Open Geospatial Consortium Inc.

Submission Date: 2014-12-09

Approval Date: 2016-02-23

Publication Date: 2016-06-10

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

URL for this OGC® document: http://docs.opengeospatial.org/is/14-014r3/14-014r3.html

Reference number of this document: 14-014r3

Version 3.0

Category: OGC® Implementation Specification

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

OGC® Catalogue Services Specification - HTTP protocol binding - Abstract Test Suite

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/14-014r3/14-014r3.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 Specification
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

i. Abstract

See OGC 12-176r7 -- OGC® Catalogue Services Specification - HTTP Protocol Binding.

ii. Keywords

See OGC 12-176r7 -- OGC® Catalogue Services Specification - HTTP Protocol Binding.

iii. Preface

See OGC 12-176r7 -- OGC® Catalogue Services Specification - HTTP Protocol Binding.

iv. Document terms and definitions

See OGC 12-176r7 -- OGC® Catalogue Services Specification - HTTP Protocol Binding.

v. Submitting organizations

The See OGC 12-176r7 -- OGC® Catalogue Services Specification - HTTP Protocol Binding.

vi. Submitters

See OGC 12-176r7 -- OGC® Catalogue Services Specification - HTTP Protocol Binding.

1 Introduction

This document defines the abstract test suite for the OGC® Catalogue Services Specification - HTTP protocol binding - Abstract Test Suite (see OGC 12-176r7).

OGC 12-176r7 defines a set of conformance classes each of which has a similarly named requirements class. This document lists each requirements class from OGC 12-176r7 and then describes the set of abstract tests that must be passed to claim conformance to the requirements class and the corresponding conformance class.

Every implementation of OGC 12-176r7 must pass all the tests for the Basic-Catalogue requirements class. All other requirements classes are optional.

Each test in this abstract test suite tests a corresponding requirement from OGC 12-176r7 that has the same sequence number as the test; for example, Test-001 tests Requirement-001 from OGC 12-176r7.

2 Basic-Catalogue conformance class

2.1 Test-001

a) Requirement: *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)

b) Test purpose: check that the server correctly sets the Content-Type HTTP header.

c) Test method: In response to a series of valid CSW requests verify that the value of the Content-Type header correctly identifies the media type of the body of the response.

2.2 Test-002

a) Requirement: 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.

b) Test purpose: check that the server correctly handles CSW request parameters whose function mirrors that of HTTP headers (e.g. outputFormat and Accept)

c) Test method: Pick one or more CSW request parameters that have analogous HTTP headers (e.g. outputFormat and Accept). Execute CSW requests where both the request parameter and the HTTP header are set to difference values and verify that the server uses the value of the request parameter.

2.3 Test-003

a) Requirement: 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.

a) Test purpose: In the absence of a CSW request parameter (e.g. outputFormat), verify that the server uses the value from the analogous HTTP header (e.g. Accept).

b) Test method: Pick one or more CSW request parameters that have analogous HTTP headers (e.g. outputFormat and Accept). Set the value of the HTTP header and execute a CSW request omitting the analogous CSW request parameter. Verify by inspecting the response that the server correctly read and used the value from the HTTP header.

2.4 Test-004

a) Requirement: 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.

b) Test purpose: If a CSW parameter and its analogous HTTP header are both present in a CSW request, verify that their values agree.

c) Test method: Pick one or more CSW request parameters that have analogous HTTP headers (e.g. outputFormat and Accept). Execute CSW requests where both the request parameter and the HTTP header are set to the same value and verify that the server generates a valid request. Execute CSW requests where the value of the request parameter and the value of its analogous HTTP header are set to difference values and verify that the server generates an InvalidParameterValue exception.

2.5 Test-005

a) Requirement: 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.

b) Test purpose: To verify that in the absence of an optional request parameter (with a defined default value) and the absence of its analogous HTTP header, the server correctly uses the defined default value for the request parameter.

c) Test method: Pick one or more CSW parameters with defined default values that also have analogous HTTP headers (e.g. outputFormat and Accept). Execute a set of requests that omit both the request parameter and the analogous HTTP header. Verify that the server is using the defined default value for the parameter by inspecting the response.

2.6 Test-006

a) Requirement: 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.

b) Test purpose: Verify that when the base URL of a server is accessed and the HTTP Accept header is not set, the server responds with an OGC capabilities document encoded in XML.

c) Test method: Resolve the base URL of a server without setting the HTTP Accept header and verify that the server responses with an XML-encoded OGC capabilities document.

2.7 Test-007

a) Requirement: 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.

b) Test purpose: Verify that when the base URL of a server is accessed and the HTTP Accept header is set to "text/xml" or "application/xml", the server responds with an OGC capabilities document encoded in XML.

c) Test method: Set the value of the HTTP Accept header to "text/xml" or "application/xml". Resolve the base URL of the server and verify that it returns a valid OGC capabilities document encoded in XML.

2.8 Test-008

a) Requirement: 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 10-032r2).

b) Test purpose: To verify that accessing the base URL to server results in an OpenSearch description document when the server declares that it supports the OpenSearch conformance class and the value of the HTTP Accept header is set to "application/opensearchdescription+xml".

c) Test method: Verify that the server implements the OpenSearch conformance class. Set the value of the HTTP Accept header to "application/opensearchdescription+xml" and resolve the base URL of the server. Verify that the response is a valid OpenSearch description document.

2.9 Test-009

a) Requirement: Servers shall, in their capabilities document, indicate the specific request encodings that can be processed using the HTTP POST method (see Table 17).

b) Test purpose: To verify that a server correctly identifies the request encodings that it can accept using the HTTP POST method.

c) Test method: Execute a GetCapabilities request. Verify that a service constraint named "PostEncoding" is present in the response and verify that the list of values for the constraint includes one of more of "SOAP", "XML" and/or "KVP".

d) Reference: 7.1.5

2.10 Test-010

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

b) Test purpose: To verify that all requests, except GetCapabilities, support the mandatory parameter service, request and version.

c) Test method: Execute a GetCapabilities request to determine the set of operation – in addition to GetCapabilities – that the server support. Execute each operation omitting one or more of the parameters service, request and version and verify that the server responds with an exception.

2.11 Test-011

a) Requirement: For KVP encoding, parameter names shall be treated as being case insensitive.

b) Test purpose: To verify that the server handles KVP-encoded parameter names in a case-insensitive manner.

c) Test method: Execute a GetCapabilities request to determine which set of KVP-encoded operations the server implements. For that set of operations execute requests where the case of the request parameter names is varied and verify that the server recognizes the parameters.

2.12 Test-012

a) Requirement: For KVP encoding, parameter values shall be treated as being case sensitive.

b) Test purpose: To verify that parameter values for KVP-encoded requests are handled in a case sensitive manner.

c) Test method: Execute a GetCapabilities request to determine which set of KVP-encoded operations the server implements. For that set of operations determine the set of request parameters than accept text values. Execute a set of requests where the case of the values of those parameters is varies and verify that the server responds with an exception message.

2.13 Test-013

a) Requirement: The “service” parameter shall be specified for all CSW requests.

b) Test purpose:

c) Test method:

2.14 Test-014

a) Requirement: Servers that implement this international standard shall set the value of the version parameter to 3.0.0.

b) Test purpose: To verify that the server recognizes version 3.0.0 as a valid CSW version.

c) Test method: Execute a GetCapabilities request and verify that a parameter domain is defined for the “version” parameter and that the list of values includes the value “3.0.0”. Execute a set of requests where the version is set to “3.0.0” and verify that a valid response is generated.

2.15 Test-035

a) Requirement: 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.

b) Test purpose: Verify that the server generates an exception when processing an unrecognized or invalid request.

c) Test method: Get a capabilities document from the server and determine the set of operations offered. Execute requests from that set that are purposely incorrect and verify that the server generated an exception. Also execute requests that not in the set of implemented operations, or made-up requests, and verify that the server generates an XML-encoded exception message.

2.16 Test-036

a) Requirement: 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).

b) Test purpose: Verify that exception messages validate against the schemas defined in OWS Common.

c) Test method: Cause the server to generate an exception and verify that the response validates against the schemas defined in clause 8 of the OWS Common Implementation specification.

2.17 Test-037

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

b) Test purpose: To verify that the server generates the correct exception codes for specific errors.

c) Test method: Formulate a set of requests that should generate the exception codes in Table 27 of OGC 06-121r9. Execute those requests and verify that the exception responses indicate the correct exception codes.

2.18 Test-038

a) Requirement: In addition to the exception codes defined in OGC 06-121r9, server that implement this standard shall also implement the exceptions defined in Table 10.

b) Test purpose: To verify that the server generates the correct exception codes for specific errors.

c) Test method: Formulate a set of requests that should generate the exception codes found in Table 10 of this standard. Execute those requests and verify that the exception responses indicate the correct exception codes.

2.19 Test-039

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

b) Test purpose: To verify that exception codes are indicated along with exception messages.

c) Test method: Formulate a set of requests that are invalid and should generate an exception response. Execute those requests and inspect the exception responses to verify that all exception messages are accompanied by an exception code. To the extent possible, verify that the correct code is being used.

2.20 Test-040

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

b) Test purpose: Verify that the server sets the HTTP status code in its response to a request.

c) Test method: Formulate a set of valid and invalid requests. Execute those requests and inspect the server's response to verify that the HTTP status code is being set.

2.21 Test-041

a) Requirement: 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.

b) Test purpose: Verify that the HTTP status code is set to 200 when a request successfully executes and that the response is valid relative to the request.

c) Test method: Formulate a set of valid CSW requests. Execute those requests and verify that the HTTP status code is set to 200. Further verify that the responses are valid as defined in this standard.

2.22 Test-042

a) Requirement: 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 11.

b) Test purpose: Verify that the serve sets the HTTP status code appropriately when an exception is encountered.

c) Test method: Formulate a set of invalid requests that should generate exceptions with HTTP status codes as defined in Table 11. Execute those requests and verify that the responses body is as described in clause 6.7 and the HTTP status codes are set as defined in Table 11.

2.23 Test-043

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

b) Test purpose: Verify that the server can response to a KVP-encoded GetCapabilities request using the HTTP GET method.

c) Test method: Formulate a valid KVP-encoded GetCapabilities request. Execute the request and verify that the server responds with a valid capabilities document as described in this standard.

2.24 Test-044

a) Requirement: 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.

b) Test purpose: Verify that the server can generate full or partial capabilities document depending of the value of the Sections parameters.

c) Test method: Formulate a set of GetCapabilities requests where the value of the Sections parameter is set to one or more of “ServiceIdentification”, “ServiceProvider”, “OperationsMetadata” and/or “Filter_Capabilities”. Inspect the response and verify that the requested sections are present.

2.25 Test-045

a) Requirement: If the Sections parameter of the GetCapabilities request is not specified, then all sections of a capabilities document listed in Table 12 shall be presented in the response.

b) Test purpose: Verify that a complete capabilities document is generated when the Sections parameter is omitted from a GetCapabilities request.

c) Test method: Execute a GetCapabilities request, omitting the Sections parameter. Inspect the response and verify that all sections, as defined in this standard, are present in the capabilities document.

2.26 Test-046

a) Requirement: 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.

b) Test purpose: Verify that all operation listed in the OperationsMetadata sections are implemented.

c) Test method: Execute a GetCapabilities request and inspect the response to obtain the list of supported operations. For each listed operation formulate and execute a valid request. In each case, verify that the server generated a valid response.

2.27 Test-047

a) Requirement: Any additional Parameter and Constraint elements, not defined in this standard, that a server introduces in its capabilities shall be safely ignorable by compliant clients.

b) Test purpose: Verify that any addition Parameter and Constraint elements listed in a servers capabilities document can be safely ignored.

c) Test method: Execute a GetCapabilities request. Inspect the response and identify all Parameter and Constraint elements not defined in this standard. Verify that they do not define information that is necessary for the normal operation of the server.

2.28 Test-048

a) Requirement: All the service constraints listed in Table 17 – 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.

b) Test purpose: Verify that the server correctly advertises that set of conformance classes it implements.

c) Test method: Execute a GetCapabilities request. Inspect the response and verify that all the service constraints listed in Table 17 are present in the capabilities document and that their values are set to either "TRUE" or "FALSE".

2.29 Test-063

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

b) Test purpose: Verify that the server understands how to the NAMESPACE parameter in a KVP-encoded request.

c) Test method: Formulate a CSW request that requires the use of the NAMESPACE parameter to bind a prefix to a namespace URL. Execute the request and verify that the server correctly parsed the value of the NAMESPACE parameter.

2.30 Test-064

a) Requirement: The value of the NAMESPACE parameter shall be a comma-separated list of lists delimited by parentheses.

b) Test purpose: Verify that the server correctly parses the value of the NAMESPACE parameter when more than one prefix is being bound to a namespace URL.

c) Test method: Formulate a CSW request that requires the use of the NAMESPACE parameter and binds several prefixes to namespace URLs. Execute the request and verify that the server correctly parsed the value of the NAMESPACE parameter.

2.31 Test-065

a) Requirement: 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).

b) Test purpose: Verify that the server correctly parses commas not being used as list separators in the value of the NAMESPACE parameter.

c) Test method: Formulate a CSW request that requires the use of the NAMESPACE parameter and who's value includes commas not being used as list separators. Verify that the server correctly parses the value of the NAMESPACE parameter.

2.32 Test-066

a) Requirement: 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.

b) Test purpose: Verify that the server correctly parses the value of NAMESPACE parameter when the value include parentheses not being used as sub-list delimiters.

c) Test method: Formulate a CSW request that required the use of the NAMESPACE parameter and whose value include parentheses not being used as sub-list delimiters. Verify that the server correctly parses the value of the NAMESPACE parameter.

