CSW interface and message vocabulary
(http://purl.org/dc/elements/1.1/) – for Dublin Core elements
(http://purl.org/dc/terms/) – for Dublin Core terms
(http://www.opengis.net/fes/2.0) – for OGC Filter vocabulary
(http://www.opengis.net/gml/3.2) – for OGC GML 3.2 vocabulary
(http://www.opengis.net/ows/2.0) – for OGC Web Service Common vocabulary
(http://www.w3.org/2005/Atom) – for the ATOM vocabulary (as per RFC 4287)

In addition, a CSW implementation may make use of additional namespaces depending on which additional information models and/or CSW profiles the server implements. For example, a server that also support the ebRIM v3.0 information model would make use of the urn:oasis:names:tc:ebxml-regrep:xsd:rim:3.0 namespace.

NOTE: These XML Schema Documents use a new namespace identifier URI instead of the previous namespace identifier URI, namely http://www.opengis.net/cat/csw/3.0 instead of http://www.opengis.net/cat/csw/2.0.2. This change was made based on the TC decision described in Clause 11 in OGC Best Practices Paper 06-135r1. That decision was based on the discussion in document 05-065r3 “Change Request for Namespaces for versions and profiles of XML Schemas”. A new namespace identifier URI is used for this Catalogue version 3.0.0 because these reasons apply to this version 3.0.0.

6.4 Obtaining service metadata

Service metadata can be explicitly requested from a conforming catalogue using the GetCapabilities request (see 7.1).

Service metadata can also be implicitly requested from a conforming catalogue by accessing the base URL of a conforming catalogue using the HTTP GET method.

How a service responds to an implicit request for service metadata depends on the value of the “Accept” HTTP headers (see HTTP/1.1).

Requirement-006

If the GET method is used to access the base URL of a conforming catalogue and the Accept header is not set then the service shall respond with an OGC capabilities document, encoded in XML, as described in 6.8.3.

Requirement-007

If the HTTP GET method is used to access the base URL of a conforming catalogue and the Accept header is set to include the MIME types "text/xml" or "application/xml" as the most desirable response then the server shall response with an OGC capabilities document encoded in XML as described in 7.1.3.

Requirement-008

If the HTTP GET method is used to access the base URL of a catalogue conforming to the OpenSearch conformance class (see Table 1) and the Accept header is set to include the MIME type "application/opensearchdescription+xml" as the most desirable response the server shall respond with an OpenSearch description document (see http://www.opensearch.org/Specifications/OpenSearch/1.1 and OGC 10-032r8).

NOTE: Obtaining service metadata in the form of an OGC capabilities document explicitly, using the GetCapabilities request (see 7.1) affords a client some degree of control over the content of the response. For example, using the GetCapabilities request, a client may request that only certain sections of an OGC capabilities document appear in the response. Obtaining service metadata implicitly, whether is in the form of an OGC capabilities document or some other format, only allows for retrieval of a complete service metadata document. There is no provision for the client to control the content of an implicitly request for service metadata.

6.5 Request encoding

6.5.1 Introduction

This standard defines up to two encodings for each CSW request. A keyword-value (KVP) pair encoding and an XML encoding. The XML-encoding may, additionally, be embedded within a SOAP envelope.

6.5.2 General model message mapping

Table 6 maps the general model operations (see OGC 12-168), to the Catalogue Service for the Web (CSW) operations. This table does not list the general model operations that are not mapped to CSW operations.

Table 6 — General model to CSW mapping

General Model Operation	CSW Operation
OGC_Service.getCapabilities	GetCapabilities
OGC_Service.GetResourceById	GetRecordById
Discovery.query	GetRecords
Discovery.getDomain	GetDomain
Manager.transaction	Transaction
Manager.hervestRecords	Harvest

6.5.3 HTTP methods

Only interactions using the HTTP GET and HTTP POST methods are described in this standard (see 6.2).

KVP-encoded requests are suitable for use with the HTTP GET method.

The HTTP POST method may be used to accept requests encoding using any of the KVP, XML or SOAP encodings defined in this standard.

Requirement-009

Servers shall, in their capabilities document, indicate the specific request encodings that can be processed using the HTTP POST method (see Table 20).

The general rules for the use of SOAP 1.2 messages (for communication between a catalogue client and a CSW) are described in chapter 11.8 of OGC 06-121r9.

6.5.4 KVP encoding rules

KVP operation parameter values shall be encoded as described in OGC 06-121r9, Clause 11.5.Common request parameters

Requirement-010

All CSW operation requests except for GetCapabilities shall include the parameters service, request and version specified in Table 29 of OGC 06-121r9.

Only one of these parameters is included in the general catalogue model, the others are specific to the HTTP protocol binding.

In KVP encoding, these common parameters in CSW operation requests are encoded as shown in Table 7.

Requirement-011

For KVP encoding, parameter names shall be treated as being case insensitive.

Requirement-012

For KVP encoding, parameter values shall be treated as being case sensitive.

Table 7 — KVP encoding of common operation request parameters

Keyword	Datatype and value	Optionality	Parameter in general model
service	Character String type Default value of "CSW"	One (mandatory)	serviceId
request	Character String type Value is operation name (e.g., "GetRecordById")	One (Mandatory)	(none)
version	Character String type Default value of "3.0.0"	One (Mandatory)	(none)

In XML encoding, all operation request elements, except for GetCapabilities, are extended from the following XML Schema fragment:

```
<xsd:complexType name="RequestBaseType" id="RequestBaseType" abstract="true">
   <xsd:attribute name="service" type="ows:ServiceType"
                  use="optional" default="CSW"/>
   <xsd:attribute name="version" type="ows:VersionType"
                  use="optional" default="3.0.0"/>
</xsd:complexType>
```

The "service" parameter is used to indicate that the request is a CSW request.

Requirement-013

The "service" parameter shall be specified for all CSW requests.

The "version" parameter is used to indicate that the associated CSW request conforms to this standard.

Requirement-014

Servers that implement this standard shall set the value of the version parameter to 3.0.0.

XML-encoded requests do not include a "request" parameter because the name of the requested operation is encoded as the name of the root XML element encoding the request.

6.5.5 Query predicate encoding

6.5.5.1 Introduction

The general model allows catalogue clients to specify the predicate language used to constrain operations. The HTTP protocol binding schemas define two predicate languages, based on the BNF defined in the general model (see OGC 12-168, clause 6.2.4), which may be used. The two predicate languages are:

a) CQL_TEXT is a text encoding of the BNF.

b) FILTER is an XML encoding of the BNF grammar and is normatively defined in the Filter Encoding Implementation Specification, version 2.0 (see OGC 09-026r1).

Requirement-015

All CSW implementations that implement the XML filter encoding syntax as defined in the Filter Encoding Implementation Specification, version 2.0 (see OGC 09-026r1) shall support at least the following filter operators:

- logical operators:
  - And, Or, Not
- comparison operators:
  - PropertyIsEqualTo, PropertyIsNotEqualTo, PropertyIsLessThan, PropertyIsGreaterThan, PropertyIsLessThanOrEqualTo, PropertyIsGreaterThanOrEqualTo, PropertyIsLike, PropertyIsBetween
- spatial operators:
  - BBOX
- temporal operators:

  TOverlaps

6.5.5.2 XML encoding

The following XML schema fragments define how the predicate language may be XML encoded in CSW operations that allow constraints to be defined (Query, Update and Delete):

```
<xsd:element name="Constraint"
             type="csw30:QueryConstraintType" id="Constraint"/>
<xsd:complexType name="QueryConstraintType" id="QueryConstraintType">
   <xsd:choice>
      <xsd:element ref="fes:Filter"/>
      <xsd:element name="CqlText" type="xsd:string"/>
   </xsd:choice>
   <xsd:attribute name="version" type="xsd:string" use="required"/>
</xsd:complexType>
```

Requirement-016

The version parameter on the Constraint element shall be used to specify a version number indicating to which version of a specification the constraint conforms.

For example, in the XML encoding, if the fes:Filter element is being used, the version parameter could be set to "2.0.0" indicating that the filter conforms to version 2.0.0 of the OGC Filter Encoding 2.0 Encoding Standard (see OGC 09-026r1).

6.5.5.3 KVP encoding

Table 8 describes the KVP-encoding for query constraints. The parameters are grouped according to the kind of searching they enable as denoted in Table 8 by the column labeled "Query class".

- "Text search" enables full text queries based on a list of specified search terms.
- "Record search" allow one or more records to be retrieved from the catalogue based on the record identifier.
- "Spatial search" enables spatial searching based on a bounding box, a center-point-radius or using an arbitrary geometry.
- "Temporal search" supports temporal searching based on a time instant or period.

Spatial search parameters are further grouped according to the kinds of spatial searching they enable denoted in Table 8 by the column labeled "Query subclass".

- "Bounding box search" enable spatial searching using a bounding box with the implied spatial operator of *Intersects*.
- "Arbitrary geometry search" parameters enable spatial searching using an arbitrary geometry and an arbitrary spatial operator specified using the "relation" parameter.
- "Proximity search" enable spatial queries using a center-point and radius with the implied spatial operator of *Intersects*.

In general a catalogue query can include parameters from any query class specified in Table 8. If parameters from more than one class are specified in a request then the implied logical operator is "AND". For "Spatial search" parameters, only parameters from one subclass may be used in a single request. This means that a KVP-encoded catalogue query cannot, for example, include the *bbox* and *geometry* parameters.

Requirement-017

A server implementation shall advertise which of the KVP parameters specified in Table 8 it implements using the Filter Capabilities section (see 7.1.3) of the server's capabilities document.

Note: the parameters q, recordIds and bbox must be implemented to satisfy the Basic-Catalogue conformance class (see Clause 2 for the conformance class)

The server's capabilities document must advertise support for the KVP parameters defined in Table 8, as follows:

- Advertising support for any of the spatial operators – Equals, Disjoint, Touches, Within, Overlaps, Crosses, Intersects, Contains, DWithin and/or Beyond – in the server's Filter Capabilities section (see 7.1.3) implies that the server supports the *geometry*, *geometry_crs* and *relation* KVP parameters (see Table 8);

- Advertising support for the DWithin and Beyond operators in the server's Filter Capabilities section (see 7.1.3) further implies support for the *distance* and *distance_uom* KVP parameters (see Table 8);
- Support for the *lat*, *lon* and *radius* KVP parameters can be specified in the SpatialCapabilities subsection of the server's Filter Capabilities section (see 7.1.3) using the extension mechanism (i.e. *extension:operation_name*) defined in the filter specification (see OGC 09-026r1, clause 7.14); and
- Advertising support for any of the temporal operators – After, Before, Begins, BegunBy, TContains, During, EndedBy, Ends, TEquals, Meets, MetBy, TOverlaps, OverlappedBy and/or AnyInteracts – in the server's Filter Capabilities section implies support for the time and TRelation KVP parameters (see Table 8).

Table 8 — KVP encoding for query constraints

Query class	Query subclass	Keyword	OpenSearch Parameter(s)	Description	Data type and value
Text search[3]	N/A	q	searchTerms	A space separated list of search terms that are used to search all text fields in a catalogue record.	Character String
Record search	N/A	recordIds	Uid	Comma separated list of record identifiers to retrieve.	Character String
Spatial search	Bounding box search.	bbox	Box	Identifies a bounding box to be used as a spatial predicate	See clause 10.2, 06-121r9
	Arbitrary geometry search	geometry	Geometry	Identifies a geometry to be used as a spatial predicate.	WKT String (see OGC 06-103r4)
		geometry_crs	N/A	The CRS used to express the value of *geometry* parameter.	URI

Query class	Query subclass	Keyword	OpenSearch Parameter(s)	Description	Data type and value
		relation	Relation	The spatial operator to apply using the value of the *geometry* parameter.	Character String. One of: Equals Disjoint Touches Within Overlaps Crosses **Intersects** (default) Contains DWithin Beyond
		distance	N/A	If the value of relation is DWithin or Beyond, contains the distance value.	Number
		distance_uom	N/A	The units of measure used to express the value of the distance parameter	Character String. (see http://unitsofmeasure.org/ucum.html)
	Proximity search	lat	lat	Latitude expressed in WGS84.	Number
		lon	lon	Longitude expression in WGS84.	Number
		radius	radius	Search radius expressed in meters along the surface of the Earth.	Number
Temporal search	N/A	time	start/end	A time instance or time period	Character String: (see http://www.w3.org/TR/NOTE-datetime)

Query class	Query subclass	Keyword	OpenSearch Parameter(s)	Description	Data type and value
		TRelation	N/A	The temporal operator to apply using the value of the *time* parameter.	Character String. One of: After Before Begins BegunBy TContains During EndedBy Ends **TEquals**[1] Meets MetBy TOverlaps OverlappedBy **AnyInteracts**[2]

1. Default for a time instance
2. Default for a time period
3. Basic text searching using the "q" parameter is expected to be processed as follows:
    a. Match against the full text of at least the text fields in the catalogue record that are mapped to the dc:title, dc:description and dc:subject child elements (see OGC 10-032r8) of the csw:Record element
    b. Matching is case-insensitive
    c. In the case of multiple search terms, if any of the fields being tested in a record, as per (a), contains at least one of the specified search terms then that record shall appear in the result set
    d. Exact phrases can be delimited using quotations (i.e. 0x22 in ASCII)

Examples:

```
Text search
   …&q=AVHRR+"Meteosat 9"&…
   NOTES:
   - If at least the dc:title, dc:description or dc:subject text fields of a
     record contain the term AVHRR, the record shall appear in the result set
   - If at least the dc:title, dc:description or dc:subject text fields of a
     record contain the term "Meteosat 9", the record shall appear in the
     result set
   - If at least the dc:title, dc:description or dc:subject text fields of a
     record contain both the terms AVHRR and "Meteosat 9", the record shall
     appear in the result set

Spatial search using a bounding box
   …&bbox=43.6050,-79.4271,43.6915,-79.3162,urn:ogc:def:EPSG::4326&…

Spatial search using an arbitrary geometry and spatial operator
   …&geometry=POLYGON((43.6050,-79.4271,43.6050,-79.3162,43.6915,-
   79.3162,43.6915,-79.4271))&relation=Overlap&…
```

```
Spatial search using a centre point and radius
   …&lat=43.6050&lon=-79.4271&radius=10000&…

Temporal search using a time period
   …&time=2012-01-10/2012-12-31&…

Temporal search using a time period and a temporal operator
   …&time=2012-12-20/2012-12-21&trelation=During&…

Temporal search for a specific time instance
   …&time=2012-12-20&…

Query combining a text search and a spatial search using a bounding box
   …&q="Canadian National Exhibition"&bbox=43.6050,-79.4271,43.6915,-
   79.3162,urn:ogc:def:EPSG::4326&…

Opensearch (see 10-032r8) template fragment using the substitution variables
   start and end to denote the start and end of a time period
   …&time={start}/{end}&…
```

Table 9 describes additional KVP parameters that may be used to express complex filters encoded using CQL and OGC Filter.

Requirement-018

Servers that implement the CONSTRAINT, CONSTRAINTLANGAUGE and CONSTRAINTLANGUAGEVERSION parameters shall advertise this in their capabilities document by advertising support for the Filter-FES-KVP-Advanced conformance class (see Table 1)

Table 9 — KVP for advanced query parameters

Keyword	Description	Data type and value	Optionality
CONSTRAINTLANGUAGE	Identifies the predicate language used for the value of the Constraint	anyURI	Zero or one (Optional) Must be specified with the Constraint
CONSTRAINTLANGUAGEVERSION	Identifies the version of the predicate language used.	Character String	Zero or one There is no default as the parameter is specified if required to indicate which version of a specification the value of the constraint parameter conforms to.
Constraint	Text of query constraint in the predicate language identified by the CONSTRAINT LANGUAGE	Character String	Zero or one (Optional) Must be specified with the CONSTRAINT LANGUAGE

Servers may implement the parameters listed in Table 9 in order to support other, non-OGC query languages (e.g Apache Lucene). All supported non-OGC query langauges shall be advertised in the server's capabilities document using the ConstraintLanguages constraint (see Table 20). Support of OGC query languages (i.e. Filter-CQL, Filter-FES-XML and Filter-FES-KVP) shall be advertised in the server's capabilities document by declaring support for the corresponding conformance classes (see Table 1 & Table 20).

6.5.6 Enabling OpenSearch

6.5.6.1 Introduction

The OpenSearch standard defines a simple API that acts as a common façade on top of catalogue (and other) services that implement different query APIs thus requiring that client software only understand the OpenSearch API rather than having to understand each individual API implemented by each catalogue a client might want to access.

The component that allows OpenSearch clients to access servers that implement this standard is the OpenSearch description document. This clause describes the relationship between the GetRecords operation (see 7.3), the OpenSearch description document and the templates parameters defined in the OpenSearch (see http://www.opensearch.org) and OpenSearch GeoSpatial and Temporal Extensions (see OGC 10-032r8) standards.

Servers that want to enable OpenSearch clients must implement the Basic-Catalogue and OpenSearch conformance classes (see Table 1).

6.5.6.2 Description document autodiscovery

Interaction with an OpenSearch compatible server commences with the discovery, by an OpenSearch client, of an OpenSearch Description document. The OpenSearch standard defines several methods of autodiscovery of description documents (see http://www.opensearch.org/Specifications/OpenSearch/1.1#Autodiscovery)

This standard defines an operation constraint named "OpenSearchDescriptionDocument" (see Table 19) that enables autodiscovery of an OpenSearch description document from within an OGC capabilities documents. The following XML fragment is an example of the use of the OpenSearchDescriptionDocument constraint:

```
  <ows20:Operation name="GetRecords">
      <ows20:DCP>
         <ows20:HTTP>
            <ows20:Get xlink:href="http://www.cswserver.com/csw?"/>
            ...
         </ows20:HTTP>
      </ows20:DCP>
      ...
      <ows20:Constraint name="OpenSearchDescriptionDocument">
         <ows20:AllowedValues>

<ows20:Value>http://www.cswserver.com/descriptionDocument.xml</ows20:Value>
         </ows20:AllowedValues>
      </ows20:Constraint>
      ...
   </ows20:Operation>
```

This standard further defines a method for implicitly obtaining service metadata, including an OpenSearch description document, by accessing the base URL of a conforming catalogue with the HTTP GET methods (see 6.4).

6.5.6.3 Description document

The function of an OpenSearch Description document is similar to that of an OGC Capabilities document in that it provides metadata about a service. Specifically an OpenSearch Description document provides metadata about the service provider and, unlike an OGC capabilities document, provides parameterized templates describing the kinds of queries that a client can ask the service to perform (see OGC 10-032r8, clause 8).

Table 8 shows the mapping between GetRecords parameters used for expressing query predicates and the substitution variables defined in the OpenSearch standard (see http://www.opensearch.org) and in OGC's OpenSearch GeoSpatial and Temporal Extensions (see OGC 10-032r8). This mapping can be used to create OpenSearch description documents allowing OpenSearch clients to access catalogues that implement this standard.

The following example illustrates the how the two sets of parameters may be used in an OpenSearch description document:

```
<OpenSearchDescription xmlns="http://a9.com/-/spec/opensearch/1.1/"
   xmlns:geo="http://a9.com/-/opensearch/extensions/geo/1.0/" xmlns:time="
http://a9.com/-/opensearch/extensions/time/1.0/">
   <ShortName>CSW Web Search</ShortName>
   <Description>OpenSearch enabled CSW catalogue.</Description>
   <Tags>web CSW opengeospatial consortium OGC</Tags>
   <Contact>admin@cswservice.com</Contact>
<!-- Search by bbox returning csw:Record (i.e. GetRecordsResponse) -->
   <Url type="application/xml"
template="http://cswservice.com/csw?service=CSW&amp;version=3.0&amp;q={searchTe
rms}&amp;maxRecords={count?}&amp;startPosition={startIndex?}&amp;bbox={geo:box?
}&amp;outputSchema=http://www.opengis.net/cat/csw/3.0&outputFormat=application/
xml"/>
   <!-- Search by bbox returning ATOM -->
   <Url type="application/atom+xml"
template="http://cswservice.com/csw?service=CSW&amp;version=3.0&amp;q={searchTe
rms}&amp;maxRecords={count?}&amp;startPosition={startIndex?}&amp;bbox={geo:box?
}&amp;outputFormat=application/atom+xml"/>
   <!-- Search by bbox returning HTML -->
   <Url type="text/html"
template="http://cswservice.com/csw?service=CSW&amp;version=3.0&amp;q={searchTe
rms}&amp;maxRecords={count?}&amp;startPosition={startIndex?}&amp;bbox={geo:box?
},urn:ogc:def:EPSG::4326&amp;outputSchema=http://www.opengis.net/cat/csw/3.0&am
p;outputFormat=application/text/html"/>
   <!-- Search by bbox and time returning ATOM -->
   <Url type="application/atom+xml"
template="http://cswservice.com/csw?service=CSW&amp;version=3.0&amp;q={searchTe
rms}&amp;maxRecords={count?}&amp;startPosition={startIndex?}&amp;bbox={geo:box?
}&amp;time={time:start}/{time:end}&amp;outputFormat=application/atom+xml"/>
   <LongName>Opengeospatial Catalogue Service Web Search</LongName>
   <Query role="example"
          searchTerms="library" geo:box="43.46,-79.63,43.87,-79.17"/>
   <Developer>
   <Attribution>Copyright 2012, cswservice.com, Inc.</Attribution>
```

```
   <SyndicationRight>open</SyndicationRight>
</OpenSearchDescription>
```

Attention is drawn to the "Url" elements that use standard CSW GetRecords requests with appropriate OpenSearch substitution variables.

6.5.6.4 Response

The canonical response to an OpenSearch query is an XML-encoded ATOM document as described in clause 9.3 of OGC 10-032r8. Table 10 maps the CSW request/response parameters (see Table 23) to the OpenSearch response parameters.

Table 10 — Mapping CSW response parameter to OpenSearch response parameters

CSW response parameter	OpenSearch response parameter
numberOfRecordsMatched	totalResults
startPosition	startIndex
numberOfRecordsReturned	itemsPerPage

6.5.6.5 Requirements for an OpenSearch enabled CSW

This clause outlines the requirements for an OpenSearch enabled catalogue. An OpenSearch enabled catalogue is one that provides all the necessary metadata and parameter support to allow it to be accessed by an OpenSearch client.

Requirement-019

An OpenSearch enabled catalogue shall conform to the Basic-Catalogue conformance class (see Table 1).

Requirement-020

An OpenSearch enabled catalogue shall conform to the OpenSearch conformance class (see Table 1).

Requirement-021

An OpenSearch enabled catalogue shall be capable of providing an OpenSearch description document when the base URL of the service is accessed using the GET method (see 6.4) and the value of "Accept" HTTP header (see HTTP/1.1) shall be set to "application/opensearchdescription+xml"

Requirement-022

An OpenSearch enabled catalogue shall, in its OpenSearch description document, include at least one URL template with:

- the value of the type attribute set to “application/xml”
- the value of the CSW parameter outputFormat set to “application/xml”
- the value of the CSW parameter outputSchema set to “http://www.opengis.net/cat/csw/3.0 “

and shall include the OpenSearch parameters {startIndex?}, {count?}, {searchTerms?} and {geo:box?} in the URL template.

Requirement-023

An OpenSearch enabled catalogue shall, in its OpenSearch description document, include at least one URL template with

- the value of the type attribute set to “application/atom+xml”
- the value of the CSW parameter outputFormat set to “application/atom+xml”

the value of the CSW parameter outputSchema is not set and including the OpenSearch parameters {startIndex?}, {count?}, {searchTerms?} and {geo:box?} in the URL template.

6.6 Response encoding

6.6.1 Introduction

The message payload described in this subclause is an XML realization of the core metadata properties described in the general model (see OGC 12-168).

Requirement-024

All implementations of the HTTP protocol binding, and application profiles derived from the HTTP protocol binding, shall support the response schema described in this subclause and realized using the csw:AbstractRecord elements.

An application profile may specify other allowable payloads that constitute the body of a response message (if applicable).

Requirement-025

Entities shall conform to a registered Internet media type, but there is otherwise no restriction on the content; the actual payload is dependent upon the information model

supported by the profile.

Requirement-026

In all cases, elements of the underlying information model shall be mapped to the core metadata properties described in the general model (see OGC 12-168) which are concretely expressed in the HTTP protocol binding using the schema described in this subclause.

6.6.2 Abstract Record

An abstract element, called csw:AbstractRecord, is declared which is the head of a substitution group representing all the views (i.e. brief, summary and full) of the core metadata properties.

