Open Geospatial Consortium

Publication Date: 2014-02-26

Approval Date: 2014-01-17

Submission Date: 2013-07-10

External identifier for this Document: http://www.opengis.net/doc/IS/wcs_processing_extension/2.0

Reference number of this Document: OGC 08-059r4

Version: 2.0

Category: OGC® Interface Standard

Editor: Peter Baumann, Jinsongdi Yu

OGC® Web Coverage Service WCS

Interface Standard - Processing Extension

Warning

This document is an OGC Member approved international standard. This document is available on a royalty free, non-discriminatory basis. 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: Interface
Document stage: Approved for public release
Document language: English

License Agreement

Contents

Page

Tables

Page

i. Abstract

The OGC *Web Coverage Service (WCS)– Processing Extension* defines an extension to the WCS Core [OGC 09-110], the *ProcessCoverages* request type, which allows clients to initiate server-side processing and filtering of coverages and to download the resulting coverage or value sets based on the query language defined in the Web Coverage Processing Service (WCPS) interface standard [OGC 08-068].

ii. Keywords

ogcdoc, wcs, wcps, processing, coverages, extension

iii. Submitting organizations

The following organizations have submitted this Interface Specification to the Open Geospatial Consortium, Inc.:

- Jacobs University Bremen
- Fuzhou University

iv. Document Contributor Contact Points

Name	Organization
Peter Baumann	Jacobs University Bremen, rasdaman GmbH
Jinsongdi Yu	Fuzhou University

v. Changes to the OGC® Abstract Specification

The OGC® Abstract Specification does not require any changes to accommodate the technical contents of this document.

vi. Future Work

Among the topics for future development are the following items:

- Add a RESTful protocol binding.

Foreword

This Web Coverage Service (WCS) Processing extension is an OGC Interface Standard which relies on Web Coverage Service (WCS) Core [OGC 09-110], based on the Web Coverage Processing Service (WCPS) Language Interface Standard [OGC 08-068].

This document includes one normative Annex.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. The Open Geospatial Consortium shall not be held responsible for identifying any or all such patent rights.

Recipients of this document are requested to submit, with their comments, notification of any relevant patent claims or other intellectual property rights of which they may be aware that might be infringed by any implementation of the standard set forth in this document, and to provide supporting documentation.

Suggested additions, changes, and comments on this draft document are welcome and encouraged. Such suggestions may be submitted by email message or by making suggested changes in an edited copy of this document.

Introduction

The OGC *Web Coverage Service (WCS)– Processing Extension* defines an extension to the WCS Core [OGC 09-110], the *ProcessCoverages* request type, which allows clients to initiate server-side processing and filtering of coverages and to download the resulting coverage or value sets based on the query language defined in the Web Coverage Processing Service (WCPS) interface standard [OGC 08-068].

NOTE This WCS Processing Extension defines only synchronous requests, following general WCS philosophy. A semantically equivalent binding [OGC 09-045] of WCPS to the Web Processing Service (WPS) [OGC 05-007r4] allows WCPS to leverage all of the WPS process control mechanisms, including asynchronous processing.

WCPS provides access to original or derived sets of geospatial coverage information, in forms that are useful for client-side rendering, input into scientific models, and other client applications. As such, WCPS includes WCS functionality and extends it with an expression language to form requests of arbitrary complexity allowing, e.g., multi-valued coverage results.

NOTE As the expressive power of the *GetCoverage* operation is a proper subset of the *ProcessCoverages* expressiveness, any *GetCoverage* request can be expressed as a *ProcessCoverages* request.

OGC® Web Coverage Service Interface Standard - Processing Extension

1 Scope

This WCS Processing extension is an OGC Interface Standard which relies on WCS Core [OGC 09-110], based on the Web Coverage Processing Service (WCPS) Language Interface Standard [OGC 08-068].

This extension of the WCS standard specifies an additional processing operation that may optionally be implemented by WCS servers. This operation, the *ProcessCoverages* request type, allows a client to request processing of multi-dimensional grid coverage data on a WCS server by means of the Web Coverage Processing Service (WCPS) language and to retrieve the results over the World Wide Web.