2.33 Test-067

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

b) Test purpose: Verify that the server correctly parses the value of the NAMESPACE parameter when a default namespace (i.e. no prefix) is being bound.

c) Test method: Formulate a CSW request that uses the NAMESPACE parameter to bind a default namespaces. Verify that the server correctly parses the value of the NAMESPACE parameter.

2.34 Test-068

a) Requirement: 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.

b) Test purpose: Verify that the server handled embedded commas as described in OGC06-121r9, clause 11.5.3.

c) Test method: Verify that the server passes the test for Requirement-065.

2.35 Test-073

a) Requirement: 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.

b) Test purpose: Verify that the server recognizes valid MIME types as the value of the outputFormat parameter. Verify that the server uses the default value of application/xml is the outputFormat parameter is not specified. Verify that the server understands the application/soap+xml MIME type if the server declares that is can handle SOAP messages (see X.X).

c) Test method: Formulate a GetRecords request that uses the outputFormat parameter but specified in invalid MIME type as its values. Execute the request

and verify that the server generates an exception. Formulate a GetRecords request that does not specify the outputFormat parameter. Execute the request and verify that the response is a valid XML document. If the server declared support for SOAP, formulate a GetRecords request where the outputFormat is set to application/soap+xml. Execute the request and verify that the response is a valid SOAP document.

2.36 Test-074

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

b) Test purpose: Verify that the server support XML as an output format.

c) Test method: Get a capabilities document from the server and verify that application/xml is listed as a valid value in the value domain for the outputFormat parameter. Formulate one or more requests where the outputFormat is set to application/xml and verify that the response document(s) is/are valid XML document(s).

2.37 Test-075

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

b) Test purpose: Verify that the server advertises the list of supported output formats in its capabilities document.

c) Test method: Get a capabilities document from the server. Inspect the document and verify that a parameter domain element (see X.X) appears in the document listing all the supported output format values.

2.38 Test-076

a) Requirement: 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.

b) Test purpose: Verify that the server generated a valid XML document that also uses xsi:schemaLocation attribute to provide the necessary information to automatically validate the document.

c) Test method: Formulate a GetRecords request that uses the outputFormat parameter and sets its value to application/xml. Execute the request and verify that a valid XML document is generated. Inspect the response document and verify that the xsi:schemaLocation attribute is specified and includes schema information necessary to automatically validate the response.

2.39 Test-077

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

b) Test purpose: Verify that the server uses the value of the outputSchema parameter to indicate the schema of the response to a GetRecords request.

c) Test method: Get a capabilities document from the server and determine the like of outputFormat values that the server supports. Formulate a GetRecords request using the outputFormat parameter and setting its value to one of the advertised values. Execute the request and verify that the response schema matches the requested output schema.

2.40 Test-078

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

b) Test purpose: Verify that one of the supported output schemas is the OGC core schema.

c) Test method: Get a capabilities document from the server. Inspect the document and determine the value domain for the outputSchema parameter for the GetRecords request. Verify that one of accepted values for the outputSchema parameter is "http://www.opengis.net/cat/csw/3.0".

2.41 Test-081

a) Requirement: 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.

b) Test purpose: Verify that the server advertises the list of supported outputSchema values using the "Parameter" element as defined in OGC 06-121r3.

c) Test method: Get a capabilities document from the server. Inspect the capabilities document and determine if there is, for the GetRecords operation, a "Parameter" element with the name "outputSchema".

2.42 Test-082

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

b) Test purpose: Verify that the server understands and uses the value of the startPosition parameter correctly.

c) Test method: Formulate a GetRecords request without using the startPosition parameter. Execute the request and take note of the response records. Modify the request and set the value of the startPosition parameter to something greater than

1 but less than the number of records returned by the original request. Execute this new request and verify that the server starts presenting records at the specified start position.

2.43 Test-083

a) Requirement: The default value of the startPosition parameter shall be 1.

b) Test purpose: Verify that the default value for the startPosition parameter is 1.

c) Test method: Formulate a GetRecords request without specifying the startPosition parameter and whose result set is known. Execute the request and verify that the server start presenting records from the first record in the known result set.

2.44 Test-084

a) Requirement: 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.

b) Test purpose: Verify that the server understands and uses the value of the maxRecords parameter correctly.

c) Test method: Formulate a GetRecords request using the maxRecords parameter and whose result set is known. Ensure that the maxRecords value is greater than 1 but less that the number of records of the known result set. Execute the request and verify that the server returns "maxRecords" records from the result set.

2.45 Test-085

a) Requirement: The default value for the maxRecords parameter shall be 10.

b) Test purpose: Verify that the server enforces that maxRecords default value of 10 records.

c) Test method: Formulate a GetRecords request that does not use the maxRecords parameter and whose result set is known and is greater than 10. Execute the request and verify that the response contains no more than 10 records.

2.46 Test-086

a) Requirement: 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.

b) Test purpose: Verify that the server can correctly respond to a GetRecords request that only returns the number of records the query will return.

c) Test method: Formulate a GetRecords request that uses the maxRecords parameter with its value set to zero and whose result set is known. Execute the request and verify that the server responds with an empty SearchResults element.

Inspect the response and verify that the record count presented in the response is correct.

2.47 Test-087

2.48 Test-088

a) Requirement: 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.

b) Test purpose: Verify that the server correctly understands and uses the typeNames parameter.

c) Test method: Get a capabilities document from the server. Inspect the capabilities document and determine the value domain for the typeNames parameter for the GetRecords request. Formulate one or more GetRecords requests using the advertised values of the typeNames parameter a verify that the server generated a valid response. Formulate a GetRecords request that deliberates uses a typeNames value that is not advertised in the server's capabilities document. Execute the request and verify that the server generates an exception message.

2.49 Test-089

a) Requirement: In such cases the application profile shall describe how multiple typeNames values should be processed.

b) Not testable in this standard.

2.50 Test-090

a) Requirement: 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.

b) Test purpose: Verify that the server correctly processes the ElementName parameter.

c) Test method: Get a capabilities document from the server and determine the list of supported output schemas. Formulate a GetRecords request that uses a recognized output schema and uses the ElementName parameter to identifiy one or more elements from the information model of the catalogue that should appear in the response. Execute the request and inspect the response to verify that the requested elements appear in the response.

2.51 Test-091

a) Requirement: 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 Subclause6.7.

b) Test purpose: Verify that the server correctly handles GetRecords requests that identify invalid elements to be presented in the response.

c) Test method: Get a capabilities document from the server and determine the list of supported output schemas. Formulate a GetRecords request that uses a recognized output schema and uses the ElementName parameter. The value of the ElementName parameter should be an element name that is not part of the requested output schema. Verify that the server responds with an exception messages that uses the exception code InvalidParameterValue.

2.52 Test-092

a) Requirement: 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 xsi:schemaLocation attribute.

b) Test purpose: Verify that, for XML-encoded responses, the server uses the xsi:schemaLocation attribute to reference schemas that can be used to validate the document.

c) Test method: Formulate a GetRecords request that uses the outputFormat parameter with the value set to "application/xml". Execute the request and verify that the server generates a valid response. Inspect the response document and verify that the xsi:schemaLocation attribute used to reference one or more schemas that can be used to validate the document.

2.53 Test-093

a) Requirement: 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.

b) Test purpose: Verify that, for XML-encoded responses, the server augments the set of information model elements requested such that a valid XML document can be generated.

c) Test method: Get a capabilities document from the server and determine the list of supported output schemas and output formats. Identify an output schemas that can be generated using an XML representation. Formulate a GetRecords request that use this output schema and also uses the ElementName parameter to identifiy one or more elements from the information model of the catalogue to be presented in the reponse. Ensure that the number of elements identified using the ElementName parameter are fewer than would be necessary to generate a valid XML-encoded response. Execute the request and verify that a valid XML document is generated. The fact that the response is valid inidicates that the server has correctly augmented the requested set of information model elements.

2.54 Test-095

a) Requirement: 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.

b) Test purpose: Verify that the server correctly processed the ElementSetName parameter.

c) Test method: Using the OGC core output schema, formulate GetRecords requests that request the brief, summary and full element sets using the ElementSetName parameter. Execute each request and verify that the server generates the correct set of elements in each case.

2.55 Test-096

a) Requirement: 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.

b) Test purpose: Verify that the server correctly processes the typeNames parameter on the ElementSetName parameter.

c) Test method: Formulate a GetRecords request that uses the ElementSetName parameter and also uses the typeNames parameter. Set the value of the typeNames parameter to be the same as one of the queried type names. Execute the request and verify that the server generates a valid response. Modify the GetRecords request to the value of the typeNames on the ElementSetName parameter to some typeNames not being queries. Verify that the server generates an exception response.

2.56 Test-097

a) Requirement: 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.

b) Test purpose: Verify that the named set of elements for all entities being queried is presented in the response if the typeNames parameter is not specified for the ElementSetName parameter.

c) Test method: Formulate a GetRecords request that uses the ElementSetName parameter to request a named set of response elements (e.g. brief). Do not specify a value for the typeNames parameter on the ElementSetName parameter. Execute the request and verify that the requested set of elements (e.g. brief) is presented for all queried entities.

2.57 Test-098

a) Requirement: 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 Subclause6.7.

b) Test purpose: Verify that the correct exception response is generated when the value of the typeNames parameter on the ElementSetName parameter does not match one of the entities being queried by the request.

c) Test method: Verify that the server passes the test for Requirement-096. Inspect the exception response generated for the test for Requirement-096 and verify that the exception code is InvalidParameterValue.

2.58 Test-099

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

b) Test purpose: Verify that the ElementSetName parameter and ElementName parameter are mutually exclusive.

c) Test method: Formulate a GetRecords request that specified both the ElementSetName and ElementName parameters. Execute the request and verify that the server generates an exception response.

2.59 Test-100

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

b) ONLY TESTABLE IN A PROFILE.

2.60 Test-101

a)

2.61 Test-102

a) Requirement: Servers shall advertise in their capabilities documents, via the ows:Parameter element (see 7.1.5, Table 15), the list of element set names that the server understands. That list shall also include the set names “brief”, “summary” and “full”.

b) Test purpose: Verify that the server advertises the list of name elements sets in its capabilities document.

c) Test method: Get a capabilities document from the server. Inspect the document and verify that a parameter constraint is defined for the ElementSetName parameter of the GetRecords request. Verify that the list of supported values for the ElementSetName parameter includes the values “brief”, “summary” and “full”. Additional names element sets may also be specified.

2.62 Test-123

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

b) Only testable in a profile of this standard.

2.63 Test-124

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

b) Test purpose: Verify that the server generates one summary records in response to a GetRecordById request that does not specify the ElementSetName parameter.

c) Test method: Formulate a GetRecordById request that does not specify the ElementSetName parameter. Execute the request and verify that the server responds with one summary record.

2.64 Test-125

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

b) Test purpose: Verify that the supported element set names are listed in the server's capabilities document.

c) Test method: Get the server's capabilities document. Inspect the document and verify that an operation parameters names "ElementSetName" is specified for the GetRecordById operation. Verify that at least the values "brief", "summary" and "full" are listed as valid values for the ElementSetName parameter. Other values are also allowed.

2.65 Test-126

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

b) Test purpose: Verify that the server correctly processes record identifiers.

c) Test method: Formulate a GetRecordById request and use the Id parameter to specified the identifier of the record to be retrieved. Execute the request and verify that the record presented in the response is the one requested.

2.66 Test-127

a) Requirement: If the identifier specified in the operation is invalid, then the operation shall fail and an InvalidParameterValue exception message should be returned as described in Subclause 6.7.

b) Test purpose: Verify that the server correctly processed record identifiers.

c) Test method: Formulate a GetRecordById request and use the Id parameter to specify a bogus record identifier. Execute the request and verify that the server responds with and exception message indicating that this is an InvalidParameterValue exception.

2.67 Test-128

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

b) Test purpose: Verify that the server verifies that the value of the outputFormat parameter is a valid MIME type.

c) Test method: Formulate a GetRecordById request that also uses the outputFormat parameter. Set the value of the outputFormat to be string that would be considered an invalid MIME type. Execute the request and verify that the server responds with an exception message.

2.68 Test-129

a) Requirement: 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.

b) Test purpose: Verify that default value for the outputFormat parameter for the GetRecordById request is an XML document.

c) Test method: Formulate a GetRecordById request that does not also use the outputFormat parameter. Execute the request and verify that the response is a valid XML document.

2.69 Test-130

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

b) Test purpose: Verify that the server advertises the list of valid output format values for the GetRecordById operation.

c) Test method: Get the server's capabilities document. Inspect the document and verify that a parameter constrain is specified for the GetRecordById operation that lists the set of supported output formats.

2.70 Test-131

a) Requirement: 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.

b) Test purpose: Verify that the server uses the xsi:schemaLocation attribute to provide information required to validate a response.

c) Test method: Formulate a GetRecordById request where the output format is set to “application/xml”. Execute the request and verify that the responses contains an xsi:schemaLocation attribute providing information required to validate the response.

2.71 Test-132

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

b) Test purpose: Verify that the server correctly processes the outputSchema parameter.

c) Test method: Formulate a GetRecordById request that also includes the outputSchema parameter. Set the value of the outputSchema parameter to be one of the supported output schemas as advertised in the server’s capabilities document. Execute the request and verify that the response is in the schema specified in the request.

2.72 Test-133

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

b) Test purpose: Verify that the server supports the output schema for the OGC core presentables.

c) Test method: Get the server’s capabilities document. Verify that the document contains a parameter constraint named “outputSchema” for the GetRecordById operations. Verify that the value “http://www.opengis.net/cat/csw/3.0” is in the value domain of the outputSchema parameter. Formulate a GetRecordById request that also includes the outputSchema parameter whose value is set to “http://www.opengis.net/cat/csw/3.0”. Execute the request and verify that the response document presents the OGC core presentables.