The following XML-Schema fragment declares the csw:AbstractRecord element:

```
<xsd:element name="AbstractRecord"
             type="csw:AbstractRecordType" abstract="true"/>
<xsd:complexType name="AbstractRecordType" abstract="true">
   <xsd:attribute name="deleted" type="xsd:boolean"
                  use="optional" default="false"/>
</xsd:complexType>
```

The attribute ”deleted“ defines if the metadata item was deleted (time of deletion can be expressed e.g. by including an element like dct:modified in the record.

6.6.3 Core queryable and returnable realization

6.6.3.1 Introduction

The common CSW record syntax is an XML-based encoding of Dublin Core metadata terms; it represents a concrete realization of the core metadata properties abstractly specified in Clause 6 of the general model (see OGC 12-168).

The full set of core properties is concretely materialized by the csw:Record element. Two addition elements, csw:BriefRecord and csw:SummaryRecord materialize the brief and summary views of the full set of core properties.

Table 11 maps the Dublin Core element names of the OGC query and response elements specified in general model (see OGC 12-168) to the concrete XML elements specified in this standard.

Table 11 — Mapping of Dublin Core names to XML element names

Dublin Core element name	OGC queryable[2] term (abstract)	XML element name[1] (to be used in query and response)	Datatype
title	Title	dc:title	character string
creator		dc:creator	character string
subject	Subject	dc:subject	character string
description	Abstract	dct:abstract	character string
publisher		dc:publisher	character string
contributor		dc:contributor	character string
date	Modified	dct:modified	date
type	Type	dc:type	character string, from a predefined set
format	Format	dc:format	character string
identifier	Identifier	dc:identifier	character string, an identifier
source	Source	dc:source	character string
language		dc:language	character string, from a predefined set (see IETF RFC 4646)
relation	Association	dc:relation	
rights		dc:rights	
coverage	BoundingBox	ows:BoundingBox	gml:Envelope
coverage	TemporalExtent	csw: TemporalExtent	gml:TimePeriod
	AnyText	csw:AnyText	character string

1. These XML elements are contained as children of the csw:Record element.
2. A blank value indicates that there is no corresponding OGC queryable term.

NOTE Other XML elements names may be substitutable for the XML elements listed in Table 11. The schema rec-dcterms.xsd contains a complete list of substitutable XML elements.

6.6.3.2 Full record

The XML encoding of the element representing the full catalogue record, csw:Record, is defined by the following XML-Schema fragment:

```
<xsd:element name="Record" type="csw30:RecordType"
             substitutionGroup="csw30:AbstractRecord"/>
<xsd:complexType name="RecordType" final="#all">
   <xsd:complexContent>
      <xsd:extension base="csw30:DCMIRecordType">
         <xsd:sequence>
            <xsd:element name="AnyText" type="csw30:EmptyType"
                         minOccurs="0" maxOccurs="unbounded"/>
            <xsd:element ref="ows:BoundingBox"
```

```
                        minOccurs="0" maxOccurs="unbounded"/>
                <xsd:element name="TemporalExtent" type="csw30:TemporalExtentType"
                        minOccurs="0" maxOccurs="unbounded"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>
```

The full record is based on the XML schemas developed for the Dublin Core Metadata Initiative with the addition of the csw:AnyText, ows:BoundingBox and csw:TemporalExtent elements.

Requirement-027

All elements contained in the csw:Record element that correspond to the abstract OGC queryable terms listed in Table 11 shall be available as queryables.

Requirement-028

In the event that a catalogue implementation does not have a value to map from its internal mdel to an OGC queryable term listed in Table 11 (and its XML realization as an element contained in csw:Record), its value shall be considered to be NULL for the purpose of querying.

Requirement-029

In the Dublin Core schema, the elements “identifier” and “title” (see Table 11) or any element that can substitute for them, are optional. However for the purposes of this specification, these elements shall be considered mandatory presentables. Thus the OGC abstract queryable terms “Identifier” and “Title” and their XML realization, dc:identifier and dc:title, are mandatory presentables.

Requirement-030

The csw:AnyText element shall only be available as a queryable, and is intended as a query target for a full text query of the catalogue's records. This element is not a presentable and shall never appear in a response message. Even though the content model for the csw:AnyText element is empty, the catalogue shall interpret its value to be the full text of all text fields in the catalogue record.

Requirement-031

The ows:BoundingBox element shall be used to express the spatial extent of a csw:Record record.

Requirement-032

The csw:TemporalExtent element shall be used to express the temporal extent of a csw:Record record.

It is anticipated that the number of optional presentables will vary from one record instance to another inside the catalogue; and so the remaining elements, or any element that can substitute for them, may appear as presentables if the catalogue has a corresponding value to present. Otherwise, elements for which no value is available may be omitted from the response.

Example csw30.Record:

```
<csw30:Record
   xmlns:csw30="http://www.opengis.net/cat/csw/3.0"
   xmlns:dc="http://purl.org/dc/elements/1.1/"
   xmlns:dct="http://purl.org/dc/terms/"
   xmlns:ows="http://www.opengis.net/ows/2.0"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.opengis.net/cat/csw/3.0
                       ../../../csw/3.0/record.xsd">
   <dc:creator>U.S. Geological Survey</dc:creator>
   <dc:contributor>State of Texas</dc:contributor>
   <dc:publisher>U.S. Geological Survey</dc:publisher>
   <dc:subject>Elevation, Hypsography, and Contours</dc:subject>
   <dc:subject>elevation</dc:subject>
   <dct:abstract>Elevation data collected for the National Elevation Dataset
(NED) based on 30m horizontal and 15m vertical accuracy.</dct:abstract>
   <dc:identifier>ac522ef2-89a6-11db-91b1-7eea55d89593</dc:identifier>
   <dc:relation>OfferedBy</dc:relation>
   <dc:source>dd1b2ce7-0722-4642-8cd4-6f885f132777</dc:source>
   <dc:rights>Copyright (c) 2004, State of Texas</dc:rights>
   <dc:type>Service</dc:type>
   <dc:title>National Elevation Mapping Service for Texas</dc:title>
   <dct:modified>2004-03-01</dct:modified>
   <dc:language>en</dc:language>
   <ows:BoundingBox>
      <ows:LowerCorner>-108.44 28.229</ows:LowerCorner>
      <ows:UpperCorner>-96.223 34.353</ows:UpperCorner>
   </ows:BoundingBox>
   <csw30:TemporalExtent>
   <csw30:begin>2001-12-01T09:30:47Z</csw30:begin>
   <csw30:end>2001-12-17T09:30:47Z</csw30:end>
   </csw30:TemporalExtent>
</csw30:Record>
```

The following example query shows the usage of the csw30:Record elements as queryables within a filter statement (default namespace here is http://www.opengis.net/cat/csw/3.0):

```
<fes:Filter>
   <fes:And>
      <fes:PropertyIsLike escapeChar="\" singleChar="?" wildCard="*">
         <fes:ValueReference>dc:title</fes:ValueReference>
```

```
            <fes:Literal>*Elevation*</fes:Literal>
         </fes:PropertyIsLike>
         <fes:PropertyIsEqualTo>
            <fes:ValueReference>dc:type</fes:ValueReference>
            <fes:Literal>Service</fes:Literal>
         </fes:PropertyIsEqualTo>
         <fes:PropertyIsGreaterThanOrEqualTo>
            <fes:ValueReference>dct:modified</fes:ValueReference>
            <fes:Literal>2012-06-01</fes:Literal>
         </fes:PropertyIsGreaterThanOrEqualTo>
         <fes:Intersects>
            <fes:ValueReference>ows:BoundingBox</fes:ValueReference>
            <gml:Envelope>
               <gml:lowerCorner>14.05 46.46</gml:lowerCorner>
               <gml:upperCorner>17.24 48.42</gml:upperCorner>
            </gml:Envelope>
         </fes:Intersects>
         <fes:BBOX>
            <fes:ValueReference>ows:BoundingBox</fes:ValueReference>
            <gml:Envelope>
               <gml:lowerCorner>14.05 46.46</gml:lowerCorner>
               <gml:upperCorner>17.24 48.42</gml:upperCorner>
            </gml:Envelope>
         </fes:BBOX>
         <fes:TOverlaps>
            <fes:ValueReference>csw:TemporalExtent</fes:ValueReference>
            <gml:TimePeriod gml:id="TP1">
               <gml:begin>
                  <gml:TimeInstant gml:id="TI1">
                  <gml:timePosition>2012-05-17T08:00:00Z</gml:timePosition>
                  </gml:TimeInstant>
               </gml:begin>
               <gml:end>
                  <gml:TimeInstant gml:id="TI2">
                  <gml:timePosition>2012-05-23T11:00:00Z</gml:timePosition>
                  </gml:TimeInstant>
               </gml:end>
            </gml:TimePeriod>
         </fes:TOverlaps>
      </fes:And>
   </fes:Filter>
```

6.6.3.3 Summary record

The XML encoding of the element representing the summary view, csw:SummaryRecord, is defined by the following XML-Schema fragment:

```
<xsd:element name="SummaryRecord" type="csw30:SummaryRecordType"
             substitutionGroup="csw30:AbstractRecord"/>
<xsd:complexType name="SummaryRecordType" final="#all">
   <xsd:complexContent>
   <xsd:extension base="csw30:AbstractRecordType">
      <xsd:sequence>
         <xsd:element ref="dc:identifier" maxOccurs="unbounded"/>
         <xsd:element ref="dc:title" maxOccurs="unbounded"/>
         <xsd:element ref="dc:type" minOccurs="0"/>
         <xsd:element ref="dc:subject" minOccurs="0" maxOccurs="unbounded"/>
         <xsd:element ref="dc:format" minOccurs="0" maxOccurs="unbounded"/>
         <xsd:element ref="dc:relation" minOccurs="0" maxOccurs="unbounded"/>
         <xsd:element ref="dct:modified" minOccurs="0" maxOccurs="unbounded"/>
         <xsd:element ref="dct:abstract" minOccurs="0" maxOccurs="unbounded"/>
```

```
            <xsd:element ref="dct:spatial" minOccurs="0" maxOccurs="unbounded"/>
            <xsd:element ref="ows:BoundingBox"
                         minOccurs="0" maxOccurs="unbounded"/>
            <xsd:element name="TemporalExtent" type="csw30:TemporalExtentType"
                         minOccurs="0" maxOccurs="unbounded"/>
         </xsd:sequence>
      </xsd:extension>
      </xsd:complexContent>
</xsd:complexType>
```

Requirement-033

For a csw:SummaryRecord, the dc:identifier and dc:title elements, or any element that can be substituted for them, are mandatory presentables and shall always appear in the response.

The remaining elements, or any element that can substitute for them, can appear as a presentable if the catalogue has a corresponding value to present.

EXAMPLE csw30.SummaryRecord:

```
<SummaryRecord
   xmlns:dc="http://purl.org/dc/elements/1.1/"
   xmlns:dct="http://purl.org/dc/terms/"
   xmlns="http://www.opengis.net/cat/csw/3.0"
   xmlns:csw="http://www.opengis.net/cat/csw/3.0"
   xmlns:ows="http://www.opengis.net/ows/2.0"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.opengis.net/cat/csw/3.0
                       ../../../csw/3.0/cswAll.xsd">
   <dc:identifier>00180e67-b7cf-40a3-861d-b3a09337b195</dc:identifier>
   <dc:title>Image2000 Product 1 (at1) Multispectral</dc:title>
   <dc:type>dataset</dc:type>
   <dc:subject>imagery</dc:subject>
   <dc:subject>baseMaps</dc:subject>
   <dc:subject>earthCover</dc:subject>
   <dc:format>BIL</dc:format>
   <dct:modified>2012-10-04 00:00:00</dct:modified>
   <dct:abstract>IMAGE2000 product 1 individual orthorectified scenes.
IMAGE2000 was  produced from ETM+ Landsat 7 satellite data and provides a
consistent European coverage of individual orthorectified scenes in national
map projection systems.</dct:abstract>
   <ows:BoundingBox>
      <ows:LowerCorner>-108.44 28.229</ows:LowerCorner>
      <ows:UpperCorner>-96.223 34.353</ows:UpperCorner>
   </ows:BoundingBox>
   <TemporalExtent>
      <begin>2001-12-01T09:30:47Z</begin>
      <end>2001-12-17T09:30:47Z</end>
   </TemporalExtent>
</SummaryRecord>
```

6.6.3.4 Brief record

The XML encoding of the element representing the brief view, csw:BriefRecord, is defined by the following XML-Schema fragment:

```
<xsd:element name="BriefRecord" type="csw30:BriefRecordType"
             substitutionGroup="csw30:AbstractRecord"/>
<xsd:complexType name="BriefRecordType" final="#all">
   <xsd:complexContent>
      <xsd:extension base="csw30:AbstractRecordType">
         <xsd:sequence>
            <xsd:element ref="dc:identifier" maxOccurs="unbounded"/>
            <xsd:element ref="dc:title" maxOccurs="unbounded"/>
            <xsd:element ref="dc:type" minOccurs="0"/>
            <xsd:element ref="ows:BoundingBox"
                         minOccurs="0" maxOccurs="unbounded"/>
         </xsd:sequence>
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>
```

Requirement-034

For the csw:BriefRecord, the dc:identifier and dc:title elements, or any element that can be substituted for them, are mandatory presentables and shall always appear in the response.

The remaining elements, or any element that can be substituted for them, can appear as a presentable if the catalogue has a corresponding value to present.

Example csw30.BriefRecord:

```
<csw30:BriefRecord
   xmlns:csw30="http://www.opengis.net/cat/csw/3.0"
   xmlns:dc="http://purl.org/dc/elements/1.1/"
   xmlns:dct="http://purl.org/dc/terms/"
   xmlns:ows="http://www.opengis.net/ows/2.0"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.opengis.net/cat/csw/3.0
                       ../../../csw/3.0/record.xsd">
   <dc:identifier>00180e67-b7cf-40a3-861d-b3a09337b195</dc:identifier>
   <dc:title>Image2000 Product 1 (at1) Multispectral</dc:title>
   <dc:type>dataset</dc:type>
</csw30:BriefRecord>
```

6.7 Exception encoding

6.7.1 XML encoding

Requirement-035

In the event that a catalogue service encounters an error while processing a request or receives an unrecognised request, it shall generate an XML document indicating that an error has occurred.

Requirement-036

The format of the XML error response is specified by, and shall validate against, the

exception response schema defined in clause 8 of the OWS Common Implementation Specification (see OGC 06-121r9).

Requirement-037

Servers shall implement the generic exception codes in Table 27 of OGC 06-121r9 for all conformance classes defined in this International Standard.

Requirement-038

In addition to the exception codes defined in OGC 06-121r9, server that implement this standard shall also implement the exceptions defined in Table 12.

An ows:ExceptionReport element may contain one or more catalogue processing exceptions specified using the ows:Exception element. The mandatory version attribute is used to indicate the version of the service exception report schema. For this version of the specification, this value is fixed at 3.0.0.

Individual exception messages are contained within the ows:ExceptionText element.

Requirement-039

The mandatory exceptionCode attribute shall be used to associate an exception code with the accompanying message.

The optional **locator** attribute may be used to indicate where an exception was encountered in the request that generated the error. Table 12 indicates what value the locator parameter should have for each exception code.

Table 12 — CSW exception codes

exceptionCode values	Meaning of code	"locator" value	Conformance class
InvalidValue	A Transaction (see 7.6) has attempted to insert or change the value of a data component in a way that violated the schema of the information model.	The "locator" parameter contains the name of the information model component being incorrectly modified.	Transaction
OperationParsingFailed	The request is badly formed and failed to be parsed by the server.	The "locator" parameter contains the value of the "handle" or "requestId" parameter if one is available. Otherwise the server may use	All (see Table 1)

		some other means to locate the error (e.g Operation name)	
OperationProcessingFailed	An error was encountered while processing the operation.	The "locator" parameter contains the value of the "handle" parameter if one is available. Otherwise the server may use some other means to locate the error (e.g. Operation name)	All (see Table 1)

Multiple exceptions may be reported in a single exception report so server implementations should endeavour to report as many exceptions as necessary to clearly describe a problem.

EXAMPLE If parsing an operation failed because the value of a parameter is invalid, the server should report an OperationParsingFailed exception and an InvalidParameterValue exception.

A number of elements defined in this document include a handle attribute and/or a requestId attribute that may be used to associate a mnemonic name with the element. If such a handle or requestId attribute exists, its value may be reported using the locator attribute of the ows:ExceptionText element in order to correlate the element with the exception text. If the handle or requestId attribute is not specified, then a catalogue is not required to specify a value for the locator attribute or may, optionally, attempt to locate the error using other means such as line numbers.

EXAMPLE The following is an example of an exception message

```
<ExceptionReport
   version="3.0.0"
   xmlns="http://www.opengis.net/ows/2.0"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.opengis.net/ows/2.0
                       http://schemas.opengis.net/ows/2.0/owsAll.xsd">
   <Exception exceptionCode="ParsingError" locator="INSERT STMT 01">
      <ExceptionText>parse error: missing closing tag for
element</ExceptionText>
   </Exception>
</ExceptionReport>
```

NOTE This example uses the exceptionCode value "ParsingError", which has not yet been specified, meaning that an error was detected in validating a XML encoded operation request.

6.7.2 HTTP status codes

Requirement-040

Servers shall set the HTTP status code (see IETF RFC 2616:1999, 6.1.1) in their responses.

Requirement-041

Upon successful processing of a CSW operation, the server shall set the HTTP status code to "200" with the response phrase being set to "OK". The body of the response shall be encoded as defined in this international standard for the corresponding CSW operation.

Requirement-042

A server that generates an exception, in response to a CSW request, shall generate a response body as described in clause 6.7 and shall include an appropriate HTTP status code as defined in Table 13.

Table 13 correlates OWS (see OGC 10-121r9) and CSW (see Table 12) exception codes with HTTP status codes (see IETF RFC 2616:1999, 6.1.1)

Table 13 –- Correlation between OWS and CSW exception codes and HTTP status codes

OGC exception code	HTTP status code	HTTP reason phrase1
InvalidValue	400	Invalid property value
OperationParsingFailed	400	Bad request
OperationProcessingFailed	403	Server processing failed
OWS Common exception codes		
OperationNotSupported	400	Not implemented
MissingParameterValue	400	Bad request
InvalidParameterValue	400	Bad request
VersionNegotiationFailed	400	Bad request
InvalidUpdateSequence	400	Bad request
OptionNotSupported	400	Not implemented
NoApplicableCode	400	Internal server error
The HTTP reason phrases are informational only (as per the HTTP specification); that is, a client should NOT be expected to parse and understand the reason phrases.		

7 Operations overview

Figure 2 shows the request/response message pairs for all the operations defined for the web catalogue service (CSW). There are three classes of operations: *service operations*

which are operations a client may use to interrogate the service to determine its capabilities; *discovery operations* which a client may use to determine the information model of the catalogue and query catalogue records; and *management operations* which are used to create or change records in the catalogue.

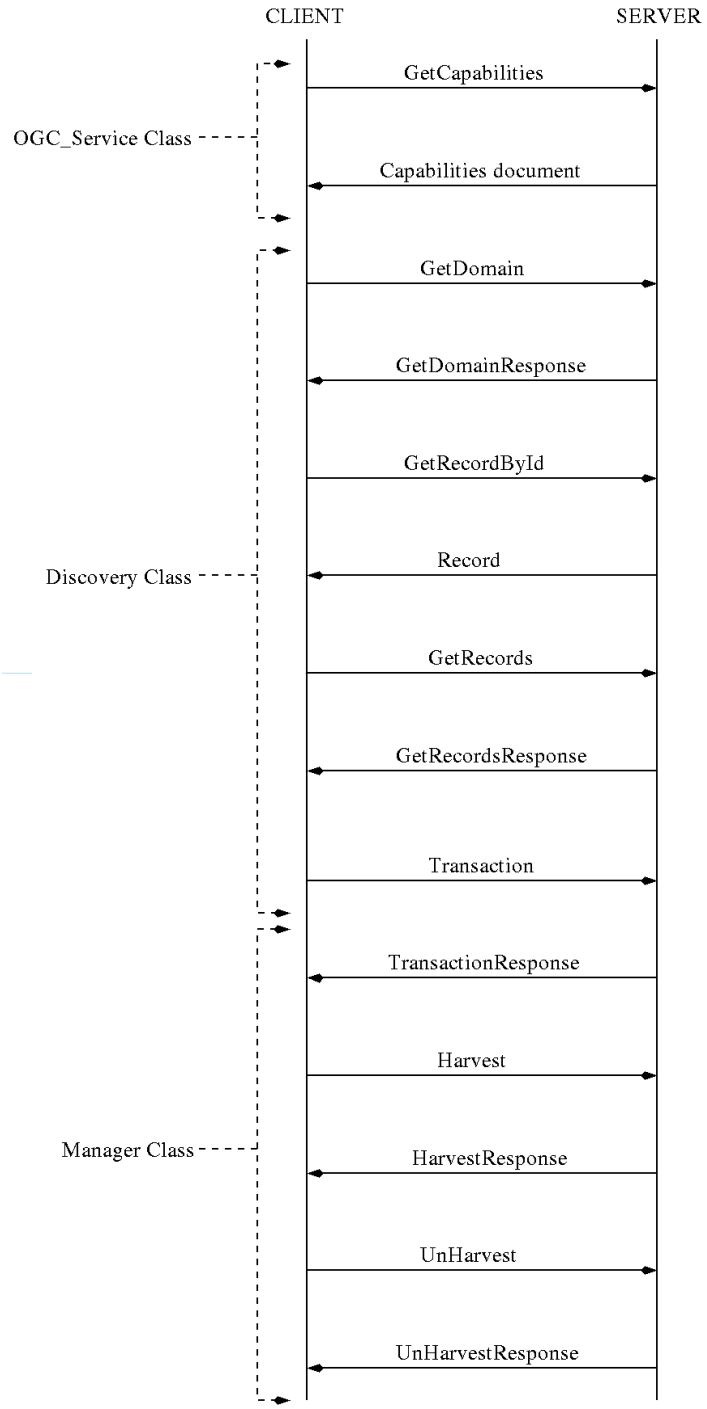

Figure 2 — Protocol sequence diagram

Figure 3 depicts a conceptual architecture to illustrate the relationship of these interfaces to service consumers and producers. The arrows represent the CSW requests that producers and consumers of metadata may generate. For example, to create metadata, a

metadata producer may invoke the **Transaction** request or the **Harvest** request. Similarly, a consumer of metadata may invoke the **GetRecords** request to query the catalogue.

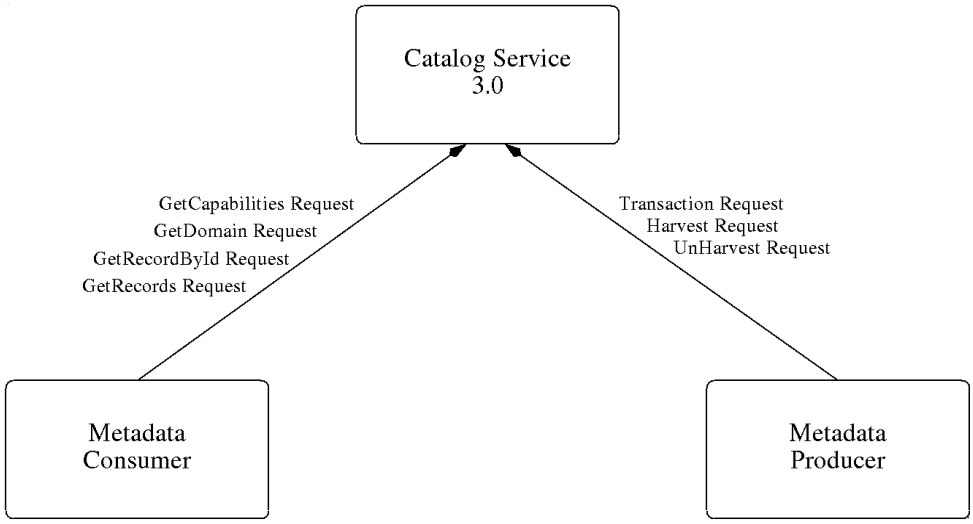

Figure 3 — Conceptual architecture

7.1 GetCapabilities operation

7.1.1 Introduction

The mandatory GetCapabilities operation allows CSW clients to retrieve service metadata from a server. The response to a GetCapabilities request is an XML document containing service metadata about the server.

7.1.2 Operation request

The GetCapabilities request is defined in Subclause 7 of the *OGC Web Services Common Specification 2.0* (see OGC 06-121r9).