Example The following WCPS expression retrieves the difference between red and green channels of coverages Modis1, Modis2, and Modis3, encoded in NetCDF (the format name may vary, depending on the name specified in the NetCDF format encoding extension specification):

```
for $c in ( Modis1, Modis2, Modis3 )
return
    encode( abs( $c.red - $c.green ), "application/x-netcdf" )
```

2 Compliance

This document establishes the following requirements and conformance class:

- *processing*, of URI http://www.opengis.net/spec/WCS_service-extension_processing/2.0/req/processing; the corresponding conformance class is *processing*, with URI http://www.opengis.net/spec/WCS_service-extension_processing/2.0/conf/processing.

Standardisation target of all requirements and conformance classes are WCS implementations (currently: servers).

Requirements URIs defined in this document are relative to http://www.opengis.net/spec/WCS_service-extension_processing/2.0/req, conformance test URIs are relative to http://www.opengis.net/spec/WCS_service-extension_processing/2.0/conf.

Annex A lists the conformance tests which shall be exercised on any software artefact claiming to implement this WCS Extension.

3 Normative references

This *OGC WCS Processing Extension* specification consists of the present document and an XML Schema. The complete specification is identified by OGC URI

http://www.opengis.net/spec/WCS_service-extension_processing/2.0, the document has OGC URI http://www.opengis.net/doc/IS/WCS_service-extension_processing/2.0.

The complete specification is available for download from http://www.opengeospatial.org/standards/wcs; additionally, the XML Schema is posted online at http://schemas.opengis.net/wcs/processing/2.0 as part of the OGC schema repository. In the event of a discrepancy between bundled and schema repository versions of the XML Schema files, the schema repository shall be considered authoritative.

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

The normative documents listed in Table 1 contain provisions that, through reference in this text, constitute provisions of this specification. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. For undated references, the latest edition of the normative document referred to applies.

Table 1 — Conformance class dependencies

Processing conformance class	Dependency document	Dependency conformance class
processing	OGC 09-146, *GML 3.2.1 Application Schema for Coverages*, version 1.0	*gml-coverage*
	OGC 09-110, *OGC® Web Coverage Service 2.0 Interface Standard - Core*, version 2.0	*core*
	OGC 08-068r2, OGC® *Web Coverage Processing Service (WCPS) Language Interface Standard*, version 2.0	*wcps*

4 Terms and definitions

For the purposes of this document, the terms and definitions given in the above references apply. In addition, the following terms and definitions apply. An arrow "□" indicates that the following term is defined in this Clause.

4.1 [WCPS] Query

String conforming to the WCPS language specification

5 Conventions

5.1 UML notation

Unified Modeling Language (UML) static structure diagrams appearing in this specification are used as described in Subclause 5.2 of OGC Web Services Common [OGC 06-121].

5.2 Data dictionary tables

The UML model data dictionary is specified herein in a series of tables. The contents of the columns in these tables are described in Subclause 5.5 of [OGC 06-121]. The contents of these data dictionary tables are normative, including any table footnotes.

5.3 Namespace prefix conventions

The following namespaces are used in this document. The prefix abbreviations used constitute conventions used here, but are **not** normative. The namespaces to which the prefixes refer are normative, however.

Table 2— Namespace mappings

Prefix	Namespace URI	Description
xsd	http://www.w3.org/2001/XMLSchema	XML Schema namespace
gml	http://www.opengis.net/gml/3.2	GML 3.2.1
gmlcov	http://www.opengis.net/gmlcov/1.0	GML 3.2.1 Application Schema for Coverages 1.0
proc	http://www.opengis.net/wcs/processing/2.0	WCS Processing Extension

5.4 Multiple representations

When multiple representations of the same information are given in a specification document these are consistent. Should this not be the case then this is considered an error, and the XML schema shall take precedence.

6 *Processing* requirements class

6.1 Overview

Clients and servers supporting this Processing Extension can communicate through WCPS requests, that is: clients can submit WCPS queries, servers can execute syntactically and semantically correct queries (as per WCPS definition), and servers can deliver the results of processing such queries to the clients which these can accept.

This Clause 6 establishes the Processing Extension core conformance class *processing* which defines how WCS clients and servers shall interact through WCPS queries.

Figure 1 shows a (slightly simplified) UML diagram summarizing the WCS Processing interface.

6.2 Modifications to *GetCapabilities*

A server announces support of the Processing Extension to a client by adding the URL identifying this extension to the list of supported extensions delivered in the Capabilities document.