2.73 Test-133a

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

b) Test purpose: Verify that the server supports the output format values for XML and ATOM.

c) Test method: Formulate a KVP-encoded GetRecordById request that also uses the OUTPUTFORMAT parameter. Set its value to “application/xml”. Execute the request and verify that the server responses with a valid XML document. Set its value to “application/atom+xml”. Execute the request and verify that the server responds with a valid ATOM entry.

2.74 Test-136

a) Requirement: 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.

b) Test purpose: Verify that the server correctly advertises the value domain for the outputSchema parameter.

c) Test method: Get the server's capabilities document. Inspect the capabilities document and verify that for the GetRecordById operation there is a parameter constraint defined for the outputSchema parameter.

2.75 Test-137

a) Requirement: 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.

b) Test purpose: Verify that the value domain for the outputSchema parameter includes the value "http://www.opengis.net/cat/csw/3.0".

c) Test method: Get the server's capabilities document. Inspect the document and verify that the for the GetRecordById operation there is a parameter constraint defined for the outputSchema parameter. Inspect the value domain for the outputSchema parameter and verify that the value "http://www.opengis.net/cat/csw/3.0" is listed.

2.76 Test-138

a) Requirement: 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.

b) Test purpose: Verify that the default value for the outputSchema parameter is correctly advertised in the capabilities document.

c) Test method: Get the server's capabilities document. Inspect the document and verify that for the GetRecordById operation there is a parameter constraint for the outputSchema parameter. Verify that the first value in the list of supported values is "http://www.opengis.net/cat/csw/3.0".

2.77 Test-139

a) Requirement: 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.

b) Test purpose: Verify that in response to a GetRecordById request the server responds with a bear record. That is a record that is not in-turn enclosed in some containing structure.

c) Test method: Formulate a set of GetRecordById requests that fetch XML representations of the OGC core presentables using the csw:Record schemas and the ATOM schema and also use the ElementSetName parameter to request names sets of elements. Execute the requests and verify that the server responds with a valid response that satisfies the specified request parameters. Verify that the response is a "bare" record – that is a representation of the record that is not enclosed in some containing structure. For the OGC core presentables represented as XML the response should be a base csw:Brief, csw:Summary or csw:Record elements. For ATOM, the response should be a base ATOM entry.

2.78 Test-141

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

b) Test purpose: Verify that the server responds correctly in the situation where a record is not found.

c) Test method: Formulate a GetRecordById request that will not find a record. Execute the request and verify that the server responds with an exception message and that the HTTP status code is set to 400.

3 OpenSearch

3.1 Pre-requisites

a) Test purpose: Verify that the server declares that it implements the OpenSearch conformance class.

b) Test method: Get the server's capabilities document. Inspect the document and identify the service constraint named "OpenSearch". If its value is set to "TRUE" proceed to satisfy the remaining requirements in this sub-clause. If its value is set to "FALSE", the server does not implement the OpenSearch conformance class and no further testing is required.

3.2 Test-008

a) Requirement: 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 10-032r2).

b) Test purpose: To verify that accessing the base URL to server results in an OpenSearch description document when the server declares that it supports the OpenSearch conformance class and the value of the HTTP Accept header is set to "application/opensearchdescription+xml".

c) Test method: Verify that the server implements the OpenSearch conformance class. Set the value of the HTTP Accept header to "application/opensearchdescription+xml" and resolve the base URL of the server. Verify that the response is a valid OpenSearch description document.

3.3 Test-021

a) Requirement: 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".

b) Test purpose: Verify that an OpenSearch description document is generated by a server that implements the OpenSearch conformance class.

c) Test method: Execute a GetCapabilities request. Inspect the response and verify the OpenSearch service constraint is defined (see Table 17) and that its value is set to "TRUE". Verify that Requirement-008 is satisfied.

3.4 Test-022

a) Requirement: 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.

b) Test purpose: Verify that the OpenSearch description document includes templates for queries that generate csw:Record responses.

c) Test method: Satisfy Requirements-021. Inspect the OpenSearch description document generated by the server and verify that it includes the URL templates described by this requirement.

3.5 Test-023

a) Requirement: 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.

b) Test purpose: Verify that the OpenSearch description document includes templates for queries that generate ATOM output.

c) Test method: Satisfy Requirements-021. Inspect the OpenSearch description document generated by the server and verify that it includes the URL templates described by this requirement.

4 GetCapabilities-XML

4.1 Pre-requisites

a) Test purpose: Verify that the server declares that it implements the GetCapabilities-XML conformance class.

b) Test method: Get the server’s capabilities document. Inspect the document and identify the service constraint named “GetCapabilities-XML”. If its value is set to “TRUE” proceed to satisfy the remaining requirements in this sub-clause. If its value is set to “FALSE”, the server does not implement the GetCapabilities-XML conformance class and no further testing is required.

4.2 Test-044

a) Requirement: 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.

b) Test purpose: Verify that the server can generate full or partial capabilities document depending of the value of the Sections parameters.

c) Test method: Formulate a set of GetCapabilities requests where the value of the Sections parameter is set to one or more of “ServiceIdentification”, “ServiceProvider”, “OperationsMetadata” and/or “Filter_Capabilities”. Inspect the response and verify that the requested sections are present.

4.3 Test-045

a) Requirement: If the Sections parameter of the GetCapabilities request is not specified, then all sections of a capabilities document listed in Table 12 shall be presented in the response.

b) Test purpose: Verify that a complete capabilities document is generated when the Sections parameter is omitted from a GetCapabilities request.

c) Test method: Execute a GetCapabilities request, omitting the Sections parameter. Inspect the response and verify that all sections, as defined in this standard, are present in the capabilities document.

4.4 Test-046

a) Requirement: 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.

b) Test purpose: Verify that all operation listed in the OperationsMetadata sections are implemented.

c) Test method: Execute a GetCapabilities request and inspect the response to obtain the list of supported operations. For each listed operation formulate and execute a valid request. In each case, verify that the server generated a valid response.

4.5 Test-047

a) Requirement: Any additional Parameter and Constraint elements, not defined in this standard, that a server introduces in its capabilities shall be safely ignorable by compliant clients.

b) Test purpose: Verify that any addition Parameter and Constraint elements listed in a servers capabilities document can be safely ignored.

c) Test method: Execute a GetCapabilities request. Inspect the response and identify all Parameter and Constraint elements not defined in this standard. Verify that they do not define information that is necessary for the normal operation of the server.

4.6 Test-048

a) Requirement: All the service constraints listed in Table 17 – 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.

b) Test purpose: Verify that the server correctly advertises that set of conformance classes it implements.

c) Test method: Execute a GetCapabilities request. Inspect the response and verify that all the service constraints listed in Table 17 are present in the capabilities document and that their values are set to either “TRUE” or “FALSE”.

4.7 Test-174

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

b) Test purpose: Verify that the server can generate a valid capabilities document when presented with an XML-encoded GetCapabilities request using the HTTP POST method.

c) Test method: Execute a KVP-encoded GetCapabilities request using the HTTP GET method. Inspect the capabilities document and verify that the server implements the GetCapabilities-XML conformance class by checking for the "GetCapabilities-XML" service constraint with its value set to "TRUE". Formulate an XML-encoded GetCapabilities request and present it to the server using the HTTP POST method. Verify that the server responds by generating a valid capabilities document.

5 GetRecordById-XML

5.1 Pre-requisites

a) Test purpose: Verify that the server declares that it implements the GetRecordById-XML conformance class.

b) Test method: Get the server's capabilities document. Inspect the document and identify the service constraint named "GetRecordById-XML". If its value is set to "TRUE" proceed to satisfy the remaining requirements in this sub-clause. If its value is set to "FALSE", the server does not implement the GetRecordById-XML conformance class and no further testing is required.

5.2 Test-123

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

b) Only testable in a profile of this standard.

5.3 Test-124

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

b) Test purpose: Verify that the server generates one summary records in response to a GetRecordById request that does not specify the ElementSetName parameter.

c) Test method: Formulate a GetRecordById request that does not specify the ElementSetName parameter. Execute the request and verify that the server responds with one summary record.

5.4 Test-125

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

b) Test purpose: Verify that the supported element set names are listed in the server's capabilities document.

c) Test method: Get the server's capabilities document. Inspect the document and verify that an operation parameters names "ElementSetName" is specified for the GetRecordById operation. Verify that at least the values "brief", "summary" and "full" are listed as valid values for the ElementSetName parameter. Other values are also allowed.

5.5 Test-126

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

b) Test purpose: Verify that the server correctly processes record identifiers.

c) Test method: Formulate a GetRecordById request and use the Id parameter to specified the identifier of the record to be retrieved. Execute the request and verify that the record presented in the response is the one requested.

5.6 Test-127

a) Requirement: If the identifier specified in the operation is invalid, then the operation shall fail and an InvalidParameterValue exception message should be returned as described in Subclause 6.7.

b) Test purpose: Verify that the server correctly processed record identifiers.

c) Test method: Formulate a GetRecordById request and use the Id parameter to specify a bogus record identifier. Execute the request and verify that the server responds with and exception message indicating that this is an InvalidParameterValue exception.

5.7 Test-128

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

b) Test purpose: Verify that the server verifies that the value of the outputFormat parameter is a valid MIME type.

c) Test method: Formulate a GetRecordById request that also uses the outputFormat parameter. Set the value of the outputFormat to be string that would be considered an invalid MIME type. Execute the request and verify that the server responds with an exception message.

5.8 Test-129

a) Requirement: 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.

b) Test purpose: Verify that default value for the outputFormat parameter for the GetRecordById request is an XML document.

c) Test method: Formulate a GetRecordById request that does not also use the outputFormat parameter. Execute the request and verify that the response is a valid XML document.

5.9 Test-130

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

b) Test purpose: Verify that the server advertises the list of valid output format values for the GetRecordById operation.

c) Test method: Get the server's capabilities document. Inspect the document and verify that a parameter constrain is specified for the GetRecordById operation that lists the set of supported output formats.

5.10 Test-131

a) Requirement: 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.

b) Test purpose: Verify that the server uses the xsi:schemaLocation attribute to provide information required to validate a response.

c) Test method: Formulate a GetRecordById request where the output format is set to "application/xml". Execute the request and verify that the responses contains an xsi:schemaLocation attribute providing information required to validate the response.

5.11 Test-132

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

b) Test purpose: Verify that the server correctly processes the outputSchema parameter.

c) Test method: Formulate a GetRecordById request that also includes the outputSchema parameter. Set the value of the outputSchema parameter to be one of the supported output schemas as advertised in the server's capabilities document. Execute the request and verify that the response is in the schema specified in the request.

5.12 Test-133

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

b) Test purpose: Verify that the server supports the output schema for the OGC core presentables.

c) Test method: Get the server's capabilities document. Verify that the document contains a parameter constraint named "outputSchema" for the GetRecordById operations. Verify that the value "http://www.opengis.net/cat/csw/3.0" is in the value domain of the outputSchema parameter. Formulate a GetRecordById request that also includes the outputSchema parameter whose value is set to "http://www.opengis.net/cat/csw/3.0". Execute the request and verify that the response document presents the OGC core presentables.

5.13 Test-133a

a) Requirement: Servers that conform to this standard shall support the outputFormat parameter values "application/xml" and "application/atom+xml".

b) Test purpose: Verify that the server supports the output format values for XML and ATOM.

c) Test method: Formulate an XML-encoded GetRecordById request that also uses the outputFormat parameter. Set its value to "application/xml". Execute the request and verify that the server responses with a valid XML document. Set its value to "application/atom+xml". Execute the request and verify that the server responds with a valid ATOM entry.

6 GetRecords-Basic-XML

6.1 Pre-requisites

a) Test purpose: Verify that the server declares that it implements the GetRecords-Basic-XML conformance class.

b) Test method: Get the server's capabilities document. Inspect the document and identify the service constraint named "GetRecords-Basic-XML". If its value is set to "TRUE" proceed to satisfy the remaining requirements in this sub-clause. If its value is set to "FALSE", the server does not implement the GetRecords-Basic-XML conformance class and no further testing is required.

6.2 Test-073

a) Requirement: 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.

b) Test purpose: Verify that the server recognizes valid MIME types as the value of the outputFormat parameter. Verify that the server uses the default value of application/xml is the outputFormat parameter is not specified. Verify that the server understands the application/soap+xml MIME type if the server declares that is can handle SOAP messages (see X.X).

c) Test method: Formulate a GetRecords request that uses the outputFormat parameter but specified in invalid MIME type as its values. Execute the request and verify that the server generates an exception. Formulate a GetRecords request that does not specify the outputFormat parameter. Execute the request and verify that the response is a valid XML document. If the server declared support for SOAP, formulate a GetRecords request where the outputFormat is set to application/soap+xml. Execute the request and verify that the response is a valid SOAP document.

6.3 Test-074

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

b) Test purpose: Verify that the server support XML as an output format.

c) Test method: Get a capabilities document from the server and verify that application/xml is listed as a valid value in the value domain for the outputFormat parameter. Formulate one or more requests where the outputFormat is set to application/xml and verify that the response document(s) is/are valid XML document(s).

6.4 Test-075

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

b) Test purpose: Verify that the server advertises the list of supported output formats in its capabilities document.

c) Test method: Get a capabilities document from the server. Inspect the document and verify that a parameter domain element (see X.X) appears in the document listing all the supported output format values.