Requirement-043

All CSW servers shall implement the HTTP GET transfer using the KVP-encoding of the GetCapabilities operation.

Requirement-174

Servers that support the GetCapabilities-XML class shall implement the HTTP POST transfer using XML-encoding of the GetCapabilities operation.

In addition to the common service metadata elements defined in Subclause 7.4 of the *OGC Web Service Common Specification 2.0* (see OGC 06-121r9), the GetCapabilities request may be used to request the sections listed in Table 14.

Table 14 — Additional section name values and meaning

Section name	Meaning
Filter_Capabilities	A Filter_Capabilities section shall be included in the service metadata to describe which elements of the predicate language are supported.

An application profile may introduce additional service information items as needed by extending the csw:CapabilitiesType definition.

7.1.3 Operation response

The response to a GetCapabilities request is an XML document. Table 15 lists the sections that may appear in the response to a GetCapabilities request.

Table 15 — Section names and contents

Section name	Contents
ServiceIdentification	Metadata about a specified CSW implementation. The contents and schema of this section shall be as specified in Subclause 7.4.4 and owsServiceIdentification.xsd of OGC 06-121r9.
ServiceProvider	Metadata about the organization offering the CSW service. The contents and schema of this section shall be as specified in Subclause 7.4.5 and owsServiceProvider.xsd of OGC 06-121r9.
OperationsMetadata	Metadata about the CSW operations offered by a specific CSW implementation, including the URLs for operation requests. The contents and schema of this section shall be as specified in Subclauses 7.4.6 and owsOperationsMetadata.xsd of OGC 06-121r9. The specific operations that may be listed in the OperationsMetadata section are specified in Subclause 7.1.4 of this document.
Filter_Capabilities	Metadata about the filter capabilities of the server if the server implements the Filter predicate encoding as defined in OGC 09-026r1.

Requirement-044

The capabilities document of a CSW service shall, depending on the value of Sections parameter of the GetCapabilities request, contain one or more of the ServiceIdentification, ServiceProvider, OperationsMetadata or Filter_Capabilities sections.

Requirement-045

If the Sections parameter of the GetCapabilities request is not specified, then all sections of a capabilities document listed in Table 15 shall be presented in the response.

7.1.4 OperationsMetadata section standard contents

Requirement-046

The OperationsMetadata section of the capabilities document shall list all operations implemented by a CSW service, as described in Subclause 7.4.6 of OGC 06-121r9.

Table 16 and Table 17 list the set of operations that may appear in the OperationsMetadata section of a CSW capabilities document. In both tables, the "Attribute name" column uses XPath notation to identify the attribute whose value is being specified. The "Attribute value**"** column indicates an allowable value for the specified attribute name, and the last column indicates what the value means.

An application profile may restrict the ExtendedCapabilities element to provide additional computational metadata (e.g., WSDL service descriptions, OWL-S resource definitions) compliant with this specification.

Table 16 lists the minimum set of operations that must appear in the OperationsMetadata section of a CSW capabilities document.

Table 16 — Required values of the OperationsMetadata section attributes

Attribute name	Attribute value	Meaning of attribute value
OperationsMetadata/Operation/@name	GetCapabilities	The GetCapabilities operation is implemented by this server.
OperationsMetadata/Operation/@name	GetRecordById	The GetRecordById operation is implemented by this server.
OperationsMetadata/Operation/@name	GetRecords	The GetRecords operation is implemented by this server.

Table 17 lists the remaining CSW operations that may be listed in the OperationsMetadata sections of a capabilities document.

Table 17 — Optional values of the OperationsMetadata section attributes

Attribute name	Attribute value	Meaning of attribute value
OperationsMetadata/Operation/@name	GetDomain	The GetDomain operation is implemented by this server.
OperationsMetadata/Operation/@name	Harvest	The Harvest operation is implemented by this server.
OperationsMetadata/Operation/@name	UnHarvest	The UnHarvest operation is implemented by this server.
OperationsMetadata.Operation.name	Transaction	The Transaction operation is implemented by this server.

As described in clause 7.4.6 of OGC 06-121r9, the OperationsMetadata section allows a server to advertise endpoints for each HTTP method that may be used to invoke an operation. However, because this standard allows servers to support both a plain XML

encoding and a SOAP encoding for operations, an issue arises concerning how a server should advertise the POST endpoint for plain XML and/or SOAP encoded requests. This is accomplished using the PostEncoding constraint (see 7.1.5.2) and the fact that the multiplicity of the ows:Post element is unbounded. If a server supports both XML and SOAP encoding then it may specify multiple ows:Post elements each containing a PostEncoding constraint describing which XML encoding is supported at the corresponding endpoint. The endpoint URLs may be different or the same. If they are the same this simply means that the server can detect, from the input stream, whether the request is plain XML-encoded or SOAP-encoded. The following XML fragment from a sample capabilities document illustrates how both SOAP and plain XML encodings can be advertised:

```
<ows20:Operation name="GetRecords">
   <ows20:DCP>
      <ows20:HTTP>
         <ows20:Post xlink:href="http://www.someserver.example.com/postEndpoint">
            <ows20:Constraint name="PostEncoding">
               <ows20:AllowedValues>
                  <ows20:Value>SOAP</ows20:Value>
               </ows20:AllowedValues>
            </ows20:Constraint>
         </ows20:Post>
         <ows20:Post xlink:href="http://www.someserver.example.com/postEndpoint">
            <ows20:Constraint name="PostEncoding">
               <ows20:AllowedValues>
                  <ows20:Value>XML</ows20:Value>
               </ows20:AllowedValues>
            </ows20:Constraint>
         </ows20:Post>
         ...
      </ows20:HTTP>
      ...
   </ows20:DCP>
   ...
</ows20:Operation>
```

In this example, the POST encoding endpoint URL is the same in both cases indicating that the server can detect the request encoding from the stream.

7.1.5 Parameter and Constraint element

The Parameter and Constraint elements, defined in Subclause 7.4.6 of OGC 06-121r9, allow additional metadata to be specified for each operation or for the service in its entirety.

The Parameter element is typically used to convey the value domain of a parameter. For example, the domain of the exceptionCode parameter could record all the codes implemented for each operation by a server. Similarly, each of the GetCapabilities operation request parameters might have its domain recorded. For example, the domain of the Sections parameter could record all the sections implemented by a server.

The Constraint element is typically used to convey some capacity limit that the server enforces. For example the Constraint element could be used to convey the default values of the maxRecords query parameter (see OGC 06-121r9, subclause 7.4.6).

Besides the Parameter and Constraint elements defined in this standard, servers may include additional – perhaps vendor specific -- Parameter and Constraint element in the capabilities document. The meaning of such Parameter and Constraint elements is not defined in this standard.

Requirement-047

Any additional Parameter and Constraint elements, not defined in this standard, that a server introduces in its capabilities shall be safely ignored by clients which are solely targeted at this specification.

7.1.5.1 Parameter domains for CSW operations

Table 18 lists the set of parameter domains that may be specified in the OperationsMetadata section of a CSW capabilities document for CSW operations.

Table 18 — Parameter domains for CSW operations

Operation Name	Parameter Name	Expected Value Type	Description/Possible Values
All operations (except GetCapabilities)	Version	string	Include the value “3.0.0”. May include the values “2.0.2” or other vendor specific version number.
GetCapabilities	AcceptVersions	string	Include the value “3.0.0”. May include the values “2.0.2” or other vendor specific version number.
GetCapabilities	AcceptFormats	MIME type	Includes the value "text/xml". May include other MIME types such as "text/html" or "text/plain" or any other vendor supported MIME type the server is capable of generating.
GetCapabilities	Sections	string	Zero or more of the following list of values: "ServiceIdentification", "ServiceProvider", "OperationsMetadata", "Filter_Capabilities"
GetRecords	typeName	Qualified xml element	Includes the value csw:Record. May include other qualified xml elements that the server of a CSW profile supports (e.g. gmd:MD_Metadata).
GetRecords GetRecordById	outputFormat	string or MIME type	Includes the value application/xml (or “application/soap+xml” for SOAP). May include any other

			string MIME type that the server supports.
GetRecords GetRecordById	outputSchema	URI	
GetRecords GetRecordById	ElementSetName	string	Includes the values “brief”, “summary” and “full”. May include any other value that the server recognizes as a named set of information model elements.
GetRecords GetRecordById	geometry_crs	URI	A list of coordinate references system identifiers that the server supports.[1,2]
Harvest	ResourceType	string	Lists the token identifying the resources types that the server is able to harvest (see Table 25).
Harvest	ResourceFormat	string	Includes the value “application/xml”. May include any other values indicating the resource formats that the server supports.
UnHarvest	outputFormat	string or MIME type	Indicates the format of the response. Includes the value “text/xml”. May include any other response format that the server supports (e.g. text/html).
NOTES: 1. For servers that support it, the value domain for this parameter may also be optionally obtained using the GetDomain operation (see 7.2). 2. The value domain for this parameter also applies to the BBOX parameter. The CRS for the BBOX parameter is not referencable as a distinct parameter name because it is encoded after the coordinate values string (see OGC 06-121r9, clause 10.2).			

7.1.5.2 Constraints for CSW operations

Table 19 lists the set of parameter domains that may be specified in the OperationsMetadata section of a CSW capabilities document for CSW operations.

Table 19 — Operation constraints

Operation Name	Constraint Name	Expected value type	Description
GetRecords	MaxRecordDefault	Integer value	Specifies the default value of the maxRecords parameter.
GetRecords	SortLevelLimit	Integer value	Defines the maximum number of properties that may be simultaneously sorted. In the event that a request contains too many properties, the server shall respond with an exception as specified in clause 6.7. If the constraint is not specified

			then there is no limit to the number of sort properties that may be specified.
GetRecords	FederatedCatalogues	anyURI	One or more base URL's of federated catalogues.
GetRecords	OpenSearchDescriptionDocument	anyURI	Provides a URI reference to an open search description document that the service offers.
Transaction	TransactionSchemas	anyURI	A list of URI's identifying that schemas upon which the Transaction operation may be applied. For XML encoded information model the URI's should be the namespace URI of each supported schema.
Harvest	HarvestHandlesAttachements	Boolean	A value of "true" indicated that the Harvest operation can handle multi-part MIME document attachments. A value of "false" indicates that attachments are not supported.
All operations	PostEncoding	String value; one or more of "SOAP", "XML", "KVP"	Indicates the request encoding that can be accepted via the HTTP POST method. The scope of this constraint depends on where it is encoded within the Operation element. If no PostEncoding constraint is specified then XML shall be assumed unless there is a service-level PostEncoding constraint specified that overrides this default (see Table 20).

7.1.5.3 Service constraints

Table 20 lists the set of service-level constraints that may be specified in the OperationsMetadata section of a CSW capabilities document.

Requirement-048

All the service constraints listed in Table 20 – that correspond to conformance classes defined in Table 1 -- shall be specified in the capabilities document of a server with the appropriate value set to indicate whether the server conforms to the corresponding class or not.

Table 20 — Service constraints

Constraint Name	Possible values and/or value types	Description
OpenSearch	boolean value; either "TRUE" or "FALSE".	Indicates that the server implements the OpenSearch conformance class.
GetCapabilities-XML	boolean value; either "TRUE" or "FALSE".	Indicates that the server implements the GetCapabilities-XML conformance class.
GetRecordById-XML	boolean value; either "TRUE" or "FALSE".	Indicates that the server implements the GetRecordsById-XML conformance class.
GetRecords-Basic-XML	boolean value; either "TRUE" or "FALSE".	Indicates that the server implements the GetRecords-Basic-XML conformance class.
GetRecords-Distributed-XML	boolean value; either "TRUE" or "FALSE".	Indicates that the server implements the GetRecords-Destributed-XML conformance class.
GetRecords-Distributed-KVP	boolean value; either "TRUE" or "FALSE".	Indicates that the server implements the GetRecords-Distributed-XML conformance class.
GetRecords-Async-XML	boolean value; either "TRUE" or "FALSE".	Indicates that the server implements the GetRecords-Async-XML conformance class.
GetRecords-Async-KVP	boolean value; either "TRUE" or "FALSE".	Indicates that the server implements the GetRecords-Async-KVP conformance class.
GetDomain-XML	boolean value; either "TRUE" or "FALSE".	Indicates that the server implements the GetDomain-XML conformance class.
GetDomain-KVP	boolean value; either "TRUE" or "FALSE".	Indicates that the server implements the GetDomain-KVP conformance class.
Transaction	boolean value; either "TRUE" or "FALSE".	Indicates that the server implements the Transaction conformance class.
Harvest-Basic-XML	boolean value; either "TRUE" or "FALSE".	Indicates that the server implements the Harvest-Basic-XML conformance class.
Harvest-Basic-KVP	boolean value; either "TRUE" or "FALSE".	Indicates that the server implements the Harvest-Basic-KVP conformance class.
Harvest-Async-XML	boolean value; either "TRUE" or "FALSE".	Indicates that the server implements the Harvest-Async-XML conformance class.
Harvest-Async-KVP	boolean value; either "TRUE" or "FALSE".	Indicates that the server implements the Harvest-Async-KVP conformance class.
Harvest-Periodic-XML	boolean value; either "TRUE" or "FALSE".	Indicates that the server implements the Harvest-Periodic-XML conformance class.
Harvest-Periodic-KVP	boolean value; either "TRUE" or "FALSE".	Indicates that the server implements the Harvest-Periodic-KVP conformance class.
Filter-CQL	boolean value; either "TRUE" or "FALSE".	Indicates that the server implements the Filter-CQL conformance class.
Filter-FES-XML	boolean value; either "TRUE" or "FALSE".	Indicates that the server implements the Filter-FES-XML conformance class.
Filter-FES-KVP-Advanced	boolean value; either "TRUE" or "FALSE"	Indicates that the server implements the Filter-FES-KVP-Advanced conformance class
SupportedGMLVersions	character string	List of GML versions that the server supports.

PostEncoding	character string; one or more of “SOAP”, “XML”, “KVP”	Indicates, at the service-level, the request encodings that can be accepted via the HTTP POST method unless otherwise indicated by a locally scoped constraint (See Table 19).
DefaultSortingAlgorithm	anyURI	A reference to a description of the default sort which is the sort that the server applies if no SortBy clause (see 7.3.4.11) is specified in a request. Absence of this constraint implies that the default sort is alphabetical by Title (see 6.6.3) in ascending order.
ConstraintLanguages	anyURI	The list of supported values for the CONSTRAINTLANGUAGE parameter (see Table 9).
CoreQueryables	character string; one or more of the OGC quryable terms (see Table 11, column 2)	The list of OGC queryable terms (see Table 11) which the server can map to its internal model for the purpose of query. Absense of this constraint implies that the server can map all the OGC core queryable terms to its internal model.
CoreSortables	character string	A list of core properties (see 6.6.3) that the server can accept in a SortBy clause (see 7.3.4.11). Absence of this constraint implies that the server can sort by any of the core properties.

EXAMPLE The following XML fragment shows how the SupportedGMLVersions constraint can be used to list the namespaces of the supported GML versions.

```
<ows20:Constraint name="SupportedGMLVersions">
   <ows20:AllowedValues>
      <ows20:Value>http://www.opengis.net/gml/3.2</ows20:Value>
      <ows20:Value>http://www.opengis.net/gml</ows20:Value>
   </ows20:AllowedValues>
</ows20:Constraint>
```

EXAMPLE The following XML fragment shows how the ConstraintLanguages constraint can be used to list the supported constraint languages.

```
<ows20:Constraint name="ConstraintLanguages">
   <ows20:AllowedValues>
      <ows20:Value>http://www.opengis.net/fes/2.0</ows20:Value>
      <ows20:Value> http://lucene.apache.org/</ows20:Value>
   </ows20:AllowedValues>
</ows20:Constraint>
```

In this example the server supports OGC’s Filter (see OGC 09-926r2) and the Apache Lucene query language. Thus the following query could be presented to the server:

```
…&CONSTRAINT_LANGUAGE=http://lucene.apache.org&CONSTRAINT=<Apache Lucene
query>&…
```

7.1.6 Examples[1]

KVP Encoding:

http://www.someserver.com/csw30.cgi?REQUEST=GetCapabilities&SERVICE=CSW&ACCEPTVERSION=3.0.0,2.0.2&outputFormat=application/xml

XML Encoding:

```
<GetCapabilities
   service="CSW"
   xmlns="http://www.opengis.net/cat/csw/3.0"
   xmlns:ows="http://www.opengis.net/ows/2.0"
   xmlns:csw30="http://www.opengis.net/cat/csw/3.0"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.opengis.net/cat/csw/3.0
                       ../../../csw/3.0/CSW30-discovery.xsd">
   <ows:AcceptVersions>
      <ows:Version>2.0.2</ows:Version>
      <ows:Version>3.0.0</ows:Version>
   </ows:AcceptVersions>
   <ows:AcceptFormats>
      <ows:OutputFormat>application/xml</ows:OutputFormat>
   </ows:AcceptFormats>
</GetCapabilities>
```

Annex B includes a sample capabilities document for a CSW server that supports only the default message payload.

7.2 GetDomain operation

7.2.1 Introduction

The optional GetDomain operation can be used to obtain information about the values of components of the information model(s) that a catalogue offers or about parameters of the API defined in this standard. The information returned by the GetDomain operation can represent the set of possible values or the set of available values for the specified component. The set of possible values represents all the values that the component can have. The set of available values represents the subset of possible values that the component actually has at the time the GetDomain operation is invoked.

For example, an information model component or request parameter defined as a 16bit positive integer has 65535 possible values but the number of distinct values actually stored in the catalogue may be much smaller.

This type of runtime information about the values of an information model component or request parameter is useful for generating user interfaces with meaningful pick lists or for generating query predicates that have a higher chance of actually identifying a result set.

---

[1] All examples in this clause are informative. In addition, the examples do not include all the XML syntax required to validate. This is done intentionally so as not to obfuscate the examples with XML syntax.

It should be noted that the **GetDomain** operation is a "best-effort" operation. That is to say that a catalogue tries to generate useful information about the requested information model component or request parameter. However, clients should be prepared to handle the case where the GetDomain operation returns no information.

7.2.2 **KVP** encoding

Table 21 specifies the keyword-value pair encoding for the GetDomain operation request.

Table 21 — KVP encoding for GetDomain operation request

Keyword [b]	Data type and value	Optionality and use	Parameter in general model
REQUEST	Character String Fixed value of "GetDomain", case insensitive	One (Mandatory) [a]	(none)
Service	Character String Fixed value of "CSW"	One (Mandatory)	serviceId
Version	Character String Fixed value of "3.0.0"	One (Mandatory)	(none)
ParameterName	List of Character String, comma separated Unordered list of names of requested parameters, of the form *OperationName.ParameterName*	Zero or one (Conditional)	parameterName
ValueReference	List of Character String, comma separated Unordered list of references to components of the information model that the catalogue is using	Zero or one (Conditional)	parameterName
resultType	Enumerated Character String Valid values are "possible" and "available"	Zero or one Default value is "available"	(none)
Filter	Character String	Zero or one	(none)
a The REQUEST parameter contains the same information as the name of the GetDomain element in XML encoding. b Parameter keywords are case insensitive for KVP encoding. Parameters values are case sensitive..			
NOTE To reduce the need for readers to refer to other parts of this document, the first three parameters listed below are copied from Table 7 .			

7.2.3 **XML encoding**

The following XML-Schema fragment defines that XML encoding for the GetDomain operation request:

```
<xsd:element name="GetDomain" type="csw30:GetDomainType" id="GetDomain"/>
<xsd:complexType name="GetDomainType" id="GetDomainType">
   <xsd:complexContent>
      <xsd:extension base="csw30:RequestBaseType">
         <xsd:sequence maxOccurs="unbounded">
            <xsd:choice>
               <xsd:sequence>
                  <xsd:element name="ValueReference">
                     <xsd:complexType>
```

```
                        <xsd:simpleContent>
                           <xsd:extension base="xsd:string">
                              <xsd:attribute name="resultType"
                               type="csw30:ResultTypeType"
                               use="optional" default="available"/>
                           </xsd:extension>
                        </xsd:simpleContent>
                     </xsd:complexType>
                  </xsd:element>
                  <xsd:element ref="fes:Filter" minOccurs="0"/>
               </xsd:sequence>
               <xsd:element name="ParameterName" type="xsd:string"/>
            </xsd:choice>
         </xsd:sequence>
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>
<xsd:simpleType name="ResultTypeType">
   <xsd:restriction base="xsd:string">
      <xsd:enumeration value="possible"/>
      <xsd:enumeration value="available"/>
   </xsd:restriction>
</xsd:simpleType>
```

7.2.4 Parameter descriptions

7.2.4.1 ValueReference parameter

Requirement-049

The ValueReference parameter shall be used to identify a property or other nested component of the information model(s) that a catalogue offers to be interrogated by the GetDomain operation.

Requirement-050

The value of the ValueReference parameter shall be a path expression identifying the specific component to be interrogated.

Requirement-051

For XML-encoded information, an XPath expression shall be used to indicate the information model component to be interrogated

Other encodings of the information model(s) offered by a catalogue may be supported (e.g. JSON) but this standard does not describe the meaning of the ValueReference parameter in such situations.

The optional resultType parameter may be used to indicate whether the response should contain the set of possible or available values for the specified component of the information model.

Valid values for that resultType parameter are “possible” and “available” with the latter being the default value.

An optional filter expression, specified using fes:Filter (see OGC 09-026r1), may be applied to the set of returned values to further refine the response. The use of fes:Filter allows, for example, a CSW to implement a suggestion capability such as that found in most search engines these days.

7.2.4.2 ParameterName parameter

Requirement-052

The ParameterName parameter shall be used to identify an interface parameter to be interrogated by the GetDomain operation.

Requirement-053

The value of the ParameterName parameter shall be a dot-path expression, starting with the operation name, that identifies the API components to be integrated.

For example, “GetRecords.outputFormat” identifies that the value domain for the outputFormat parameter of the GetDomain operation is being requested.

A single GetDomain request can include both ValueReference (see 7.2.4.1) and ParameterName parameters.

