Open Geospatial Consortium

Publication Date: 2014-02-26

Approval Date: 2013-12-13

Submission Date: 2013-06-21

Reference number of this Document: OGC 12-040

Reference URL for this document: http://www.opengis.net/doc/IS/WCS-service-extension-range-subsetting/1.0

Version: 1.0

Category: OGC® Interface Standard

Editor: Peter Baumann, Jinsongdi Yu

OGC® Web Coverage Service Interface Standard - Range Subsetting Extension

Document type:	OGC Standard
Document subtype:	Interface
Document stage:	Approved
Document language:	English

Contents

Page

Tables

Page

i. Abstract

This document specifies parameters to the OGC Web Coverage Service (WCS) *GetCoverage* request which allow extraction of specific fields, according to the range type specification, from the range set of a coverage during server-side processing of a coverage in a *GetCoverage* request.

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.

ii. Keywords

ogcdoc, wcs, Big Data, range_subsetting, extension, band extraction, channel extraction

iii. Terms and definitions

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

iv. Submitting organizations

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

- Jacobs University Bremen
- Fuzhou University

v. Document Contributor Contact Points

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

vi. Revision history

Date	Release	Author	Paragraph modified	Description
2012-04-20	0.0.1	Peter Baumann	All	Created
2013-01-17	0.0.2	Peter Baumann	6+; Annex A	Conformance classes for the protocol bindings
2013-12-17	1.0.0	Peter Baumann	Several	Editorial finalization

vii. Changes to the OGC® Abstract Specification

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

viii. Future Work

Among the topics for future development are the following items:

- Establish a conformance class for subsetting of array-valued range types, i.e., fields whose definition in the coverage range type is described by a `<swe:DataArray>`.

- Establish a conformance class for subsetting of nested range types, i.e., fields whose definition in the coverage range type is described by a an arbitrary nesting of `<swe:DataRecord>` and `<swe:DataArray>` components.

Foreword

This WCS Range Subsetting extension is an OGC Interface Standard which relies on WCS Core [OGC 09-110r4] and the GML Application Schema for Coverages [OGC 09-146r2].

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.

Introduction

The OGC *Web Coverage Service (WCS) – Range Subsetting Extension* defines an extension to the WCS Core [OGC 09-110r4] to request and obtain a coverage with the original domain set but a reduced (extracted) range set as compared to the coverage addressed.

Selection is based on the coverage's range type definition where identifiable components are given; in some domains these range components defined in the range type are referred to as "channels", "bands", or "variables". Various methods are provided for specifying the range components to be retained in the result, such as individual components, lists of components, intervals (based on the range type order), and combinations thereof.

NOTE This standard does not address encoding issues such as whether the number and data type of range fields selected can be represented by the encoding format chosen. See the WCS format encoding extensions for details such aspects.

OGC® Web Coverage Service Interface Standard - Range Subsetting Extension

1 Scope

This OGC WCS Range Subsetting Extension defines retrieval of selected range components from coverages offered by a WCS server. According to the GML 3.2.1 Application Schema – Coverages [OGC 09-146r2], all range cells (commonly called "pixel" or "voxel" in various application domains) of a coverage share a common type, the range type. This range type is built on the *DataRecord* and *DataArray* type construction mechanisms established in SWE Common [OGC 08-094r1]; in other words, a range cell can be a record, or an array, or a nesting of these constructors.

This OGC WCS Range Subsetting Extension establishes how a client can request an extract from such composite range elements from a WCS server. The core conformance class of this standard, *record-subsetting*, defines extraction from unnested records (such as hyperspectral satellite imagery). In a future revision of this standard, optional conformance class *array-subsetting* will define such extraction from nested arrays, and optional conformance class *nested-subsetting*, will define access to nested record and array range types.