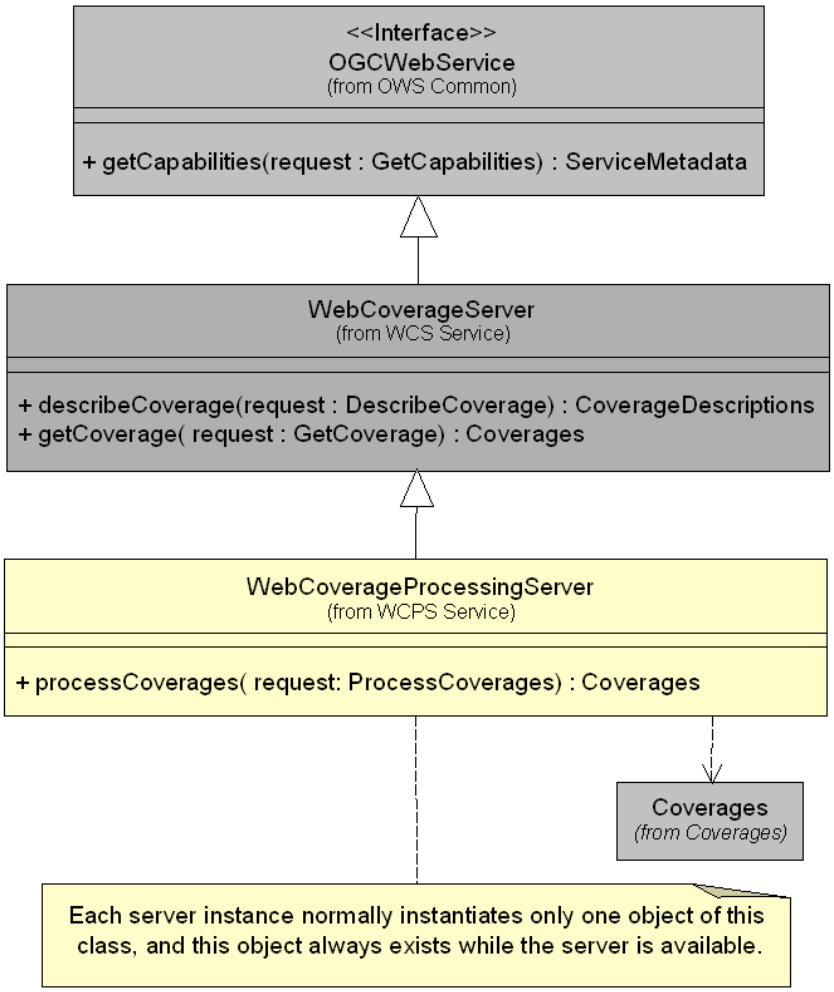

Figure 1—WCS Processing interface UML diagram (simplified)

Requirement 1 extension-identifier:
A WCS service implementing conformance class *processing* of this Processing Extension **shall** include the following URI in the `Profile` element of the `ServiceIdentification` in a *GetCapabilities* response:`http://www.opengis.net/spec/WCS_service-extension_processing/2.0/conf/processing` .
Dependency: WCS Core [OGC 09-110],
`http://www.opengis.net/spec/WCS/2.0/req/core`

6.3 Modifications to *DescribeCoverage*

None.

6.4 Modifications to *GetCoverage*

None.

6.5 ProcessCoverages

6.5.1 Overview

The additional request type *ProcessCoverages* allows to submit a WCPS request string and obtain a processing result consisting of a – possibly empty – sequence of data items (such as scalars, features, coverages).

The *ProcessCoverages* request has one mandatory parameter containing the WCPS query string and zero or more optional parameters (like scalars or features, including the feature subtype coverage) acting as ad-hoc input to a parametrized query.

Queries can be parametrized. Parameters are indicated by a decimal number, prefixed with a "$" character, e.g., $1 and $2. The same number may occur more than once. Values for the parameters are provided separately with the request, outside the query string. The server performs textual substitution (following http entity resolution).

6.5.2 *ProcessCoverages* request

Requirement 2 processCoverages:
A *ProcessCoverages* request **shall** adhere to Figure 2, Table 3, and the XML schema defined for this Processing Extension.

