Open Geospatial Consortium Inc.

Date: 2005-09-16

Reference number of this document: OGC 05-007r4

Version: 0.4.0

Category: OpenGIS® Discussion Paper

Editors: Peter Schut, Arliss Whiteside

OpenGIS® Web Processing Service

Warning

This 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.

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:	OpenGIS® Discussion Paper
Document subtype:	Draft new Implementation Specification
Document stage:	Approved Discussion Paper
Document language:	English

Contents

Page

Figures

Page

Tables

Page

i. Preface

This document specifies the interface to a Web Processing Service (WPS). This document is the result of work undertaken to support the Canadian Geospatial Data Infrastructure (CGDI), and in particular the National Land and Water Information Service (NLWIS), and the National Forest Information Service (NFIS). The specification was first implemented as a prototype in 2004 by Agriculture and Agri-Food Canada (AAFC). In the first half of 2005, it was the subject of a successful OGC Interoperability Experiment

Suggested additions, changes, and comments on this recommendation paper 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. Document terms and definitions

This document uses the specification terms defined in Subclause 5.3 of [OGC 05-008], which is based on the ISO/IEC Directives, Part 2. Rules for the structure and drafting of International Standards. In particular, the word “shall” (not “must”) is the verb form used to indicate a requirement to be strictly followed to conform to this specification

iii. Submitting organizations

The following organizations submitted this document to the Open Geospatial Consortium Inc.

GeoConnections / Natural Resources Canada

PCI Geomatics

iv. Document contributor contact points

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

Name	Organization
Peter Schut	Agriculture and Agri-Food Canada
Xiaoyaun Geng	Agriculture and Agri-Food Canada
Maru Newby	Agriculture and Agri-Food Canada
Stephane Fellah	Image Matters LLC
Stephen Keens	PCI Geomatics
Weisheng Li	PCI Geomatics
Martin Kyle	Galdos Systems
Christian Kiehle	RWTH Aachen University
Christian Heier	Wupperverband
Mike Adair	Natural Resources Canada
Nicole Ostlaender	University of Muenster - Institute for Geoinformatics
Harald Borsutzky	University of Muenster - Institute for Geoinformatics
Arliss Whiteside	BAE Systems Electronics and Integrated Solutions

v. Revision history

Date	Release	Editor	Primary clauses modified	Description
05 May 2004	0.1.0	Peter. Schut	All	Initial document, formatted for OGC template
22 May 2004	0.1.0	Peter Schut	All	Cleaned up some problems, added informative examples in Annex B
21 Oct. 2004	0.2.0	Stephane Fellah	Content	Rewrite the schema and the Table of Contents
22 Nov. 2004	0.2.0	Xiaoyuan Geng	All	Created document using the latest OGC template, the initial draft, and schema
24 Dec. 2004	0.2.1	Peter Schut	All	Minor corrections and revisions throughout, additions of human readable explanations of schemas
11 April 2005	0.2.3	Peter Schut	All	Upgrade based on results to date of WPSie.
05 April 2005	0.3.0	Peter Schut	All	Upgrade based on results to date of WPSie and alignment with OWS Common
13 July 2005	0.4.0	Peter Schut	6 & 7	Complete documentation of each element and renaming of elements to eliminate confusion caused by abstractions
1 Sept 2005	0.4.0	Arliss Whiteside	All	Added UML diagrams, aligned with new schemas.

Date	Release	Editor	Primary clauses modified	Description
16 Sept 2005	0.4.0	Peter Schut and Arliss Whiteside	All	Final editing and cleanup

vi. Changes to the OGC Abstract Specification

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

vii. Future work

Improvements in this document seem desirable to:

a) Improve or delete use of HTTP GET transfer of the Execute operation request.

b) Specify a way to reference values of possible process inputs that are maintained by a WPS server, and a way for clients to find those possible process input values.