6.5 Test-076

a) Requirement: 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.

b) Test purpose: Verify that the server generated a valid XML document that also uses xsi:schemaLocation attribute to provide the necessary information to automatically validate the document.

c) Test method: Formulate a GetRecords request that uses the outputFormat parameter and sets its value to application/xml. Execute the request and verify that a valid XML document is generated. Inspect the response document and verify that the xsi:schemaLocation attribute is specified and includes schema information necessary to automatically validate the response.

6.6 Test-077

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

b) Test purpose: Verify that the server uses the value of the outputSchema parameter to indicate the schema of the response to a GetRecords request.

c) Test method: Get a capabilities document from the server and determine the like of outputFormat values that the server supports. Formulate a GetRecords request using the outputFormat parameter and setting its value to one of the advertised values. Execute the request and verify that the response schema matches the requested output schema.

6.7 Test-078

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

b) Test purpose: Verify that one of the supported output schemas is the OGC core schema.

c) Test method: Get a capabilities document from the server. Inspect the document and determine the value domain for the outputSchema parameter for the GetRecords request. Verify that one of accepted values for the outputSchema parameter is "http://www.opengis.net/cat/csw/3.0".

6.8 Test-081

a) Requirement: 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.

b) Test purpose: Verify that the server advertises the list of supported outputSchema values using the "Parameter" element as defined in OGC 06-121r3.

c) Test method: Get a capabilities document from the server. Inspect the capabilities document and determine if there is, for the GetRecords operation, a "Parameter" element with the name "outputSchema".

6.9 Test-082

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

b) Test purpose: Verify that the server understands and uses the value of the startPosition parameter correctly.

c) Test method: Formulate a GetRecords request without using the startPosition parameter. Execute the request and take note of the response records. Modify the request and set the value of the startPosition parameter to something greater than

1 but less than the number of records returned by the original request. Execute this new request and verify that the server starts presenting records at the specified start position.

6.10 Test-083

a) Requirement: The default value of the startPosition parameter shall be 1.

b) Test purpose: Verify that the default value for the startPosition parameter is 1.

c) Test method: Formulate a GetRecords request without specifying the startPosition parameter and whose result set is known. Execute the request and verify that the server start presenting records from the first record in the known result set.

6.11 Test-084

a) Requirement: 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.

b) Test purpose: Verify that the server understands and uses the value of the maxRecords parameter correctly.

c) Test method: Formulate a GetRecords request using the maxRecords parameter and whose result set is known. Ensure that the maxRecords value is greater than 1 but less that the number of records of the known result set. Execute the request and verify that the server returns "maxRecords" records from the result set.

6.12 Test-085

a) Requirement: The default value for the maxRecords parameter shall be 10.

b) Test purpose: Verify that the server enforces that maxRecords default value of 10 records.

c) Test method: Formulate a GetRecords request that does not use the maxRecords parameter and whose result set is known and is greater than 10. Execute the request and verify that the response contains no more than 10 records.

6.13 Test-086

a) Requirement: 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.

b) Test purpose: Verify that the server can correctly respond to a GetRecords request that only returns the number of records the query will return.

c) Test method: Formulate a GetRecords request that uses the maxRecords parameter with its value set to zero and whose result set is known. Execute the request and verify that the server responds with an empty SearchResults element.

Inspect the response and verify that the record count presented in the response is correct.

6.14 Test-087

6.15 Test-088

a) Requirement: 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.

b) Test purpose: Verify that the server correctly understands and uses the typeNames parameter.

c) Test method: Get a capabilities document from the server. Inspect the capabilities document and determine the value domain for the typeNames parameter for the GetRecords request. Formulate one or more GetRecords requests using the advertised values of the typeNames parameter a verify that the server generated a valid response. Formulate a GetRecords request that deliberates uses a typeNames value that is not advertised in the server's capabilities document. Execute the request and verify that the server generates an exception message.

6.16 Test-089

a) Requirement: In such cases the application profile shall describe how multiple typeNames values should be processed.

b) Not testable in this standard.

6.17 Test-090

a) Requirement: 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.

b) Test purpose: Verify that the server correctly processes the ElementName parameter.

c) Test method: Get a capabilities document from the server and determine the list of supported output schemas. Formulate a GetRecords request that uses a recognized output schema and uses the ElementName parameter to identifiy one or more elements from the information model of the catalogue that should appear in the response. Execute the request and inspect the response to verify that the requested elements appear in the response.

6.18 Test-091

a) Requirement: 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 Subclause6.7.

b) Test purpose: Verify that the server correctly handles GetRecords requests that identify invalid elements to be presented in the response.

c) Test method: Get a capabilities document from the server and determine the list of supported output schemas. Formulate a GetRecords request that uses a recognized output schema and uses the ElementName parameter. The value of the ElementName parameter should be an element name that is not part of the requested output schema. Verify that the server responds with an exception messages that uses the exception code InvalidParameterValue.

6.19 Test-092

a) Requirement: 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 xsi:schemaLocation attribute.

b) Test purpose: Verify that, for XML-encoded responses, the server uses the xsi:schemaLocation attribute to reference schemas that can be used to validate the document.

c) Test method: Formulate a GetRecords request that uses the outputFormat parameter with the value set to "application/xml". Execute the request and verify that the server generates a valid response. Inspect the response document and verify that the xsi:schemaLocation attribute used to reference one or more schemas that can be used to validate the document.

6.20 Test-093

a) Requirement: 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.

b) Test purpose: Verify that, for XML-encoded responses, the server augments the set of information model elements requested such that a valid XML document can be generated.

c) Test method: Get a capabilities document from the server and determine the list of supported output schemas and output formats. Identify an output schemas that can be generated using an XML representation. Formulate a GetRecords request that use this output schema and also uses the ElementName parameter to identifiy one or more elements from the information model of the catalogue to be presented in the reponse. Ensure that the number of elements identified using the ElementName parameter are fewer than would be necessary to generate a valid XML-encoded response. Execute the request and verify that a valid XML document is generated. The fact that the response is valid inidicates that the server has correctly augmented the requested set of information model elements.

6.21 Test-095

a) Requirement: 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.

b) Test purpose: Verify that the server correctly processed the ElementSetName parameter.

c) Test method: Using the OGC core output schema, formulate GetRecords requests that request the brief, summary and full element sets using the ElementSetName parameter. Execute each request and verify that the server generates the correct set of elements in each case.

6.22 Test-096

a) Requirement: 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.

b) Test purpose: Verify that the server correctly processes the typeNames parameter on the ElementSetName parameter.

c) Test method: Formulate a GetRecords request that uses the ElementSetName parameter and also uses the typeNames parameter. Set the value of the typeNames parameter to be the same as one of the queried type names. Execute the request and verify that the server generates a valid response. Modify the GetRecords request to the value of the typeNames on the ElementSetName parameter to some typeNames not being queries. Verify that the server generates an exception response.

6.23 Test-097

a) Requirement: 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.

b) Test purpose: Verify that the named set of elements for all entities being queried is presented in the response if the typeNames parameter is not specified for the ElementSetName parameter.

c) Test method: Formulate a GetRecords request that uses the ElementSetName parameter to request a named set of response elements (e.g. brief). Do not specify a value for the typeNames parameter on the ElementSetName parameter. Execute the request and verify that the requested set of elements (e.g. brief) is presented for all queried entities.

6.24 Test-098

a) Requirement: 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 Subclause6.7.

b) Test purpose: Verify that the correct exception response is generated when the value of the typeNames parameter on the ElementSetName parameter does not match one of the entities being queried by the request.

c) Test method: Verify that the server passes the test for Requirement-096. Inspect the exception response generated for the test for Requirement-096 and verify that the exception code is InvalidParameterValue.

6.25 Test-099

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

b) Test purpose: Verify that the ElementSetName parameter and ElementName parameter are mutually exclusive.

c) Test method: Formulate a GetRecords request that specified both the ElementSetName and ElementName parameters. Execute the request and verify that the server generates an exception response.

6.26 Test-100

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

b) ONLY TESTABLE IN A PROFILE.

6.27 Test-101

a) Requirement: If the ElementSetName parameter is not set in a request the service shall respond with one or more *summary* records.

b) Test purpose: Verify that the default query behaviour is to generate summary records.

c) Test method: Formulate a GetRecords request that does not use the ElementSetName or ElementName parameters. Execute the request and verify that the server generates a valid response and that response validates against the summary record schema.

6.28 Test-102

a) Requirement: Servers shall advertise in their capabilities documents, via the ows:Parameter element (see 7.1.5, Table 15), the list of element set names that the server understands. That list shall also include the set names “brief”, “summary” and “full”.

b) Test purpose: Verify that the server advertises the list of name elements sets in its capabilities document.

c) Test method: Get a capabilities document from the server. Inspect the document and verify that a parameter constraint is defined for the ElementSetName parameter of the GetRecords request. Verify that the list of supported values for the ElementSetName parameter includes the values "brief", "summary" and "full". Additional names element sets may also be specified.

7 GetRecords-Distributed-XML

7.1 Pre-requisites

a) Test purpose: Verify that the server declares that it implements the GetRecords-Distributed-XML conformance class.

b) Test method: Get the server's capabilities document. Inspect the document and identify the service constraint named "GetRecords-Distributed-XML". If its value is set to "TRUE" proceed to satisfy the remaining requirements in this sub-clause. If its value is set to "FALSE", the server does not implement the GetRecords-Distributed-XML conformance class and no further testing is required.

7.2 Test-069

a) Requirement: Servers that support the GetRecords-Distributed-XML conformance class shall implement support for the requestId parameter.

b) Test purpose: Verify that the server parses and uses the value of the requestId parameter.

c) Test method: Formulate a distributed XML-encoded GetRecords request that uses the requestId parameter. Execute the request(s) and verify that the server has parsed and correctly used the value of the requestId parameter.

7.3 Test-070

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

b) Test purpose: Verify that the server validates that the value of the requestId parameter is a valid UUID (as per X.667).

c) Test method: Formulate an XML-encoded GetRecords request that uses the requestId parameter and set the value of the parameter to be an invalid UUID. Verify that the server raises an exception in this case.

7.4 Test-071

a) Requirement: 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.

b) Test purpose: Verify that the value of the requestId parameter is included in the response document.

c) Test method: Formulate an XML-encoded GetRecords request that uses the requestId parameter. Execute the request and verify that the value of the parameter is included in the response document.

7.5 GetRecords-Distributed-KVP

7.6 Test-069

a) Requirement: Servers that support the GetRecords-Distributed-KVP conformance class shall implement support for the REQUESTID parameter.

b) Test purpose: Verify that the server parses and uses the value of the REQUESTID parameter.

c) Test method: Formulate a distributed KVP-encoded GetRecords request that uses the REQUESTID parameter. Execute the request and verify that the server has parsed and correctly used the value of the REQUESTID parameter.

7.7 Test-070

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

b) Test purpose: Verify that the server validates that the value of the REQUESTID parameter is a valid UUID (as per X.667).

c) Test method: Formulate a KVP-encoded GetRecords request that uses the REQUESTID parameter and set the value of the parameter to be an invalid UUID. Verify that the server raises an exception in this case.

7.8 Test-071

a) Requirement: 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.

b) Test purpose: Verify that the value of the REQUESTID parameter is included in the response document.

c) Test method: Formulate a KVP-encoded GetRecords request that uses the requestId parameter. Execute the request and verify that the value of the parameter is included in the response document.

8 GetRecords-Async-XML

8.1 Pre-requisites

a) Test purpose: Verify that the server declares that it implements the GetRecords-Async-XML conformance class.

b) Test method: Get the server's capabilities document. Inspect the document and identify the service constraint named "GetRecords-Async-XML". If its value is set to "TRUE" proceed to satisfy the remaining requirements in this sub-clause. If its value is set to "FALSE", the server does not implement the GetRecords-Async-XML conformance class and no further testing is required.

8.2 Test-072

a) Requirement: 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.

b) Test purpose: Verify that the server generates a request identifier if one is not provided in a GetRecords request.

c) Test method: Formulate an asynchronous XML-encoded and do not use the requestId parameter. Execute the request(s) and verify that the server generates a value for the requestId parameter in the response.

8.3 Test-111

a) Requirement: f 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).

b) Test purpose: Verify that the server processes the GetRecords request synchronously if the ResponseHandler parameter is not specified.

c) Test method: Formulate an XML-encoded GetRecords request that does not use the ResponseHandler parameter. Execute the request and verify that the response is not an acknowledgement messages but rather the actual response to the request. Verify that the response is valid.

8.4 Test-112

a) Requirement: 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.

b) Test purpose: Verify that the server processes the GetRecords request asynchronously if the ResponseHandler parameter is specified.

c) Test method: Formulate an XML-encoded GetRecords request that also include the ResponseHandler parameter. Execute the request and verify that the response is an acknowledgement messages.

8.5 Test-113

a) Requirement: 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 ResponseHandler parameter.

b) Test purpose: Verify that after processing and asynchronous GetRecords request the server sends a valid response document or an exception messages (if the request failed).

c) Test method: Formulate a set of XML-encoded GetRecords requests that use the ResponseHandler parameter and that also generate valid responses as well as exception responses. Execute the requests and verify that in each case the server responds with an acknowledgment message. Wait to get the results from the server and verify that the correct responses were given (either a valid response or an exception as the case may be).

8.6 Test-114

a) Requirement: 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.

b) Test purpose: Verify that the server correctly processes multiple response handlers.

c) Test method: Formulate an XML-encoded GetRecords request that also includes multiple ResponseHandler parameters. Execute the request and verify that the server responds with an acknowledgement message. Verify that the multiple specified handlers are notified when the results of the operation are ready. Verify, for each case, that the responses are valid with respect to the request.