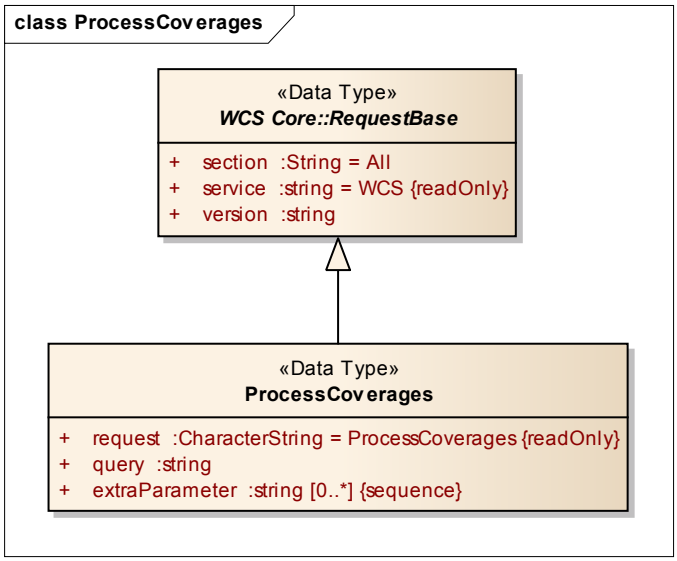

Figure 2 —`ProcessCoverages` UML diagram

Note The request protocol version in the `service` parameter is that of the WCS Core.

Table 3 — Components of `ProcessCoverages` structure

Name	Definition	Data type	Multiplicity
`request`	Request type, fixed to "ProcessCoverages"	`CharacterString`	one (mandatory)

`query`	WCPS query string to be executed on the server	`String`	one (mandatory)
`extra-Parameter`	Optional query parameters	`any`	zero or more (optional)

Example User wants to do an on-the-fly masking of a satellite image to suppress all pixels representing clouds. The cloud mask, which has been obtained from elsewhere (such as a meteorological archive) is submitted with the query as an extra parameter, which is combined with the server-side stored coverage to compute the masked result coverage.

6.5.3 *ProcessCoverages* response

Depending on the result type of the WCPS query sent, the result of a successful evaluation is either a list of (possibly composite) values, or a list of coverages. This result list is represented by a multipart/mixed message; the representation of each component is given by the encoding specified in WCPS.

Requirement 3 processCoverages-response-expansion:
The response to a successful *ProcessCoverages* request **shall** be obtained by evaluating the query according to the WCPS standard, following proper substitution of all extra parameters in the query.

Requirement 4 processCoverages-response-coverage-mimetype:
The response to a successful *ProcessCoverages* request **shall** have a MIME type of "multipart/mixed" as defined in IETF RFC 2046, Section 5.1.3 [2].

Requirement 5 processCoverages-response-coverage-encoding:
The contents of the response to a successful *ProcessCoverages* request **shall** be encoded as defined in IETF RFC 2046, Section 5.1.3 [2].

Requirement 6 processCoverages-response-coverage-encoding2:
Each coverage contained in the response to a successful *ProcessCoverages* request with a coverage result type **shall** be encoded as specified in the WCPS request query's `encode()` function.

Example The result of the following query is a multipart/mixed message where each part consists of a JPEG2000 image:

```
for $c in ( Scene1 )
return encode( $c.red, "image/jp2" )
```

6.6 Exceptions

Requirement 7 processCoverages-exception:
When a WCPS server encounters an error while evaluating a *ProcessCoverages* request it **shall** return an exception report message as indicated in Table 4 with a `locator` value as specified.

Table 4 — Exception codes for *ProcessCoverages* request

exceptionCode value	HTTP code	Meaning of exception code	locator value
SyntaxError	400	Operation request is syntactically malformed.	Offending token and position of error, optionally token expected
SemanticError	400	Operation request is semantically wrong.	Details on error reason

6.7 Encodings

6.7.1 GET/KVP Encoding

Parameters are mapped to keys and values, respecting the GET/KVP encoding rules.

Requirement 8 processCoverages-getkvp:
A *ProcessCoverages* request using the GET/KVP protocol **shall** adhere to the following rules:
- request parameter is encoded with key "REQUEST" and value "ProcessCoverages";
- request parameter query is encoded with key "QUERY" and its value representing a printable string;
- extra parameters in the request have a key value of "1", "2", etc. using decimal ASCII digits.
- the request adheres to Subclause 6.1 of the WCS GET/KVP Extension [OGC 09-147r2].

Note it may be unwieldy or even impossible to transmit complex, large data structures like coverage as parameters using GET/KVP; it is recommended to use a POST encoding instead. A main use case is simple scalar parameter passing.