7.2.5 Response

7.2.5.1 Encoding

The following XML-Schema fragment defines the response to a GetDomain operation.

```
<xsd:element name="GetDomainResponse"
             type="csw30:GetDomainResponseType" id="GetDomainResponse"/>
<xsd:complexType name="GetDomainResponseType" id="GetDomainResponseType">
   <xsd:annotation>
      <xsd:documentation xml:lang="en">
         Returns the actual values for some property. In general this is
         a subset of the value domain (that is, set of permissible values),
         although in some cases these may be the same.
      </xsd:documentation>
   </xsd:annotation>
   <xsd:sequence>
      <xsd:element name="DomainValues"
                   type="csw30:DomainValuesType" maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="DomainValuesType" id="DomainValuesType">
```

```
   <xsd:sequence>
      <xsd:choice>
         <xsd:element name="ValueReference" type="xsd:string"/>
         <xsd:element name="ParameterName" type="xsd:anyURI"/>
      </xsd:choice>
      <xsd:choice minOccurs="0">
         <xsd:element name="ListOfValues"
                      type="csw30:ListOfValuesType"/>
         <xsd:element name="ConceptualScheme"
                      type="csw30:ConceptualSchemeType" maxOccurs="unbounded"/>
         <xsd:element name="RangeOfValues"
                      type="csw30:RangeOfValuesType"/>
      </xsd:choice>
   </xsd:sequence>
   <xsd:attribute name="type" type="xsd:QName" use="required"/>
   <xsd:attribute name="resultType"
                  type="csw30:ResultTypeType" use="required"/>
</xsd:complexType>
<xsd:complexType name="ListOfValuesType" id="ListOfValuesType">
   <xsd:sequence>
      <xsd:element name="Value" maxOccurs="unbounded">
         <xsd:complexType>
            <xsd:complexContent>
               <xsd:extension base="xsd:anyType">
                  <xsd:attribute name="isDefault" type="xsd:boolean"
                                 use="optional" default="false"/>
                  <xsd:attribute name="count"
                                 type="xsd:nonNegativeInteger"
                                 use="optional"/>
                  <xsd:attribute name="uom"
                                 type="gml:UomIdentifier"
                                 use="optional"/>
               </xsd:extension>
            </xsd:complexContent>
         </xsd:complexType>
      </xsd:element>
   </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="ConceptualSchemeType" id="ConceptualSchemeType">
   <xsd:sequence>
      <xsd:element name="Name" type="xsd:string"/>
      <xsd:element name="Document" type="xsd:anyURI"/>
      <xsd:element name="Authority" type="xsd:anyURI"/>
   </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="RangeOfValuesType" id="RangeOfValuesType">
   <xsd:sequence>
      <xsd:element name="MinValue" type="xsd:anyType" minOccurs="0"/>
      <xsd:element name="MaxValue" type="xsd:anyType" minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

7.2.5.2 Parameter descriptions

7.2.5.2.1 Echoed request parameters

Requirement-054

The response to a GetDomain operation shall echo the ValueReference or ParameterName used to invoke the operation.

7.2.5.2.2 List of values

Requirement-055

If the response to a GetDomain request is an enumerated list of values, then the ListOfValue element shall be employed to encode that information.

One value in the response list may be identified as the default value by using the isDefault parameter and setting its value to “true”.

Requirement-056

In the event that a default value is indicated only one value in the response shall be identified as such.

In the event that the value of the resultType parameter in the GetDomain request is set to “available” (see 7.2.4.1), each value in the response list may include a count of the number of instances of that value.

Requirement-057

If a value in the response is a measurement, its units of measure shall be indicated using the uom parameter.

7.2.5.2.3 Conceptual Schema

Requirement-058

If the response to a GetDomain request is a controlled vocabulary or authoritative list of values, then the ConceptualSchema element shall be employed to encode that information.

Requirement-059

The reference to a controlled vocabulary shall include its name, a link to document describing the controlled vocabulary and a link to an authority responsible for maintaining that controlled vocabulary.

7.2.5.2.4 Range of values

Requirement-060

If the response to a GetDomain request is a range of values, then the RangeOfValues element shall be employed to encode that information.

Requirement-061

The boundaries of the range shall be identified using the MinValue and MaxValue parameters.

The absence of one or both of MinValue or MaxValue indicates an unbounded range in that direction.

7.2.5.2.5 Empty response

Requirement-062

An empty response from the server shall be taken to mean that the catalogue was unable to determine anything about the specified value reference or parameter.

7.2.6 Examples

KVP encoded example:

```
http://www.someserver.com/csw/csw.cgi?request=GetDomain&version=3.0.0&p
arameterName=GetRecords.outputFormat
```

XML encoded example:

```
<GetDomain
   service="CSW"
   version="3.0.0"
   xmlns="http://www.opengis.net/cat/csw/3.0"
   xmlns:csw30="http://www.opengis.net/cat/csw/3.0"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.opengis.net/cat/csw/3.0
                       ../../../csw/3.0/CSW30-discovery.xsd">
   <ParameterName>GetRecords.outputFormat</ParameterName>
</GetDomain>
```

7.3 GetRecords operation

7.3.1 Introduction

The primary means of resource discovery in the general model is the Discovery.query operation. In the HTTP protocol bindingthis operation is implemented as the mandatory **GetRecords** operation, which does a search operation.

7.3.2 KVP encoding

Table 22 specifies the keyword-value pair encoding for the **GetRecords** operation request. Additional keyword-value pair parameters, defined in Table 8 and in Table 9, may also be used with the GetRecords operation to encode query predicates.

This KVP encoding of the GetRecords operation is suitable for use with the HTTP GET method.

NOTE To reduce the need for readers to refer to other parts of this document, the first three parameters listed below are copied from Table 7.

Table 22 — KVP encoding for GetRecords operation request

Keyword [d]	Data type and value	Optionality and use	Parameter in general model
REQUEST	Character String Fixed value of *GetRecords*, case insensitive	Mandatory [a]	(none)
Service	Character String Default value of "CSW"	Mandatory	serviceId
Version	Character String Default value of 3.0.0	Mandatory	(none)
NAMESPACE	List of Character String, comma separated Used to specify a namespace and its prefix Format is xmlns([prefix=]namespace-url). If the prefix is not specified then this is the default namespace.	Optional [b] If qualified names are used in a request all prefixes must be declared by a namespace specification. Only if no qualified name is used the NAMESPACE parameter can be omitted. In this case all elements are in the default namespace[f].	(none)
requestId	URI	Optional Include when client chooses to assign requestId. **This parameter is mandatory in the case of a distributed search.**	requestId

Keyword [d]	Data type and value	Optionality and use	Parameter in general model
outputFormat	Character String Value is Mime type	Optional Default value is *application/xml.* For SOAP *application/soap+ xml* is mandatory.	returnFormat
outputSchema	Any URI.	Optional Default value is *http://www.opengis.net/cat/csw/3.0.*	responseSchema
startPosition	Positive Integer	Optional Default value is 1	cursorPosition
maxRecords	Union: Non-negative Integer	Optional Default value is 10	iteratorSize
typeNames	List of Character String, comma separated Unordered List of object types implicated in the query	Mandatory	collectionID
ElementSetName	Character String	Conditional One (but not both) of ElementSetName or ElementName must be specified.	responseElements
ElementSetName_TypeName	Character String	Conditional May only be specified if the ElementSetName parameters is specified.	responseElements
ElementName	List of Character String	Conditional One (but not both) of ElementSetName or ElementName must be specified.	responseElements

Keyword [d]	Data type and value	Optionality and use	Parameter in general model
SortBy	List of Character String, comma separated Ordered list of names of metadata elements to use for sorting the response Format of each list item is *metadata_element_name:A* indicating an ascending sort or *metadata_ element_name:D* indicating descending sort	Optional Default action is to sort the result according to its default sort which must be declared in the capabilities document (see DefaultSortAlgorithm, Table 20). If no default sort is specified in the capabilities document then it is assumed that the server will sort responses alphabetically by Title (see 6.6.3) in ascending order. If names of one or more metadata elements are provided ordering is done as follows: first order is based on first metadata-element (order: A or D), if records exist with the same value for the first meta-data element ordering of those records is done based on second metadata-element (order: A or D), and so on...	sortField and sortOrder
DistributedSearch	Boolean	Optional Default value is FALSE	queryScope
HopCount	Integer	Optional Include only if DistributedSearch parameter is included Default value is *2*	hopCount

Keyword [d]	Data type and value	Optionality and use	Parameter in general model
ClientId	Any URI	Conditional Mandatory if DistributedSearch set to TRUE	clientId
DistributedSearchId	Any URI	Conditional Mandatory if DistributedSearch set to TRUE	distributedSearchId
DistributedSearchIdTimout	This timeout parameter defines (in sec) how long the distributedSearchId should be valid, meaning how long a server involved in distributed search should minimally store information related to the distributedSearchId	Optional Default value is 600. Makes only sense in case that DistributedSearch set to TRUE	distributedSearchIdTimout
FederatedCatalogues	List of comma separated structures of the format: FederatedCatalogues (fC(URL1, [timeout1]), fC(URL2, [timeout1]),…fC(URLn, [timeoutn])). The URLs have to be escaped. [] means optional.[e]	Conditional Optional if DistributedSearch set to TRUE. Don´t include when DistributedSearch set to FALSE	federatedCatalogues
ResponseHandler	Comma separated list of anyURI	Optional If not included, process request synchronously	responseHandler

a. The REQUEST parameter contains the same information as the name of the GetRecords element in XML encoding.

b. The NAMESPACE parameter contains the same information as the xmlns attributes which may be used for encoding namespace information in XML encoding.

c. The CONSTRAINTLANGUAGE parameter contains the same information as the root element of t0he content of the <Constraint> element indicates the predicate language being used in XML encoding.

d. Parameter keywords are case insensitive for KVP encoding. Parameters values are case sensitive.

e. As defined in OWS Common a URL prefix is defined as a string including, in order, the scheme ("http" or "https"), Internet Protocol hostname or numeric address, optional port number, path, mandatory question mark '?'

f. The default name space is dictated by the value of the outputSchema parameter. For the default value of the outputSchema parameter, the default namespace is http://www.opengis.net/cat/csw/3.0. For an Atom response, where no outputSchema is specified, the default namespace is http://www.w3.org/2005/Atom. For other output schemas, the default namespace will be whatever that schema defines as its default namespace (e.g. in the case of ebRIM it would be urn:oasis:names:tc:ebxml-regrep:xsd:rim:3.0)

7.3.3 XML encoding

The following XML-Schema fragment defines the XML encoding of the **GetRecords** operation request:

```
<xsd:element name="GetRecords" type="csw30:GetRecordsType" id="GetRecords"/>
<xsd:complexType name="GetRecordsType" id="GetRecordsType">
     <xsd:complexContent>
        <xsd:extension base="csw30:RequestBaseType">
           <xsd:sequence>
              <xsd:element name="DistributedSearch"
                           type="csw30:DistributedSearchType" minOccurs="0"/>
              <xsd:element name="ResponseHandler" type="xsd:anyURI"
                           minOccurs="0" maxOccurs="unbounded"/>
              <xsd:choice>
                 <xsd:element ref="csw30:AbstractQuery"/>
                 <xsd:any namespace="##other" processContents="strict"/>
              </xsd:choice>
           </xsd:sequence>
           <xsd:attribute name="requestId" type="xsd:anyURI" use="optional"/>
           <xsd:attributeGroup ref="csw30:BasicRetrievalOptions"/>
        </xsd:extension>
     </xsd:complexContent>
  </xsd:complexType>
  <xsd:attributeGroup name="BasicRetrievalOptions" id="BasicRetrievalOptions">
     <xsd:attribute name="outputFormat" type="xsd:string"
                    use="optional" default="application/xml"/>
     <xsd:attribute name="outputSchema" type="xsd:anyURI" use="optional"
                    default="http://www.opengis.net/cat/csw/3.0"/>
     <xsd:attribute name="startPosition" type="xsd:positiveInteger"
                    use="optional" default="1"/>
     <xsd:attribute name="maxRecords" type="xsd:nonNegativeInteger"
                    use="optional" default="10"/>
  </xsd:attributeGroup>
     <xsd:complexType name="DistributedSearchType" id="DistributedSearchType">
     <xsd:sequence>
        <xsd:element name="federatedCatalogues"
                     type="csw30:FederatedCatalogueType"
                     minOccurs="0" maxOccurs="unbounded"/>
     </xsd:sequence>
     <xsd:attribute name="hopCount" type="xsd:positiveInteger"
                    use="optional" default="2"/>
     <xsd:attribute name="clientId" type="xsd:anyURI" use="required"/>
     <xsd:attribute name="distributedSearchId"
                    type="xsd:anyURI" use="required"/>
     <xsd:attribute name="distributedSearchIdTimout" type="xsd:unsignedLong"
                    use="optional" default="600"/>
  </xsd:complexType>
  <xsd:complexType name="FederatedCatalogueType" id="FederatedCatalogueType">
     <xsd:attribute name="catalogueURL" type="xsd:anyURI" use="required"/>
     <xsd:attribute name="timeout" type="xsd:unsignedLong" use="optional"/>
  </xsd:complexType>
  <xsd:element name="AbstractQuery" type="csw30:AbstractQueryType"
               abstract="true" id="AbstractQuery"/>
  <xsd:complexType name="AbstractQueryType"
                   abstract="true" id="AbstractQueryType"/>
  <xsd:element name="Query" type="csw30:QueryType"
               substitutionGroup="csw30:AbstractQuery" id="Query"/>
  <xsd:complexType name="QueryType" id="QueryType">
     <xsd:complexContent>
        <xsd:extension base="csw30:AbstractQueryType">
           <xsd:sequence>
              <xsd:choice>
                 <xsd:element ref="csw30:ElementSetName"/>
                 <xsd:element name="ElementName" type="xsd:string"
                              maxOccurs="unbounded"/>
              </xsd:choice>
```

```
            <xsd:element ref="csw30:Constraint" minOccurs="0"/>
            <xsd:element ref="fes:SortBy" minOccurs="0"/>
         </xsd:sequence>
         <xsd:attribute name="typeNames"
                        type="csw30:TypeNameListType" use="required"/>
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>
<xsd:simpleType name="TypeNameListType" id="TypeNameListType">
   <xsd:list itemType="xsd:QName"/>
</xsd:simpleType>
<xsd:element name="Constraint"
             type="csw30:QueryConstraintType" id="Constraint"/>
<xsd:complexType name="QueryConstraintType" id="QueryConstraintType">
   <xsd:choice>
      <xsd:element ref="fes:Filter"/>
      <xsd:element name="CqlText" type="xsd:string"/>
   </xsd:choice>
   <xsd:attribute name="version" type="xsd:string" use="required"/>
</xsd:complexType>
<xsd:element name="ElementSetName"
             type="csw30:ElementSetNameType" id="ElementSetName"/>
<xsd:complexType name="ElementSetNameType" id="ElementSetNameType">
   <xsd:simpleContent>
      <xsd:extension base="csw30:ElementSetType">
         <xsd:attribute name="typeNames"
                        type="csw30:TypeNameListType" use="optional"/>
      </xsd:extension>
   </xsd:simpleContent>
</xsd:complexType>
<xsd:simpleType name="RequiredElementSetNamesType"
                id="RequiredElementSetNamesType">
   <xsd:restriction base="xsd:string">
      <xsd:enumeration value="brief"/>
      <xsd:enumeration value="summary"/>
      <xsd:enumeration value="full"/>
   </xsd:restriction>
</xsd:simpleType>
<xsd:simpleType name="ElementSetType">
   <xsd:union memberTypes="xsd:string csw30:RequiredElementSetNamesType"/>
</xsd:simpleType>
```

7.3.4 Parameter descriptions

7.3.4.1 NAMESPACE parameter

Requirement-063

The NAMESPACE parameter shall be used in KVP-encoded requests to bind namespace prefixes to namespace URLs for qualified names specified in other parameters.

For example, the **typeNames** parameter may include qualified names of the form *namespace prefix:name*.

Requirement-064

The value of the NAMESPACE parameter shall be a comma-separated list of lists

delimited by parentheses.

Example:

NAMESPACE=(*namespace prefix*, *namespace_url*),(*namespace_prefix*,*namespace_url*),…

Requirement-065

For the value of the NAMESPACE parameter, literal commas shall be used as list-item separators. Commas not being used as list separators shall be encoded as %2C (see OGC 06-121r9, clause 11.5.3).

Requirement-066

For the value of the NAMESPACE parameter, literal parentheses shall be used to enclose sub-lists consisting of pairs of values each binding a namespace prefix to a namespace url. Parentheses not being used to enclose namespace declarations shall be encoded as %28 for an open parenthesis and %29 for a closing parenthesis.

Requirement-067

An empty namespace prefix string shall be used to bind the default namespace and as in XML, only one default namespace may be bound.

Example:

NAMESPACE=(,http://www.server.com),(ns1,http://www.ns1.com)

Requirement-068

As per OWS common (see OGC 06-121r9, clause 11.5.3), raw commas shall be used to encode list separators. Embedded commas shall be encoded as %2C.

This parameter is not required for the XML-encoded requests since XML includes another mechanism for binding namespace prefixes.

7.3.4.2 requestId parameter

The **requestId** parameter may be included to uniquely identify a request message.

Requirement-069

Servers that support the GetRecords-Distributed-XML and/or GetRecords-Distributed-KVP conformance class shall implement support for the requestId parameter.

Requirement-070

The value of the requestId parameter shall be a UUID (Universally Unique Identifier) generated using the mechanism described in the X.667 standard.

Requirement-071

If the client specifies a value for the **requestId** parameter in a request, the server shall include that value in its response to the request.

Requirement-072

In the event that a server supports the GetRecords-Async-XML and/or GetRecords-Async-KVP conformance classes and a request is received that includes a ResponseHandler parameter but no requestId parameter, the server shall generate a requestId and include it in the Acknowledgement message (see 7.3.4.14) and in the eventual response.

7.3.4.3 outputFormat parameter

The **outputFormat** parameter is used to control the format of the output that is generated in response to a **GetRecords** request.

Requirement-073

The value of the outputFormat parameter shall be a MIME type. The default value, "**application/xml**", means that the output shall be an XML document. For SOAP *application/soap+xml* is mandatory.

Requirement-074

All catalogues that conform to this standard shall support XML as an output format indicated by the value application/xml.

Other output formats may be supported and may include output formats such as TEXT (MIME type *text/plain*), JSON (MIME type application/json), HTML (MIME type *text/html*) or XML without an associated schema (MIME type *text/xml*). This standard does not provide any description of meaning for output formats other than "application/xml" and "*application/soap+xml"* in the case SOAP is being used.

Requirement-075

The list of output formats that a CSW instance provides shall be advertised in the Capabilities document.

Requirement-076

In the case where the output format is *application/xml*, the CSW shall generate an XML document that validates against a schema document that is specified in the output document via the **xsi:schemaLocation** attribute defined in XML.

7.3.4.4 outputSchema parameter

Requirement-077

The outputSchema parameter shall be used to indicate the schema of the output that is generated in response to a GetRecords request.

Requirement-078

Servers that conform to this standard shall support the outputSchema parameter value "http://www.opengis.net/cat/csw/3.0.

Requirement-079

The default value for the outputSchema parameter shall be "http://www.opengis.net/cat/csw/3.0" indicating that response document shall conform to the schema of the core properties (see 6.6.3).

Requirement-080

The outputFormat value "application/atom+xml" shall be used to indicate that the schema of the response document shall conform to the ATOM schema as described in OGC 10-032r8.

Implementation of this standard and/or application profiles may define additional values for the outputSchema parameter but this standard does not describe how a service will operate upon such schemas.

It is recommended that any additional values defined for the outputSchema parameter should be the target namespace of the additionally supported output schemas and should

include a version number. For example, the value *urn:oasis:names:tc:ebxml-regrep:rim:xsd:2.5* might be used to indicate an ebRIM v2.5 output schema, while the value *urn:oasis:names:tc:ebxml-regrep:rim:xsd:3.0* might be used to indicate an ebRIM v3.0 output schema.

Requirement-081

The list of supported outputSchema values shall be advertised in the capabilities document of the service using the Parameter element as outlined in OGC 06-121r3.

7.3.4.5 startPosition parameter

Requirement-082

The optional startPosition parameter shall be used to indicate at which record position the catalogue should start generating output.

Requirement-083

The default value of the startPosition parameter shall be 1.

The default value of *1* means that the catalogue should starts presenting results at the first record in the result set.

7.3.4.6 maxRecords attribute

Requirement-084

The optional maxRecords parameter shall be used to define the maximum number of records that should be presented from the result set of a query.

Requirement-085

The default value for the maxRecords parameter shall be 10.

Requirement-086

If the value of the maxRecords parameter is set to zero, the server shall return a GetRecordsResponse element containing an empty SearchResults element that indicates the estimated size of the result set.

7.3.4.7 typeNames parameter

Requirement-088

The mandatory typeNames parameter shall be used to list one or more names of queryable entities in the catalogue's information model that may be constrained in the predicate of the query.

In the case of XML realization of the OGC core metadata properties (see 6.6), the element csw:Record is the only queryable entity.

Other information models may include more than one queryable component. For example, queryable components for the XML realization of the ebRIM include *rim:Service*, *rim:ExtrinsicObject* and *rim:Association.*

Requirement-089

In such cases the application profile shall describe how multiple **typeNames** values should be processed.

In addition, all or some of these queryable entity names may be specified in the query to define which metadata record elements the query should present in the response to the GetRecords operation.

NOTE The typeNames parameter is different from the Type core queryable metadata property defined in Subclause 6.6.3. The typeNames parameter is composed of one or more names of queryable entities in the information model of the catalogue. The core queryable Type is used to indicate the type or class of a resource being described by the catalogue. Typically the value of the Type property is taken from some controlled vocabulary.

7.3.4.8 ElementName or ElementSetName parameter

Requirement-090

The optional ElementName parameter shall be used to specify one or more metadata record elements, from the output schema specified using the outputSchema parameter, that the query shall present in the response to the GetRecords operation.

Since Subclause 6.6.3 realizes the core metadata properties using XML schema, the value of the ElementName parameter would be an XPath expression perhaps using qualified names. In the general case, a complete XPath expression may be required to correctly reference an element in the information model of the catalogue.

EXAMPLE 1 Using an XPath expression to reference a property in the information model of a catalogue.

```
… <csw:Query typeNames="rim:ExtrinsicObject rim:Association">
     <csw:ElementName>/rim:ExtrinsicObject/@status</csw:ElementName>
```

```
    <csw:ElementName>/rim:ExtrinsicObject/@home</csw:ElementName>
    …
</csw:Query> …
```

However, in the case where the typeNames attribute on the Query element contains a single value, the catalogue can infer the first step in the path expression and it can be omitted. This is usually the case when querying the core metadata properties since the only queryable target is csw:Record.

EXAMPLE 2 A query example where the first step of the path expression is inferred.

```
<csw:Query typeNames="csw:Record">
   <csw:ElementName>dc:identifier</csw:ElementName>
   <csw:ElementName>dct:modified</csw:ElementName>
   …
</csw:Query>
…
```

Requirement-091

If the metadata record element names are not from the schema specified using the outputSchema parameter, then the service shall raise an InvalidParameterValue exception as described in Subclause 6.7.

Requirement-092

As mentioned in Subclause7.3.4.3, if the outputFormat parameter is set to *application/xml*, then the response to the GetRecords operation shall validate against a schema document that is referenced in the response using the xmlns:schemaLocation attribute.

Requirement-093

If the set of metadata record elements that the client specifies in the query is insufficient to generate a valid XML response document, a CSW shall augment the list of elements presented to the client in order to be able to generate a valid XML document.

Thus a client application should expect to receive more than the requested elements if the output format is set to generate an XML document.

Requirement-095

Well known sets of elements may be named, in which case the ElementSetName parameter can be used (e.g., brief, summary or full) to indicate which named set the service shall present to the client.

In the case where the query includes more than one entity name as the value of the typeNames attribute on the Query element, the typeNames attribute on the ElementSetName element can be used to discriminate which element set or sets should be presented.

Requirement-096

The names specified for the typeNames attribute on the ElementSetName element shall be a proper subset of the names specified as the value of the typeNames attribute on the Query element.

Requirement-097

If the typeNames attribute is not included on the ElementSetName element, then the named element sets for all entities specified as the value of the typeNames attribute on the Query element shall be presented.

Requirement-098

If any entity name specified as a value of the typeNames attribute on the ElementSetName element does not match a name specified as a value of the typeNames attribute on the Query element, then the server shall raise an InvalidParameterValue exception as described in Subclause 6.7.

EXAMPLE 3 The following XML fragment shows how the typeNames attribute on the ElementSet element can be used to indicate which element set should be presented when more than one entity in the information model of the catalogue is being queried. In this case, the query is performing a join between the rim:Service, rim:Classification and rim:Association entities but presenting the Brief element set of the rim:Service entity.