8.7 Test-115

a) Requirement: 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.

b) Test purpose: Verify that the acknowledgement message for an asynchronous GetRecords request echos the exact test of the client’s request.

c) Test method: Formulate an XML-encoded GetRecords request that also includes the ResponseHandler parameter. Execute the request and verify that the server responds with an acknowledgement message. Inspect the acknowledgement message and verify that it contains the text of the original GetRecords request.

8.8 Test-116

a) Requirement: 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).

b) Test purpose: Verify that the server checks the request syntax when executing an asynchronous request.

c) Test method: Formulate an XML-encoded GetRecords request that also uses the ResponseHandler parameter but is also an invalid request. Execute the request and verify that the server responses immediately with an exception message.

8.9 Test-117

a) Requirement: 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.

b) Test purpose: Verify that a GetRecords request is executed synchronously if the ResponseHandler parameter is not present.

c) Test method: Formulate an XML-encoded GetRecords request that does not use the ResponseHandler parameter. Execute the request and verify that the server responds with a valid response that is not an acknowledgment message.

9 GetRecords-Async-KVP

9.1 Pre-requisites

a) Test purpose: Verify that the server declares that it implements the GetRecords-Async-KVP conformance class.

b) Test method: Get the server's capabilities document. Inspect the document and identify the service constraint named "GetRecords-Async-KVP". If its value is set to "TRUE" proceed to satisfy the remaining requirements in this sub-clause. If its value is set to "FALSE", the server does not implement the GetRecords-Async-KVP conformance class and no further testing is required.

9.2 Test-072

a) Requirement: 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 request identifier and include it in the Acknowledgement message (see 7.3.4.14) and in the eventual response.

b) Test purpose: Verify that the server generates a request identifier if one is not provided in a GetRecords request.

c) Test method: Formulate an asynchronous KVP-encoded and do not use the REQUESTID parameter. Execute the request and verify that the server generates a value for the requestId parameter in the response.

9.3 Test-111

a) Requirement: 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).

b) Test purpose: Verify that the server processes the GetRecords request synchronously if the RESPONSEHANDLER parameter is not specified.

c) Test method: Formulate an KVP-encoded GetRecords request that does not use the RESPONSEHANDLER parameter. Execute the request and verify that the response is not an acknowledgement messages but rather the actual response to the request. Verify that the response is valid.

9.4 Test-112

a) Requirement: 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.

b) Test purpose: Verify that the server processes the GetRecords request asynchronously if the RESPONSEHANDLER parameter is specified.

c) Test method: Formulate a KVP-encoded GetRecords request that also include the RESPONSEHANDLER parameter. Execute the request and verify that the response is an acknowledgement messages.

9.5 Test-113

a) Requirement: 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 ResponseHandler parameter.

b) Test purpose: Verify that after processing and asynchronous GetRecords request the server sends a valid response document or an exception messages (if the request failed).

c) Test method: Formulate a set of KVP-encoded GetRecords requests that use the RESPONSEHANDLER parameter and that also generate valid responses as well as exception responses. Execute the requests and verify that in each case the server responds with an acknowledgment message. Wait to get the results from the server and verify that the correct responses were given (either a valid response or an exception as the case may be).

9.6 Test-114

a) Requirement: 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.

b) Test purpose: Verify that the server correctly processes multiple response handlers.

c) Test method: Formulate a KVP-encoded GetRecords request that includes a list of values for the RESPONSEHANDLER parameter. Execute the request and verify that the server responds with an acknowledgement message. Verify that the multiple specified handlers are notified when the results of the operation are ready. Verify, for each case, that the responses are valid with respect to the request.

9.7 Test-115

a) Requirement: 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.

b) Test purpose: Verify that the acknowledgement message for an asynchronous GetRecords request echos the exact test of the client's request.

c) Test method: Formulate a KVP-encoded GetRecords request that also includes the RESPONSEHANDLER parameter. Execute the request and verify that the server responds with an acknowledgement message. Inspect the acknowledgement message and verify that it contains the text of the original GetRecords request – in this case the URL used to invoke the request.

9.8 Test-116

a) Requirement: 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).

b) Test purpose: Verify that the server checks the request syntax when executing an asynchronous request.

c) Test method: Formulate a KVP-encoded GetRecords request that also uses the RESPONSEHANDLER parameter but is also an invalid request. Execute the request and verify that the server responses immediately with an exception message.

9.9 Test-117

a) Requirement: 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.

b) Test purpose: Verify that a GetRecords request is executed synchronously if the RESPONSEHANDLER parameter is not present.

c) Test method: Formulate a KVP-encoded GetRecords request that does not use the RESPONSEHANDLER parameter. Execute the request and verify that the server responds with a valid response that is not an acknowledgment message.

10 GetDomain-XML

10.1 Pre-requisites

a) Test purpose: Verify that the server declares that it implements the GetDomain-XML conformance class.

b) Test method: Get the server's capabilities document. Inspect the document and identify the service constraint named "GetDomain-XML". If its value is set to "TRUE" proceed to satisfy the remaining requirements in this sub-clause. If its value is set to "FALSE", the server does not implement the GetDomain-XML conformance class and no further testing is required.

10.2 Test-049

a) Requirement: 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.

b) Test purpose: Verify that the ValueReference parameter can be used to obtain runtime value domains for components of the information model.

c) Test method: Formulate a set of XML-encoded GetDomain requests where the value of the ValueReference parameter is one or more elements from the set of OGC core queryables. Execute the requests and verify that the responses are valid.

10.3 Test-050

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

b) Test purpose: To verify that the server recognizes path expression as the value of the ValueReference parameter.

c) Test method: Verify that the ValueReference values used to verify Requirement-049 are in fact path expressions that identify the components from the set of OGC core queryables.

10.4 Test-051

a) Requirement: For XML-encoded information, an XPath expression shall be used to indicate the information model component to be interrogated.

b) Test purpose: To verify that XPath expression are recognized as the value of the ValueReference parameter.

c) Test method: Verify that the path expressions used to verify Requirement-050 are XPath expressions.

10.5 Test-052

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

b) Test purpose: Verify that API parameters may be interrogated using the ParameterName parameter of the GetDomain operation.

c) Test method: Formulate a set of XML-encoded GetDomain requests using the ParameterName parameter where the value of the parameter is the name of API parameters for CSW operations (e.g. GetRecords.outputFormat). Execute the requests and verify that the responses are valid.

10.6 Test-053

a) Requirement: 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.

b) Test purpose: Verify that the server correctly recognizes dot-path expressions to identify API parameters in a GetDomain request.

c) Test method: Verify that the server passes the test for Requirement-052. Verify that the path expressions used as the value of the ParameterName parameter are dot-path expressions (e.g. GetDomain.outputFormat).

10.7 Test-054

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

b) Test purpose: Verify that the ValueReference or ParameterName value used in a GetDomain request is echoed in the response.

c) Test method: Inspect the responses from the tests for Requirement-051 and Requirement-053 and verify that the value of the ValueReference and/or ParameterName parameters are echoed in the response.

10.8 Test-055

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

b) Test purpose: Verify that an enumerated list of domain values is correctly encoded in the response to a GetDomain operations.

c) Test method: Formulate an XML-encoded GetDomain request that should result in a list of enumerated values. Execute the request and inspect the response to verify that the "ListOfValues" element was used to encode the list of domain values.

10.9 Test-056

a) Requirement: In the event that a default value is indicated, only one value in the response shall be identified as such

b) Test purpose: Verify that only one default value appears as part of the value domain for an API parameter or information model component.

c) Test method: Formulate an XML-encoded GetDomain operation the interrogates a parameter or information model component that has a default value. Execute the request and verify that the response encodes a single default value.

10.10 Test-057

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

b) Test purpose: Verify that units are included when the API parameter or information model component being interrogated is a measurement.

c) Test method: Formulate an XML-encoded GetDomain request for an API parameter or information model component that is a measurement. Execute the request and verify that units of measure are included in the response.

10.11 Test-058

a) Requirement: 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.

b) Test purpose: To verify that controlled vocabularies or authoritative lists of values are correct encoded in the response to a GetDomain request.

c) Test method: Formulate an XML-encoded GetDomain request that is supposed to return a controlled vocabulary or authoritative list. Execute the request and verify in the response that the ConceptualSchema element is used to encode the value domain.

10.12 Test-059

a) Requirement: 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.

b) Test purpose: Verify that GetDomain responses that reference a controlled vocabulary contain all the necessary information to property identify the controlled vocabulary.

c) Test method: Verify that the server passes the test for Requirement-058. Inspect the response from the Requirement-058 test and verify that the name, a link to a document describing the controlled vocabulary and a link to the authority responsible for maintaining the controlled vocabulary are all present in the response.

10.13 Test-060

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

b) Test purpose: Verify that a range of values is correctly encoded in the response to a GetDomain request.

c) Test method: Formulate an XML-encoded GetDomain request that is supposed to return a range of value. Execute the request and verify that the RangeOfValues element is used in the response to encode the range of values.

10.14 Test-061

a) Requirement: The boundaries of the range shall be identified using the MinValue and MaxValue parameters.

b) Test purpose: Verify that ranges are correctly encoded in a GetDomain response that returns a range of values.

c) Test method: Verify that the server passes the test for Requirement-060. Inspect the response and verify that the MinValue and MaxValue parameter are correctly used to define the range of values.

10.15 Test-062

a) Requirement: 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.

b) Test purpose: Verify that the server generates an empty response when the server does not have any domain information about a parameter or information model component.

c) Test method: Formulate an XML-encoded GetDomain request that is not supposed to return a domain for a parameter or information model component. Execute the request and verify that the server generates and empty response -- that is a GetDomainResponse element with no value domain information presented.

11 GetDomain-KVP

11.1 Pre-requisites

a) Test purpose: Verify that the server declares that it implements the GetDomain-KVP conformance class.

b) Test method: Get the server's capabilities document. Inspect the document and identify the service constraint named "GetDomain-KVP". If its value is set to "TRUE" proceed to satisfy the remaining requirements in this sub-clause. If its value is set to "FALSE", the server does not implement the GetDomain-KVP conformance class and no further testing is required.

11.2 Test-049

a) Requirement: 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.

b) Test purpose: Verify that the VALUEREFERENCE parameter can be used to obtain runtime value domains for components of the information model.

c) Test method: Formulate a set of KVP-encoded GetDomain requests where the value of the VALUEREFERENCE parameter is one or more elements from the set of OGC core queryables. Execute the requests and verify that the responses are valid.

11.3 Test-050

a) Requirement: The value of the VALUEREFERENCE parameter shall be a path expression identifying the specific component to be interrogated.

b) Test purpose: To verify that the server recognizes path expression as the value of the VALUEREFERENCE parameter.

c) Test method: Verify that the VALUEREFERENCE values used to verify Requirement-049 are in fact path expressions that identify the components from the set of OGC core queryables.

11.4 Test-051

a) Requirement: For XML-encoded information, an XPath expression shall be used to indicate the information model component to be interrogated.

b) Test purpose: To verify that XPath expressions are recognized as the value of the VALUEREFERENCE parameter.

c) Test method: Verify that the path expressions used to verify Requirement-050 are XPath expressions.

11.5 Test-052

a) Requirement: The PARAMETERNAME parameter shall be used to identify an interface parameter to be interrogated by the GetDomain operation.

b) Test purpose: Verify that API parameters may be interrogated using the PARAMETERNAME parameter of the GetDomain operation.

c) Test method: Formulate a set of KVP-encoded GetDomain requests using the PARAMETERNAME parameter where the value of the parameter is the name of API parameters for CSW operations (e.g. GetRecords.outputFormat). Execute the requests and verify that the responses are valid.

11.6 Test-053

a) Requirement: 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.

b) Test purpose: Verify that the server correctly recognizes dot-path expressions to identify API parameters in a GetDomain request.

c) Test method: Verify that the server passes the test for Requirement-052. Verify that the path expressions used as the value of the ParameterName parameter are dot-path expressions (e.g. GetDomain.outputFormat).

11.7 Test-054

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

b) Test purpose: Verify that the VALUEREFERENCE or PARAMETERNAME value used in a GetDomain request is echoed in the response.

c) Test method: Inspect the responses from the tests for Requirement-051 and Requirement-053 and verify that the value of the VALUEREFERENCE and/or PARAMETERNAME parameters are echoed in the response.

11.8 Test-055

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

b) Test purpose: Verify that an enumerated list of domain values is correctly encoded in the response to a GetDomain operations.

c) Test method: Formulate a KVP-encoded GetDomain request that should result in a list of enumerated values. Execute the request and inspect the response to verify that the "ListOfValues" element was used to encode the list of domain values.

11.9 Test-056

a) Requirement: In the event that a default value is indicated, only one value in the response shall be identified as such

b) Test purpose: Verify that only one default value appears as part of the value domain for an API parameter or information model component.

c) Test method: Formulate a KVP-encoded GetDomain operation the interrogates a parameter or information model component that has a default value. Execute the request and verify that the response encodes a single default value.

11.10 Test-057

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

b) Test purpose: Verify that units are included when the API parameter or information model component being interrogated is a measurement.

c) Test method: Formulate a KVP-encoded GetDomain request for an API parameter or information model component that is a measurement. Execute the request and verify that units of measure are included in the response.

11.11 Test-058

a) Requirement: 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.

b) Test purpose: To verify that controlled vocabularies or authoritative lists of values are correct encoded in the response to a GetDomain request.

c) Test method: Formulate a KVP-encoded GetDomain request that is supposed to return a controlled vocabulary or authoritative list. Execute the request and verify in the response that the ConceptualSchema element is used to encode the value domain.

11.12 Test-059

a) Requirement: 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.

b) Test purpose: Verify that GetDomain responses that reference a controlled vocabulary contain all the necessary information to property identify the controlled vocabulary.

c) Test method: Verify that the server passes the test for Requirement-058. Inspect the response from the Requirement-058 test and verify that the name, a link to a document describing the controlled vocabulary and a link to the authority responsible for maintaining the controlled vocabulary are all present in the response.