Requirement 9 processCoverages-getkvp-extraParameter-sequence:
In a *ProcessCoverages* request using the GET/KVP protocol, extra parameters values **shall** have keys whose numerical value is identical to their position number in the query.

Requirement 10 processCoverages-getkvp-query:
In a *ProcessCoverages* request using the GET/KVP protocol, the `query` **shall** be expressed in WCPS Abstract Syntax [OGC 08-068r2].

Example The following is a syntactically valid WCPS request using the GET/KVP protocol (line breaks introduced for editorial reasons only):

```
http://myserver.com/wcs?
    SERVICE=WCS&VERSION=2.0&REQUEST=ProcessCoverages&
    query=for%20$c%20in%20%28Scene1%29%
          return%20encode%28%20$c.red+42,%20%22image%2Fjp2%22%29
```

6.7.2 XML/POST Encoding

Requirement 11 processCoverages-xmlpost:
A *ProcessCoverages* request using the XML/POST protocol **shall** adhere to Subclause 6.1 of the WCS XML/POST Extension [OGC 09-148r1].

Requirement 12 processCoverages-xmlpost-encoding:
In a *ProcessCoverages* request using the XML/POST protocol, the query **shall** be represented by a `proc:ProcessCoverages` element.

Requirement 13 processCoverages-xmlpost-extraParameters:
In a *ProcessCoverages* request using the XML/POST protocol, each extra parameters **shall** be represented by a separate `proc:extraParameter` element.

Requirement 14 processCoverages-xmlpost-extraparameters-sequence:
In a *ProcessCoverages* request using the XML/POST protocol, a query **shall** contain, for each positional parameter in the query, a matching `proc:extraParameter` in the `proc:ProcessCoverages` element in proper sequence of the parameter placeholders occurrence in the query.

Example The XML fragment below resembles the same example as the GET/KVP request above.

```
<proc:ProcessCoverages>
    <proc:query>
        for $c in ( Scene1 )
        return encode( $c.red + $1, "$2" )
    </proc:query>
    <proc:extraParameter>
        42
    </proc:extraParameter>
    <proc:extraParameter>
        image/jp2
    </proc:extraParameter>
</proc:ProcessCoverages>
```

Note The XML code representing a query input XML object my contain xlink references. This may be used, for example, to encode an input coverage's range set in an attachment of a multipart/mixed request.

6.7.3 SOAP Encoding

Requirement 15 processCoverages-soap:
A *ProcessCoverages* request using the SOAP protocol **shall** adhere to the WCS SOAP Extension [OGC 09-149r1].

Example See previous Subclause for an XML example.

Requirement 16 processCoverages-soap-encoding:
In a *ProcessCoverages* request using the SOAP protocol, the query **shall** be represented by a `proc:ProcessCoverages` element.

Requirement 17 processCoverages-soap-parameters:
In a *ProcessCoverages* request using the SOAP protocol, a query **shall** contain, for each positional parameter in the query, a matching `proc:extraParameter` in the `proc:ProcessCoverages` element.

Example See previous Subclause for an XML example.

Bibliography

[1] IETF RFC 2616, *Hypertext Transfer Protocol -- HTTP/1.1*. IETF, 1999

[2] IETF RFC 2046, *Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types*. IETF, 1996

[3] IETF RFC 4180, *Common Format and MIME Type for Comma-Separated Values (CSV) Files*. IETF, 2005

Annex A
(normative)

Abstract test suite

A Processing Extension implementation must satisfy the following system characteristics to be conformant with this specification.

Test identifiers below are relative to http://www.opengis.net/spec/WCS/2.0/WCS_service-extension_processing/2.0/conf.

A.1 Conformance Test Class: *processing*

The OGC URI identifier of this conformance class is:
http://www.opengis.net/spec/WCS/2.0/conf/WCS_service-extension_processing/2.0/conf.

A.1.1 Extension identifier

Test id: **extension-identifier:**
Test Purpose: A WCS service implementing conformance class *processing* of this Processing Extension **shall** include the following URI in the `Profile` element of the `ServiceIdentification` in a *GetCapabilities* response:`http://www.opengis.net/spec/WCS_service-extension_processing/2.0/conf/processing` .
Dependency: WCS Core [OGC 09-110], `http://www.opengis.net/spec/WCS/2.0/req/core`

Test method: Determine the supported extension via a valid *GetCapabilities* request.