2 Conformance

This document establishes the following requirements and conformance classes:

- *record-subsetting*, of URI http://www.opengis.net/spec/WCS_service-extension_range-subsetting/1.0/req/record-subsetting; the corresponding conformance class is *record-subsetting*, with URI http://www.opengis.net/spec/WCS_service-extension_range-subsetting/1.0/conf/record-subsetting.

  This is the mandatory core conformance class for this extension.

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_range-subsetting/1.0/req, conformance test URIs are relative to http://www.opengis.net/spec/WCS_service-extension_range-subsetting/1.0/conf.

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

3 Normative references

This *OGC WCS Range Subsetting Extension* standard consists of the present document and an XML Schema. The complete standard is identified by OGC URI http://www.opengis.net/spec/WCS_service-extension_range-subsetting/1.0, the document has OGC URI http://www.opengis.net/doc/IS/WCS_service-extension_range-subsetting/1.0.

The standard is available for download from http://www.opengeospatial.org/standards/wcs; additionally, the XML Schema is posted online at http://schemas.opengis.net/wcs/range-subsetting/1.0 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 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

Range subsetting conformance class	Dependency document	Dependency con-formance class
record-subsetting	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*

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.

- None defined here -

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 06-121r9].

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 06-121r9]. 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 Application Schema for Coverages
wcs	http://www.opengis.net/wcs/2.0	WCS Core
rsub	http://www.opengis.net/wcs/range-subsetting/1.0	WCS Range Subsetting 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 *record-subsetting* requirements class

6.1 Overview

This Clause 6 establishes the mandatory WCS Range Subsetting Extension core conformance class, *record-subsetting*.This conformance class specifies how to request and obtain coverages where multiple range field components are extracted from a record-valued range field, i.e., a range field whose definition in the coverage range type is described by a `<swe:DataRecord>`.

Note This is the core requirements class because the range type of a coverage, as per GMLCOV [09-146r2], always is a `<swe:DataRecord>`.

The records are flat, i.e., no nesting of record components is addressed here.

6.2 Modifications to *GetCapabilities*

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

Requirement 1 extension-identifier:
A WCS service implementing conformance class *record-subsetting* of this Range Subsetting Extension **shall** include the following URI in the `Profile` element of the `ServiceI-`

`dentification` in any *GetCapabilities* response:
`http://www.opengis.net/spec/WCS_service-extension_range-subsetting/1.0/conf/record-subsetting`

6.3 Modifications to *DescribeCoverage*

None.

6.4 Modifications to *GetCoverage*

6.4.1 *GetCoverage* request

A range subsetting request parameter consists of a sequence of range component selections. Such a selection item can either be a range component name as defined in the range type of the coverage queries, or a range interval specifying a sequence consisting of all range components between the first (start) element of the interval and the last (end) element of the interval, including start and end.

Requirement 2 getCoverage-request-syntax:
A *GetCoverage* request **shall** adhere to Figure 1, Table 3, and the XML schema defined for this Range Subsetting Extension whereby, in the XML request encoding, the *GetCoverage* `wcs:Extension` element **shall** contain exactly one `RangeSubset` element.