NOTE 1 Version 0.1.0 included the ability to identify local datasets that could be used as inputs to a process. The current version 0.4.0 allows a local payload to be used by using a "reference" URL that starts with the prefix "cid:", but this is not documented. The ability for a server to provide local datasets should be reinstated, or an alternative specified.

c) Add more parameters to the ProcessDescription data structure, returned by the DescribeProcess operation, such as more parameters to elucidate the input and output meanings. For example, the Meaning data structure defined in the DomainType UML package might be added. Alternately, the meanings of input and output Titles or Identifiers might be defined on a shared site or registry using RDF.

d) Add more parameters to the ExecuteResponse data structure, such as more parameters for lengthy processes in the Status alternatives (instead of relying on unspecified contents in the status messages).

e) Specify specific uses of the Metadata contents or reference, in the ProcessBrief data structure returned by the DescribeProcess and GetCapabilities operations. For example, keywords could be defined and included to indicate the process "Application" and "Category".

NOTE 2 Earlier versions included lists of keywords in the ProcessBrief data structure, to specify the process "Application" and "Category". However, no definitive list of such keywords was agreed upon, and maintaining them in the WPS specification is problematic. It may be better if this list were maintained separately in RDF on the OGC website

Foreword

The Web Processing Service (WPS) was originally named Geoprocessing Service (OGC document number 04-043). This document replaces that early draft document.

This document includes four annexes; Annexes A, B, and D are normative, and Annex C is informative.

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

Introduction

This document specifies the interface to a Web Processing Service (WPS). A WPS can be configured to offer any sort of GIS functionality to clients across a network, including access to pre-programmed calculations and/or computation models that operate on spatially referenced data. A WPS may offer calculations as simple as subtracting one set of spatially referenced numbers from another (e.g., determining the difference in influenza cases between two different seasons), or as complicated as a global climate change model. The data required by the WPS can be delivered across a network, or available at the server.

This interface specification provides mechanisms to identify the spatially-referenced data required by the calculation, initiate the calculation, and manage the output from the calculation so that it can be accessed by the client. This Web Processing Service is targeted at processing both vector and raster data.

OpenGIS® Web Processing Service

1 Scope

This document specifies the interface to a general purpose Web Processing Service (WPS). A WPS provides client access across a network to pre-programmed calculations and/or computation models that operate on spatially referenced data. The calculation can be extremely simple or highly complex, with any number of data inputs and outputs.

This document does not specify the specific processes that could be implemented by a WPS. Instead, it specifies a generic mechanism that can be used to describe and web-enable any sort of geospatial process. To achieve interoperability, each process must be specified in a separate document, which might be called an Application Profile of this specification.

This document does not specify any specific data required or output by the WPS. Instead, it identifies a generic mechanism to describe the data inputs required and produced by a process. This data can be delivered across the network, or available at the server. This data can include image data formats such as GeoTIFF, or data exchange standards such as Geography Markup Language (GML) or Geolinked Data Access Service (GDAS).

This document does not address the archival, cataloguing, discovery, or retrieval of information that has been created by a WPS.

2 Conformance

Conformance with this specification shall be checked using all the relevant tests specified in Annex A (normative).

3 Normative references

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

ISO 19105:2000, *Geographic information — Conformance and Testing*

OGC 05-059r1, *OWS Common 1.0 Change request – Make element and type names public*

OGC 05-008, *OpenGIS® Web Services Common Specification*

This OWS Common Specification contains a list of normative references that are also applicable to this Implementation Specification.

In addition to this document, this draft specification includes several normative XML Schema Document files as specified in Annex B.

4 Terms and definitions

For the purposes of this specification, the definitions specified in Clause 4 of the OWS Common Implementation Specification [OGC 05-008] shall apply. In addition, the following terms and definitions apply.

4.1
input
data provided to a **process**

4.2
literal
any process input or output whose value can be represented in a character string, supplemented by metadata as needed

4.3
literal (XML encoding)
any process input or output whose value can be represented in a xsd:string supplemented by XML attributes as needed

NOTE A literal process input or output can be a character string, integer, general number, URI, measure, etc.

4.4
map
pictorial representation of geographic data

4.5
process
model or calculation that is made available at a **service instance**

4.6
output
result returned by a **process**

5 Conventions

5.1 Abbreviated terms

Most of the abbreviated terms listed in Subclause 5.1 of the OWS Common Implementation Specification [OGC 05-008] apply to this document, plus the following abbreviated terms.

