Open Geospatial Consortium Inc.

Date: 2009-09-16

Reference number of this document: 08-091r6

Version: 0.0.8

Category: OpenGIS® IS Corrigendum

Editor: Peter Schut

Corrigendum for OpenGIS Implementation Standard Web Processing Service (WPS) 1.0.0

Warning

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

Document type:	OpenGIS® IS
Document subtype:	Implementation Standard Corrigendum
Document stage:	Public
Document language:	English

Contents

Page

i. Preface

This document is a corrigendum for OGC Document 05-007r7.

ii. Document terms and definitions

This document uses terms defined in Subclause 5.3 of [OGC 05-008]. 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.

iii. Document contributor contact points

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

Name	Organization
Theodor Förster	ITC
Christian Heier	Wupperverband
Steven Keens	PCI Geomatics
Christian Kiehle	lat/long GmbH
Rachel ONeil	ESRI Canada
Nicole Ostlaender	Joint Research Centre (JRC)
Joan Maso Pau	Universitat Autònoma de Barcelona (CREAF)
Peter Schut (editor)	Agriculture and Agri-Food Canada
Arliss Whiteside	BAE Systems - National Security Solutions

iv. Revision history

Date	Release	Editor	Primary clauses modified	Description
20080609	0.0.1	P. Schut		First draft
20080714	0.0.2	P. Schut	2.1, 2.5, 2.6	Second draft
20080715	0.0.3	P. Schut	2.7	Added content
20080925	0.0.4	P. Schut	all	Renumbered clauses, corrected and added content
20081023	0.0.5	P. Schut	iii, 2.13	Added section 2.13 and updated contacts
20090126	0.0.6	P. Schut	2.14	Added section 2.14
20090302	0.0.7	P. Schut	2.8, 2.14	Corrected example in 2.8

Foreword

This document provides the details for a corrigendum for the existing OpenGIS Standard for the Web Processing Service version 1.0.0 and does not modify that standard. The current OpenGIS Implementation Standard that this document provides revision notes for is 05-007r7.

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 Inc. 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

This document defines the corrigendum change notes for Web Processing Service (WPS) version 1.0.0 Standard which was approved by the OGC membership on 2007-08-11. As a result of the Corrigendum process, there were edits and enhancements made to this standard to correct typographic errors, schema errors, examples, or some deficiency that prevented proper use of this standard. This document provides the details of those edits and corrections.

Web Processing Service 1.0.0

1 Scope

This corrigendum corrects and clarifies the contents of the Web Processing Service 1.0.0 Standard. Corrections apply to the explanatory text and examples and do not affect the structure or syntax of the schemas.

2 Corrigendum Description

2.1 Correct content of Table 10

On page 10, in Table 3, remove footnote "a".

This change corrects the fact that this footnote does not apply to this table.

2.2 Clarify valid occurrences of storeSupported=true

On page 22, insert the following sentence at the beginning of footnote c of Table 16 of section 9.3.1, and on line 84 at the end of the documentation of the statusSupported attribute in the wpsDescribeProcess_response schema:

> In order for a server to support status reporting it must also be able to store process outputs. I.e. "statusSupported" can be "true" only if "storeSupported" is also "true".

This change clarifies that the server is not allowed to advertise support for status reporting if it does not offer storage of the Execute response document.

2.3 Clarify the Execute HTTP GET request KVP encoding example

On page 39, add the following note to Section 10.2.2 after the example on the top of the page:

> Note: In the example shown above, the data input with identifier "Object" does not include a value for that input. Instead, it contains a reference to a web-accessible location from which that value can be retrieved. This input could also have included the optional attributes "mimetype", "encoding", and "schema".

This change clarifies why this example is a valid Execute request.

2.4 Replace executeResponseLocation with statusLocation

On pages 36, 42 and 43, and on lines 91 and 102 in the documentation of elements of the wpsExecuteRequest schema, change all occurrences of:

executeResponseLocation

to:

statusLocation

The text of the specification indicates that an attribute called executeResponseLocation *should exist in the Execute response document. This attribute is actually named* statusLocation*. While changing the name of the attribute to executeResponseLocation would clarify the specification in some places, retention of the existing name ensures that there is no change to the schema and any existing applications.*

2.5 Correct content of Table 52

On page 37, in Table 52, remove footnote "a" from Identifier and insert footnote "a" for mimeType, encoding, and schema.

This change corrects the objects to which this footnote applies.

2.6 Clarify valid combinations of attributes in KVP Execute requests

On page 38, add a note to Table 53 to state:

> One and only one of *ResponseDocument* or *RawDataOutput* may be present in the Execute request. If *ResponseDocument* is present then one or more of *storeExecuteResponse*, *lineage*, and *status* may also be present (depending on the capabilities of the server). If *RawDataOutput* is present then *storeExecuteResponse*, *lineage*, and *status* are not valid as part of the request.