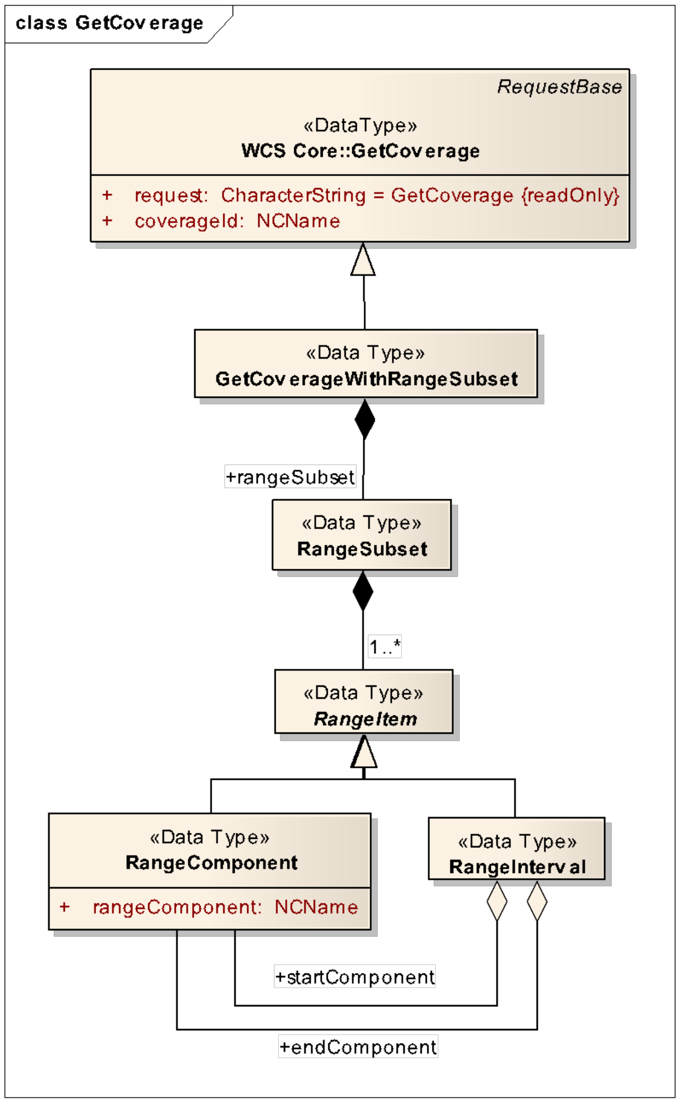

Figure 1 — `GetCoverage` with *range subsetting* support UML diagram

Table 3 — Components of `RSub::GetCoverageWithRangeSubset` structure

Name	Definition	Data type	Multiplicity
`RangeItem`	List of range components to be extracted	`RangeComponent` or `RangeInterval`	one or more (mandatory)
`RangeComponent`	Range component name	`RangeComponent`	one (mandatory)
`RangeInterval`	Pair of range interval lower and upper bound	Pair of `RangeCom-ponent`	one (mandatory)

Requirement 3 getCoverage-subsetting-list:
The `RSub::RangeSubset` parameter in a *GetCoverage* request, if present, **shall** have as its

value a non-empty `RSub::RangeItem` list as specified in Table 2, whereby each `RSub::RangeItem` is either a `RSub::RangeComponent` or a `RSub::RangeInterval`.

Note this implies that a mixing of `rangeComponent` and `rangeInterval` is possible – see example following Requirement 12.

The range subsetting parameter may only refer to existing range type components of the target coverage.

Requirement 4 getCoverage-existing-component:
In the `RSub::RangeSubset` parameter of a *GetCoverage* request, for each `RSub::RangeComponent` there **shall** exist a corresponding component name in the range type of the coverage addressed.

Note Duplicate range components are allowed; for example, a single component can be tripled into an RGB representation of that (grayscale) "channel". In GET/KVP syntax (cf. Subclause 0) transforming the red channel into RGB can be written as follows:

```
…& RANGESUBSET=red,red,red &…
```

Interval addressing is a short-hand for enumerating a sequence of components in the order they appear in the original coverage's range type.

Requirement 5 getCoverage-subsetting-expansion:
In the `RangeSubset` parameter of a *GetCoverage* request, if present, an `RSub::RangeInterval` with `RSub::startComponent` *a* and `RSub::endComponent` *b* **shall** be equivalent to a sequence of $r_1, \ldots, r_n$ where $r_1=a$ and $r_n=b$ and each r_i is a range component name in the original coverage's range type and the sequence is sorted in document order of the original coverage's range type.

Example Assume that the original coverage has a range type definition consisting of the component sequence (band1, band2, band3, band4, band5). Then, a `RangeSubset` selection of

```
band2:band4
```

is equivalent to a selection of `band2,band3,band4`.