Test passes if the URI required is contained.

A.1.2 ProcessCoverages

Test id: **processCoverages:**
Test Purpose: A *ProcessCoverages* request **shall** adhere to Figure 2, Table 3, and the XML schema defined for this Processing Extension.

Test method: Send valid *ProcessCoverages* requests to the server under test, with:

- Zero extraParameters;
- One extraParameter;
- More than one extraParameter.

Check that the result is correct.

Test passes if all individual tests pass.

A.1.3 ProcessCoverages response expansion

Test id: **processCoverages:**
Test Purpose: A *ProcessCoverages* request **shall** adhere to Figure 2, Table 3, and the XML schema defined for this Processing Extension.

Test method: Send valid *ProcessCoverages* requests to the server under test, with:

- Zero extraParameters;
- One extraParameter;
- More than one extraParameter.

Check that the result is correct.

Test passes if all individual tests pass.

A.1.4 ProcessCoverages response coverage mimetype

Test id: **Requirement 18 processCoverages-response-expansion:**
Test Purpose: The response to a successful *ProcessCoverages* request **shall** be obtained by evaluating the query according to the WCPS standard, following proper substitution of all extra parameters in the query.

processCoverages-response-coverage-mimetype:
The response to a successful *ProcessCoverages* request **shall** have a MIME type of "multipart/mixed" as defined in IETF RFC 2046, Section 5.1.3 [2].

Test method: Send valid *ProcessCoverages*requests to the server under test which deliver coverage results.

Test passes if responses have a MIME type as required.

A.1.5 ProcessCoverages response coverage encoding

Test id: **processCoverages-response-coverage-encoding:**
Test Purpose: The contents of the response to a successful *ProcessCoverages* request **shall** be encoded as defined in IETF RFC 2046, Section 5.1.3 [2].

Test method: Send valid *ProcessCoverages*requests to the server under test which deliver

- An empty coverage result list;
- A non-empty coverage result list.

Test passes if all responses are encoded as required.

A.1.6 ProcessCoverages response coverage encoding2

Test id: **processCoverages-response-coverage-encoding2:**
Test Purpose: Each coverage contained in the response to a successful *ProcessCoverages* request with a coverage result type **shall** be encoded as specified in the WCPS request query's `encode()` function.

Test method: Send valid *ProcessCoverages*requests to the server under test which deliver a non-empty coverage result list.

Test passes if all responses are encoded as required.

A.1.7 ProcessCoverages exception

Test id: **processCoverages-exception:**
Test Purpose: When a WCPS server encounters an error while evaluating a *ProcessCoverages* request it **shall** return an exception report message as indicated in Table 4 with a `locator` value as specified.

Test method: For each exception referenced in the requirement: Send an erroneous *ProcessCoverages* request to the server under test provoking this exception, as per its definition. Check for proper exception reporting.

Test passes if all individual tests pass.

A.1.8 ProcessCoverages get kvp

Test id: **processCoverages-getkvp:**
Test Purpose: A *ProcessCoverages* request using the GET/KVP protocol **shall** adhere to the following rules:
- request parameter is encoded with key "REQUEST" and value "ProcessCoverages";
- request parameter query is encoded with key "QUERY" and its value representing a printable string;
- extra parameters in the request have a key value of "1", "2", etc. using decimal ASCII digits.
- the request adheres to Subclause 6.1 of the WCS GET/KVP Extension [OGC 09-147r2].

Test method: Send a valid GET/KVP *ProcessCoverage* request with said keys and values, verify that the response is not an exception.

Test passes if result is as expected.

A.1.9 ProcessCoverages get kvp extraParameter sequence

Test id: **processCoverages-getkvp-extraParameter-sequence:**
Test Purpose: In a *ProcessCoverages* request using the GET/KVP protocol, extra parameters values **shall** have keys whose numerical value is identical to their position number in the query.

Test method: Send GET/KVP *ProcessCoverage* requests with extra parameters contain:

- keys whose numerical value is identical to their position number in the query. Verify that request succeeds.
- keys whose numerical value is not identical to their position number in the query. Verify that request fails.

Test passes if result is as expected.