This change explicitly indicates that encoding using KVP is consistent with XML encoding of Execute requests. It clarifies the valid combinations of KVP attributes and eliminates the need to refer to other portions of the specification and schema in order to understand KVP encoding of Execute requests.

2.7 Correct the Execute DataInput parameter KVP syntax reference example

On page 40, replace the Reference Example at the end of section 10.2.2.1.1 currently shown as:

fieldName=xml@Format=text/xml@Encoding=utf-8@Schema=xsd@asReference=true

with

fieldName=polygon@mimeType=text/xml@encoding=utf-8@schema=http://foo.bar/foo.xsd

This change corrects syntax errors in the example (@Format and @asReference are not valid in WPS version 1.0.0).

2.8 Correct and clarify the Chaining of Requests using KVP section

Starting on page 40, replace section 10.2.2.2 with the following text in order to correct errors in the examples and enhance clarity:

HTTP requests using KVP encoding shall support the chaining of requests whereby calls to other web services can be encoded within the value of the xlink:href attribute for a data input. HTTP encoding of these embedded calls is required to ensure that WPS Execute requests can be received and processed unambiguously. WPS instances shall decode the value of an xlink:href attribute and submit the decoded value to the network as an HTTP GET request.

An example of service chaining via KVP follows. Consider the following WPS Execute request that identifies an Input named BufferObject to a Buffer process:

> http://foo.bar.1/wps?version=1.0.0&request=Execute&service=WPS&Identifier=Buffer&DataInputs=BufferObject%3D%40xlink%3Ahref%3Dhttp%253A%252F%252Ffoo.bar2%252Fwps%253FService%253DWPS%2526Version%253D1.0.0%2526Request%253DExecute%2526Identifier%253DShpConvertToGML%2526DataInputs%253DShapefileToConvert%25253D%252540xlink%25253Ahref%25253Dhttp%2525253A%2525252F%2525252Ffoo.bar.3%2525252Fshapefile%40mimeType%3Dtext%252Fxml%40encoding%3Dutf-8%40schema%3Dhttp%253A%252F%252Ffoo.bar%252Ffoo.xsd%3BBufferDistance%3D100%40datatype%3Dinteger%40uom%3Dmeter

After fully decoding the DataInputs value (highlighted above in grey), the WPS at foo.bar.1 determines that one of the inputs is a BufferObject found at a URL identified in an xlink:href. Consequently, it shall submit that URL to the Internet as an HTTP GET request:

> http://foo.bar2/wps?Service=WPS&Version=1.0.0&Request=Execute&Identifier=ShpConvertToGML&DataInputs=ShapefileToConvert%3D%40xlink%3Ahref%3Dhttp%253A%252F%252Ffoo.bar.3%252Fshapefile

Likewise, this Execute request identifies an Input named "ShapefileToConvert" which is to be retrieved from an xlink:href, so after fully decoding the DataInputs value (highlighted above in grey), the WPS at foo.bar.2 shall submit the following URL to the Internet as an HTTP GET request.

> http://foo.bar.3/shapefile

The response from foo.bar.3 allows the WPS at foo.bar2 to complete the execution of the ShpConvertToGML process. When foo.bar.2 returns its Execute response to foo.bar.1, the WPS at foo.bar.1 will be able to complete the execution of the Buffer process.

As indicated in this example, chaining requires increased re-encoding with each level of the chain. The deeper the chain, the more complex the encoding of the deepest links in the request will be.

Service chaining via KVP is most suitable for chaining two or three simple processes together. In deeper chains or longer-running and more complex processes, the use of a workflow engine can facilitate error handling and permit better control of the processing of the chain.

2.9 Correct content of Table 55

On page 45 in Table 55, correct the text of footnote "a" by replacing the work "four" with "five".

This change corrects the number of objects to which this footnote applies.

2.10 Correct content of Table 61

On page 47 in Table 61, replace the object Name "format" with "mimeType".

This change corrects the name of this object and makes it consistent with the contents of the earlier table it references.

2.11 Correct documentation in the wpsExecuteResponse schema

Replace the first sentence in line 26 in the documentation of the elements of the ExecuteResponse *element in the wpsExecuteResponse schema which currently states:*

> However, the response to an Execute request will not include this element in the special case where the output is a single complex value result and the Execute request indicates that "store" is "false". Instead, the server shall return the complex result (e.g., GIF image or GML) directly, without encoding it in the ExecuteResponse. If processing fails in this special case, the normal ExecuteResponse shall be sent, with the error condition indicated. This option is provided to simplify the programming required for simple clients and for service chaining.

with:

> However, the response to an Execute request will not include this element in the special case where the output is a single complex value result and the Execute request indicates the response form should be a "RawDataOutput". Instead, the server shall return the complex result (e.g., GIF image or GML) directly, without encoding it in the ExecuteResponse. If processing fails in this special case, the normal ExecuteResponse shall be sent, with the error condition indicated. This option is provided to simplify the programming required for simple clients and for service chaining.