Requirement 6 getCoverage-subsetting-interval-order:
In a `RSub::RangeInterval` with `RSub::startComponent` *a* and `RSub::endComponent` *b* contained in the `RangeSubset` parameter of a *GetCoverage* request, range component *a* **shall** be before *b* or equal to *b* (in document order) in the range component sequence of the `RangeType` of the coverage addressed.

Note This requirement forbids range intervals where the lower bound of the interval is higher than the upper bound, in terms of the range type definition.

Example Using the previous RGB example, the following is disallowed and will lead to an exception:

```
blue:red
```

6.4.2 *GetCoverage* response

The range subsetting parameter effectuates that only those range components mentioned are returned to the client, and in the sequence requested (which may be different from the sequence of components in the range type definition of the coverage queried).

Requirement 7 getCoverage-response-no-subsetting:
The response to a successful *GetCoverage* request containing no `RSub::RangeSubset` parameter **shall** consist of a coverage whose range type contains exactly the original coverage's range type elements in the proper sequence of this list.

Requirement 8 getCoverage-response-components:
The response to a successful *GetCoverage* request containing a `RSub::RangeSubset` parameter consisting of list r_1, …, r_n of range component names, for some *n*>0, **shall** consist of a coverage whose range type contains exactly the original coverage's range type elements identified by r_1, …, r_n, in the proper sequence of this list.

Example Assume that the original coverage has a range type definition consisting of the component sequence (band1, band2, band3, band4, band5). Then, a `RangeSubset` selection of

```
band1,band5,band3
```

will result in a coverage with range type sequence (band1,band5,band3).

Requirement 9 getCoverage-response-contents:
The response to a successful *GetCoverage* request containing a `RSub::RangeSubset` parameter consisting of list r_1, …, r_n of range component names, for some *n*>0, **shall** consist of a coverage whose range set contains, for each domain coordinate of the coverage returned, exactly those components of the original coverage's range set values which are identified by r_1, …, r_n, in the proper sequence of this list.

Example Assume that the original coverage has a range type definition consisting of the component sequence (band1, band2, band3, band4, band5) and a single range value (1,2,3,4,5). Then, a `RangeSubset` selection of `band1,band5,band3` will result in a coverage containing the range value (1,5,3).

6.5 Parameter Encoding

6.5.1 Overview

This Clause establishes conformance classes specifying how requests containing range subsetting operations shall be encoded, depending on the protocol binding used by the request.

6.5.2 GET/KVP Encoding

Requirement 10 request-encoding-kvp-list:
In a *GetCoverage* request encoded in GET/KVP, the encoding of a `RangeSubset` parameter consisting of list r_1, …, r_n of `RSub::RangeItems`, for some *n*>1, **shall** consist of an ordered, comma-separated list of the r_i in their proper order.

Requirement 11 request-encoding-kvp-component:
In a GetCoverage request encoded in GET/KVP, the encoding of an `RSub::rangeComponent` in a `RSub::RangeSubset` parameter **shall** consist of this component name.

Requirement 12 request-encoding-kvp-interval:
In a GetCoverage request encoded in GET/KVP, the encoding of an `RSub::RangeInterval` in a `RSub::RangeSubset` parameter **shall** consist of the `RSub::startComponent` name followed by a colon (":") followed by the `RSub::endComponent` name.