11.13 Test-060

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

b) Test purpose: Verify that a range of values is correctly encoded in the response to a GetDomain request.

c) Test method: Formulate a KVP-encoded GetDomain request that is supposed to return a range of value. Execute the request and verify that the RangeOfValues element is used in the response to encode the range of values.

11.14 Test-061

a) Requirement: The boundaries of the range shall be identified using the MinValue and MaxValue parameters.

b) Test purpose: Verify that ranges are correctly encoded in a GetDomain response that returns a range of values.

c) Test method: Verify that the server passes the test for Requirement-060. Inspect the response and verify that the MinValue and MaxValue parameter are correctly used to define the range of values.

11.15 Test-062

a) Requirement: 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.

b) Test purpose: Verify that the server generates an empty response when the server does not have any domain information about a parameter or information model component.

c) Test method: Formulate a KVP-encoded GetDomain request that is not supposed to return a domain for a parameter or information model component. Execute the request and verify that the server generates and empty response -- that is a GetDomainResponse element with no value domain information presented.

12 Transaction

12.1 Pre-requisites

a) Test purpose: Verify that the server declares that it implements the Transaction conformance class.

b) Test method: Get the server's capabilities document. Inspect the document and identify the service constraint named "Transaction". If its value is set to "TRUE" proceed to satisfy the remaining requirements in this sub-clause. If its value is set to "FALSE", the server does not implement the Transaction conformance class and no further testing is required.

12.2 Test-142

a) Requirement: 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 16).

b) Test purpose: Verify that the server advertises the set of supported schemas that may be used with the Transaction operation.

c) Test method: Get the server's capabilities document. Inspect the document and verify that the server implements the Transaction conformance class. Verify that the operation constraint named "TransactionSchema" is specified for the Transaction operation.

12.3 Test-143

a) Requirement: 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 16).

b) Test purpose: Verify that the server can accept records encoded using one of its advertised schemas in a Transaction operation.

c) Test method: Get the server's capabilities document and identify the set of schemas that the server support for the Transaction operations. Formulate Transactions requests that include Insert actions and contains records encoded using one of the support schemas. Execute the request and verify that the server responds with a valid success response. Formulate a GetRecords or GetRecordById request to retrieve the just-inserted record. Verify that the record was correctly saved by the server.

12.4 Test-144

a) Requirement: If an entire record is being replace, 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 16) in the server's capabilities document.

b) Test purpose: Verify that the server can accept the representation of a complete record and can use that representation to replace the an existing record.

c) Test method: Formulate a Transaction request that includes an Update action. The Update action should be formulated to replace an existing catalogue record. Execute the request and verify that the server responds with a valid success response. Formulate a GetRecords or GetRecordById request to retrieve the just-replaced record. Inspect that response and verify that the server correctly processed the Update action.

12.5 Test-145

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

b) Test purpose: To verify that the server can update parts of existing record without replacing the entire record.

c) Test method: Formulate a Transaction request with an Update action that modify one or more parts of an existing catalogue record. Execute the request and verify that the server response with a success response. Formulate a GetRecords or GetRecordById request to retrieve the just-updated record. Verify that the specified changes were correctly made.

12.6 Test-146

a) Requirement: 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.

b) Test purpose: Verify that the server does not accept an Update action without a csw:Constraint element defining the scope of the action.

c) Test method: Formulate a Transaction operation that includes an Update action that does not also include predicates, via the csw:Constraint element, that limit the scope of the action. Execute the request and verify that the server responds with an exception message.

12.7 Test-147

a) Requirement: 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.

b) Test purpose: Verify that the server does not accept a Delete action without a csw:Constraint element defining the scope of the action.

c) Test method: Formulate a Transaction operation that includes a Delete action that does not also include predicates, via the csw:Constraint element, that limit the scope of the action. Execute the request and verify that the server responds with an exception message.

12.8 Test-148

a) 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.

b) Test purpose: Verify that in response to a Transaction operation the server identifies the number of records inserted, updated and/or deleted.

c) Test method: Formulate a Transaction request that inserts, updates and deleted record from the catalogue. Execute the request and verify that the server responds with a valid success response. Inspect the response document and verify that the server correctly identified the number of records inserted, update and/or deleted by the operation.

12.9 Test-149

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

b) Test purpose: Verify that the server includes the identifiers of any new records added to the catalogue using the Transaction operation.

c) Test method: Formulate a Transaction request and uses the Insert action to add new records to the catalogue. Execute the request and verify that the server responds with a valid success response. Inspect the response document and verify that it includes the identifiers of the newly created records.

12.10 Test-150

a) Requirement: 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.

b) Test purpose: Verify that the server uses the brief representation of a newly added record in the response to a Transaction operation.

c) Test method: Inspect the response document from the test for Requirement-149. Verify that for each added record the server includes the brief representation of that record in the response document.

12.11 Test-151

a) Requirement: 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.

b) Test purpose: Verify that for a Transaction request with Insert actions the server correctly correlated the actions in the request with the brief record representations presented in the response.

c) Test method: Formulate a Transaction request that uses the Insert action to add more than one new records to the catalogue. Execute the request and verify that the server responds with a valid success response. Inspect the response document and identify the order in which the brief representations of the newly added records are presented. Verify that is the same order in which the Insert actions that created the records appeared in the Transaction operation. Verify that the mapping between Insert actions in the Transaction operation is 1-to-1 with the brief records presented in the response.

13 Harvest-Basic-XML

13.1 Pre-requisites

a) Test purpose: Verify that the server declares that it implements the Harvest-Basic-XML conformance class.

b) Test method: Get the server's capabilities document. Inspect the document and identify the service constraint named "Harvest-Basic-XML". If its value is set to "TRUE" proceed to satisfy the remaining requirements in this sub-clause. If its value is set to "FALSE", the server does not implement the Harvest-Basic-XML conformance class and no further testing is required.

13.2 Test-152

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

b) Test purpose: Verify that the server advertises the set of metadata representation that is can ingest using the Harvest operation.

c) Test method: Get the server's capabilities document. Inspect the document an verify that a parameter constraint named "ResourceType" is specified for the Harvest operation and that is contains a list of URI's indicating the set of metadata representations that the server can harvest.

13.3 Test-153

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

b) Test purpose: Verify that the server validates the value of the ResourceFormat parameter to verify that it is a valid MIME type.

c) Test method: Formulate an XML-encoded Harvest request that also uses the ResourceFormat parameter. Set the value of the parameter using a string that is not considered a valid MIME type. Execute the request and verify that the server responds with a valid exception message.

13.4 Test-154

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

b) Test purpose: Verify that the server correctly assume that the resource format is XML if the ResourceFormat parameter is not specified on the request.

c) Test method: Formulate an XML-encoded Harvest request that does not use the ResourceFormat parameter. Set the value of the Source parameter to point to some non-XML representation of metadata. Execute the request and verify that the server responds with a valid exception message indicating that it has recognized that the specified source is not an XML document.

13.5 Test-159

a) Requirement: 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 HarvestHandlesAttachments constraint.

b) Test purpose: Verify that the server declares that it can handle attachments.

c) Test method: Get the server's capabilities document. Inspect the document and verify that an operation constraint named "HarvestHandlesAttachments" is specified and its value is set to TRUE.

13.6 Test-160

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

b) Test purpose: Verify that the server correctly handles multipart attachments.

c) Test method: Formulate an XML-encoded Harvest request that includes an attachment. Ensure that the attachment conforms to the requirements of IETF RFC 2387. Execute the request and verify that the server correctly recognizes and processed the attachment.

d) References: Requirement-161, Requirement-162

13.7 Test-161

a) Requirement: 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.

b) Test method: Verify that the server recognizes the first part of a multipart MIME attachment as the XML-encoded Harvest request.

c) Test method: Formulate an XML-encoded Harvest request with an attachment. Ensure that the request itself is not the first part of the attachment. Execute the request and verify that the server responds with a valid exception messages.

13.8 Test-162

a) Requirement: 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.

b) Test purpose: Verify that the server correctly identified the various parts of a Harvest request with attachments.

c) Test method: Formulate an XML-encoded Harvest request with an attachment. Ensure that the Content-ID header uniquely identifies each entity in the message part of the request (i.e. the request itself plus the attached metadata resource). Further ensure that the Source parameter of the Harvest request uses the "cid" scheme to refer to the attached metadata resource. Execute the request and verify that the server response with a valid success response.

13.9 Test-163

a) Requirement: 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.

b) Test purpose: Verify that the server persistently stores a metadata resource that has been attached to a Harvest request.

c) Test method: Formulate an XML-encoded Harvest request with an attachment. Execute the request and verify that the server responds with a valid success response. Retrieve the full harvested record from the catalogue using the OGC core presentable schema. Ensure that the record includes a dcmes:relation link pointing to the originally attached metadata resource.

13.10 Test-164

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

b) Test purpose: Verify that the metadata resource attached to a Harvest request can be retrieved from the server using the link provided by the server via the dcmes:relation link.

c) Test method: Formulate an XML-encoded Harvest request with an attachment. Execute the request and verify that the server responds with a valid success response. Using the OCG core presentables schema, retrieve the harvested record from the catalogue. Verify that the record include a dcmes:relation element with a link to the originally attached resource. Resolve the link and verify that the retrieved resource is in-fact the original attachments included in the Harvest request.

13.11 Test-167

a) Requirement: If the specified resource URI has not been previously harvested then the server shall raise an InvalidParameterValue exception (see Table 10).

b) Test purpose: Verify that the UnHarvest operation raises an exception if asked to unharvest a resource that the has not been previously harvested.

c) Test method: Formulate an XML-encoded UnHarvest request that uses the Source parameter and points to a non-existed resource or a resource that has not been previously harvested by the server. Execute the request and verify that the server responds with a valid exception message.

14 Harvest-Basic-KVP

14.1 Pre-requisites

a) Test purpose: Verify that the server declares that it implements the Harvest-Basic-KVP conformance class.

b) Test method: Get the server's capabilities document. Inspect the document and identify the service constraint named "Harvest-Basic-KVP". If its value is set to "TRUE" proceed to satisfy the remaining requirements in this sub-clause. If its value is set to "FALSE", the server does not implement the Harvest-Basic-KVP conformance class and no further testing is required.

14.2 Test-152

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

b) Test purpose: Verify that the server advertises the set of metadata representation that is can ingest using the Harvest operation.

c) Test method: Get the server's capabilities document. Inspect the document an verify that a parameter constraint named "ResourceType" is specified for the Harvest operation and that is contains a list of URI's indicating the set of metadata representations that the server can harvest.

14.3 Test-153

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

b) Test purpose: Verify that the server validates the value of the RESOURCEFORMAT parameter to verify that it is a valid MIME type.

c) Test method: Formulate a KVP-encoded Harvest request that also uses the RESOURCEFORMAT parameter. Set the value of the parameter using a string that is not considered a valid MIME type. Execute the request and verify that the server responds with a valid exception message.

14.4 Test-154

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

b) Test purpose: Verify that the server correctly assume that the resource format is XML if the RESOURCEFORMAT parameter is not specified on the request.

c) Test method: Formulate a KVP-encoded Harvest request that does not use the RESOURCEFORMAT parameter. Set the value of the Source parameter to point to some non-XML representation of metadata. Execute the request and verify that the server responds with a valid exception message indicating that it has recognized that the specified source is not an XML document.

15 Harvest-Async-XML

15.1 Pre-requisites

a) Test purpose: Verify that the server declares that it implements the Harvest-Async-XML conformance class.

b) Test method: Get the server's capabilities document. Inspect the document and identify the service constraint named "Harvest-Async-XML". If its value is set to "TRUE" proceed to satisfy the remaining requirements in this sub-clause. If its value is set to "FALSE", the server does not implement the Harvest-Async-XML conformance class and no further testing is required.

15.2 Test-155

a) Requirement: 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).

b) Test purpose: Verify that the Harvest operation executes synchronously if the ResponseHandler parameter is not specified.

c) Test method: Formulate an XML-encoded Harvest request that does not specify the ResourceHandler parameter. Execute the request and verify that the server responds with a valid success or exception message.

15.3 Test-156

a) Requirement: 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.

b) Test purpose: Verify that the server correctly processes an asynchronous Harvest request.

c) Test method: Formulate an XML-encoded Harvest request that also uses the ResponseHandler parameter. Execute the request and verify that the server responds with a valid acknowledgement message.

15.4 Test-157

a) Requirement: 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.

b) Test purpose: Verify that the server correctly handles the response to an asynchronous Harvest operation.

c) Test method: Formulate an XML-encoded Harvest request that also uses the ResponseHandler parameter. Execute the request and verify that the server responds with a valid acknowledgement message. Wait to be notified by the server at the response handler. Verify that the notification is a valid HarvestResponse message indicating the Harvest operation completed successfully or valid exception message indicating that the Harvest operations failed.

15.5 Test-158

a) Requirement: 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.

b) Test purpose: Verify that the server correctly processes an asynchronous harvest request with multiple response handlers.

c) Test method: Formulate an XML-encoded Harvest request that also uses the ResponseHandler parameter and specifies multiple response handler targets. Execute the request and verify that the server responds with a valid acknowledgement message. Wait for all the specified response handlers to be notified by the server that the operation has completed. Verify that each notification is a valid HarvestResponse message indicating the server successfully completed the operation or an exception message indicating the operation failed.