CGDI Canadian Geospatial Data Infrastructure

GDAS Geolinked Data Access Service

5.2 UML notation

Most diagrams that appear in this specification are presented using the Unified Modeling Language (UML) static structure diagram, as described in Subclause 5.2 of [OGC 05-008].

5.3 Used parts of other documents

This document uses significant parts of document [OGC 05-008]. To reduce the need to refer to that document, this document copies some of those parts with small modifications. To indicate those parts to readers of this document, the largely copied parts are shown with a light grey background (15%).

5.4 Platform-neutral and platform-specific specifications

As specified in Clause 10 of OpenGIS® Abstract Specification Topic 12 "OpenGIS Service Architecture" (which contains ISO 19119), this document includes both Distributed Computing Platform-neutral and platform-specific specifications. This document first specifies each operation request and response in platform-neutral fashion. This is done using a table for each data structure, which lists and defines the parameters and other data structures contained. These tables serve as data dictionaries for the UML model in Annex C, and thus specify the UML model data type and multiplicity of each listed item.

EXAMPLES 1 Platform-neutral specifications are contained in Subclauses 8.3.1, 8.3.3, 9.2.1, 9.3.1, 10.2.1, 10.3.1, and 10.3.2.

The specified platform-neutral data could be encoded in many alternative ways, each appropriate to one or more specific DCPs. This document now specifies only encoding appropriate for use of HTTP GET transfer of operations requests (using KVP encoding), and for use of HTTP POST transfer of operations requests (using XML or KVP encoding). However, the same operation requests and responses (and other data) could be encoded for other specific computing platforms, including SOAP/WSDL.

EXAMPLES 2 Platform-specific specifications for KVP encoding are contained in Subclauses 9.2.1 and 10.2.2.

EXAMPLES 3 Platform-specific specifications for XML encoding are contained in Subclauses 8.3.2, 8.3.4, 9.2.3, 9.3.2, 10.2.3, and 10.3.3.

6 WPS overview

The specified Web Processing Service (WPS) provides client access to pre-programmed calculations and/or computation models that operate on spatially referenced data. The data required by the service can be delivered across a network, or available at the server. This data can use image data formats or data exchange standards such as Geography Markup Language (GML) or Geolinked Data Access Service (GDAS). The calculation can be as simple as subtracting one set of spatially referenced numbers from another (e.g. determining the difference in influenza cases between two different seasons), or as complicated as a global climate change model.

The WPS interface specifies three operations that can be requested by a client and performed by a WPS server, all mandatory implementation by all servers. Those operations are:

a) GetCapabilities – This operation allows a client to request and receive back service metadata (or Capabilities) documents that describe the abilities of the specific server implementation. This operation also supports negotiation of the specification version being used for client-server interactions.

b) DescribeProcess – This operation allows a client to request and receive back detailed information about one or more process(es) that can be executed by an Execute operation, including the input parameters and formats, and the outputs.

c) Execute – This operation allows a client to run a specified process implemented by the WPS, using provided input parameter values and returning the outputs produced.

These operations have many similarities to other OGC Web Services, including the WMS, WFS, and WCS. Many of these interface aspects that are common with other OWSs are thus specified in the OpenGIS® Web Services Common Implementation Specification [OGC 05-008]. Many of these common aspects are normatively referenced herein, instead of being repeated in this specification.

Figure 1 is a simple UML diagram summarizing the WPS interface. This class diagram shows that the WPS interface class inherits the getCapabilities operation from the OGCWebService interface class, and adds the DescribeProcess and Execute operations. (This capitalization of names uses the OGC/ISO profile of UML.) A more complete UML model of the WPS interface is provided in Annex C (informative).