Example 1 The following KVP request snippet extracts the component named “red” from the coverage addressed (assuming that these components are defined in the coverage’s range type):
`…& RANGESUBSET=red &…`

Example 2 The following KVP request snippet extracts the three components named “nir”, “red”, and “green” from the coverage addressed (assuming that these components are defined in the coverage’s range type):
`…& RANGESUBSET=nir,red,green &…`

Example 3 The following KVP request snippet extracts the three components named “red”, “green”, and “blue” from the coverage addressed (assuming that these components are defined in the coverage’s range type), but changes its sequence over the original coverage range type definition:
`…& RANGESUBSET=green,red,blue &…`

Example 4 The following KVP request snippet extracts a number of components from the coverage adressed, starting from component “nir” up to component “green”.Assuming that the coverage’s range typecontains, in contiguous XML document order, “nir”, “red”, and “green”, the resulting coverage will contain these three range components “nir”, “red”, and “green”:
`…& RANGESUBSET=nir:green &…`

Example 5 The following KVP request snippet shows mixing of the previously exemplified component selection; for a range type consisting of band01 to band36 (with the obvious order of bands), the request results in a record containing the components band01, band03, band04, band05, band19,band20,band21:
`…& RANGESUBSET=band01,band03:band05,band19:band21 &…`

6.5.3 POST/XML Encoding

Requirement 13 request-encoding-xml:
In a *GetCoverage* request encoded in XML/POST, the encoding of an `RSub::rangeSubset` **shall** consist of a structure as defined in the XML schema accompanying this standard.

Example The following XML `rsub:rangeSubset` selects band1, band3, band4, and band5 from a coverage whose range type contains bands1 through band5:

```
<wcs:GetCoverage xmlns:wcs=http://www.opengis.net/wcs/2.0
    xmlns:gml=http://www.opengis.net/gml/3.2
    xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
    xmlns:crs=http://www.opengis.net/wcs_service-
extension_crs/1.0
    service="WCS" version="2.0.1">
  <wcs:Extension>
    <rsub:RangeSubset>
      <rsub:RangeItem>
        <rsub:RangeComponent>band1</rsub:RangeComponent>
      </rsub:RangeItem>
      <rsub:RangeItem>
        <rsub:RangeInterval>
          <rsub:startComponent>band3</rsub:startComponent>
          <rsub:endComponent>band5</rsub:endComponent>
        </rsub:RangeInterval>
      </rsub:RangeItem>
    </rsub:RangeSubset>
  </wcs:Extension>
```

```
    <wcs:CoverageId>C0001</wcs:CoverageId>
</wcs:GetCoverage>
```

6.5.4 SOAP Encoding

Requirement 14 request-encoding-soap:
In a *GetCoverage* request encoded in SOAP, the encoding of an `RSub::RangeSubset` **shall** consist of a structure as defined in Figure 1, Table 2.

Example See previous Subclause.

6.6 Exceptions

Table 4 — Exception codes for use of `rangeSubset`

`exceptionCode` value	**HTTP code**	**Meaning of exception code**	**`locator` value**
`NoSuchField`	404	One or more range field names indicated in the request are not defined in range type of the coverage addressed	First offending range field name in the parameter list
`IllegalFieldSequence`	404	In a range field interval, the lower limit is above the upper limit, in range type document order	First offending range field name in the parameter list

Bibliography

[1] OGC 08-094r1, OGC® SWE Common Data Model Encoding Standard, version 2.0

Annex A
(normative)

Abstract test suite

A Range Subsetting 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_range-subsetting/1.0/conf.

A.1 Conformance Test Class: *record-subsetting*

The OGC URI identifier of this conformance class is:
http://www.opengis.net/spec/WCS/2.0/conf/WCS_service-extension_range-subsetting/1.0/conf/record-subsetting.

A.1.1 Record-subsetting extension identifier

Test id:	**extension-identifier:**
Test Purpose:	A WCS service implementing conformance class *record-subsetting* of this Range Subsetting Extension **shall** include the following URI in the `Profile` element of the `ServiceIdentification` in any *GetCapabilities* response: `http://www.opengis.net/spec/WCS_service-extension_range-subsetting/1.0/conf/record-subsetting`
Test method:	Send a *GetCapabilities* request to server under test, verify that the response contains a `Profile` element with said URI. Test passes if all individual tests pass.