A.1.10 ProcessCoverages get kvp query

Test id: **processCoverages-getkvp-query:**
Test Purpose: In a *ProcessCoverages* request using the GET/KVP protocol, the `query` **shall** be expressed in WCPS Abstract Syntax [OGC 08-068r2].

Test method: Send a GET/KVP *ProcessCoverage* request with key "REQUEST" and value expressed in WCPS Abstract Syntax.

Test passes if result is as expected.

A.1.11 ProcessCoverages xml post

Test id: **processCoverages-xmlpost:**
Test Purpose: A *ProcessCoverages* request using the XML/POST protocol **shall** adhere to Subclause 6.1 of the WCS XML/POST Extension [OGC 09-148r1].

Test method: Determine the supported extension via a valid *GetCapabilities* request.

Test passes if the URI required is contained.

A.1.12 ProcessCoverages xml post encoding

Test id: **processCoverages-xmlpost-encoding:**
Test Purpose: In a *ProcessCoverages* request using the XML/POST protocol, the query **shall** be represented by a `proc:ProcessCoverages` element.

Test method: Send valid *ProcessCoverages* requests using the WCS XML/POST protocol. Check that the result is correct.

Test passes if all individual tests pass.

A.1.13 ProcessCoverages xml post extraParameters

Test id: **processCoverages-xmlpost-extraParameters:**
Test Purpose: In a *ProcessCoverages* request using the XML/POST protocol, each extra parameters **shall** be represented by a separate `proc:extraParameter` element.

Test method: Send a valid XML/POST *ProcessCoverages* requests with each extra parameter represented by a separate `proc:extraParameter` element . Check that the result is correct.

Test passes if all individual tests pass.

A.1.14 ProcessCoverages xml post extraParameters sequence

Test id: **processCoverages-xmlpost-extraparameters-sequence:**
Test Purpose: In a *ProcessCoverages* request using the XML/POST protocol, a query **shall** contain, for each positional parameter in the query, a matching `proc:extraParameter` in the `proc:ProcessCoverages` element in proper sequence of the parameter placeholders occurrence in the query.

Test method: Send XML/POST *ProcessCoverage* requests with extra parameters contain:

- a matching `proc:extraParameter` in the `proc:ProcessCoverages` element in proper sequence. Verify that request succeeds.
- a matching `proc:extraParameter` in the `proc:ProcessCoverages` element in improper sequence. Verify that request fails.

Test passes if result is as expected.

A.1.15 ProcessCoverages soap

Test id: **processCoverages-soap:**
Test Purpose: A *ProcessCoverages* request using the SOAP protocol **shall** adhere to the WCS SOAP Extension [OGC 09-149r1].

Test method: Determine the supported extension via a valid *GetCapabilities* request.

Test passes if the URI required is contained.

A.1.16 ProcessCoverages soap encoding

Test id: **processCoverages-soap-encoding:**
Test Purpose: In a *ProcessCoverages* request using the SOAP protocol, the query **shall** be represented by a `proc:ProcessCoverages` element.
Test method: Send valid *ProcessCoverages* requests using the WCS SOAP protocol. Check that the result is correct.

Test passes if all individual tests pass.

A.1.17 ProcessCoverages soap parameters

Test id: **processCoverages-soap-parameters:**
Test Purpose: In a *ProcessCoverages* request using the SOAP protocol, a query **shall** contain, for each positional parameter in the query, a matching `proc:extraParameter` in the `proc:ProcessCoverages` element.

Test method: Send SOAP *ProcessCoverage* requests with extra parameters contain:

- a matching `proc:extraParameter` in the `proc:Process-Coverages` element in proper sequence. Verify that request succeeds.
- a mismatching `proc:extraParameter` in the `proc:Process-Coverages` element. Verify that request fails.

Test passes if result is as expected.

-- end of ATS –

Annex B: Revision history

Date	Release	Author	Paragraph modified	Description
2005-06-06	0.0.1	Peter Baumann	All	Initial draft
2008-06-07	0.0.9	Peter Baumann	Several	Incorporated change requests by Arliss Whiteside
2009-01-10	1.0.0	Peter Baumann	Editorial	Finalized for release as standard
2013-02-06	2.0.0	Peter Baumann	All	Rephrased for WCS 2.0 and core/extension model
2013-06-17	2.0.0	Jinsongdi Yu	Req 13, 14, ATS	Update ATS
2013-07-10	2.0.0	Peter Baumann, Jinsongdi Yu	Various	Final editing