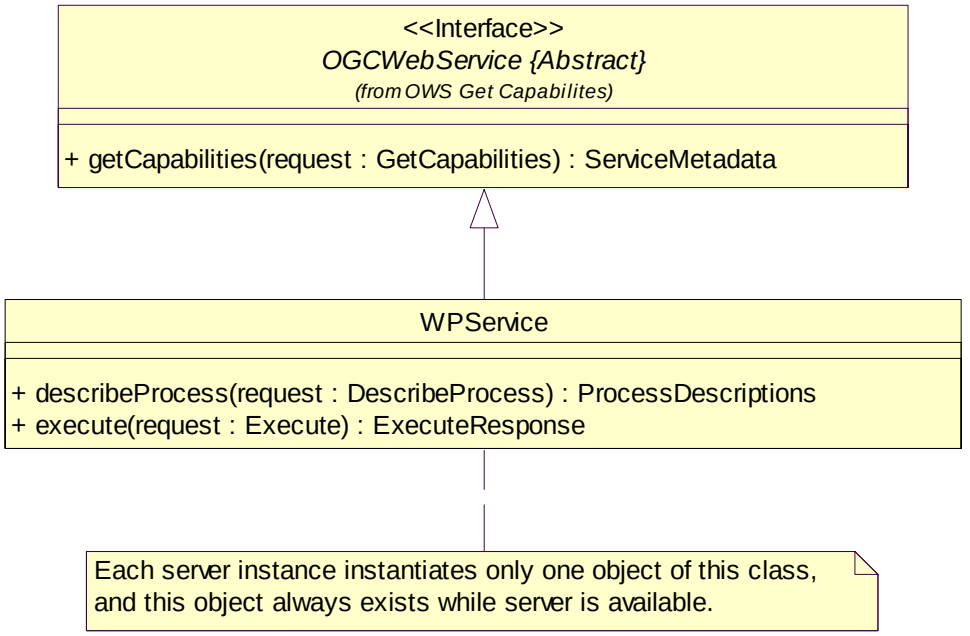

Figure 1 — WPS interface UML diagram

NOTE In this UML diagram, the request and response for each operation is shown as a single parameter that is a data structure containing multiple lower-level parameters, which are discussed in subsequent clauses. The UML classes modelling these data structures are included in the complete UML model in Annex C.

Each of the three WPS operations is described in more detail in subsequent clauses.

7 Shared aspects

7.1 Introduction

This clause specifies aspects of WPS behavior that are shared by multiple operations.

7.2 Shared data structures

This clause specifies some of the data structures and parameters used by multiple operation requests and responses specified in the following clauses. The data structure names, parameter names, meanings, data types, and multiplicity shall be as specified in Table 1 and Table 2.

NOTE 1 The 3 parameters listed below (with partial grey backgrounds) are partially copied from Table 23 in Subclause 10.6.1 of [OGC05-008].

Table 1 — Parameters in Description data structure

Name	Definition	Data type and value	Multiplicity and use
Identifier	Unambiguous identifier or name of a process, input, or output, unique for this server	ows:CodeType, as adaptation of MD_Identifier class in ISO 19115	One (mandatory)
Title	Title of a process, input, or output, normally available for display to a human	Character string type, not empty	One (mandatory)
Abstract	Brief narrative description of a process, input, or output, normally available for display to a human	Character string type, not empty	Zero or one (optional) Include when available and useful

Table 2 — Parts of ProcessBrief data structure

Name	Definition	Data type and value	Multiplicity and use
Identifier	Inherited from Description data structure, see Table 1, applied to a process	ows:CodeType	One (mandatory)
Title		Character string type	One (mandatory)
Abstract		Character string type	Zero or one (optional)
Metadata	Reference to more metadata about this process	ows:Metadata, see Table 23 of OGC 05-008	Zero or one (optional) Include when useful
process Version	Release version of process (not of WPS specification)	ows:VersionType, see OGC 05-008	Zero or one (optional) Include when needed to identify process version

7.3 Operation request encoding

The encoding of operation requests shall use HTTP GET with KVP encoding and HTTP POST with XML encoding as specified in Clause 11 of [OGC 05-008]. Table 3 summarizes the three Service operations and their encoding methods defined in this specification.