15.6 Test-165

a) Requirement: 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).

b) Test purpose: Verify that the server immediately validates an asynchronous Harvest request.

c) Test method: Formulate an XML-encoded Harvest request that also includes the ResponseHandler parameter. Execute the request and verify that the server

responds with a valid acknowledgement messages. Formulate an invalid Harvest request that also includes the ResponseHandler parameter. Execute the request and verify that the server responds with a valid exception message.

15.7 Test-166

a) Requirement: 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.

b) Test purpose: Verify that the server correctly processes a synchronous Harvest request.

c) Test method: Formulate an XML-encoded Harvest request. Execute the request and verify that the server responds with a valid HarvestResponse messages if the request succeeded or a valid exception message id the request failed.

15.8 Test-168

a) Requirement: 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).

b) Test purpose: Verify that the server correctly processes a synchronous UnHarvest request.

c) Test method: Formulate an XML-encoded UnHarvest request. Execute the request and verify that the server responds with a valid UnHarvestResponse message if the request succeeded or a valid exception message if the request failed.

15.9 Test-169

a) Requirement: 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.

b) Test purpose: Verify that the server correctly processes an asynchronous UnHarvest request.

c) Test method: Formulate an XML-encoded UnHarvest request that also include the ResponseHandler parameter. Execute the request and verify that the server responds with a valid acknowledgement messages. Formulate another UnHarvest request that also includes the responseHandler parameter but is also invalid. Execute the request and verify that the server responds with a valid exception message.

15.10 Test-170

a) Requirement: 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.

b) Test purpose: Verify that the server correctly generates a notification messages when an asynchronous UnHarvest request has completed.

c) Test method: Formulate an XML-encoded UnHarvest request that also uses the ResponseHandler parameter. Execute the request and verify that the server responds with a valid acknowledgement message. Wait for the operation to complete and verify that the server sends a valid UnHarvestResponse message or a valid exception messages to the specified response handler.

15.11 Test-171

a) Requirement: 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.

b) Test purpose: Verify that the server correctly handle multiple response handlers.

c) Test method: Formulate an XML-encoded UnHarvest request that also uses more than one ResponseHandler parameter. Execute the request and verify that the server responds with a valid acknowledgement message. Wait for the operation to complete and verify that for each response handler the server sends a valid UnHarvestResponse message or a valid exception messages to that handler.

15.12 Test-172

a) Requirement: 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).

b) Test purpose: Verify that the server immediately validates an asynchronous UnHarvest request.

c) Test method: Formulate and XML-encoded asynchronous UnHarvest operation that specifies a non-existed harvest Source. Execute the request and verify that the server responds with a valid exception message.

15.13 Test-173

a) Requirement: 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.

b) Test purpose: Verify that the server correctly processes a synchronous UnHarvest operation.

c) Test method: Formulate an XML-encoded Harvest request that does not use the ResponseHandler parameter. Execute the request and verify that the server response with a valid success message of a valid exception messages is the request failed.

16 Harvest-Async-KVP

16.1 Pre-requisites

a) Test purpose: Verify that the server declares that it implements the Harvest-Async-KVP conformance class.

b) Test method: Get the server's capabilities document. Inspect the document and identify the service constraint named "Harvest-Async-KVP". If its value is set to "TRUE" proceed to satisfy the remaining requirements in this sub-clause. If its value is set to "FALSE", the server does not implement the Harvest-Async-KVP conformance class and no further testing is required.

16.2 Test-155

a) Requirement: 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).

b) Test purpose: Verify that the Harvest operation executes synchronously if the ResponseHandler parameter is not specified.

c) Test method: Formulate a KVP-encoded Harvest request that does not specify the RESPONSEHANDLER parameter. Execute the request and verify that the server responds with a valid success or exception message.

16.3 Test-156

a) Requirement: 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.

b) Test purpose: Verify that the server correctly processes an asynchronous Harvest request.

c) Test method: Formulate a KVP-encoded Harvest request that also uses the RESPONSEHANDLER parameter. Execute the request and verify that the server responds with a valid acknowledgement message.

16.4 Test-157

a) Requirement: 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.

b) Test purpose: Verify that the server correctly handles the response to an asynchronous Harvest operation.

c) Test method: Formulate an XML-encoded Harvest request that also uses the RESPONSEHANDLER parameter. Execute the request and verify that the server responds with a valid acknowledgement message. Wait to be notified by the server at the response handler. Verify that the notification is a valid HarvestResponse message indicating the Harvest operation completed successfully or valid exception message indicating that the Harvest operations failed.

16.5 Test-158

a) Requirement: 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.

b) Test purpose: Verify that the server correctly processes an asynchronous harvest request with multiple response handlers.

c) Test method: Formulate a KVP-encoded Harvest request that also uses the RESPONSEHANDLER parameter and specifies multiple response handler targets. Execute the request and verify that the server responds with a valid acknowledgement message. Wait for all the specified response handlers to be notified by the server that the operation has completed. Verify that each notification is a valid HarvestResponse message indicating the server successfully completed the operation or an exception message indicating the operation failed.

16.6 Test-165

a) Requirement: 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).

b) Test purpose: Verify that the server immediately validates an asynchronous Harvest request.

c) Test method: Formulate a KVP-encoded Harvest request that also includes the RESPONSEHANDLER parameter. Execute the request and verify that the server responds with a valid acknowledgement messages. Formulate an invalid Harvest request that also includes the RESPONSEHANDLER parameter. Execute the request and verify that the server responds with a valid exception message.

16.7 Test-166

a) Requirement: 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.

b) Test purpose: Verify that the server correctly processes a synchronous Harvest request.

c) Test method: Formulate a KVP-encoded Harvest request that does not use the RESPONSEHANDLER parameter. Execute the request and verify that the server responds with a valid HarvestResponse messages if the request succeeded or a valid exception message id the request failed.

16.8 Test-168

a) Requirement: 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).

b) Test purpose: Verify that the server correctly processes a synchronous UnHarvest request.

c) Test method: Formulate a KVP-encoded UnHarvest request. Execute the request and verify that the server responds with a valid UnHarvestResponse message if the request succeeded or a valid exception message if the request failed.

16.9 Test-169

a) Requirement: 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.

b) Test purpose: Verify that the server correctly processes an asynchronous UnHarvest request.

c) Test method: Formulate a KVP-encoded UnHarvest request that also includes the RESPONSEHANDLER parameter. Execute the request and verify that the server responds with a valid acknowledgement messages. Formulate another UnHarvest request that also includes the RESPONSEHANDLER parameter but is also invalid. Execute the request and verify that the server responds with a valid exception message.

16.10 Test-170

a) Requirement: 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.

b) Test purpose: Verify that the server correctly generates a notification messages when an asynchronous UnHarvest request has completed.

c) Test method: Formulate a KVP-encoded UnHarvest request that also uses the RESPONSEHANDLER parameter. Execute the request and verify that the server responds with a valid acknowledgement message. Wait for the operation to complete and verify that the server sends a valid UnHarvestResponse message or a valid exception messages to the specified response handler.

16.11 Test-171

a) Requirement: 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.

b) Test purpose: Verify that the server correctly handle multiple response handlers.

c) Test method: Formulate a KVP-encoded UnHarvest request that also uses the RESPONSEHANDLER parameter with a value composed of a list of handler URIs. Execute the request and verify that the server responds with a valid acknowledgement message. Wait for the operation to complete and verify that for each response handler the server sends a valid UnHarvest response.

16.12 Test-172

a) Requirement: 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).

b) Test purpose: Verify that the server immediately validates an asynchronous UnHarvest request.

c) Test method: Formulate a KVP-encoded asynchronous UnHarvest operation that specifies a non-existed harvest source. Execute the request and verify that the server responds with a valid exception message.

16.13 Test-173

a) Requirement: 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.

b) Test purpose: Verify that the server correctly processes a synchronous UnHarvest operation.

c) Test method: Formulate a KVP-encoded Harvest request that does not use the RESPONSEHANDLER parameter. Execute the request and verify that the server response with a valid success message of a valid exception messages is the request failed.

17 Harvest-Periodic-XML

17.1 Pre-requisites

a) Test purpose: Verify that the server declares that it implements the Harvest-Periodic-XML conformance class.

b) Test method: Get the server's capabilities document. Inspect the document and identify the service constraint named "Harvest-Periodic-XML". If its value is set to "TRUE" proceed to satisfy the remaining requirements in this sub-clause. If its value is set to "FALSE", the server does not implement the Harvest-Periodic-XML conformance class and no further testing is required.

17.2 Test-175

a) Requirement: 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.

b) Test purpose: Verify that the server correctly, periodically reharvests a resource if the original Harvest request included the HarvestInterval parameter.

c) Test method: Formulate an XML-encoded Harvest request that also uses the HarvestInterval parameter to specify the reharvest period. Execute the request and verify that the server responds with a valid success response. Between the time the request was execute and the next harvest interval change the resource being harvested. Wait for the server to reharvest the resource. Retrieve the resource from the catalogue and verify that the server correctly reharvested the resource by verify that the changes not appear in the catalogue.

17.3 Test-176

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

b) Test purpose: Verify that a Harvest request without the HarvestInterval parameter harvested the specified resource once and does not try to periodically reharvest the resource.

c) Test method: ???

18 Harvest-Periodic-KVP

18.1 Pre-requisites

a) Test purpose: Verify that the server declares that it implements the Harvest-Periodic-KVP conformance class.

b) Test method: Get the server's capabilities document. Inspect the document and identify the service constraint named "Harvest-Periodic-KVP". If its value is set to "TRUE" proceed to satisfy the remaining requirements in this sub-clause. If its

value is set to “FALSE”, the server does not implement the Harvest-Periodic-KVP conformance class and no further testing is required.

18.2 Test-175

a) Requirement: 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.

b) Test purpose: Verify that the server correctly, periodically reharvests a resource if the original Harvest request included the HARVESTINTERVAL parameter.

c) Test method: Formulate a KVP-encoded Harvest request that also uses the HARVESTINTERVAL parameter to specify the reharvest period. Execute the request and verify that the server responds with a valid success response. Between the time the request was execute and the next harvest interval change the resource being harvested. Wait for the server to reharvest the resource. Retrieve the resource from the catalogue and verify that the server correctly reharvested the resource by verify that the changes not appear in the catalogue.

18.3 Test-176

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

b) Test purpose: Verify that a Harvest request without the HARVESTINTERVAL parameter harvests the specified resource once and does not try to periodically reharvest the resource.

c) Test method: ???

19 Filter-CQL

19.1 Pre-requisites

a) Test purpose: Verify that the server declares that it implements the Filter-CQL conformance class.

b) Test method: Get the server’s capabilities document. Inspect the document and identify the service constraint named “Filter-CQL”. If its value is set to “TRUE” proceed to satisfy the remaining requirements in this sub-clause. If its value is set to “FALSE”, the server does not implement the Filter-CQL conformance class and no further testing is required.

19.2 Test-104

a) Requirement: 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.

b) Test purpose: Verify that a CQL predicate can correctly interpret an XML Path expressions used to reference values from a catalogue information model that is represented as XML.

c) Test method: Formulate a KVP-encoded GetRecords request that uses CQL reference components from the XML representation of the OGC core queryables (i.e. csw:Record) using XPath expressions. Execute that request and verify that the server responds with a valid response document.

20 Filter-FES-XML

20.1 Pre-requisites

a) Test purpose: Verify that the server declares that it implements the Filter-FES-XML conformance class.

b) Test method: Get the server's capabilities document. Inspect the document and identify the service constraint named "Filter-FES-XML". If its value is set to "TRUE" proceed to satisfy the remaining requirements in this sub-clause. If its value is set to "FALSE", the server does not implement the Filter-FES-XML conformance class and no further testing is required.

20.2 Test-015

a) Requirement: 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

a) Test purpose: To verify that a server that support XML-encoded filters, supports the minimum set of filter capabilities.

b) Test method: Execute a GetCapabilities request and inspect the FilterCapabilities section to verify that the minimum set of filter operators is supported.

20.3 Test-016

a) Requirement: 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.

b) Test purpose: To verify that the service enforces the use of the version parameter on the Constraint element.

c) Test method: Execute a GetCapabilities request and verify that the server implements one or more XML-encoded requests that use a filter (e.g. GetRecords). Execute a request, using the Constraint element, but omit the "version" parameter. Verify that the server generates an exception message.

d) Reference: 6.5.5.2

20.4 Test-103

a) Requirement: 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.

b) Test purpose: To verify that the server supports the use of XPath expressions to reference components of information models that have XML representations.

c) Test method: Formulate an XML-encoded GetRecords requests that use XPath expressions, consistent with the minimum XPath subset defined in OGC 09-026r1 (see Requirement-105), to reference components in the information model. Execute the requests and verify that the server generates valid responses.

20.5 Test-105

a) Requirement: 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 the minimum XML Path Language (XPath) defined in clause 7.4.4 of OGC 09-026r1.

b) Test purpose: Verify that the server supports the minimum set of XPath capabilities defined in the FES.

c) Test method: Verify that the server passes test A.14 of OGC 09-026r1.

20.6 Test-108

a) Requirement: If no sort is specified then the server will sort the result according to its default sort which must be declared in the capabilities doc (see Table X).

b) Test purpose: Verify that if the server advertises a default sort, that sort is used to present the results of a GetRecords request that does not include a sorting clause.

c) Test method: Get the server's capabilities document. Inspect the document and see if the server defines a service constraint named DefaultSortingAlgorithm. If

the service constraint is not specified than the service has passed the test for this requirement. If the server constraint is specified then formulate an XML-encoded GetRecords request that does not include a sorting clause. Execute that request and verify that the results are presented in the order specified by the value of DefaultSortingAlgorithm service constraint.