The text of the documentation in the schema indicates incorrectly that that an attribute called store *in the Execute request controls the response form. The response form is*

actually controlled by the presence of the RawDataOutput *or the* ResponseDocument *element as described in the Standard.*

2.12 Correct documentation in the wpsDescribeProcess_Response schema

Remove the third sentence of line 383 in the documentation of the ComplexOutput element of the wpsDescribeProcess_Response schema which currently states:

> [..] When this output form is indicated, the process produces only a single output, and "store" is "false, the output shall be returned directly, without being embedded in the XML document that is otherwise provided by execute operation response. [..]

The text of the documentation in the schema indicates incorrectly that that an attribute called store *in the Execute request controls the response form. The response form is actually controlled by the presence of the* RawDataOutput *or* ResponseDocument *elements as described in the Standard. The details of how this works are superfluous in the description of this element.*

2.13 Correct content of Annex D, Section D.2 – SOAP encoding of WPS requests and responses

Starting on page 69, replace section D.2 with the following text in order to correct errors in the examples and enhance clarity:

WPS requests and responses encoded in SOAP shall use SOAP document-style encoding (also called message-style or document-literal encoding), as described in OWS 06-094.

An example of a DescribeProcess request follows:

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema">
        <soap:Body>
                <wps:DescribeProcess service="WPS" version="1.0.0"
                  xmlns="ttp://www.opengeospatial.net/wps"
                  xmlns:ows="http://www.opengeospatial.net/ows"
                  xmlns:xlink="http://www.w3.org/1999/xlink"
                  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                  xsi:schemaLocation="http://www.opengeospatial.net/wps
                    \wpsDescribeProcess.xsd">
                        <ows:Identifier>intersection</ows:Identifier>
                        <ows:Identifier>union</ows:Identifier>
                </wps:DescribeProcess>
        </soap:Body>
</soap:Envelope>
```

For Execute requests, the use of SOAP is only valid when ResponseForm is of type ResponseDocument. SOAP Execute requests which specify RawDataOutput shall generate a SOAP error.

An example of an Execute request follows.

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema">
        <soap:Body>

         <wps:Execute service="WPS" version="1.0.0"
          xmlns:wps="http://www.opengis.net/wps/1.0.0"
          xmlns:ows="http://www.opengis.net/ows/1.1"
          xmlns:xlink="http://www.w3.org/1999/xlink"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:schemaLocation="http://www.opengis.net/wps/1.0.0
          ..\wpsExecute_request.xsd">
              <ows:Identifier>Buffer</ows:Identifier>
              <wps:DataInputs>
                  <wps:Input>
                      <ows:Identifier>InputPolygon</ows:Identifier>
                      <ows:Title>Playground area</ows:Title>
                      <wps:Reference xlink:href=
                        "http://foo.bar/some_WFS_request.xml"/>
                  </wps:Input>
                  <wps:Input>
                      <ows:Identifier>BufferDistance</ows:Identifier>
                      <ows:Title>Distance which people will walk to get to a
                        playground.</ows:Title>
                      <wps:Data>
                          <wps:LiteralData>400</wps:LiteralData>
                      </wps:Data>
                  </wps:Input>
              </wps:DataInputs>
              <wps:ResponseForm>
                  <wps:ResponseDocument storeExecuteResponse="true">
                      <wps:Output asReference="true">
                          <ows:Identifier>BufferedPolygon</ows:Identifier>
                          <ows:Title>Area serviced by playground </ows:Title>
                          <ows:Abstract>Area within which most users of
                            this playground will live</ows:Abstract>
                      </wps:Output>
                  </wps:ResponseDocument>
              </wps:ResponseForm>
          </wps:Execute>

    </soap:Body>
</soap:Envelope>
```

2.14 Add section 10.2.2.1.2 - Explanation and example of HTTP GET encoding for WPS Execute requests

Starting on page 40, insert the following section in order to enhance clarity:

10.2.2.1.2 Explanation and example of HTTP GET encoding

Consider a WPS Execute request intended to pass the following parameters:

- Version = 1.0.0
- Language = en-CA
- Request = Execute
- Identifier = Buffer
- DataInputs
    - InputPolygon
        - xlink:href = http://foo.bar/some_WFS_request.xml
        - method = POST
        - mimeType = text/xml
        - encoding = UTF-8
        - schema = http://foo.bar/gml_polygon_schema.xsd
    - BufferDistance = 400
        - uom = feet
- ResponseDocument
    - BufferedPolygon
        - asReference = true
- storeExecuteResponse = true
- lineage = true
- status = true