Table 3 — Operation request encoding

Operation name	Request encoding
GetCapabilities (mandatory)	KVP
DescribeProcess (mandatory)	KVP and optional XML
Execute (mandatory)	XML and optional KVP

8 GetCapabilities operation (mandatory)

8.1 Introduction

The mandatory GetCapabilities operation allows clients to retrieve service metadata from a server. The response to a GetCapabilities request shall be a XML document containing service metadata about the server, including brief metadata describing all the processes implemented. This clause specifies the XML document that a WPS server must return to describe its capabilities.

8.2 Operation request

The GetCapabilities operation request shall be as specified in Subclauses 7.2 and 7.3 of [OGC 05-008]. The value of the "service" parameter shall be "WPS". The allowed set of service metadata (or Capabilities) XML document section names and meanings shall be as specified in Table 4.

NOTE All the sections listed below except ProcessOfferings are copied from Table 3 in [OGC 05-008]

Table 4 — Section name values and meanings

Section name	Meaning
ServiceIdentification	Return ServiceIdentification element in service metadata document
ServiceProvider	Return ServiceProvider metadata element in service metadata document
OperationsMetadata	Return OperationsMetadata element in service metadata document
ProcessOfferings	Return ProcessOfferings metadata element in service metadata document
All	Return complete service metadata document, containing all elements

The "Multiplicity and use" column in Table 1 of [OGC 05-008] specifies the optionality of each listed parameter in the GetCapabilities operation request. Table 5 specifies the implementation of those parameters by WPS clients and servers.

Table 5 — Implementation of parameters in GetCapabilities operation request

Name	Multiplicity	Client implementation	Server implementation
service	One (mandatory)	Each parameter shall be implemented by all clients, using specified value.	Each parameter shall be implemented by all servers, checking that each parameter is received with specified value.
request	One (mandatory)		
AcceptVersions	Zero or one (optional)	Should be implemented by all software clients, using specified values.	Shall be implemented by all servers, checking if parameter is received with specified value(s).
Sections	Zero or one (optional)	Each parameter should NOT be implemented by clients.	Each parameter shall NOT be implemented by all servers. No output format other than XML is allowed.
updateSequence	Zero or one (optional)		
AcceptFormats	Zero or one (optional)		

All WPS servers shall implement HTTP GET transfer of the GetCapabilities operation request, using KVP encoding. HTTP POST transfer of the GetCapabilities operation request shall NOT be implemented by WPS servers.

EXAMPLE To request a WPS capabilities document, a client could issue the following KVP encoded GetCapabilities operation request with near-minimum contents:

```
http://foo.bar/foo?
    service=WPS&
    request=getCapabilities&
    AcceptVersions="0.4.0"
```

8.3 GetCapabilities operation response

8.3.1 Normal response

The service metadata document shall contain the four sections specified in Table 6. Depending on the values in the Sections parameter of the GetCapabilities operation request, any combination of these sections can be requested and shall be returned when requested.

NOTE The first three sections listed in the following table are largely copied from Table 7 in Subclause 7.4.2 of [OGC 05-008].

Table 6 — Section name values and contents

Section name	Contents
ServiceIdentification	Metadata about this specific server. The schema of this section shall be the same as for all OWSs, as specified in Subclause 7.4.3 and owsServiceIdentification.xsd of [OGC 05-008].
ServiceProvider	Metadata about the organization operating this server. The schema of this section shall be the same for all OWSs, as specified in Subclause 7.4.4 and owsServiceProvider.xsd of [OGC 05-008].
OperationsMetadata	Metadata about the operations specified by this service and implemented by this server, including the URLs for operation requests. The basic contents and organization of this section shall be almost the same as for all OWSs, as specified in Subclause 7.4.5 and owsOperationsMetadata.xsd of [OGC 05-008], modified as specified in Subclause 7..3.2 below
ProcessOfferings	Unordered list of brief descriptions of the processes offered by this WPS server, as specified in Subclause 8.3.3 below.

In addition to these sections, each service metadata document shall include the mandatory “version” and optional updateSequence parameters specified in Table 6 in Subclause 7.4.1 of [OGC 05-008].