```
<GetRecords
   service="CSW"
   version="3.0.0"
   outputFormat="application/xml"
   outputSchema="urn:oasis:names:tc:ebxml-regrep:xsd:rim:3.0"
   xmlns="http://www.opengis.net/cat/csw/3.0"
   xmlns:csw="http://www.opengis.net/cat/csw/3.0"
   xmlns:rim="urn:oasis:names:tc:ebxml-regrep:xsd:rim:3.0"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.opengis.net/cat/csw/3.0
                       ../../../csw/3.0/CSW30-discovery.xsd"
                       urn:oasis:names:tc:ebxml-regrep:xsd:rim:3.0
                       http://docs.oasis-open.org/regrep/v3.0/schema/rim.xsd">
   <Query typeNames="rim:Service rim:Classification rim:Association">
      <ElementSetName typeNames="rim:Service">brief</ElementSetName>
      <!-- ... -->
   </Query>
</GetRecords>
```

The ElementName and ElementSetName parameters are mutually exclusive.

Requirement-099

Either an ElementSetName parameter OR one or more ElementName parameters shall be specified in a query.

The predefined set names of *brief*, *summary* and *full* are meant to represent different level of detail of the source record with *brief* representing the least amount of detail and *full* representing all the metadata record elements.

Requirement-100

The sets of metadata record element names that correspond to *brief*, *summary* and *full* shall be defined in an Application Profile.

Requirement-102

Servers shall advertise in their capabilities documents, via the ows:Parameter element (see 7.1.5, Table 18), the list of element set names that the server understands. That list shall also include the set names “brief”, “summary” and “full”.

7.3.4.9 Predicate languages

Each request encoding (XML and KVP) has a specific mechanism for specifying the predicate language that will be used to constrain a query.

In the XML encoding, the element csw:Constraint (see 6.5.5.2) is used to define the query predicate. The root element of the content of the csw:Constraint (see 6.5.5.2) element defines the predicate language that is being used. Two possible root elements are fes:Filter (see OGC 09-026r1) for the OGC XML filter encoding, and csw:CqlText (see 6.5.6.2) for a common query language string. An example predicate specification in the XML encoding is:

```
<Constraint>
   <CqlText>prop1!=10</CqlText>
</Constraint>
```

or, using the a Filter expression:

```
<Constraint>
   <fes:Filter>
      <fes:Not>
         <fes:PropertyIsEqualTo>
            <fes:ValueReference>prop1</fes:ValueReference>
            <fes:Literal>10</fes:Literal>
         </fes:PropertyIsEqualTo>
      </fes:Not>
```

```
   </fes:Filter>
</Constraint>
```

In the KVP encoding, the parameter CONSTRAINTLANGUAGE is used to specify the predicate language being used while the ConstraintVersion is used to specify the version number of the predicate language.

The Constraint parameter is used to specify the actual predicate. For example, to specify a CQL predicate, the following parameters would be set in the KVP encoding:

```
...CONSTRAINTLANGUAGE=urn:ogc:def:queryLanguage:OGC-
CSW:CQL_TEXT&CONSTRAINT=”prop1!=10”...
```

The value of the CONSTRAINT parameter in the KVP encoding may also include an XML encoded Filter expression although some clients may impose length limitations.

7.3.4.10 OGC filter syntax

7.3.4.10.1 Introduction

The XML implementation of the query language BNF from the general model (see OGC 12-168) may be found in OGC document 09-026r1. The intent of the XML encoding of the OGC common query language is that it be easily parsable using readily available XML parsers and be easily translatable into a target predicate language such as a SQL where clause or an XQuery predicate.

7.3.4.10.2 Provide functional extensibility

One feature of the OGC common query language and the XML implementation is that the predicate language is functionally extensible. This means that functions may be added to the filter predicate language without having to change the underlying schema. The relevant schema fragment from OGC 09-026r1 is:

```
<xsd:element name="Function"
             type="fes:FunctionType"
             substitutionGroup="fes:expression"/>
<xsd:complexType name="FunctionType">
   <xsd:sequence>
      <xsd:element ref="fes:expression"
                   minOccurs="0" maxOccurs="unbounded"/>
   </xsd:sequence>
   <xsd:attribute name="name" type="xsd:string" use="required"/>
</xsd:complexType>
```

According to the schema fragment, any function may be added to the filter predicate language simply by specifying its name and including zero or more ogc expressions as content of the Function element which represent the arguments of the function. The following example shows how a function may be called using the filter syntax:

```
<Function name=”MAX”>
   <PropertyName>DEPTH</PropertyName>
</Function>
```

In this example, the MAX() function is invoked to find the maximum value of the property DEPTH.

Any function may be called using the filter syntax as long as the function is advertised in the filter capabilities section (see 7.1.3, OGC 09-026r1) of an OGC capabilities document.

7.3.4.10.3 Precedence

The XML notation does not provide parentheses to indicate operator precedence as specified in the BNF. The Filter Specification uses the nested structure of the XML notation to indicate this relationship.

In this example, a more complex scalar predicate is encoded using the logical operators AND and OR. The example is equivalent to the expression ((FIELD1=10 OR FIELD1=20) AND (STATUS="VALID")):

```
<Filter xmlns="http://www.opengis.net/fes/2.0"
        xmlns:foo="http://www.someverser.com/foo/">
  <And>
    <Or>
      <PropertyIsEqualTo>
        <ValueReference>foo:FIELD1</ValueReference>
        <Literal>10</Literal>
      </PropertyIsEqualTo>
      <PropertyIsEqualTo>
         <ValueReference>foo:FIELD1</ValueReference>
         <Literal>20</Literal>
      </PropertyIsEqualTo>
    </Or>
    <PropertyIsEqualTo>
      <ValueReference>foo:STATUS</ValueReference>
      <Literal>VALID</Literal>
    </PropertyIsEqualTo>
  </And>
</Filter>
```

7.3.4.10.4 Tight and loose queries

The following examples show the implementation of tight and loose queries using the Filter grammar.[2] In both cases the query is directed to a federation of image catalogues that are compliant to an application profile where an optional searchable attribute "cloudcover" is defined as the percentage of the target obscured by clouds.

In Case 1 the investigator wants to get only images he is sure he can use so he requests only images where the cloud cover is less than 5%. This is the normal case for querying known schema.

```
<fes:Filter xmlns:fes="http://www.opengis.net/fes/2.0">
   <fes:PropertyIsLessThan>
      <fes:ValueReference>cloudcover</fes:ValueReference>
      <fes:Literal>5</fes:Literal>
```

[2] This capability is included in version 2.0 of the OGC Filter Encoding Specification (see OGC 09-026r1).

```
   </fes:PropertyIsLessThan>
</fes:Filter>
```

In Case 2 the investigator is aware that 5% cloud cover is very rare over the target area but he requires only images with less than 5% cloud cover. In this case he wants any images he might be able to use so he requests images which meets his criteria and images where cloud cover in unknown because the catalogue has chosen not to include cloudcover in its searchable attribute set.

```
<fes:Filter xmlns:fes="http://www.opengis.net/fes/2.0">
   <fes:Or>
      <fes:PropertyIsLessThan>
         <fes:ValueReference>cloudcover</fes:ValueReference>
         <fes:Literal>5</fes:Literal>
      </fes:PropertyIsLessThan>
      <fes:PropertyValueDoesNotExist>
         <fes:ValueReference>cloudcover</fes:ValueReference>
      </fes:PropertValueDoesNotExist>
   </fes:Or>
</fes:Filter>
```

7.3.4.10.5 Property references

The fes:ValueReference element is used in Filter expressions to reference properties from the information model of a catalogue.

Requirement-103

If the catalogue's information model is being realized as an XML document with an accompanying XML Schema, then XPath expressions shall be used to reference the various metadata properties.

This is the case when the core metadata properties that are realized as an XML document, using XML Schema, as described in Subclause 6.6.3.

Requirement-104

This implies the XPath expressions shall be used to reference the various core metadata properties in Filter expressions. This is also the case for predicates encoded using CQL text if the predicates reference the XML realization of the core metadata properties.

EXAMPLE 1 The following examples shows an XPath expression being used in a Filter to reference one of the core queryable properties described in Subclause 6.6.3.

```
<csw:Query typeNames="csw:Record">
   <csw:Constraint>
      <fes:Filter>
         <fes:During>
            <fes:ValueReference>/csw:Record/dct:modified</fes:ValueReference>
            <gml:TimePeriod gml:id="TP1">
               <gml:begin>
                  <gml:TimeInstant gml:id="TI1">
```

```
                  <gml:timePosition>2005-05-17T00:00:00Z</gml:timePosition>
               </gml:TimeInstant>
            </gml:begin>
            <gml:end>
               <gml:TimeInstant gml:id="TI2">
                  <gml:timePosition>2005-05-23T00:00:00Z</gml:timePosition>
               </gml:TimeInstant>
            </gml:end>
         </gml:TimePeriod>
      </fes:During>
   </fes:Filter>
</csw:Constraint>
</csw:Query>
```

This following is the same expression represented using CQL text but referencing the XML realization of the core metadata properties.

```
<csw:Query typeNames="csw:Record">
   <csw:Constraint>
      <CqlText>/csw:Record/dct:modified BETWEEN 2006-01-01 AND 2006-05-
30</CqlText>
   </csw:Constraint>
</csw:Query>
```

In certain cases, such as when the typeNames attribute on the Query element only contains the name of a single entity, the root path step may be omitted since the catalogue is able to infer what the first step in the path would be.

EXAMPLE 2 This is same example as above but using an inferred initial path step.

```
<csw:Query typeNames="csw:Record">
   <csw:Constraint>
      <fes:Filter>
         <fes:During>
            <fes:ValueReference>dct:modified</fes:ValueReference>
            <gml:TimePeriod gml:id="TP1">
               <gml:begin>
                  <gml:TimeInstant gml:id="TI1">
                     <gml:timePosition>2005-05-17T00:00:00Z</gml:timePosition>
                  </gml:TimeInstant>
               </gml:begin>
               <gml:end>
                  <gml:TimeInstant gml:id="TI2">
                     <gml:timePosition>2005-05-23T00:00:00Z</gml:timePosition>
                  </gml:TimeInstant>
               </gml:end>
            </gml:TimePeriod>
         </fes:During>
      </fes:Filter>
   </csw:Constraint>
</csw:Query>
```

The CQL text encoding of the same query would be:

```
<csw:Query typeNames="csw:Record">
   <csw:Constraint>
      <CqlText>dct:modified BETWEEN 2006-01-01 AND 2006-05-30</CqlText>
   </csw:Constraint>
</csw:Query>
```

Requirement-105

All CSW implementations that implement the XML filter encoding syntax as defined in the Filter Encoding Implmentation Specification, version 2.0 (see OGC 09-026r1) shall support the minimum XML Path Language (XPath) defined in clause 7.4.4 of OGC 09-026r1.

7.3.4.11 SortBy parameter

The result set may be sorted by specifying one or more metadata record elements upon which to sort.

Requirement-106

In KVP encoding, the SORTBY parameter shall be used to specify the list of sort elements.

Requirement-107

The value for the KVP-encoded SORTBY parameter shall be a comma-separated list of pairs metadata record element names upon which to sort the result set and the letters "A" or "D" indicating the sort order. Literal commas shall be used to delimit list elements. Commas that are not list-element delimiters shall be encoded as %2C.

Example: The following example specifies a sort on two properties named PROP1 and PROP2. PROP1 is sorted in ascending order. PROP2 is sorted in descending order.

…SORTBY=PROP1,A,PROP2,D …

Requirement-108

If no sort is specified then the server shall sort the results according to its default sort which shall be declared in the capabilities doc (see Table 20).

Requirement-109

If no sort is specified and if no default sort is specified in the capabilities document then it is assumed that the server will sort responses alphabetically by Title in ascending order.

Requirement-110

For XML encoded requests, the ogc:SortBy element shall be used to specify a list of sort metadata record elements. The attribute sortOrder is used to specify the sort order for each element. Valid values for the sortOrder attribute are *ASC* indicating an ascending

sort and *DESC* indicating a descending sort.

7.3.4.12 DistributedSearch parameter

Distributed search is explained in detail in Annex B within the Catalogue general model document.

The DistributedSearch parameter may be used to indicate that the query should be distributed. The default query behaviour, if the DistributedSearch parameter is set to *FALSE* (or is not specified at all), is to execute the query on the local server. In the XML encoding, if the DistributedSearch element is not specified then the query is executed on the local server.

The hopCount parameter controls the distributed query behaviour by limiting the maximum number of message hops before the search is terminated. Each catalogue decrements this value by one when the request is received and does not propagate the request if hopCount=0.

The clientId parameter is an Id which uniquely identifies the requestor.

The distributedSearchId is an Id which uniquely identifies a complete client initiated distributed search sequence/session.

The distributedSearchIdTimout defines how long the distributedSearchId should be valid, meaning how long a server involved in distributed search should minimally store information related to the distributedSearchId.

Catalogues may advertise, in its capabilities document, the URL-prefix[3] of the getCapabilities HTTP-GET operation of those catalogues to which a query will be distributed in the case of a distributed search. This is done by using an operation constraint called “FederatedCatalogues” within the capabilities document (see 7.1.5.2)

EXAMPLE The following XML fragment shows how the FederatedCatalogues constraint can be used to list the catalogues on which distributed queries are executed.

```
<ows:Constraint name="FederatedCatalogues">
   <ows:Value>http://gdi-de.sdisuite.de/soapServices/CSWStartup?</ows:Value>
   <ows:Value>http://anotherCat.com/CSWStartup?</ows:Value>
</ows:Constraint>
```

For a specific catalogue server these are at a maximum all catalogues to which a query will be distributed if no restriction to specific federated catalogues is defined by the client.

[3] As defined in OWS Common a URL prefix is defined as a string including, in order, the scheme ("http" or "https"), Internet Protocol hostname or numeric address, optional port number, path, mandatory question mark '?'

To restrict the number of catalogues of a federation which should be searched upon in a distributed query an optional list of those catalogues can be provided within the federatedCatatalogues parameter.

7.3.4.13 ResponseHandler parameter

The ResponseHandler parameter is a flag that indicates how a GetRecords operation is processed by a CSW.

Requirement-111

If the ResponseHandler parameter is not present, then the GetRecords operation shall be processed synchronously meaning that the client sends the GetRecords request to the server and waits to receive a valid response or exception message (see 6.7).

The problem with this mode of operation is that the client may timeout waiting for the server to process the GetRecords request.

Requirement-112

If the ResponseHandler parameter is present, the GetRecords operation shall be processed asynchronously. In this case, the server shall respond immediately to a client's request with an Acknowledgment message (see 7.4.4.13) that indicates that the request has been received and validated.

The GetRecords request may then be processed taking as much time are required to complete the operation.

Requirement-113

When the asynchronous request is completed the server shall send a GetRecordsResponse message or an exception message (see 6.7) to the URI(s) specified as the value of the ResponseHanlder parameter.

This standard does not define operations or method to check on the intermediate status of an asynchronous request.

Requirement-114

The presence of multiple response handlers indicates that the server should sent the GetRecords response to multiple URIs. This is analogous to sending an email to multiple recipients or copying the response to multiple ftp targets.

7.3.4.14 Acknowledgement message

The following XML Schema fragment defines a generic response acknowledgement message used for asynchronous operations:

```
<xsd:element name="Acknowledgement" id="Acknowledgement"
             type="csw30:AcknowledgementType"/>
<xsd:complexType name="AcknowledgementType" id="AcknowledgementType">
   <xsd:sequence>
      <xsd:element name="EchoedRequest" type="csw30:EchoedRequestType"/>
      <xsd:element name="RequestId" type="xsd:anyURI" minOccurs="0"/>
   </xsd:sequence>
   <xsd:attribute name="timeStamp" type="xsd:dateTime" use="required"/>
</xsd:complexType>
<xsd:complexType name="EchoedRequestType" id="EchoedRequestType">
   <xsd:sequence>
      <xsd:any namespace="##any" processContents="lax"/>
   </xsd:sequence>
</xsd:complexType>
```

Requirement-115

The acknowledgment message shall echo the exact text of the client's request, using the EchoedRequest element, and may include an optionally generated request identifier using the RequestId element.

The echoed request may be the XML text of the request message for XML-encoded requests or, for KVP-encoded requests, the URL for the request.

The echoed request is used to correlate the acknowledgement message with the originating request.

7.3.5 Response

7.3.5.1 Response handling

A server can respond in one of two ways to a successfully processed GetRecords operation depending on the presence or absence of the ResponseHandler parameter.

Requirement-116

If the ResponseHandler parameter is specified for a GetRecords request, the server shall verify the request syntax and immediately respond to the client with an Acknowledgment message (see 7.3.4.14).

Requirement-117

If the ResponseHandler parameter is not present, the server shall process the GetRecords request immediately and respond to the waiting client with a valid GetRecordsResponse message (see 7.3.5.3), if the operation succeeded, or an exception message (see 6.7) if the

operation failed.

7.3.5.2 Response messages

This clause describes the mandatory responses to a GetRecords request that all compliance servers must support.

Requirement-118

In response to a GetRecord request where the value of the outputFormat parameter is "application/xml" (or "*application/soap+xml")* and the value of the outputSchema parameter is "http://www.opengis.net/cat/csw/3.0" a server shall respond with a csw30:GetRecordsResponse XML document as described in this clause.

Requirement-119

In response to a GetRecords request where the value of the outputFormat parameter is "application/atom+xml" and the value of the outputSchema parameter is not set, a server shall respond with an XML document that conforms to the ATOM schema as described in OGC 10-032r8.

A CSW service may support other combinations of values for the outputFormat and outputSchema parameters but the meaning of such combinations is not described in this standard.

7.3.5.3 XML encoding

The following XML-schema fragment defines the schema of the response to a GetRecords operation where the value of the outputFormat parameter is set to "application/xml" and the value of the outputSchema parameter is set to "http://www.opengis.net/cat/csw/3.0".

```
<xsd:element name="GetRecordsResponse" id="GetRecordsResponse"
                type="csw30:GetRecordsResponseType"/>
   <xsd:complexType name="GetRecordsResponseType">
      <xsd:sequence>
         <xsd:element name="RequestId" type="xsd:anyURI" minOccurs="0"/>
         <xsd:element name="SearchStatus" type="csw30:RequestStatusType"/>
         <xsd:element name="SearchResults" type="csw30:SearchResultsType"/>
      </xsd:sequence>
      <xsd:attribute name="version" type="xsd:string" use="optional"/>
   </xsd:complexType>
   <xsd:complexType name="RequestStatusType" id="RequestStatusType">
      <xsd:attribute name="timestamp" type="xsd:dateTime" use="optional"/>
   </xsd:complexType>
   <xsd:simpleType name="ResultsStatusType" id="ResultsStatusType">
      <xsd:restriction base="xsd:string">
         <xsd:enumeration value="subset"/>
```

```
      <xsd:enumeration value="complete"/>
      <xsd:enumeration value="processing"/>
      <xsd:enumeration value="none"/>
   </xsd:restriction>
</xsd:simpleType>
<xsd:complexType name="SearchResultsType" id="SearchResultsType">
   <xsd:sequence>
      <xsd:choice>
         <xsd:element ref="csw30:AbstractRecord"
                      minOccurs="0" maxOccurs="unbounded"/>
         <xsd:any namespace="##other" processContents="strict"
                  minOccurs="0" maxOccurs="unbounded"/>
      </xsd:choice>
      <xsd:element ref="csw30:FederatedSearchResultBase"
                   minOccurs="0" maxOccurs="unbounded"/>
   </xsd:sequence>
   <xsd:attribute name="resultSetId" type="xsd:anyURI" use="optional"/>
   <xsd:attribute name="elementSet"
                  type="xsd:string" use="optional"/>
   <xsd:attribute name="recordSchema" type="xsd:anyURI" use="optional"/>
   <xsd:attribute name="numberOfRecordsMatched"
                  type="xsd:nonNegativeInteger" use="required"/>
   <xsd:attribute name="numberOfRecordsReturned"
                  type="xsd:nonNegativeInteger" use="required"/>
   <xsd:attribute name="nextRecord"
                  type="xsd:nonNegativeInteger" use="optional"/>
   <xsd:attribute name="expires" type="xsd:dateTime" use="optional"/>
   <xsd:attribute name="elapsedTime" type="xsd:unsignedLong"
                  use="optional"/>
   <xsd:attribute name="status" type="csw30:ResultsStatusType"
                  use="optional" default="subset"/>
</xsd:complexType>
<xsd:element name="FederatedSearchResultBase"
             type="csw30:FederatedSearchResultBaseType"
             abstract="true" id="FederatedSearchResultBase"/>
<xsd:complexType name="FederatedSearchResultBaseType"
                 abstract="true" id="FederatedSearchResultBaseType">
   <xsd:attribute name="catalogueURL" type="xsd:anyURI" use="required"/>
</xsd:complexType>
<xsd:element name="FederatedSearchResult"
             type="csw30:FederatedSearchResultType"
             substitutionGroup="csw30:FederatedSearchResultBase"
             id="FederatedSearchResult"/>
<xsd:complexType name="FederatedSearchResultType"
                 id="FederatedSearchResultType">
   <xsd:complexContent>
      <xsd:extension base="csw30:FederatedSearchResultBaseType">
         <xsd:sequence>
            <xsd:element name="searchResult"
                         type="csw30:SearchResultsType"/>
         </xsd:sequence>
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>
<xsd:element name="FederatedException"
             type="csw30:FederatedExceptionType"
             substitutionGroup="csw30:FederatedSearchResultBase"
             id="FederatedException"/>
<xsd:complexType name="FederatedExceptionType" id="FederatedExceptionType">
   <xsd:complexContent>
      <xsd:extension base="csw30:FederatedSearchResultBaseType">
         <xsd:sequence>
```

```
            <xsd:element ref="ows:ExceptionReport" maxOccurs="unbounded"/>
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
```

Three levels of detail may be contained within a csw:GetRecordsResponse element.

First, the RequestId element may be used to correlate the response to a GetRecords request for which a value was defined for the requestId attribute.

Requirement-120

The SearchStatus element shall be present and indicates the status of the response.

The search status consists of a timestamp attribute indicating when the result set was created.

Requirement-121

The content of the SearchResults element shall be the set of records returned by the GetRecords operation.

Table 23 describes the attributes that can appear on the **SearchResults** element.

Table 23 — SearchResults parameters

Parameters	Description
resultSetId	A server-generated identifier for the result set. May be used in subsequent GetRecords operations to further refine the result set. If the server does not implement this capability then the attribute should be omitted.
elementSet	The element set returned (e.g. brief, summary or full).
recordSchema	A reference to the type or schema of the records returned.
rumberOfRecordsMatched	Number of records found by the GetRecords operation.
numberOfRecordsReturned	Number of records actually returned to client. This may not be the entire result set since some servers may limit the number of records returned to limit the size of the response package transmitted to the client. Subsequent queries may be executed to see more of the result set. The *nextRecord* attribute will indicate to the client where to begin the next query.
nextRecord	Start position of next record. A value of 0 means all records have been returned.
Expires	An ISO 8601 format date indicating when the result set will expire. If this value is not specified then the result set expires immediately.
elapsedTime	runtime information (msec) of the search within the federated catalogue
federatedSearchResult	The results of every catalogue involved in a distributed search result are grouped within the *federatedSearchResult* element (which is of type *FederatedSearchResultType*) of the *searchResults*. Every

	federatedSearchResult element includes the *catalogueURL* (the URL-prefix[4] of the getCapabilities HTTP-GET operation of the catalogue). This URL is also required for a subsequent getRecordByID request to be sent by the client. Further the *federatedSearchResult* element again includes an element of the type *SearchResultsType* so that trees of results can be described. If a federated catalogue has thrown an exception an entry of type *FederatedExceptionType* is included instead of type *FederatedSearchResultType*. *FederatedExceptionType* includes the URL-prefix of the getCapabilities HTTP-GET operation of the catalogue (*catalogueURL*) as well as one or more elements of type ows:ExceptionReport.

7.3.6 ATOM response

Requirement-122

When the value of the outputFormat parameter is set to "application/atom+xml" and the value of the outputScherma parameter is not set, a CSW shall generate an XML document conforming to the ATOM schema as described in OGC document 10-032r8.

NOTE: Because of limitations in how operations can be described using WSDL, the WSDL files that accompany this standard (see 7.9) show the csw:GetRecordsResponse element as the only valid response to a GetRecords (see 7.3) request. Implementors wishing to use the accompanying WSDL files should be aware of this limitation.

A CSW service may also support other values for the outputFormat parameter, with respect to ATOM, but this standard does not describe their meaning. An example of such a value might be "text/html" which could mean that the server should generate an HTML representation of an ATOM feed or "application/json" which could mean that the server should generate a JSON representation of an ATOM feed.

7.3.7 Examples

KVP encoded example

```
http://www.someserver.com/csw/csw.cgi?request=GetRecords&version=3.0.0&
outputFormat=application/xml&outputSchema=http://www.opengis.net/cat/cs
w/3.0&namespace=csw:http://www.opengis.org/cat/csw&ResponseHandler="mai
lto:pvretano@cubewerx.com"&typeName=csw:Record&elementSetName=brief&con
straintlanguage=urn:ogc:def:queryLanguage:OGC-
CSW:CQLTEXT&constraint="csw:AnyText Like '%pollution%'"
```

XML encoded request

```
<GetRecords
   service="CSW"
   version="3.0.0"
```

[4] As defined in OWS Common a URL prefix is defined as a string including, in order, the scheme ("http" or "https"), Internet Protocol hostname or numeric address, optional port number, path, mandatory question mark '?'