A.1.2 GetCoverage request syntax

Test id:	**getCoverage-request-syntax:**
Test Purpose:	A *GetCoverage* request **shall** adhere to Figure 1, Table 3, and the XML schema defined for this Range Subsetting Extension whereby, in the XML request encoding, the *GetCoverage* `wcs:Extension` element **shall** contain exactly one `RangeSubset` element.
Test method:	Send *GetCoverage* requests testing server response on the cases distinguished in said reference. Check proper response. Test passes if all individual tests pass.

A.1.3 GetCoverage subsetting list

Test id: **getCoverage-subsetting-list:**

Test Purpose: The `RSub::RangeSubset` parameter in a *GetCoverage* request, if present, **shall** have as its value a non-empty `RSub::RangeItem` list as specified in Table 2, whereby each `RSub::RangeItem` is either a `RSub::RangeComponent` or a `RSub::RangeInterval`.

Test method: Send *GetCoverage* requests to the service under test, evaluate whether responses are adequate. Exercise tests for each of the following situations:

- empty `RSub::rangeItem`
- `RSub::rangeItem` is a `RSub::rangeComponent`
- `RSub::rangeItem` is a `RSub::rangeInterval`

Test passes if all individual tests pass.

A.1.4 GetCoverage existing component

Test id: **getCoverage-existing-component:**

Test Purpose: In the `RSub::RangeSubset` parameter of a *GetCoverage* request, for each `RSub::RangeComponent` there **shall** exist a corresponding component name in the range type of the coverage addressed.

Test method: Send *GetCoverage* requests to the service under test, evaluate whether responses are adequate. Exercise tests for each of the following situations:

- For each `RSub::rangeComponent` there exists a corresponding component name in the range type of the coverage addressed
- There is a component name does not in the range type of the coverage addressed

Test passes if all individual tests pass.

A.1.5 GetCoverage subsetting expansion

Test id: **getCoverage-subsetting-expansion:**

Test Purpose: In the `RangeSubset` parameter of a *GetCoverage* request, if present, an `RSub::RangeInterval` with `RSub::startComponent` a and `RSub::endComponent` b **shall** be equivalent to a sequence of r_1, …, r_n where $r_1=a$ and $r_n=b$ and each r_i is a range component name in the original coverage's range type and the sequence is sorted in document order of the original coverage's range type.

Test method: Send a *GetCoverage* request with an `RSub::rangeInterval` to the service under test, check proper response.

Test passes if all individual tests pass.

A.1.6 GetCoverage interval order

Test id: **getCoverage-subsetting-interval-order:**

Test Purpose: In a `RSub::RangeInterval` with `RSub::startComponent` *a* and `RSub::endComponent` *b* contained in the `RangeSubset` parameter of a *GetCoverage* request, range component *a* **shall** be before *b* or equal to *b* (in document order) in the range component sequence of the `RangeType` of the coverage addressed.

Test method: Send a *GetCoverage* request with a correct `RSub::rangeInterval` to the service under test; check that result is not an exception.

Send a *GetCoverage* request with an `RSub::rangeInterval` containing a lower interval bound higher than the upper bound, according to range type document order; check that result is an `IllegalFieldSequence` exception.

Test passes if all individual tests pass.

A.1.7 GetCoverage response

Test id: **getCoverage-response-no-subsetting:**

Test Purpose: The response to a successful *GetCoverage* request containing no `RSub::RangeSubset` parameter **shall** consist of a coverage whose range type contains exactly the original coverage's range type elements in the proper sequence of this list.

Test method: Send a *GetCoverage* requestcontaining no `RSub::rangeSubset` parameter to the service under test, check that the result consists of a coverage whose range type contains exactly the original coverage's range type elements in the proper sequence of this list.

Test passes if all individual tests pass.