8.3.2 OperationsMetadata section contents

For the WPS, the OperationsMetadata section shall be almost the same as for all OGC Web Services, as specified in Subclause 7.4.5 and owsOperationsMetadata.xsd of [OGC 05-008]. The only change shall be substitution of the ows:DomainType from the owsDomainType.xsd for the ows:DomainType in ows:OperationsMetadata.xsd of [OGC 05-008].

In the OperationsMetadata section, the mandatory values of various (XML) attributes shall be as specified in Table 7. In Table 7, the "Attribute name" column uses dot-separator notation to identify parts of a parent item. The "Attribute value**"** column references an operation parameter, in this case an operation name, and the meaning of including that value is listed in the right column.

Table 7 — Required values of OperationsMetadata section attributes

Attribute name	Attribute value	Meaning of attribute value
Operation.name	GetCapabilities	The GetCapabilities operation is implemented by this server.
	DescribeProcess	The DescribeProcess operation is implemented by this server.
	Execute	The Execute operation is implemented by this server.

In addition to the required values listed in Table 5, there are many optional values of the "name" attributes and "value" elements in the OperationsMetadata section, which may be included when considered useful. Most of these attributes and elements are for recording the domains of various parameters and quantities.

EXAMPLE 1 The domain of the exceptionCode parameter could record all the codes implemented for each operation by that specific server. Similarly, each of the GetCapabilities operation optional request parameters might have its domain recorded.

8.3.3 ProcessOfferings section

The ProcessOfferings section of a WPS service metadata document shall contain a brief description of each of the processes offered by the service. More specifically, the ProcessOfferings section shall include the subsections specified in Table 8.

Table 8 — Parts of ProcessOfferings section

Name	Definition	Data type	Multiplicity and use
Process Brief	Brief description of process, not including input and output parameters	ProcessBrief data structure, see Table 2	One or more (mandatory) One for each process implemented by server

NOTE The UML class diagram contained in Subclause C.4 provides a graphical view of the contents of the ProcessOfferings section listed in Table 8.

The "Multiplicity and use" columns in Table 6 through Table 16 in [OGC 05-008], and in Table 1, Table 2, and Table 6 through Table 8 of this document, specify the optionality of each listed parameter and data structure in the GetCapabilities operation response. All the "mandatory" parameters and data structures shall be implemented by all WPS servers, using a specified value(s).

As indicated in Table 5 of this document, the "updateSequence", Sections, and Accept Formats parameters defined in Table 6 of [OGC 05-008] shall not be implemented by WPS servers. The GetCapabilities operation response from a server shall thus always be encoded in XML, contain all sections, and not contain the "updateSequence" parameter. All WPS clients shall be implemented to allow such GetCapabilities responses.

All other “optional” parameters and data structures, in the GetCapabilities operation response, should be implemented by all OWS servers using specified values, whenever and wherever each is considered useful metadata for that server.

8.3.4 Capabilities document XML encoding

A XML schema fragment for a WPS service metadata document is:

```
   <element name="Capabilities">
      <annotation>
         <documentation>WPS GetCapabilities operation response. This
document provides clients with service metadata about a specific
service instance, including metadata about the processes that can be
executed. Since the server does not implement the updateSequence and
Sections parameters, the server shall always return the complete
Capabilities document, without the updateSequence parameter.
</documentation>
      </annotation>
      <complexType>
         <complexContent>
            <extension base="ows:CapabilitiesBaseType">
               <sequence>
                  <element ref="wps:ProcessOfferings"/>
               </sequence>
            </extension>
         </complexContent>
      </complexType>
   </element>
   <!-- ========================================================== -->
   <element name="ProcessOfferings">
      <annotation>
         <documentation>List of brief descriptions of the processes
offered by this WPS server. </documentation>
      </annotation>
      <complexType>
         <sequence>
            <element name="Process" type="wps:ProcessBriefType"
maxOccurs="unbounded">
               <annotation>
                  <documentation>Unordered list of one or more brief
descriptions of all the processes offered by this WPS server.
</documentation>
               </annotation>
            </element>
         </sequence>
      </complexType>
   </element>
```