```
   maxRecords="5"
   startPosition="1"
   outputFormat="application/xml"
   outputSchema="http://www.opengis.net/cat/csw/3.0"
   xmlns:dc="http://purl.org/dc/elements/1.1/"
   xmlns:dct="http://purl.org/dc/terms/"
   xmlns="http://www.opengis.net/cat/csw/3.0"
   xmlns:csw="http://www.opengis.net/cat/csw/3.0"
   xmlns:fes="http://www.opengis.net/fes/2.0"
   xmlns:ows="http://www.opengis.net/ows/2.0"
   xmlns:gml="http://www.opengis.net/gml/3.2"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.opengis.net/cat/csw/3.0
                       ../../../csw/3.0/cswAll.xsd
                       http://www.opengis.net/gml/3.2
                       http://schemas.opengis.net/gml/3.2.1/gml.xsd">

<ResponseHandler>ftp://www.myserver.com/pub/MyQuery_Resp.xml</ResponseHandler>
   <Query typeNames="Record">
      <ElementSetName>brief</ElementSetName>
      <Constraint version="1.1.0">
         <fes:Filter>
            <fes:PropertyIsLike wildCard="%" singleChar="_" escapeChar="\">
               <fes:ValueReference>AnyText</fes:ValueReference>
               <fes:Literal>%polution%</fes:Literal>
            </fes:PropertyIsLike>
         </fes:Filter>
      </Constraint>
   </Query>
</GetRecords>
```

Distributed Search example request:

```
<GetRecords service="CSW" version="3.0.0" maxRecords="5" startPosition="1"
outputFormat="application/xml"
outputSchema="http://www.opengis.net/cat/csw/3.0"
xmlns:dc="http://purl.org/dc/elements/1.1/"
xmlns:dct="http://purl.org/dc/terms/"
xmlns="http://www.opengis.net/cat/csw/3.0"
xmlns:csw="http://www.opengis.net/cat/csw/3.0"
xmlns:fes="http://www.opengis.net/fes/2.0"
xmlns:ows="http://www.opengis.net/ows/2.0"
xmlns:gml="http://www.opengis.net/gml/3.2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opengis.net/cat/csw/3.0
                     ../../../csw/3.0/cswAll.xsd
                    http://www.opengis.net/gml/3.2
                    http://schemas.opengis.net/gml/3.2.1/gml.xsd">
   <DistributedSearch distributedSearchId="123" clientId="Client" hopCount="3">
      <federatedCatalogues catalogueURL="http://www.conterra.de/csw?"
timeout="30000"/>
   </DistributedSearch>
   <Query typeNames="Record">
      <ElementSetName typeNames="csw:Record">full</ElementSetName>
      <Constraint version="1.1.0">
         <fes:Filter>
            <fes:And>
               <fes:PropertyIsLike escapeChar="\" singleChar="?" wildCard="*">
                  <fes:ValueReference>dc:title</fes:ValueReference>
                  <fes:Literal>*Elevation*</fes:Literal>
               </fes:PropertyIsLike>
               <fes:PropertyIsEqualTo>
```

```
                  <fes:ValueReference>dc:type</fes:ValueReference>
                  <fes:Literal>Service</fes:Literal>
               </fes:PropertyIsEqualTo>
               <fes:PropertyIsGreaterThanOrEqualTo>
                  <fes:ValueReference>dct:modified</fes:ValueReference>
                  <fes:Literal>2012-06-01</fes:Literal>
               </fes:PropertyIsGreaterThanOrEqualTo>
               <fes:Intersects>
                  <fes:ValueReference>ows:BoundingBox</fes:ValueReference>
                  <gml:Envelope>
                     <gml:lowerCorner>14.05 46.46</gml:lowerCorner>
                     <gml:upperCorner>17.24 48.42</gml:upperCorner>
                  </gml:Envelope>
               </fes:Intersects>
               <fes:BBOX>
                  <fes:ValueReference>ows:BoundingBox</fes:ValueReference>
                  <gml:Envelope>
                     <gml:lowerCorner>14.05 46.46</gml:lowerCorner>
                     <gml:upperCorner>17.24 48.42</gml:upperCorner>
                  </gml:Envelope>
               </fes:BBOX>
               <fes:TOverlaps>
                  <fes:ValueReference>csw:TemporalExtent</fes:ValueReference>
                  <gml:TimePeriod gml:id="TP1">
                     <gml:begin>
                        <gml:TimeInstant gml:id="TI1">
                           <gml:timePosition>2012-05-
17T08:00:00Z</gml:timePosition>
                        </gml:TimeInstant>
                     </gml:begin>
                     <gml:end>
                        <gml:TimeInstant gml:id="TI2">
                           <gml:timePosition>2012-05-
23T11:00:00Z</gml:timePosition>
                        </gml:TimeInstant>
                     </gml:end>
                  </gml:TimePeriod>
               </fes:TOverlaps>
            </fes:And>
         </fes:Filter>
      </Constraint>
   </Query>
</GetRecords>
```

Distributed Search example response (no items in resultset):

```
<GetRecordsResponse
   xmlns:dc="http://purl.org/dc/elements/1.1/"
   xmlns:dct="http://purl.org/dc/terms/"
   xmlns="http://www.opengis.net/cat/csw/3.0"
   xmlns:csw="http://www.opengis.net/cat/csw/3.0"
   xmlns:fes="http://www.opengis.net/fes/2.0"
   xmlns:ows="http://www.opengis.net/ows/2.0"
   xmlns:gml="http://www.opengis.net/gml/3.2"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.opengis.net/cat/csw/3.0
                       ../../../csw/3.0/cswAll.xsd
                       http://www.opengis.net/gml/3.2
                       http://schemas.opengis.net/gml/3.2.1/gml.xsd">
   <RequestId>http://www.altova.com</RequestId>
   <SearchStatus timestamp="2006-12-17T09:30:47-05:00"/>
   <SearchResults
```

```
      resultSetId="http://www.altova.com"
      elementSet="brief"
      recordSchema="http://www.opengis.net/cat/csw/3.0"
      numberOfRecordsMatched="0"
      numberOfRecordsReturned="0"
      nextRecord="1"
      status="complete"
      elapsedTime="6000">
      <FederatedSearchResult catalogueURL="http://www.esa.int/csw?">
         <searchResult
            numberOfRecordsReturned="0"
            numberOfRecordsMatched="0"
            status="complete"
            elapsedTime="5000">
            <FederatedSearchResult
               catalogueURL="http://www.eumetsat.int/csw?">
               <searchResult
                  numberOfRecordsReturned="0"
                  numberOfRecordsMatched="0"
                  status="complete"
                  elapsedTime="1000">
                  <FederatedException
                     catalogueURL="http://www.somehost.int/csw?">
                     <ows:ExceptionReport version="3.0.0">
                        <ows:Exception exceptionCode="someCode">
                           <ows:ExceptionText>someError</ows:ExceptionText>
                        </ows:Exception>
                     </ows:ExceptionReport>
                  </FederatedException>
               </searchResult>
            </FederatedSearchResult>
            <FederatedSearchResult
               catalogueURL="http://www.dlr.de/csw?">
               <searchResult
                  numberOfRecordsReturned="0"
                  numberOfRecordsMatched="0"
                  status="complete"
                  elapsedTime="2000">
               </searchResult>
            </FederatedSearchResult>
         </searchResult>
      </FederatedSearchResult>
   </SearchResults>
</GetRecordsResponse>
```

7.4 GetRecordById operation

7.4.1 Introduction

The mandatory GetRecordById request retrieves the default representation of catalogue records using its identifier. The GetRecordById operation is an implementation of the GetResourceByIdoperation from the general model and of the GetResourceById operation from OGC 06-121r9, defined in Subclause 9.3. This operation presumes that a previous query has been performed in order to obtain the identifiers that may be used with this operation. For example, records returned by a GetRecords operation may contain references to other records in the catalogue that may be retrieved using the GetRecordById operation. This operation is also a subset of the GetRecords operation,

and is included as a convenient short form for retrieving and linking to records in a catalogue.

The definition of a distributed search capability for this operation is not required as a previous getRecords response returns sufficient information to access directly the responsible catalogue server.

7.4.2 KVP encoding

Table 24 specifies the keyword value pair encoding for the GetRecordById operation request.

NOTE To reduce the need for readers to refer to other parts of this document, the first three parameters listed below are copied from Table 7.

Table 24 — KVP encoding for GetRecordById operation request

Keyword [b]	Data type and value	Optionality and use	Parameter in general model
REQUEST	Character String Fixed value of "GetRecordById", case insensitive.	One (Mandatory) [a]	(none)
Service	Character String Default value of "CSW"	One (Mandatory)	serviceId
version	Character String Default value of 3.0.0	One (Mandatory)	(none)
ElementSetName	CodeList with allowed values: "brief", "summary" or "full"	Zero or one (Optional) Default value is "summary"	responseElements
outputFormat	Character String Value is a MIME type The only values that are required to be supported are *application/xml* (and "*application/soap+xml"* for SOAP). Other supported values, that may include *text/html* and *text/plain*, must be advertised in the capabilities document.	Zero or one (Optional) Default value is *application/xml*	returnFormat
outputSchema	URI Reference to the preferred schema of the response	Zero or one (Optional) Default value depends on schema of catalogue's information model	responseSchema
Id	anyURI	One (Mandatory)	

a The REQUEST parameter contains the same information as the name of the < GetRecordById> element in XML encoding.

b Parameter keywords are case insensitive for KVP encoding. Parameters values are case sensitive.

7.4.3 XML encoding

The following XML-Schema fragment defines the XML encoding for the GetRecordById operation request:

```
<xsd:element name="GetRecordById"
             type="csw30:GetRecordByIdType" id="GetRecordById"/>
<xsd:complexType name="GetRecordByIdType" id="GetRecordByIdType">
   <xsd:complexContent>
      <xsd:extension base="csw30:RequestBaseType">
         <xsd:sequence>
            <xsd:element name="Id" type="xsd:anyURI"/>
            <xsd:element ref="csw30:ElementSetName" minOccurs="0"/>
         </xsd:sequence>
         <xsd:attribute name="outputFormat" type="xsd:string"
                        use="optional" default="application/xml"/>
         <xsd:attribute name="outputSchema" type="xsd:anyURI" use="optional"/>
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>
```

7.4.4 Parameter descriptions

7.4.4.1 ElementSetName parameter

The ElementSetName parameter is used to specify a named, predefined set of metadata record elements from each source record that should be presented in the response to the operation. The predefined set names of *brief*, *summary* and *full* are meant to represent different level of detail of the source record with *brief* representing the least amount of detail and *full* representing all the metadata record elements.

Requirement-123

The sets of metadata record element names that correspond to *brief*, *summary* and *full* shall be defined in an Application Profile.

Requirement-124

If the ElementSetName parameter is not set in a request the service shall respond with one *summary* record.

Requirement-125

Other set names may also be used but these shall be listed in the server's capabilities document using the Parameter element.

7.4.4.2 Id parameter

Requirement-126

The value of the Id parameter shall be the identifier of a record that the CSW is to present to the client.

In the XML encoding, one Id element may be used to specify the record identifier to retrieve.

Requirement-127

If the identifier specified in the operation is invalid (not matching, not found), then the operation shall fail and an InvalidParameterValue exception message should be returned as described in Subclause 6.7.

The GetRecordById operation returns a bare response (see Requirement-139) and as such does not return a response container such as the GetRecordsResponse (see 7.3.5) of the GetRecords operation (see 7.3). As a result, and as stated in Requirement-127, an invalid identifier results in an exception message. The value of the identifier parameter for the GetRecordById operation is considered invalid if it is empty, badly formed as validated by the server or does not match any records.

7.4.4.3 outputFormat parameter

The outputFormat parameter is used to control the format of the output that is generated in response to a GetRecordById request.

Requirement-128

The value of the outputFormat parameter shall be a MIME type.

Requirement-129

The default value for the outputFormat parameters shall be "application/xml" (or "*application/soap+xml"* for SOAP*)* indicating the output shall be an XML document.

Other output formats may be supported and may include output formats such as TEXT (MIME type *text/plain*), or HTML (MIME type *text/html*).

Requirement-130

The list of output formats that a CSW instance provides shall be advertised in the capabilities document.

Requirement-131

In the case where the output format is “application/xml”, the CSW shall generate an XML document that validates against a schema document that is specified in the output document via the xsi:schemaLocation attribute defined in XML.

7.4.4.4 outputSchema parameter

The outputSchema parameter is included in the GetRecordById operation to allow clients to request a representation of a catalogue record other than the default representation.

By default, the GetRecordById operation generates a response that validates against the schema of the information model that the catalogue implements. This is the default representation of a catalogue record.

Requirement-132

The outputSchema parameter shall be used to indicate the schema of the output that is generated in response to a GetRecordById request.

Requirement-133

Servers that conform to this standard shall support the outputSchema parameters value “http://www.opengis.net/cat/csw/3.0”.

Requirement-133a

Servers that conform to this standard shall support the outputFormat parameter values “application/xml” and “application/atom+xml”.

Requirement-134

The default value for the outputSchema parameter shall be “http://www.opengis.net/cat/csw/3.0” indicating that response document shall conform to the schema of the core properties (see 6.6.3).

Requirement-135

The outputFormat value “application/atom+xml” shall be used to indicate that the schema of the response document shall conform to the ATOM XML schema.

Requirement-136

The list of valid values for the outputSchema parameter for the GetRecordById operations shall be advertised in the server's capabilities document using the Parameter element within the Operation element.

Requirement-137

The list of valid values for the outputSchema parameter advertised in the capabilities document shall include the value "http://www.opengis.net/cat/csw/3.0", representing the schema of the common queryable and returnable elements that all CSW implementation must support.

Requirement-138

The default value of the outputSchema parameter – which is also the default representation of a catalogue record -- shall be the first value listed in the server's capabilities document as the list of valid values for the outputSchema parameter for the GetRecordById operation.

EXAMPLE 1 The following XML fragment lists the possible values for the outputSchema parameter for a catalogue that only support the OGC common schema:

```
<Operation name="GetRecordById>
    ...
    <Parameter name="outputSchema">
       <Value>http://www.opengis.net/cat/csw/3.0</Value>
    </Parameter>
    ...
</Operation>
```

EXAMPLE 2 The following XML fragment lists the possible values for the outputSchema parameter for a catalogue that uses ebRIM v3.0 as its underlying information model:

```
<Operation name="GetRecordById>
    ...
    <Parameter name="outputSchema">
       <Value>urn:oasis:names:tc:ebxml-regrep:xsd:rim:3.0</Value>
       <Value>http://www.opengis.net/cat/csw/3.0</Value>
    </Parameter>
    ...
</Operation>
```

7.4.5 Response

Requirement-139

The response to a GetRecordById request shall be a bare record from one of the information models supported by the server. A bare record is one without a containing

element such as that defined for the response to a GetRecords request (see 6.7). The schema and format of the response is determined by the values of the outputFormat (see 7.3.4.3) and outputSchema (see 7.3.4.4) parameters. Brief- and summary-representations must be (XML-schema) valid subsets of the full bare record.

Requirement-140

If the value of the outputFormat parameter is set to "application/atom+xml" and the value of the outputScherma parameter is not set a CSW shall only generate an ATOM "entry" XML element response.

Requirement-141

If no matching record is found for the Id provided within a GetRecordById request an Exception shall be returned and the HTTP status code set to 400.

7.4.6 Examples

KVP encoded request:

http://www.someserver.com/csw/csw.cgi?request=GetRecordById&version=http://www.someserver.com/csw/csw.cgi?request=GetRecordById&version=3.0.0&id=REC-10

XML encoded request:

```
<GetRecordById
   service="CSW"
   version="3.0.0"
   outputFormat="application/xml"
   outputSchema="http://www.opengis.net/cat/csw/3.0"
   xmlns="http://www.opengis.net/cat/csw/3.0"
   xmlns:csw30="http://www.opengis.net/cat/csw/3.0"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.opengis.net/cat/csw/3.0
                       ../../../csw/3.0/cswAll.xsd
                       http://www.opengis.net/gml/3.2
                       http://schemas.opengis.net/gml/3.2.1/gml.xsd">
   <Id>REC-10</Id>
</GetRecordById>
```

Another example request:

```
<GetRecordById
   service="CSW"
   version="3.0.0"
   outputFormat="application/xml"
   outputSchema="http://www.opengis.net/cat/csw/3.0"
   xmlns="http://www.opengis.net/cat/csw/3.0"
   xmlns:csw30="http://www.opengis.net/cat/csw/3.0"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.opengis.net/cat/csw/3.0
```

```
                        ../../../csw/3.0/cswAll.xsd
                        http://www.opengis.net/gml/3.2
                        http://schemas.opengis.net/gml/3.2.1/gml.xsd">
   <Id>efc40467-284d-4fee-af2a-522c717e7165</Id>
   <ElementSetName>summary</ElementSetName>
</GetRecordById>
```

Without the declaration of an Element Set or Output Schema, the server will respond with a default encoding of the result as *summary* record.

XML encoded response:

```
<csw30:SummaryRecord
   xmlns:csw30="http://www.opengis.net/cat/csw/3.0"
   xmlns:dc="http://purl.org/dc/elements/1.1/"
   xmlns:dct="http://purl.org/dc/terms/"
   xmlns:ows="http://www.opengis.net/ows/2.0"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.opengis.net/cat/csw/3.0
                        ../../../csw/3.0/record.xsd">
   <dc:identifier>00180e67-b7cf-40a3-861d-b3a09337b195</dc:identifier>
   <dc:title>Image2000 Product 1 (at1) Multispectral</dc:title>
   <dc:type>dataset</dc:type>
   <dc:subject>imagery</dc:subject>
   <dc:subject>baseMaps</dc:subject>
   <dc:subject>earthCover</dc:subject>
   <dc:format>BIL</dc:format>
   <dct:modified>2004-10-04 00:00:00</dct:modified>
   <dct:abstract>IMAGE2000 product 1 individual orthorectified scenes.
IMAGE2000 was  produced from ETM+ Landsat 7 satellite data and provides a
consistent European coverage of individual orthorectified scenes in national
map projection systems.</dct:abstract>
</csw30:SummaryRecord>
```

7.5 Record locking

This standard does not define a locking interface, instead relying on the underlying repository to mediate concurrent access to catalogue records. A profile of this standard may define a locking interface if the implementation community demands this functionality.

7.6 Transaction operation

7.6.1 Introduction

The optional Transaction operation defines an interface for creating, modifying and deleting catalogue records.

Requirement-142

The specific information model(s) that can be operated upon by the Transaction operation shall be declared in the capabilities document of a server using the TransactionSchema constraint (see Table 19).

7.6.2 KVP encoding

There is no KVP encoding for transaction operation request, because there is no convenient way to encode the transaction payloads using keyword-value pairs. In addition, the actual text of a transaction message may be very long, again making it inconvenient to use KVP encoding.

7.6.3 XML encoding

7.6.3.1 Overview

The following XML schema fragment defines the XML encoding of the Transaction operation request:

```
<xsd:element name="Transaction" type="csw30:TransactionType" id="Transaction"/>
<xsd:complexType name="TransactionType" id="TransactionType">
   <xsd:complexContent>
      <xsd:extension base="csw30:RequestBaseType">
         <xsd:sequence>
            <xsd:choice maxOccurs="unbounded">
               <xsd:element name="Insert" type="csw30:InsertType"/>
               <xsd:element name="Update" type="csw30:UpdateType"/>
               <xsd:element name="Delete" type="csw30:DeleteType"/>
            </xsd:choice>
         </xsd:sequence>
         <xsd:attribute name="verboseResponse" type="xsd:boolean"
                        use="optional" default="false"/>
         <xsd:attribute name="requestId" type="xsd:anyURI" use="optional"/>
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>
```

The Transaction element defines an atomic unit of work and is a container for one or more insert, update and/or delete actions.

The requestId attribute may be used by a client application to associate a user-defined identifier with the operation.

The verboseResponse attribute is a boolean that may be used by a client to indicate to a server the amount of detail to generate in the response. A value of FALSE means that a CSW should generate a terse or brief transaction response. A value of TRUE, or the absence of the attribute, means that the normal detailed transaction response should be generated. The schema of transaction response is defined in Subclause 7.6.4.

7.6.3.2 Insert action

The following XML-Schema fragment defines an insert action:

```
<xsd:complexType name="InsertType" id="InsertType">
   <xsd:sequence>
      <xsd:any namespace="##other"
               processContents="strict" maxOccurs="unbounded"/>
   </xsd:sequence>
   <xsd:attribute name="typeName" type="xsd:QName" use="optional"/>
   <xsd:attribute name="handle" type="xsd:ID" use="optional"/>
</xsd:complexType>
```

The Insert element is a container for one or more records that are to be inserted into the catalogue.

Requirement-143

The schema of the record(s) to be inserted shall conform to one of the schemas for an information model that the catalogue supports as advertised using the TransactionSchema constraint on the Transaction operations (see Table 19).

The handle attribute is an additional parameter not defined in the general model. It is used in the XML encoding to associate a mnemonic name with each action contained in a Transaction element for the purpose of error handling. If a CSW encounters an error processing a transaction request and the handle attribute is defined, the CSW can localize the source of the problem for the client by specifying the handle in the exception response as described in Subclause 6.7.

7.6.3.3 Update action

The following XML Schema fragment defines an update action:

```
<xsd:complexType name="UpdateType" id="UpdateType">
   <xsd:sequence>
      <xsd:choice>
         <xsd:any namespace="##other" processContents="strict"/>
         <xsd:sequence>
            <xsd:element ref="csw30:RecordProperty" maxOccurs="unbounded"/>
            <xsd:element ref="csw30:Constraint"/>
         </xsd:sequence>
      </xsd:choice>
   </xsd:sequence>
   <xsd:attribute name="typeName" type="xsd:QName" use="optional"/>
   <xsd:attribute name="handle" type="xsd:ID" use="optional"/>
</xsd:complexType>
<xsd:element name="RecordProperty" type="csw30:RecordPropertyType"/>
<xsd:complexType name="RecordPropertyType">
   <xsd:sequence>
      <xsd:element name="Name" type="xsd:string"/>
      <xsd:element name="Value" type="xsd:anyType" minOccurs="0">
   </xsd:sequence>
</xsd:complexType>
```

The Update element may be used to replace an entire record in the catalogue with a new instance of that record or it may be used to update specific values in a catalogue record.

Requirement-144

If an entire record is being replaced, an instance of that record shall appear as the content of the Update element. The schema of the new record instance shall validate against one of the schemas listed as the value of the TransactionSchema constraint (see Table 19) in the server's capabilities document.

Requirement-145

If specific record properties are being updated then a sequence of RecordProperty elements shall appear as the content of the Update element.

The RecordProperty element contains a Name element and a Value element. The Name element is used to reference a value to be updated by name. For XML-encoded records, the value of the Name element may be an XPath expressions (see W3C XML Path Language) identifying a specific value in the record to be changed. The Value element contains the replacement value that will be used to update the record in the catalogue.

The set of records affected or scope of an Update action is determined by the contents of the csw:Constraint element. The csw:Constraint element is defined in Subclause 6.5.5.2.

Requirement-146

In order to prevent all records in a catalogue from inadvertently being updated, the csw:Constraint (see 6.5.5.2) element shall be specified.

The optional typeNames attribute may be used to specify the collection name from which records will be updated.

The handle attribute is described in subclause 7.6.3.2.

7.6.3.4 Delete action

The following XML Schema fragment defines a delete action:

```
<xsd:complexType name="DeleteType" id="DeleteType">
   <xsd:sequence>
      <xsd:element ref="csw30:Constraint"/>
   </xsd:sequence>
   <xsd:attribute name="typeName" type="xsd:QName" use="optional"/>
   <xsd:attribute name="handle" type="xsd:ID" use="optional"/>
</xsd:complexType>
```

The Delete element contains a csw:Constraint element (see 6.5.5.2) that identifies a set of records that are to be deleted from the catalogue.

Requirement-147

The csw:Constraint (see 6.5.5.2) element shall be specified in order to prevent every record in the catalogue from inadvertently being deleted.

The typeNames attribute is used to specify the collection name from which records will be deleted.

The handle attribute is described in subclause 7.6.3.2.

7.6.4 Response

The following XML Schema fragment defines the response to a Transaction request:

```
<xsd:element name="TransactionResponse"
             type="csw30:TransactionResponseType" id="TransactionResponse"/>
<xsd:complexType name="TransactionResponseType">
   <xsd:sequence>
      <xsd:element name="TransactionSummary"
                   type="csw30:TransactionSummaryType"/>
      <xsd:element name="InsertResult" type="csw30:InsertResultType"
                   minOccurs="0" maxOccurs="unbounded"/>
   </xsd:sequence>
   <xsd:attribute name="version" type="xsd:string" use="optional"/>
</xsd:complexType>
<xsd:complexType name="TransactionSummaryType" id="TransactionSummaryType">
   <xsd:sequence>
      <xsd:element name="totalInserted"
                   type="xsd:nonNegativeInteger" minOccurs="0"/>
      <xsd:element name="totalUpdated"
                   type="xsd:nonNegativeInteger" minOccurs="0"/>
      <xsd:element name="totalDeleted"
                   type="xsd:nonNegativeInteger" minOccurs="0"/>
   </xsd:sequence>
   <xsd:attribute name="requestId" type="xsd:anyURI" use="optional"/>
</xsd:complexType>
<xsd:complexType name="InsertResultType" id="InsertResultType">
   <xsd:sequence>
      <xsd:element ref="csw30:BriefRecord" maxOccurs="unbounded"/>
   </xsd:sequence>
   <xsd:attribute name="handleRef" type="xsd:anyURI" use="optional"/>
</xsd:complexType>
```

Requirement-148

The response to a Transaction operation shall include a summary of the transaction by indicating the number records created, updated or deleted by the transaction.

Requirement-149

The response to a Transaction operation shall include the identifiers of any new records created by the transaction.

Requirement-150

The InsertResult element may appear zero or more times in the transaction response and shall report a brief representation of each new record, including the record identifier, created in the catalogue.

Requirement-151

The records shall be reported in the same order in which the Insert elements appear in a transaction request and shall map 1-to-1.

Optionally, the handle attribute may be used to correlate a particular Insert element in the Transaction request with an InsertResult element found in the transaction response.

7.7 Harvest operation

7.7.1 Introduction

The general model defines two operations in the Manager class that may be used to create or update records in the catalogue. They are the transaction operation and the harvestResources operation. The transaction operation may be used to "push" data into the catalogue and is defined in Subclause 7.6. This subclause defines the optional Harvest operation, which is an operation that "pulls" data into the catalogue. That is, this operation only references the data to be inserted or updated in the catalogue, and it is the job of the catalogue service to resolve the reference, fetch that data, and process it into the catalogue.

The Harvest operation had two modes of operation, controlled by a flag in the request. The first mode of operation is a synchronous mode in which the CSW receives a Harvest request from the client, processes it immediately, and sends the results to the client while the client waits. The second mode of operation is asynchronous in that the server receives a Harvest request from the client, and sends the client an immediate acknowledgement that the request has been successfully received. The server can then process the Harvest request whenever it likes, taking as much time as is required and then send the results of the processing to a URI specified in the original Harvest request. This latter mode of operation is included to support Harvest requests that could run for a period of time longer than most HTTP timeout’s will allow.

Processing a Harvest request means that the CSW resolves the URI pointing to the metadata resource, parses the resource, and then creates or modifies metadata records in the catalogue in order to register the resource. This operation may be performed only once or periodically depending on how the client invokes the operation.

7.7.2 KVP encoding

Table 25 specifies the keyword-value pair encoding for the Harvest operation request.

NOTE To reduce the need for readers to refer to other parts of this document, the first three parameters listed below are copied from Table 7.

Table 25 — KVP encoding for Harvest operation request

Keyword [c]	Data type and value	Optionality and use	Parameter in general model
REQUEST	Character String Fixed value of *Harvest*, case insensitive	One (Mandatory) [a]	(none)
service	Character String Default value of "CSW"	One (Mandatory)	serviceId
version	Character String Default value of 3.0.0	One (Mandatory)	(none)
NAMESPACE	List of Character String, comma separated Used to specify a namespace and its prefix Format is xmlns([*prefix=]url)*. If the *prefix* is not specified then this is the default namespace.	Zero or one (Optional) [b] If qualified names are used in a request all prefixes must be declared by a namespace specification. Only if no qualified name is used the NAMESPACE parameter can be omitted. In this case all elements are in the default namespace.	(none)
Source	URI Reference to the source from which the resource is to be harvested	One (Mandatory)	Source
ResourceType	Character String Reference to the type of resource being harvested, see Subclause7.7.4.2	One (Mandatory)	Type
ResourceFormat	Character String MIME type indicating format of the resource being harvested	Zero or one (Optional) Default value is *application/xml*	resourceFormat
ResponseHandler	URL A reference to a person or entity that the CSW should respond to when it has completed processing a Harvest request asynchronously	Zero or one (Optional) If not included, process request synchronously	responseHandler
HarvestInterval	Period Must conform to ISO8601 Period syntax.	Zero or one (Optional) If not specified, then harvest only once in response to the request.	harvestInterval

a The REQUEST parameter contains the same information as the name of the <Harvest> element in XML encoding.

b The NAMESPACE parameter contains the same information as the xmlns attributes which may be used to convey namespace information in XML encoding.

c Parameter keywords are case insensitive for KVP encoding. Parameters values are case sensitive..

7.7.3 XML encoding

The following XML-Schema fragment defines the XML encoding for a Harvest operation request:

```
<xsd:element name="Harvest" type="csw30:HarvestType" id="Harvest"/>
<xsd:complexType name="HarvestType">
   <xsd:complexContent>
      <xsd:extension base="csw30:RequestBaseType">
         <xsd:sequence>
            <xsd:element name="Source" type="xsd:anyURI"/>
            <xsd:element name="ResourceType" type="xsd:string"/>
            <xsd:element name="ResourceFormat" type="xsd:string"
                         default="application/xml" minOccurs="0"/>
            <xsd:element name="HarvestInterval" type="xsd:duration"
                         minOccurs="0"/>
            <xsd:element name="ResponseHandler" type="xsd:anyURI"
                         minOccurs="0" maxOccurs="unbounded"/>
         </xsd:sequence>
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>
```

7.7.4 Parameter descriptions

7.7.4.1 Source parameter

The Source parameter is used to specify a URI reference to the metadata resource to be harvested.

7.7.4.2 ResourceType parameter

The ResourceType parameter references a document that defines the structure of the resource being harvested. For high interoperability, this resource should be an XML document, and the ResourceType parameter string value should be a URI that references the structure of that XML document (i.e., its XML Schema namespace identifier URI). If a server can harvest resources in the schema of an information model it supports, the ResourceType URI should be the same as the outputSchema parameter URI defined for the GetRecords operation.

Table 26 defines a set of URIs that may be used to identify well know metadata formats. CSW implementations and CSW application profiles may support other values of the ResourceType parameter for harvesting additional artifact types. If creating new ResourceType URNs is needed, the format of the values may be *urn:ogc:def:resourceType:CSW:token,* where *token* is a placeholder for a value that denotes the specific resource type.

Table 26 — URIs for well known metadata standards

URI	Description
http://www.opengis.net/wms	WMS capability document, all current versions
http://www.opengis.net/wfs	WFS capability document, all current versions
http://www.opengis.net/wcs	WCS capability document, all current versions
http://www.opengis.net/cat/csw	CSW capability document, all current versions
http://www.opengis.net/wmts	WMTS capability document, all current versions
http://www.fgdc.gov/metadata/csdgm	Content Standard for Digital Geospatial Metadata (CSDGM), Vers. 2 (FGDC-STD-001-1998)
http://www.auslig.gov.au/dtd/anzmeta-1.3.dtd	Australian Spatial Data Infrastructure Standard
http://www.isotc211.org/schemas/2005/gmd/	ISO19139 document
http://metadata.dod.mil/mdr/ns/DDMS/1.3/	DEPARTMENT OF DEFENSE DISCOVERY METADATA STANDARD (DDMS)

Requirement-152

A compliant server shall, in its Capabilities document, advertise the resource type URIs it recognizes using the Parameter element within the Operation element.

EXAMPLE The following XML fragment illustrates how a catalogue server could advertise which resource types it can harvest:

```
<ows:Operation name="Harvest">
   ...
   <ows:Parameter name="ResourceType">
      <ows:Value>http://www.opengis.net/wms</ows:Value>
      <ows:Value>http://www.opengis.net/wfs</ows:Value>
      <ows:Value>http://www.opengis.net/wms</ows:Value>
      <ows:Value>http://www.opengis.net/wmts</ows:Value>
      <ows:Value>http://www.opengis.net/cat/csw</ows:Value>
      <ows:Value>urn:ogc:def:resourceType:CSW:FGDC</ows:Value>
      <ows:Value>http://www.auslig.gov.au/dtd/anzmeta-1.3.dtd</ows:Value>
      <ows:Value>http://metadata.dod.mil/mdr/ns/DDMS/1.3/</ows:Value>
      <ows:Value>audio/mpeg</ows:Value>
      <ows:Value>application/pdf</ows:Value>
   </ows:Parameter>
   ...
</ows:Operation>
```

7.7.4.3 ResourceFormat parameter

The ResourceFormat parameter is used to indicate the encoding used for the resource being harvested. This parameter is included to support the harvesting of metadata resources available in various formats such as plain text, XML or HTML.

Requirement-153

The value of the ResourceFormat parameter shall be a MIME type.

Requirement-154

If the ResourceFormat parameter is not specified for the Harvest request then the default value of *application/xml* shall be assumed.

7.7.4.4 ResponseHandler parameter

The ResponseHandler parameter is a flag that indicates how the Harvest operation is processed by a CSW.

Requirement-155

If the parameter is not present, then the Harvest operation shall be processed synchronously meaning that the client sends the Harvest request to a CSW and then waits to receive a HarvestResponse or exception message (see 6.7).

The problem with this mode of operation is that the client may timeout waiting for the server to process the Harvest request.

Requirement-156

If the parameter is present, the Harvest operation shall be processed asynchronously. In this case, the server shall respond immediately to the client's request with an Acknowledgement message (see 7.3.4.14) that indicates that the request has been received and validated

The Harvest request may then be processed at some later time taking as much time as is required to complete the operation.

Requirement-157

When the asynchronous request is completed the server shall send a HarvestResponse message or an exception message (see 6.7) to the URI(s) specified as the value of the ResponseHandler parameter.

This standard does not define any operations or method to check on the intermediate processing status of an asynchronous request.

Requirement-158

The presence of multiple response handlers indicates that the server shall send the

response to the Harvest operation to multiple URI(s). This is analogous to sending an email message to multiple recipients or copying the response to multiple ftp servers.

7.7.4.5 HarvestInterval Parameter

Requirement-175

The HarvestInterval parameter shall be used to specify the period of time, in ISO 8601 period format, that should elapse before a CSW attempts to re-harvest the specified resource thus refreshing it copy of a resource.

Requirement-176

If no HarvestInterval parameter is specified then the resource shall be harvested only once in response to the Harvest request.

7.7.5 Harvest with attachments

7.7.5.1 Introduction

In some cases the resource to be harvested cannot be referenced by a URI and may, optionally, be included with the Harvest request. In such cases the resource may be attached to the Harvest request using a multipart message.

Requirement-159

The ability to handle attachments on the Harvest operation is optional and servers that support it shall declare this in their capabilities document using the HarvestHandlesAttachements constraint.

7.7.5.2 Multipart attachments

Requirement-160

When attaching items to a Harvest request, the multipart/related content type (see IETF RFC 2387) shall be used.

Multipart media types such as this one are intended for compound messages that consist of several interrelated parts; such entities comprise a 'root' part plus any number of other parts.

Requirement-161

The first part of a multipart MIME attachment shall be the XML-encoded Harvest

request. Subsequent parts shall contain the documents referenced by the Harvest operation.

A repository item is included in a message part, with the Content-Type and Content-ID headers of that part set as shown in the following example:

```
Content-Type: multipart/related; boundary=j9a8fd4r; type="application/xml:

--j9a8fd4r
Content-Type: application/xml
Content-ID: <id0001>

<?xml version="1.0"?>
<csw:Harvest ...>
   <csw:Source>cid:od0002</csw:Source>
   ...
</csw:Harvest>

--j9a8fd4r
Content-Type: application/xml
Content-ID: <id0002>

<?xml version="1.0"?>
...
```

Requirement-162

In multipart attachments, the Content-ID header field shall be used to uniquely identify MIME entities in the message part. The csw:Source element shall contain the value which shall be a URL conforming to the 'cid' scheme. The URL value shall refer to a specific body part of a message as described in RFC 2392.

7.7.5.3 Attachment persistence

Requirement-163

Servers that support the Harvest operation with attachments shall arrange to store the attached resource in some persistent manner and shall make the attached resource accessible via the dcmes:relation element in a csw:Record response.

Requirement-164

The value of the dcmes:relation element shall be a URL that, when resolved, retrieves the attachment from the servers repository.

Example:

```
<csw30:Record
```

```
   xmlns:csw30="http://www.opengis.net/cat/csw/3.0"
   xmlns:dc="http://purl.org/dc/elements/1.1/"
   xmlns:dct="http://purl.org/dc/terms/"
   xmlns:ows="http://www.opengis.net/ows/2.0"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.opengis.net/cat/csw/3.0
                       ../../../csw/3.0/record.xsd">
   <dc:identifier>00180e67-b7cf-40a3-861d-b3a09337b195</dc:identifier>
   <dc:title>Image2000 Product 1 (at1) Multispectral</dc:title>
   <dct:modified>2004-10-04 00:00:00</dct:modified>
   <dct:abstract>IMAGE2000 product 1 individual orthorectified scenes.
IMAGE2000 was  produced from ETM+ Landsat 7 satellite data and provides a
consistent European coverage of individual orthorectified scenes in national
map projection systems.</dct:abstract>
   <dc:type>dataset</dc:type>
   <dc:subject>imagery</dc:subject>
   <dc:subject>baseMaps</dc:subject>
   <dc:subject>earthCover</dc:subject>
   <dc:format>BIL</dc:format>
   <dc:creator>Vanda Lima</dc:creator>
   <dc:language>en</dc:language>
   <dc:relation>http://www.someserver.com/repository/00180e67-b7cf-40a3-861d-
b3a09337b195</dc:relation>
   <ows:WGS84BoundingBox>
      <ows:LowerCorner>14.05 46.46</ows:LowerCorner>
      <ows:UpperCorner>17.24 48.42</ows:UpperCorner>
   </ows:WGS84BoundingBox>
</csw30:Record>
```

7.7.6 Response

The following XML-Schema fragment defines the HarvestResponse message:

```
<xsd:element name="HarvestResponse"
             type="csw30:HarvestResponseType" id="HarvestResponse"/>
<xsd:complexType name="HarvestResponseType" id="HarvestResponseType">
   <xsd:choice>
      <xsd:element ref="csw30:Acknowledgement"/>
      <xsd:element ref="csw30:TransactionResponse"/>
   </xsd:choice>
</xsd:complexType>
```

A server can respond in one of two ways to a successfully processed Harvest operation depending on the presence or absence of the ResponseHandler parameter.

Requirement-165

If the ResponseHandler parameter is specified for a Harvest request, the server shall verify the request syntax and immediately respond to the client with an Acknowledgment message (see 7.3.4.14).

Requirement-166

If the ResponseHandler parameter is not present, the CSW server shall process the Harvest request immediately and respond to the waiting client with a valid HarvestResponse message, if the operation succeeded, or an exception message (see 6.7)

if the operation failed.

If the Harvest attempt is successful, the response may include summary representations of the newly created or modified catalogue object(s). This response is the same as the TransactionResponse (see 7.6.4).

7.7.7 Examples

KVP encoded example:

```
http://www.myserver.com/csw/csw.cgi?request=Harvest&version="3.0.0"&sou
rce=http://www.yourserver.com/metadata.xml&resourcetype=http://www.fgdc
.gov/metadata/csdgm&resourceformat=application/xml&responsehandler=mail
to:pvretano@cubewerx.com&harvestinterval=P2W
```

XML encoded example:

```
<Harvest
   service="CSW"
   version="3.0.0"
   xmlns="http://www.opengis.net/cat/csw/3.0"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.opengis.net/cat/csw/3.0
                       ../../../csw/3.0/cswAll.xsd">
   <Source>http://www.yourserver.com/metadata.xml</Source>
   <ResourceType>http://www.fgdc.gov/metadata/csdgm</ResourceType>
   <ResourceFormat>application/xml</ResourceFormat>
   <HarvestInterval>P14D</HarvestInterval>
   <ResponseHandler> ftp://ftp.myserver.com/HarvestResponses</ResponseHandler>
</Harvest>
```

7.8 UnHarvest operation

7.8.1 Introduction

This subclause defines the optional UnHarvest operation which deletes resources from the catalogue that had been previously added using the Harvest operation (see 7.7). The UnHarvest operation is an extension of the general model (see OGC 12-168r4) introduced by this standard. The following figure defines the UML for this operation.

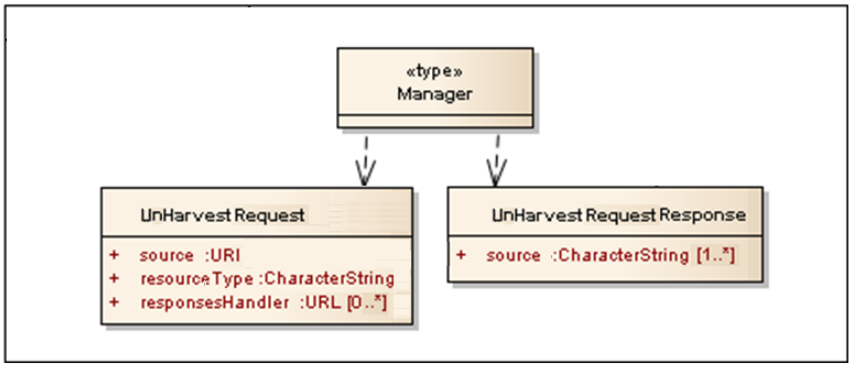

Figure 4 — UnHarvest operation

The UnHarvest operation had two modes of operation, controlled by a flag in the request. The first mode of operation is a synchronous mode in which the CSW receives an UnHarvest request from the client, processes it immediately, and sends the response to the client. The second mode of operation is asynchronous in that the server receives an UnHarvest request from the client, and sends the client an immediate acknowledgement that the request has been successfully received. The server then processes the UnHarvest request, taking as much time as is required, and when completed sends the results of the processing to the URI specified in the UnHarvest request. This latter mode of operation is included to support UnHarvest requests that may run for a period of time longer than most HTTP timeout's will allow.

7.8.2 KVP encoding

Table 27 specifies the keyword-value pair encoding for the UnHarvest operation.

NOTE To reduce the need for readers to refer to other parts of this document, the first three parameters listed below are copied from Table 7.

Table 27 — KVP encoding for UnHarvest operation request

Keyword [b]	Data type and value	Optionality and use	Parameter in general model
REQUEST	Character String Fixed value of *Harvest*, case insensitive	One (Mandatory) [a]	(none)
service	Character String Default value of "CSW"	One (Mandatory)	serviceId
version	Character String Default value of 3.0.0	One (Mandatory)	(none)
Source	URI Reference to the source from which the resource is to be harvested	One (Mandatory)	Source
ResourceType	Character String Reference to the type of resource being harvested, see Subclause7.7.4.2	One (Mandatory)	Type
ResponseHandler	URL A reference to a person or entity that the CSW should respond to when it has completed processing Harvest request asynchronously	Zero or one (Optional) If not included, process request synchronously	responseHandler

a The REQUEST parameter contains the same information as the name of the <UnHarvest> element in XML encoding.

b Parameter keywords are case insensitive for KVP encoding. Parameters values are case sensitive..

7.8.3 XML encoding

The following XML-Schema fragment defines the XML encoding for a UnHarvest operation request:

```
<xsd:element name="UnHarvest" type="csw30:UnHarvestType" id="UnHarvest"/>
<xsd:complexType name="UnHarvestType">
   <xsd:complexContent>
      <xsd:extension base="csw30:RequestBaseType">
         <xsd:sequence>
            <xsd:element ref="csw30:Source" maxOccurs="unbounded"/>
            <xsd:element name="ResourceType" type="xsd:string"/>
            <xsd:element name="ResponseHandler" type="xsd:anyURI"
                         minOccurs="0" maxOccurs="unbounded"/>
         </xsd:sequence>
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>
```

7.8.4 Parameter descriptions

7.8.4.1 Source parameter

The Source parameter is used to specify the URI reference of a metadata resource that was previously harvested using the Harvest operation (see 7.7).

Requirement-167

If the specified resource URI has not been previously harvested then the server shall raise an InvalidParameterValue exception (see Table 12).

7.8.4.2 ResourceType parameter

The ResourceType parameter is described in clause 7.7.4.2.

7.8.4.3 ResponseHandler parameter

The ResponseHandler parameter is a flag that indicates how the UnHarvest operation should be processed by a CSW server.

Requirement-168

If the ResponseHandler parameter is not present, then the UnHarvest operation shall be processed synchronously meaning that the client sends the UnHarvest request to a CSW and then waits to receive a valid UnHarvestResponse or an exception message (see 6.7).

The problem with this mode of operation is that the client may timeout waiting for the server to process the UnHarvest request.

Requirement-169

If the ResponseHandler parameter is present, the UnHarvest operation shall be processed

asynchronously. In this case, the server shall respond immediately to a client's request with an Acknowledgement message (see 7.3.4.14) that indicates that the request has been received and validated.

The UnHarvest request may then be processed using as much time as is required to complete the operation.

Requirement-170

When the asynchronous request has been completed the server shall send an UnHarvestResponse message or an exception message (see 6.7) to the URI(s) specified as the value of the ResponseHandler parameter.

This standard does not define any operations or methods to check on the intermediate processing status of an asynchronous request.

Requirement-171

The presence of multiple response handlers indicates that the server shall send the response to UnHarvest operations to multiple URI(s). This analogous to sending an email message to multiple recipients or copying the response to multiple ftp servers.

7.8.5 Response

The following XML-Schema fragment defines the UnHarvestResponse message:

```
<xsd:element name="UnHarvestResponse"
             type="csw30:UnHarvestResponseType" id="UnHarvestResponse">
   <xsd:annotation>
      <xsd:documentation xml:lang="en">
         The response to an UnHarvest request is simply a list of
         csw30:Source elements echoing what has been unharvested.
      </xsd:documentation>
   </xsd:annotation>
</xsd:element>
<xsd:complexType name="UnHarvestResponseType" id="UnHarvestResponseType">
   <xsd:sequence>
      <xsd:element ref="csw30:Source" maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>
```

The response to an UnHarvest request since echoes the source URI of the un-harvested resource indicating that the operation completed successfully.

A server can respond in one of two ways to an UnHarvest request depending on the presence or absence of the ResponseHandler parameter.

Requirement-172

If the ResponseHandler parameter is present, then the CSW server shall verify the request syntax and immediately respond to the client with an Acknowledgment message (see 7.3.4.13).

Requirement-173

If the ResponseHandler parameter is not present, the CSW server shall process the UnHarvest request immediately and respond to the waiting client with a valid UnHarvestResponse message, if the operation succeeded, or an exception message (see 6.7) if the operation failed.

7.8.6 Examples

KVP encoded example:

```
http://www.myserver.com/csw/csw.cgi?request=UnHarvest&version="3.0.0"&s
ource=http://www.yourserver.com/metadata.xml&resourcetype=http://www.fg
dc.gov/metadata/csdgm
```

XML encoded example:

```
<UnHarvest
   service="CSW"
   version="3.0.0"
    xmlns:dc="http://purl.org/dc/elements/1.1/"
   xmlns:dct="http://purl.org/dc/terms/"
   xmlns="http://www.opengis.net/cat/csw/3.0"
   xmlns:csw="http://www.opengis.net/cat/csw/3.0"
   xmlns:fes="http://www.opengis.net/fes/2.0"
   xmlns:ows="http://www.opengis.net/ows/2.0"
   xmlns:gml="http://www.opengis.net/gml/3.2"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.opengis.net/cat/csw/3.0
                        ../../../csw/3.0/cswAll.xsd
                       http://www.opengis.net/gml/3.2
                       http://schemas.opengis.net/gml/3.2.1/gml.xsd">
   <Source
resourceType="http://www.fgdc.gov/metadata/csdgm">http//www.yourserver.com/meta
data.xml</Source>
   <ResponseHandler> ftp://ftp.myserver.com/HarvestResponses</ResponseHandler>
</UnHarvest>
```

7.9 XML Schemas

The CSW abilities specified in this Clause directly and indirectly use several XML Schema Documents, included in the zip file version with this document. These XML Schema files are:

a) cswAll.xsd

b) cswCommon.xsd

c) cswGetCapabilities.xsd

d) cswGetRecords.xsd

e) cswGetRecordById.xsd

f) cswGetDomain.xsd

g) cswHarvest.xsd

h) cswTransaction.xsd

i) cswUnHarvest.xsd

j) record.xsd

k) rec-dcmes.xsd

l) rec-dcterms.xsd

These XML Schema files are posted at the URL http://schemas.opengis.net/cat/csw/3.0 for electronic access. In the event of a discrepancy between the attached and online versions of the XML Schema files, the online files are considered normative.

These new XML Schemas build on the XML Schemas defined in the Filter Encoding Implementation Specification (see OGC 09-026r1), and the OWS Common Implementation Specification (see OGC 06-121r9), and described in those documents.

Annex A
(normative)

Abstract conformance test suite

The abstract test suite for this standard can be found in OGC document 14-014r3, "OpenGIS Catalogue Services Specification – HTTP Protocol Binding – Abtract Test Suite."

Annex B
(informative)

Example CSW capabilities document