A.1.8 GetCoverage response component

Test Purpose: **getCoverage-response-components:**
The response to a successful *GetCoverage* request containing a `RSub::RangeSubset` parameter consisting of list $r_1, \ldots, r_n$ of range component names, for some n>0, **shall** consist of a coverage whose range type contains exactly the original coverage's range type elements identified

by r_1, …, r_n, in the proper sequence of this list.

Test method: Send a *GetCoverage* requestcontaining containing a `RSub::rangeSubset` parameter consisting of list r_1, …, r_n of range component names to the service under test, check that the result consists of a coverage whose range type contains exactly the original coverage's range type elements identified by r_1, …, r_n, in the proper sequence of this list.

Test passes if all individual tests pass.

A.1.9 GetCoverage response content

Test id: **getCoverage-response-contents:**
Test Purpose: The response to a successful *GetCoverage* request containing a `RSub::RangeSubset` parameter consisting of list r_1, …, r_n of range component names, for some *n*>0, **shall** consist of a coverage whose range set contains, for each domain coordinate of the coverage returned, exactly those components of the original coverage's range set values which are identified by r_1, …, r_n, in the proper sequence of this list.

Test method: Send a *GetCoverage* requestcontaining containing a `RSub::rangeSubset` parameter consisting of list r_1, …, r_n of range component names to the service under test, check that the result consists of a coverage whose range set contains, for each domain coordinate of the coverage returned, exactly those components of the original coverage's range set values which are identified by r_1, …, r_n, in the proper sequence of this list.

Test passes if all individual tests pass.

A.1.10 KVP request list encoding

Test id: **request-encoding-kvp-list:**
Test Purpose: In a *GetCoverage* request encoded in GET/KVP, the encoding of a `RangeSubset` parameter consisting of list r_1, …, r_n of `RSub::RangeItems`, for some *n*>1, **shall** consist of an ordered, comma-separated list of the r_i in their proper order.

Test method: Send aGET/KVP *GetCoverage* request containing the encoding of a `rangeSubset` parameter consisting an ordered, comma-separated list of `RSub::rangeItems` (n>1). Check proper response.

Test passes if all individual tests pass.

A.1.11 KVP request component encoding

Test id: **request-encoding-kvp-component:**

Test Purpose: In a GetCoverage request encoded in GET/KVP, the encoding of an `RSub::rangeComponent` in a `RSub::RangeSubset` parameter **shall** consist of this component name.

Test method: Send a GET/KVP *GetCoverage* request containing the component name of of an `RSub::rangeComponent`. Check proper response.

Test passes if all individual tests pass.

A.1.12 KVP request interval encoding

Test id: **request-encoding-kvp-interval:**
Test Purpose: In a GetCoverage request encoded in GET/KVP, the encoding of an `RSub::RangeInterval` in a `RSub::RangeSubset` parameter **shall** consist of the `RSub::startComponent` name followed by a colon (":") followed by the `RSub::endComponent` name.

Test method: Send a GET/KVP *GetCoverage* request containing the encoding of an `RSub::rangeInterval` parameter consisting of the `RSub::start-Component` name followed by a colon (":") followed by the `RSub::end-Component` name. Check proper response.

Test passes if all individual tests pass.

A.1.13 XML/Post request encoding

Test id: **request-encoding-xml:**
Test Purpose: In a *GetCoverage* request encoded in XML/POST, the encoding of an `RSub::rangeSubset` **shall** consist of a structure as defined in the XML schema accompanying this standard

Test method: Send XML/POST*GetCoverage* requests testing server response on the cases distinguished in said reference. Check proper response.

Test passes if all individual tests pass.

A.1.14 SOAP request encoding

Test id: **request-encoding-soap:**
Test Purpose: In a *GetCoverage* request encoded in SOAP, the encoding of an `RSub::RangeSubset` **shall** consist of a structure as defined in Figure 1, Table 2.

Test method: Send SOAP *GetCoverage* requests testing server response on the cases distinguished in said reference. Check proper response.

Test passes if all individual tests pass.

-- end of ATS –