As indicated, this XML schema fragment extends ows:CapabilitiesBaseType in owsCommon.xsd of [OGC 05-008], which uses the owsServiceIdentification.xsd, owsServiceProvider.xsd, and owsOperationsMetadata.xsd schemas specified in [OGC 05-008]. Specific to the WPS is the ProcessOfferings section, which contains brief descriptions of the one or more processes that can be executed by this server. That section uses the following XML schema fragment:


```
   <complexType name="DescriptionType">
      <annotation>
         <documentation>Description of a WPS process, input, or output
object. </documentation>
      </annotation>
      <sequence>
         <element ref="ows:Identifier">
            <annotation>
               <documentation>Unambiguous identifier or name of a
process, unique for this server, or unambiguous identifier or name of
an input or output, unique for this process. </documentation>
            </annotation>
         </element>
         <element ref="ows:Title">
            <annotation>
               <documentation>Title of a process, input, or output,
normally available for display to a human. </documentation>
            </annotation>
         </element>
         <element ref="ows:Abstract" minOccurs="0">
            <annotation>
               <documentation>Brief narrative description of a
process, input, or output, normally available for display to a human.
</documentation>
            </annotation>
         </element>
      </sequence>
   </complexType>
   <!-- ================================================== -->
   <complexType name="ProcessBriefType">
      <annotation>
         <documentation>Brief description of a Process, designed for
Process discovery. </documentation>
      </annotation>
      <complexContent>
         <extension base="wps:DescriptionType">
            <sequence>
               <element ref="ows:Metadata" minOccurs="0"
maxOccurs="unbounded">
                  <annotation>
                     <documentation>Optional unordered list of
additional metadata about this process. A list of optional and/or
required metadata elements for this process could be specified in a
specific Application Profile for this service. </documentation>
                  </annotation>
               </element>
            </sequence>
            <attribute name="processVersion" type="ows:VersionType"
use="optional">
               <annotation>
                  <documentation>Release version of this Process,
included when a process version needs to be included for clarification
about the process to be used. It is possible that a WPS supports a
process with different versions due to reasons such as modifications of
process algorithms. Notice that this is the version identifier for the
process, not the version of the WPS interface. </documentation>
               </annotation>
            </attribute>
```



```
         </extension>
      </complexContent>
   </complexType>
```

This schema fragment is part of the attached normative wpsCommon.xsd XML Schema Document. All these XML Schema Documents contain documentation of the meaning of each element, attribute, and type, and this documentation shall be considered normative as specified in Subclause 11.6.3 of [OGC 05-008].

An example of a ProcessOfferings section is included at the end of the following Capabilities document example. In order to obtain detailed information about a process, the DescribeProcess operation can be used.