```
<?xml version="1.0" encoding="UTF-8"?>
<csw:Capabilities version="3.0.0" xmlns="http://www.opengis.net/cat/csw/3.0"
xmlns:csw="http://www.opengis.net/cat/csw/3.0"
xmlns:fes="http://www.opengis.net/fes/2.0"
xmlns:gml="http://www.opengis.net/gml/3.2"
xmlns:ows20="http://www.opengis.net/ows/2.0"
xmlns:gmd="http://www.isotc211.org/2005/gmd"
xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opengis.net/cat/csw/3.0
                        ../../../csw/3.0/cswAll.xsd
                        http://www.opengis.net/gml/3.2
                        http://schemas.opengis.net/gml/3.2.1/gml.xsd
                        http://www.w3.org/1999/xlink
                        http://www.w3.org/1999/xlink.xsd">
   <ows20:ServiceIdentification>
      <ows20:Title>Catalogue Service for Spatial Information</ows20:Title>
      <ows20:Abstract>terraCatalog 3.2 based OGC CSW 3.0 Catalogue Service for
OGC core and ISO metadata (describing geospatial services, datasets and
series)</ows20:Abstract>
      <ows20:Keywords>
         <ows20:Keyword>OGC</ows20:Keyword>
         <ows20:Keyword>CSW</ows20:Keyword>
         <ows20:Keyword>Catalog Service</ows20:Keyword>
         <ows20:Keyword>metadata</ows20:Keyword>
         <ows20:Keyword>CSW</ows20:Keyword>
         <ows20:Type
codeSpace="http://www.someGeospatialVocabulary.com">theme</ows20:Type>
      </ows20:Keywords>
      <ows20:ServiceType>CSW</ows20:ServiceType>
      <ows20:ServiceTypeVersion>3.0.0</ows20:ServiceTypeVersion>
   </ows20:ServiceIdentification>
   <ows20:ServiceProvider>
      <ows20:ProviderName>con terra GmbH</ows20:ProviderName>
      <ows20:ProviderSite xlink:type="simple"
xlink:href="http://www.conterra.de"/>
      <ows20:ServiceContact>
         <ows20:IndividualName/>
         <ows20:PositionName/>
         <ows20:ContactInfo>
            <ows20:Phone>
               <ows20:Voice>+49-251-7474-400</ows20:Voice>
               <ows20:Facsimile>+49-251-7474-100</ows20:Facsimile>
            </ows20:Phone>
            <ows20:Address>
               <ows20:DeliveryPoint>Marting-Luther-King-Weg
24</ows20:DeliveryPoint>
               <ows20:City>Muenster</ows20:City>
               <ows20:AdministrativeArea>NRW</ows20:AdministrativeArea>
               <ows20:PostalCode>48165</ows20:PostalCode>
               <ows20:Country>Germany</ows20:Country>
```

```
<ows20:ElectronicMailAddress>conterra@conterra.de</ows20:ElectronicMailAddress>
            </ows20:Address>
            <ows20:OnlineResource xlink:href="mailto:conterra@conterra.de"/>
         </ows20:ContactInfo>
      </ows20:ServiceContact>
   </ows20:ServiceProvider>
   <ows20:OperationsMetadata>
      <ows20:Operation name="GetCapabilities">
         <ows20:DCP>
            <ows20:HTTP>
               <ows20:Post
xlink:href="http://www.sdisuite.de/terraCatalog/soapService/services/CSWDiscove
ry"/>
               <ows20:Get
xlink:href="http://www.sdisuite.de/terraCatalog/soapServices/CSWStartup?"/>
            </ows20:HTTP>
         </ows20:DCP>
         <ows20:Parameter name="AcceptVersions">
            <ows20:AllowedValues>
               <ows20:Value>3.0.0</ows20:Value>
               <ows20:Value>2.0.2</ows20:Value>
            </ows20:AllowedValues>
         </ows20:Parameter>
      </ows20:Operation>
      <ows20:Operation name="GetRecords">
         <ows20:DCP>
            <ows20:HTTP>
               <ows20:Post
xlink:href="http://www.sdisuite.de/terraCatalog/soapService/services/CSWDiscove
ry"/>
            </ows20:HTTP>
         </ows20:DCP>
         <ows20:Parameter name="typeName">
            <ows20:AllowedValues>
               <ows20:Value>csw:Record</ows20:Value>
               <ows20:Value>gmd:MD_Metadata</ows20:Value>
            </ows20:AllowedValues>
         </ows20:Parameter>
         <ows20:Parameter name="outputFormat">
            <ows20:AllowedValues>
               <ows20:Value>application/xml</ows20:Value>
            </ows20:AllowedValues>
         </ows20:Parameter>
         <ows20:Parameter name="outputSchema">
            <ows20:AllowedValues>
               <ows20:Value>http://www.opengis.net/cat/csw/3.0</ows20:Value>
               <ows20:Value>http://www.isotc211.org/2005/gmd</ows20:Value>
            </ows20:AllowedValues>
         </ows20:Parameter>
         <ows20:Parameter name="ElementSetName">
            <ows20:AllowedValues>
               <ows20:Value>brief</ows20:Value>
               <ows20:Value>summary</ows20:Value>
               <ows20:Value>full</ows20:Value>
            </ows20:AllowedValues>
         </ows20:Parameter>
         <ows20:Constraint name="SupportedGMLVersions">
            <ows20:AllowedValues>
               <ows20:Value> http://www.opengis.net/gml/3.2</ows20:Value>
               <ows20:Value> http://www.opengis.net/gml</ows20:Value>
            </ows20:AllowedValues>
```

```
         </ows20:Constraint>
         <ows20:Constraint name="OpenSearchDescriptionDocument">
            <ows20:AllowedValues>

<ows20:Value>http://www.sdisuite.de/descriptionDocument.xml</ows20:Value>
            </ows20:AllowedValues>
         </ows20:Constraint>
      </ows20:Operation>
      <ows20:Operation name="GetRecordById">
         <ows20:DCP>
            <ows20:HTTP>
               <!-- This is an example of an operation that has both a
                    SOAP and XML-with-POST endpoint.  In this case the
                    two endpoints are different, however, they could
                    as well been the same URL.  If the endpoint were the
                    same it would imply that the service is able to detect
                    from the XML stream whether the request is a SOAP
                    request or a plain XML request -->
               <ows20:Post
xlink:href="http://www.sdisuite.de/terraCatalog/soapServiceURL">
                  <ows20:Constraint name="PostEncoding">
                     <ows20:AllowedValues>
                        <ows20:Value>SOAP</ows20:Value>
                     </ows20:AllowedValues>
                  </ows20:Constraint>
               </ows20:Post>
               <ows20:Post
xlink:href="http://www.sdisuite.de/terraCatalog/postServiceURL">
                  <ows20:Constraint name="PostEncoding">
                     <ows20:AllowedValues>
                        <ows20:Value>XML</ows20:Value>
                     </ows20:AllowedValues>
                  </ows20:Constraint>
               </ows20:Post>
               <ows20:Get
xlink:href="http://www.sdisuite.de/terraCatalog/soapServices/CSWStartup?"/>
            </ows20:HTTP>
         </ows20:DCP>
         <ows20:Parameter name="ElementSetName">
            <ows20:AllowedValues>
               <ows20:Value>brief</ows20:Value>
               <ows20:Value>summary</ows20:Value>
               <ows20:Value>full</ows20:Value>
            </ows20:AllowedValues>
         </ows20:Parameter>
         <ows20:Parameter name="outputSchema">
            <ows20:AllowedValues>
               <ows20:Value>http://www.opengis.net/cat/csw/2.0.2</ows20:Value>
            </ows20:AllowedValues>
         </ows20:Parameter>
      </ows20:Operation>
      <ows20:Operation name="GetDomain">
         <ows20:DCP>
            <ows20:HTTP>
               <ows20:Post
xlink:href="http://www.sdisuite.de/terraCatalog/soapService/services/CSWDiscove
ry"/>
            </ows20:HTTP>
         </ows20:DCP>
      </ows20:Operation>
      <ows20:Operation name="Transaction">
         <ows20:DCP>
```

```
            <ows20:HTTP>
               <ows20:Post
xlink:href="http://www.sdisuite.de/terraCatalog/soapService/services/CSWManager
"/>
            </ows20:HTTP>
         </ows20:DCP>
         <ows20:Constraint name="TransactionSchemas">
            <ows20:AllowedValues>
               <ows20:Value>http://www.isotc211.org/2005/gmd</ows20:Value>
            </ows20:AllowedValues>
         </ows20:Constraint>
      </ows20:Operation>
      <ows20:Operation name="Harvest">
         <ows20:DCP>
            <ows20:HTTP>
               <ows20:Post
xlink:href="http://www.sdisuite.de/terraCatalog/soapService/services/CSWManager
"/>
            </ows20:HTTP>
         </ows20:DCP>
      </ows20:Operation>
      <!-- OpenSearch support ? -->
      <ows20:Constraint name="OpenSearch">
         <ows20:AllowedValues>
            <ows20:Value>true</ows20:Value>
         </ows20:AllowedValues>
      </ows20:Constraint>
      <!-- GetCapabilities-XML support ? -->
      <ows20:Constraint name="GetCapabilities-XML">
         <ows20:AllowedValues>
            <ows20:Value>true</ows20:Value>
         </ows20:AllowedValues>
      </ows20:Constraint>
      <!-- GetRecordById-XML support ? -->
      <ows20:Constraint name="GetRecordById-XML">
         <ows20:AllowedValues>
            <ows20:Value>true</ows20:Value>
         </ows20:AllowedValues>
      </ows20:Constraint>
      <!-- GetRecords-Basic-XML support ? -->
      <ows20:Constraint name="GetRecords-Basic-XML">
         <ows20:AllowedValues>
            <ows20:Value>true</ows20:Value>
         </ows20:AllowedValues>
      </ows20:Constraint>
      <!-- GetRecords-Distributed-XML support ? -->
      <ows20:Constraint name="GetRecords-Distributed-XML">
         <ows20:AllowedValues>
            <ows20:Value>true</ows20:Value>
         </ows20:AllowedValues>
      </ows20:Constraint>
      <!-- GetRecords-Distributed-KVP support ? -->
      <ows20:Constraint name="GetRecords-Distributed-KVP">
         <ows20:AllowedValues>
            <ows20:Value>true</ows20:Value>
         </ows20:AllowedValues>
      </ows20:Constraint>
      <!-- GetRecords-Async-XML support ? -->
      <ows20:Constraint name="GetRecords-Async-XML">
         <ows20:AllowedValues>
            <ows20:Value>true</ows20:Value>
         </ows20:AllowedValues>
```

```
</ows20:Constraint>
<!-- GetRecords-Async-KVP support ? -->
<ows20:Constraint name="GetRecords-Async-KVP">
   <ows20:AllowedValues>
      <ows20:Value>true</ows20:Value>
   </ows20:AllowedValues>
</ows20:Constraint>
<!-- GetDomain-XML support ? -->
<ows20:Constraint name="GetDomain-XML">
   <ows20:AllowedValues>
      <ows20:Value>true</ows20:Value>
   </ows20:AllowedValues>
</ows20:Constraint>
<!-- GetDomain-KVP support ? -->
<ows20:Constraint name="GetDomain-KVP">
   <ows20:AllowedValues>
      <ows20:Value>true</ows20:Value>
   </ows20:AllowedValues>
</ows20:Constraint>
<!-- Harvest-Basic-XML support ? -->
<ows20:Constraint name="Harvest-Basic-XML">
   <ows20:AllowedValues>
      <ows20:Value>true</ows20:Value>
   </ows20:AllowedValues>
</ows20:Constraint>
<!-- Harvest-Basic-KVP support ? -->
<ows20:Constraint name="Harvest-Basic-KVP">
   <ows20:AllowedValues>
      <ows20:Value>true</ows20:Value>
   </ows20:AllowedValues>
</ows20:Constraint>
<!-- Harvest-Async-XML support ? -->
<ows20:Constraint name="Harvest-Async-XML">
   <ows20:AllowedValues>
      <ows20:Value>true</ows20:Value>
   </ows20:AllowedValues>
</ows20:Constraint>
<!-- Harvest-Async-KVP support ? -->
<ows20:Constraint name="Harvest-Async-KVP">
   <ows20:AllowedValues>
      <ows20:Value>true</ows20:Value>
   </ows20:AllowedValues>
</ows20:Constraint>
<!-- Harvest-Periodic-XML support ? -->
<ows20:Constraint name="Harvest-Periodic-XML">
   <ows20:AllowedValues>
      <ows20:Value>true</ows20:Value>
   </ows20:AllowedValues>
</ows20:Constraint>
<!-- Harvest-Periodic-KVP support ? -->
<ows20:Constraint name="Harvest-Periodic-KVP">
   <ows20:AllowedValues>
      <ows20:Value>true</ows20:Value>
   </ows20:AllowedValues>
</ows20:Constraint>
<!-- Filter-CQL support ? -->
<ows20:Constraint name="Filter-CQL">
   <ows20:AllowedValues>
      <ows20:Value>true</ows20:Value>
   </ows20:AllowedValues>
</ows20:Constraint>
<!-- Filter-FES support ? -->
```

```
      <ows20:Constraint name="Filter-FES">
         <ows20:AllowedValues>
            <ows20:Value>true</ows20:Value>
         </ows20:AllowedValues>
      </ows20:Constraint>
      <!-- Filter-KVP support ? -->
      <ows20:Constraint name="Filter-KVP">
         <ows20:AllowedValues>
            <ows20:Value>true</ows20:Value>
         </ows20:AllowedValues>
      </ows20:Constraint>
      <!-- CSW-Response support ? -->
      <ows20:Constraint name="CSW-Response">
         <ows20:AllowedValues>
            <ows20:Value>true</ows20:Value>
         </ows20:AllowedValues>
      </ows20:Constraint>
      <!-- ATOM-response support ? -->
      <ows20:Constraint name="ATOM-response">
         <ows20:AllowedValues>
            <ows20:Value>true</ows20:Value>
         </ows20:AllowedValues>
      </ows20:Constraint>
      <ows20:Constraint name="CoreQueryables">
         <ows20:AllowedValues>
            <ows20:Value>Title</ows20:Value>
            <ows20:Value>Subject</ows20:Value>
            <ows20:Value>Abstract</ows20:Value>
            <ows20:Value>Modified</ows20:Value>
            <ows20:Value>Type</ows20:Value>
            <ows20:Value>Format</ows20:Value>
            <ows20:Value>Identifier</ows20:Value>
            <ows20:Value>Association</ows20:Value>
            <ows20:Value>BoundingBox</ows20:Value>
         </ows20:AllowedValues>
      </ows20:Constraint>
      <ows20:Constraint name="CoreSortables">
         <ows20:AllowedValues>
            <ows20:Value>Title</ows20:Value>
            <ows20:Value>Type</ows20:Value>
            <ows20:Value>Modified</ows20:Value>
         </ows20:AllowedValues>
      </ows20:Constraint>
      <ows20:Constraint name="DefaultSortingAlgorithm">
         <ows20:AllowedValues>
<ows20:Value>http://www.sdisuite.de/terraCatalog/documentation/descriprionOfSor
talgorithm.html</ows20:Value>
         </ows20:AllowedValues>
      </ows20:Constraint>
      <!-- enter the supported federated catalogues for distributed search as
values below -->
      <ows20:Constraint name="FedeartedCatalogues">
         <ows20:AllowedValues>
            <ows20:Value/>
         </ows20:AllowedValues>
      </ows20:Constraint>
      <ows20:Constraint name="WSDL">
         <ows20:AllowedValues>

<ows20:Value>http://www.sdisuite.de/terraCatalog/soapService/services/CSWDiscov
ery?wsdl</ows20:Value>
         </ows20:AllowedValues>
```

```
      </ows20:Constraint>
      <ows20:Constraint name="OpenSearch">
         <ows20:AllowedValues>
            <ows20:Value>http://www.sdisuite.de/terraCatalog</ows20:Value>
         </ows20:AllowedValues>
      </ows20:Constraint>
   </ows20:OperationsMetadata>
   <fes:Filter_Capabilities xmlns:ows11="http://www.opengis.net/ows/1.1">
      <fes:Conformance>
         <fes:Constraint name="">
            <ows11:AllowedValues>
               <ows11:Value/>
            </ows11:AllowedValues>
         </fes:Constraint>
      </fes:Conformance>
      <fes:Spatial_Capabilities>
         <fes:GeometryOperands>
            <fes:GeometryOperand name="gml:Envelope"/>
            <fes:GeometryOperand name="gml:Point"/>
            <fes:GeometryOperand name="gml:LineString"/>
            <fes:GeometryOperand name="gml:Polygon"/>
         </fes:GeometryOperands>
         <fes:SpatialOperators>
            <fes:SpatialOperator name="BBOX"/>
            <fes:SpatialOperator name="Beyond"/>
            <fes:SpatialOperator name="Contains"/>
            <fes:SpatialOperator name="Crosses"/>
            <fes:SpatialOperator name="Disjoint"/>
            <fes:SpatialOperator name="DWithin"/>
            <fes:SpatialOperator name="Equals"/>
            <fes:SpatialOperator name="Intersects"/>
            <fes:SpatialOperator name="Overlaps"/>
            <fes:SpatialOperator name="Touches"/>
            <fes:SpatialOperator name="Within"/>
         </fes:SpatialOperators>
      </fes:Spatial_Capabilities>
      <fes:Extended_Capabilities>
         <fes:AdditionalOperators>
            <fes:Operator name="Between"/>
            <fes:Operator name="EqualTo"/>
            <fes:Operator name="GreaterThan"/>
            <fes:Operator name="GreaterThanEqualTo"/>
            <fes:Operator name="LessThan"/>
            <fes:Operator name="LessThanEqualTo"/>
            <fes:Operator name="Like"/>
            <fes:Operator name="NotEqualTo"/>
            <fes:Operator name="NullCheck"/>
         </fes:AdditionalOperators>
      </fes:Extended_Capabilities>
   </fes:Filter_Capabilities>
</csw:Capabilities>
```

Annex C: Revision History

The revision history for 2.0 and earlier versions of the OGC Catalogue Services specification is provided in Annex B of the general model (see OGC 12-168). The revision history for this HTTP protocol part of the OGC Catalogue Services specification is:

Date	Release	Author	Paragraph(s) modified	Description
2007-05-25	3.0.0	Whiteside	All	First draft, majority copied from 2.0.2
2008-10-28	3.0.0	Voges	6.8.2, 6.8.3, 6.8.4.13, 6.8.5, 6.8.6, 6.9.1, table 15,16	Improvement of distributed search: Functional extensions to the discovery request and response messages (which define elements that allow for the retrieval and comprehension of a distributed result set).
2008-11-12	3.0.0	Voges	6.8.2	To have a consistent API, both the KVP and XML encodings MUST support the equivalent parameters for the same operation. “constraint version” was REQUIRED by the GetRecords request’s XML schema, but in the KVP encoding it was not mentioned. (see CR-06-112).
2008-11-14	3.0.0	Voges	6.8.2, 6.8.4.2	Made maxRecords a union to number and the string 'unlimited' indicating that all records shall be returned.
2008-12-05	3.0.0	Voges	6.8.2, 6.12.2	Clarification on usage of qualified names in KVP requests
2008-12-11	3.0.0	Voges	6.8.2, 6.8.4.12	Some clarifications for constraint, constraintlanguage and sortBy parameters.
2009-01-06	3.0.0	Voges	6.8.4.2, 6.9.2	Some clarifications on the usage of the requestId parameter in getRecords requests and inclusion of the service and version parameters within the KVP encoding of the GetRecordById
2010-05.11	3.0.0	Voges	6.8.3, 6.11.3	Elementname now of type xsd:string instead of xsd:Qname (see CR 08-177, Harmonized UpdateType, InsertType, DeleteType concerning the typeName attribute, AbstractRecordType has now optional attribute “deleted”, getRecordByID response shall be the raw response in its original format, conformant with the outputFormat and outputSchema. GetRecordByIdResponse deleted. Changed different dc:Record schema snippets and examples

Date	Release	Author	Paragraph(s) modified	Description
2010-05-31	3.0.0	Voges	various	Actualized all lots of XML(-schema) snippets
2010-08-25	3.0.0	Voges	Table 5, 6.2.5.3.2	Some clarifications on the usage of core queryables, including example
2010-08-31	3.0.0	Voges	6.5.4	Added a section how to reference the supported GML versions
2010-10-06	3.0.0	Voges	6.5.4, 6.3.2 6.3.7	Section 6.5.4 defines now that no GML version is mandatory. Aligned Exception reporting and SOAP Exception handling with OWS Common 2.0.
2011-04-07	3.0.0	Voges	6.9	As we switched getRecordById to return the "raw" metadata xml (instead getRecordyById response) on a getRecordById request -> it becomes impossible to request more than one metadata-object by id: otherwise a list of metadata elements without a common root node would be returned which is not XML conformant.
2012-02-22	3.0.0	Voges	all	Starting defining Requirement Classes, checked alignment with OWS Common 2.0

Date	Release	Author	Paragraph(s) modified	Description
2012-11-02	3.0.0	Vretanos	all	GetDomain operation was made more general being allowing it to be used to interrogate any request parameter and information model component. Clarifications in the standard about supporting other output formats. his was always possible but not as clearly explained as it should have been. The CONSTRAINTLANGUAGE parameter is now of type anyURI and the following identifiers have been defined: FILTER -> http://www.opengis.net/fes/2.0 CQL -> http://www.opengis.net/csw/3.0/cql Added a set of conformance classes with requirements - this includes conformance classes for handle async requests Remove resultType parameter since there are other means for performing resultType=hits (i.e. maxRecords=0), etc. Refactor schemas so to follow common file naming practices. Synchronized schema fragments in the document with the refactored schemas. Fixed a number of inconsistencies between the XML and KVP encodings of operations. Support added for HTTP/POST with attachments for Harvesting of resources that cannot be referenced via URL. Support added for an UnHarvest operation. - this is complimentary operation for Harvest
2015-04-21	3.0.0	Reed	Numerous	Edits and comments related to preparing the document for publication,

Date	Release	Author	Paragraph(s) modified	Description
2012-12-14	3.0.0	Voges	all	- now possible to advertise the valid query and sortable terms in the Capabilities, - updated sample Capabilities: defined, if there is a specific sorting algorithm for GetRecords responses, how to express this in the Capabilities - updated and reformatted all xml-samples in the document - Added queryable TemporalExtent (Type gml:TimePeriod - must support minimally operator TVOverlaps) and TemporalExtent in csw:Record response (newly defined type Type csw30:TemporalExtentType) - should be discussed. - WSDL updates - Reviewed the whole document (without specific focus on transaction interface)
2013-01-02	3.0.0	Vretanos	Clause 6	- update KVP encoding for filter to allow better allignement with OpenSearch - move the CONSTRAINT, CONSTRAINT_LANGAUGE and CONSTRAINT_VERSION parameters into their own conformance class called Filter-FES-KVP-Advanced - add a discussion about the requirements for OpenSearch - split clause 6 into two clauses - reorder clause 6 to better present the material - renumber Tables - cross check references - update terms and definitions
2013-01-08	3.0.0	Voges	7.9, Annex B	Updated sections on xml schemata, updated capabilities example, some reformattings, updated xml schemata.
2013-04-24	3.0.0	Voges	Diverse	Dropped „Implements the Filter-CQL conformance class“ for OpenSearch ConformanceClass, added MimeType “application/soap+xml” for SOAP bindings, some clarifications on the usage of Atom for GetRecordById

Date	Release	Author	Paragraph(s) modified	Description
2014-09-02	3.0.0	Vretanos	Diverse	Did updates regarding solutions to RFC Comments (see separate comments document on public comment period)
2014-10-01	3.0.0	Voges	Diverse	Mapping OGC_Service.GetResourceById to GetRecordById, changed OSDD example, replaced operations "present" and "search" by “GetResourceById” and “query”, other minor and editorial changes
2014-10-20	3.0.0	Vretanos	Diverse	Different fixes
2014-11-25	3.0.0	Carl Reed	Diverse	Some minor edits
2014-12-02	3.0.0	Bigagli	Diverse	URI check, minor editorial
2015-01-09	3.0.0	Voges	ATS	Updated Reference to ATS
2015-07-09	3.0.0	Bigagli	Diverse	Revision comments from TB11
2016-02-09	3.0.0	Simmons	All	Edits and pref for publication