20.7 Test-109

a) Requirement: 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.

b) Test purpose: Verify that the server performs an alphabetic sort based on the record's Title if no sorting clause is specified in the request and the server does not advertise a default sorting algorithm.

c) Test method: Get the server's capabilities document and verify that the server does not define a service constraint names DefaultSortingAlgorithm. Formulate an XML-encoded GetRecords request that does not include a sorting clause. Execute the request and verify that the records in the response are sorted based on the Title of each record.

20.8 Test-110

a) Requirement: 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.

b) Test purpose: Verify that the server correctly parses and XML-encoded sorting clause.

c) Test method: Get the server's capabilities document and verify that the server implements the GetRecords-Basic-XML conformance class. Formulate an XML-encoded GetRecords request that also includes a sorting clause. Execute the request and verify that the response records are sorted as specified in the request.

21 Filter-FES-KVP

21.1 Pre-requisites

a) Test purpose: Verify that the server declares that it implements the Filter-FES-KVP conformance class.

b) Test method: Get the server's capabilities document. Inspect the document and identify the service constraint named "Filter-FES-KVP". If its value is set to "TRUE" proceed to satisfy the remaining requirements in this sub-clause. If its value is set to "FALSE", the server does not implement the Filter-FES-KVP conformance class and no further testing is required.

21.2 Test-017

a) Requirement: A server implementation is not required to support all the KVP parameters specified in Table 6 but shall advertise which parameters it implements using the Filter Capabilities section (see 7.1.3) of the server's capabilities document.

b) Test purpose: To verify that the supported filter capabilities are correctly advertised in the capabilities document.

c) Test method: Execute a GetCapabilities request and determine the set of KVP-encoded operations that the server implements that also use a filter (e.g. GetRecords). Inspect the FilterCapabilities section of the capabilities document and determine the list of supported operators. For the set of identified KVP-encoded operations, execute requests that use operators NOT advertised in the FilterCapabilities section. Verify that the server responds with an exception messages.

d) Reference: 6.5.5.3

22 Filter-FES-KVP-Advanced

22.1 Pre-requisites

a) Test purpose: Verify that the server declares that it implements the Filter-FES-KVP-Advanced conformance class.

b) Test method: Get the server's capabilities document. Inspect the document and identify the service constraint named "Filter-FES-KVP-Advanced". If its value is set to "TRUE" proceed to satisfy the remaining requirements in this sub-clause. If its value is set to "FALSE", the server does not implement the Filter-FES-KVP-Advanced conformance class and no further testing is required.

22.2 Test-018

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

b) Test purpose: To verify that a server that declares to implement the Filter-FES-KVP-Advanced conformance class advertises this fact in its capabilities document.

c) Test method: Get the server's capabilities document. Inspect the documents and determine the set of KVP-encoded operations that the server implements that also use a filter (e.g. GetRecords). Execute a set of requests for these operations using the CONSTRAINT, CONSTRAINTLANGUAGE and CONSTRAINTLANGUAGEVERSION parameters and verify that the server generates valid responses.

22.3 Test-106

a) Requirement: In KVP encoding, the SORTBY parameter shall be used to specify the list of sort elements

b) Test purpose: Verify that the server correctly processed the SORTBY parameter.

c) Test method: Formulate a set of KVP-encoded GetRecords requests that use the SORTBY parameter. Execute the requests and verify that the server generates valid responses.

22.4 Test-107

a) Requirement: 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.

b) Test purpose: Verify that the server correctly parses the value of SORTBY parameter.

c) Test method: Formulate a set of KVP-encoded GetRecords requests that use the SORTBY parameter and sort on more than one property and that possibly use a mix of sort orders (ascending or descending). Execute the request and verify that the server generates a valid response and that the sort order of the records is as specified by the SORTBY parameter.

22.5 Test-108

a) Requirement: If no sort is specified then the server will sort the result according to its default sort which must be declared in the capabilities doc (see Table X).

b) Test purpose: Verify that if the server advertises a default sort, that sort is used to present the results of a GetRecords request that does not include a sorting clause.

c) Test method: Get the server's capabilities document. Inspect the document and see if the server defines a service constraint named DefaultSortingAlgorithm. If the service constraint is not specified than the service has passed the test for this requirement. If the server constraint is specified then formulate a KVP-encoded GetRecords request that does not include a sorting clause. Execute that request and verify that the results are presented in the order specified by the value of DefaultSortingAlgorithm service constraint.

22.6 Test-109

a) Requirement: 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.

b) Test purpose: Verify that the server performs an alphabetic sort based on the record's Title if no sorting clause is specified in the request and the server does not advertise a default sorting algorithm.

c) Test method: Get the server's capabilities document and verify that the server does not define a service constraint names DefaultSortingAlgorithm. Formulate a KVP-encoded GetRecords request that does not include a sorting clause. Execute the request and verify that the records in the response are sorted based on the Title of each record.

23 CSW-Response

23.1 Pre-requisites

a) Test purpose: Verify that the server declares that it implements the CSW-Response conformance class.

b) Test method: Get the server's capabilities document. Inspect the document and identify the service constraint named "CSW-Response". If its value is set to "TRUE" proceed to satisfy the remaining requirements in this sub-clause. If its value is set to "FALSE", the server does not implement the CSW-Response conformance class and no further testing is required.

23.2 Test-024

a) Requirement: 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.

b) Test purpose: Verify that all implementation of the HTTP protocol binding support the OGC core queryables and presentables are realized by the csw:AbstractRecord XML element.

c) Test method: Execute a set of queries where the request output schema is "http://www.opengis.net/cat/csw/3.0" and verify that the response validates against the record.xsd schema.

23.3 Test-025

a) Requirement: 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.

23.4 Test-026

a) Requirement: 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.

23.5 Test-027

a) Requirement: All elements contained in the csw:Record element that correspond to the abstract OGC queryable terms listed in Table 9 (see OGC 12-176r2) shall be available as queryables.

b) Test purpose: Verify that any of the core queryables may be used in GetRecords requests.

c) Test method: Formulate a set of GetRecords requests that include predicates that reference the core queryables and verify that the server generates valid responses.

23.6 Test-028

a) Requirement: In the event that a catalogue implementation does not have a value to map to an element contained in csw:Record, its value shall be considered to be NULL for the purpose of querying.

23.7 Test-029

a) Requirement: In the Dublin Core schema, the elements “identifier” and “title” (see Table 9) or any element that can substitute for them, are optional. However for the purposes of this specification, these elements shall be considered mandatory queryables and presentables. Thus the OGC abstract queryable terms “Identifier” and “Title” and their XML realization, dc:identifier and dc:title, are mandatory queryables and presentables.

b) Test purpose: Verify that all responses include the dc:identifier and dc:title elements (or synonyms thereof).

c) Test method: Query records from the catalogue where the output schema is set to http://www.opengis.net/csw/csw/3.0. Inspect the responses and verify that is all cases the elements dc:identifier and dc:title are present.

23.8 Test-030

a) Requirement: 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.

b) Test purpose: Verify that the server understands the csw:AnyText queryable.

c) Test method: Formulate query requests that use csw:AnyText in a predicate. Verify that the server generates a valid response.

23.9 Test-031

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

b) Test purpose: Verify that the ows:BoundingBox element is used to encode the spatial extent of a catalogue record.

c) Test method: Execute a set of queries to retrieve catalogue records. Inspect the responses and verify that any spatial extent is expressed using the ows:BoundingBox element.

23.10 Test-032

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

b) Test purpose: Verify that the csw:TemporalExtent element is used to encode the temporal extent of a catalogue record.

c) Test method: Execute a set of queries to retrieve catalogue records. Inspect the responses and verify that any temporal extent is expressed using the csw:TemporalExtent element.

23.11 Test-033

a) Requirement: 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.

b) Test purpose: Verify that the mandatory elements dc:identifier and dc:title are presented in a summary record.

c) Test method: Execute a set of queries where the result type is set to "summary". Verify that all responses contain the mandatory dc:identifier and dc:title elements.

23.12 Test-034

a) Requirement: 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.

b) Test purpose: Verify that the mandatory elements dc:identifier and dc:title are presented in a brief record.

c) Test method: Execute a set of queries where the result type is set to "brief". Verify that all responses contain the mandatory dc:identifier and dc:title elements.

23.13 Test-079

a) Requirement: 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).

b) Test purpose: Verify that the default value for the output schema is "http://www.opengis.net/cat/csw/3.0".

c) Test method: Get a capabilities document from the server. Inspect the document and determine the value domain for the outputSchema parameter for the GetRecords request. Verify that the value "http://www.opengis.net/cat/csw/3.0" is one of the value in the domain and verify that it is indicated to be the default value. Formulate a GetRecords request and omit the outputSchema parameter and either omit the outputFormat parameter of set its value to "application/xml". Execute the request and verify that the response is a valid XML document and that it validates against the OGC core schema.

23.14 Test-118

a) Requirement: In response to a GetRecords 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.

b) Test purpose: Verify that the server can generate the OGC core presentables encoded in XML in response to a GetRecords request.

c) Test method: Formulate a GetRecords request where the output format is set to "application/xml" and the output schema is set to "http://www.opengis.net/cat/csw/3.0". Execute the request and verify that the server responds with a valid GetRecordsResponse XML document. Verify that the content of response is the OGC core presentables.

23.15 Test-120

a) Requirement: The SearchStatus element shall be present and indicates the status of the response.

b) Test purpose: Verify that the response to a GetRecords request include a SearchStatus element.

c) Test method: Formulate a GetRecords request. Execute the request and verify that the response includes a SearchStatus element.

23.16 Test-121

a) Requirement: The content of the SearchResults element shall be the set of records returned by the GetRecords operation.

23.17 Test-134

a) Requirement: 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).

b) Test purpose: Verify that the server implements the default value for the outputSchema parameter.

c) Test method: Formulate a GetRecordById request that does not include the outputSchema parameter or the outputFormat parameter. Execute the request and verify that the server responds with a valid XML document that validates against the schema for the OGC core presentables.

24 ATOM-response

24.1 Pre-requisites

a) Test purpose: Verify that the server declares that it implements the ATOM-response conformance class.

b) Test method: Get the server's capabilities document. Inspect the document and identify the service constraint named "ATOM-response". If its value is set to "TRUE" proceed to satisfy the remaining requirements in this sub-clause. If its value is set to "FALSE", the server does not implement the ATOM-response conformance class and no further testing is required.

24.2 Test-019

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

b) Test purpose: To verify that an OpenSearch-enabled catalogue also implements the Basic-Catalogue conformance class.

c) Test method: Execute and GetCapabilities request. Inspect the response and verify that the "OpenSearch" service constraint (see Table 17) is specified and its values is "TRUE". Further, verify that the server satisfies all the requirements for the Basic-Catalogue conformance class specified in Table 2.

d) Reference: 6.5.6.5

24.3 Test-020

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

b) Test purpose: Verify that an OpenSearch enabled catalogue satisfies all the requirements of the OpenSearch conformance class.

c) Test method: Execute a GetCapabilities request and inspect the response document to verify that the "OpenSearch" service constraint (see Table 17) is specified and that its value is set to "TRUE". Verify that the server satisfies all the requirements for the OpenSearch conformance class specified in Table 2.

24.4 Test-021

a) Requirement: 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".

b) Test purpose: Verify that an OpenSearch description document is generated by a server that implements the OpenSearch conformance class.

c) Test method: Execute a GetCapabilities request. Inspect the response and verify the OpenSearch service constraint is defined (see Table 17) and that its value is set to "TRUE". Verify that Requirement-008 is satisfied.

24.5 Test-022

a) Requirement: 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.

b) Test purpose: Verify that the OpenSearch description document includes templates for queries that generate csw:Record responses.

c) Test method: Satisfy Requirements-021. Inspect the OpenSearch description document generated by the server and verify that it includes the URL templates described by this requirement.

24.6 Test-023

a) Requirement: 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.

b) Test purpose: Verify that the OpenSearch description document includes templates for queries that generate ATOM output.

c) Test method: Satisfy Requirements-021. Inspect the OpenSearch description document generated by the server and verify that it includes the URL templates described by this requirement.

24.7 Test-080

a) Requirement: 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 10-032r2.

b) Test purpose: Verify that the server support an ATOM response to a GetRecords request.

c) Test method: Get a capabilities document from the server. Inspect the document and determine the value domain for the outputFormat parameter of the GetRecords request. Verify that one of the advertised values is “application/atom+xml”. Formulate a GetRecords request omitting the outputSchema parameter and using the outputFormat parameter. Set the value of the outputFormat parameter to “application/atom+xml”. Execute the request and verify that the response is an ATOM feed of the records that satisfy the request.

24.8 Test-119

a) Requirement: 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 10-032r2.

b) Test purpose: Verify that the server can generate a valid ATOM feed in response to a GetRecords request.

c) Test method: Formulate a GetRecords requests where the outputSchema parameter is not set and the outputFormat parameter is set of “application/atom+xml”. Execute the request and verify that the server responds with a valid ATOM feed.

24.9 Test-122

a) Requirement: When the value of the outputFormat parameter is set to “application/atom+xml” and the value of the outputSchema parameter is not set, a CSW shall generate an XML document conforming to the ATOM schema as described in OGC document 10-032.

b) Test purpose: Verify that the server can generate a valid ATOM feed in response to a GetRecords request.

c) Test method: Verify that the server passes Requirement-119.

24.10 Test-135

a) Requirement: 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.

b) Test purpose: Verify that the server can respond to a GetRecordById request with an ATOM response.

c) Test method: Formulate a GetRecordById request that uses the outputFormat parameter but omits the outputSchema parameter. Set the value of the outputFormat parameter to "application/atom+xml". Execute the request and verify that the server responds with a valid ATOM entry.

24.11 Test-140

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

b) Test purpose: Verify that the server generates a valid ATOM response.

c) Test method: Verify that the server passes the test for Requirement-135.