For HTTP GET, this request would be encoded as follows:

http://foo.bar/foo?Version=1.0.0&Language=en-CA&Request=Execute&Identifier=Buffer&DataInputs=InputPolygon%3D%40xlink%3Ahref%3Dhttp%253A%252F%252Ffoo.bar%252Fsome_WFS_request.xml%40method%3DPOST%40mimeType%3Dtext%252Fxml%40encoding%3DUTF-8%40schema%3Dhttp%253A%252F%252Ffoo.bar%252Fgml_polygon_schema.xsd%3B%20BufferDistance%3D400%40uom%3Dfeet&ResponseDocument=BufferedPolygon%3D%40asReference%3Dtrue&storeExecuteResponse=true&lineage=true&status=true

Upon receipt of this request, the HTTP server will use the "?" character to identify the base URL from the list of parameters, the "&" character to identify the separation of one parameter from the next, and the "=" character to identify the separation of a parameter name from its value. This will result in the following assignment of values to parameters.

Parameter	Value
Version	1.0.0
Language	en-CA
Request	Execute
Identifier	Buffer
DataInputs	InputPolygon%3D%40xlink%3Ahref%3Dhttp%253A%252F%252Ffoo.bar%252Fsome_WFS_request.xml%40method%3DPOST%40mimeType%3Dtext%252Fxml%40encoding%3DUTF-8%40schema%3Dhttp%253A%252F%252Ffoo.bar%252Fgml_polygon_schema.xsd%3B%20BufferDistance%3D400%40uom%3Dfeet
ResponseDocument	BufferedPolygon%3D%40asReference%3Dtrue
storeExecuteResponse	true
lineage	true
status	true

For the *DataInputs* and *ResponseDocument* (or *RawDataOutput*) parameters, the WPS must then apply URL decoding, resulting in this case to the following assignment of values for these parameters:

Parameter	Decoded Value
DataInputs	InputPolygon=@xlink:href=http%3A%2F%2Ffoo.bar%2Fsome_WFS_request.xml@method=POST@mimeType=text%2Fxml@encoding=UTF-8@schema=http%3A%2F%2Ffoo.bar%2Fgml_polygon_schema.xsd; BufferDistance=400@uom=feet
ResponseDocument	BufferedPolygon=@asReference=true

For each of these parameters, the WPS will now use the first "=" character to identify the separation of an input/output name from its value, and the ";" character to identify the separation of one input/output value from the next, resulting in the following assignments:

Parameter	Input/Output Name	Value
DataInputs	InputPolygon	@xlink:href=http%3A%2F%2Ffoo.bar%2Fsome_WFS_request.xml@method=POST@mimeType=text%2Fxml@encoding=UTF-8@schema=http%3A%2F%2Ffoo.bar%2Fgml_polygon_schema.xsd
	BufferDistance	400@uom=feet
ResponseDocument	BufferedPolygon	@asReference=true

Next, within the value of each input/output, the "@" character shall be used to identify the separation of attributes from their input/output value and from one another, and the "=" character to separate an attribute name from its value. This produces the following assignments.

Parameter	Input/Output Name	Attribute Name	Value
DataInputs	InputPolygon		
		xlink:href	http%3A%2F%2Ffoo.bar%2Fsome_WFS_request.xml
		method	POST
		mimeType	text%2Fxml
		encoding	UTF-8
		schema	http%3A%2F%2Ffoo.bar%2Fgml_polygon_schema.xsd
	BufferDistance		400
		uom	feet
Response Document	BufferedPolygon		
		asReference	true

Finally, for all attribute values, the WPS will apply URL decoding, revealing the intended values of each of these parameters:

Parameter	Input/Output Identifier	Attribute Name	Value
DataInputs	InputPolygon		
		xlink:href	http://foo.bar/some_WFS_request.xml
		method	POST
		mimeType	text/xml
		encoding	UTF-8
		schema	http://foo.bar/gml_polygon_schema.xsd
	BufferDistance		400
		uom	feet
Response Document	BufferedPolygon		
		asReference	true

Construction of a valid Execute request follows the reverse process, namely the client must:

1. Create URL-encoded representations for all Input and Output values as well as their associated attribute values.
2. For each Input/Output, concatenate its encoded value, an "@" character, the name of an attribute, an "=" character, and the encoded value of that attribute. Repeat from the "@" character for additional attributes as necessary.

3. Create the value of the DataInputs/ResponseDocument/RawDataOutput parameter by concatenate the first Input/Output identifier, an "=" character, and its concatenated value from step 2 above. Add inputs/outputs as required, separated by a ";" character.
4. URL-encode the strings resulting from step 3 above as part of the conventional practice of encoding the values of all parameters that form part of a URL.