8.3.5 Capabilities document example

In response to GetCapabilities operation request, a WPS server might generate a document that looks like:

```
<?xml version="1.0" encoding="UTF-8"?>
<Capabilities xmlns:xlink="http://www.w3.org/1999/xlink"
version="0.4.0" xmlns="http://www.opengeospatial.net/wps"
xmlns:ows="http://www.opengeospatial.net/ows"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opengeospatial.net/wps
..\wpsGetCapabilities.xsd">
   <ows:ServiceIdentification>
      <ows:Title>AAFC GDAS-based WPS server</ows:Title>
      <ows:Abstract>AAFC GDAS-based WPS server developed for the OGC
WPSie.</ows:Abstract>
      <ows:Keywords>
         <ows:Keyword>WPS</ows:Keyword>
         <ows:Keyword>AAFC</ows:Keyword>
         <ows:Keyword>geospatial</ows:Keyword>
         <ows:Keyword>geoprocessing</ows:Keyword>
      </ows:Keywords>
      <ows:ServiceType>WPS</ows:ServiceType>
      <ows:ServiceTypeVersion>0.2.0</ows:ServiceTypeVersion>
      <ows:ServiceTypeVersion>0.1.0</ows:ServiceTypeVersion>
      <ows:Fees>NONE</ows:Fees>
      <ows:AccessConstraints>NONE</ows:AccessConstraints>
   </ows:ServiceIdentification>
   <ows:ServiceProvider>
      <ows:ProviderName>Agriculture and Agri-Food
Canada</ows:ProviderName>
      <ows:ProviderSite xlink:href="http://gis.agr.gc.ca/"/>
      <ows:ServiceContact>
         <ows:IndividualName>Peter Schut</ows:IndividualName>
         <ows:PositionName>Information System
Scientist</ows:PositionName>
         <ows:ContactInfo>
            <ows:Phone>
               <ows:Voice>+1 613 759-1874</ows:Voice>
               <ows:Facsimile>+1 613 759-1937</ows:Facsimile>
            </ows:Phone>
            <ows:Address>
```



```
                  <ows:DeliveryPoint>Room 1135, Neatby Building, 960,
Carling Avenue</ows:DeliveryPoint>
                  <ows:City>Ottawa</ows:City>
                  <ows:AdministrativeArea>ON</ows:AdministrativeArea>
                  <ows:PostalCode>K1AOC6</ows:PostalCode>
                  <ows:Country>Canada</ows:Country>

   <ows:ElectronicMailAddress>schutp@agr.gc.ca</ows:ElectronicMailAddre
ss>
               </ows:Address>
            </ows:ContactInfo>
         </ows:ServiceContact>
   </ows:ServiceProvider>
   <ows:OperationsMetadata>
      <ows:Operation name="GetCapabilities">
         <ows:DCP>
            <ows:HTTP>
               <ows:Get
xlink:href="http://wms1.agr.gc.ca/GeoPS/GeoPS?"/>
            </ows:HTTP>
         </ows:DCP>
      </ows:Operation>
      <ows:Operation name="DescribeProcess">
         <ows:DCP>
            <ows:HTTP>
               <ows:Get
xlink:href="http://wms1.agr.gc.ca/GeoPS/GeoPS?"/>
               <ows:Post
xlink:href="http://wms1.agr.gc.ca/GeoPS/GeoPS"/>
            </ows:HTTP>
         </ows:DCP>
      </ows:Operation>
      <ows:Operation name="Execute">
         <ows:DCP>
            <ows:HTTP>
               <ows:Get
xlink:href="http://wms1.agr.gc.ca/GeoPS/GeoPS?"/>
               <ows:Post
xlink:href="http://wms1.agr.gc.ca/GeoPS/GeoPS"/>
            </ows:HTTP>
         </ows:DCP>
      </ows:Operation>
   </ows:OperationsMetadata>
   <ProcessOfferings>
      <Process processVersion="1.0">
         <ows:Identifier>buffer</ows:Identifier>
         <ows:Title>Buffer a polygon feature</ows:Title>
         <ows:Abstract>Buffer  the polygon coordinates found in one
GML stream by a given buffer distance, and output the results in
GML.</ows:Abstract>
         <ows:Metadata xlink:title="buffer" />
         <ows:Metadata xlink:title="polygon" />
      </Process>
   </ProcessOfferings>
</Capabilities>
```

8.3.6 GetCapabilities exceptions

When a WPS server encounters an error while performing a GetCapabilities operation, it shall return an exception report message as specified in Clause 8 of [OGC 05-008]. The allowed exception codes shall include those listed in Table 5 of Subclause 7.4.1 of [OGC 05-008], if the updateSequence parameter is implemented by the server.

9 DescribeProcess operation (mandatory)

9.1 Introduction

The mandatory DescribeProcess operation allows WPS clients to request and receive back a full description of one or more processes that can be executed by the Execute operation. This description includes the input parameters and formats, plus the output formats. This description can be used to automatically build a user interface to capture the parameter values to be used to execute a process instance.

9.2 DescribeProcess operation request

9.2.1 DescribeProcess request parameters

A request to perform the DescribeProcess operation shall include the parameters listed and defined in Table 9. This table specifies the UML data type, source of values, and multiplicity of each listed parameter, plus the meaning to servers when each optional parameter is not included in the operation request. Although some values listed in the "Name" column appear to contain spaces, they shall not contain spaces.

NOTE 1 To reduce the need for readers to refer to other documents, the first three parameters listed below are largely copied from Table 21 in Subclause 9.2.1 of [OGC 05-008]. The Identifier parameter is partially copied from Table 23 in Subclause 10.6.1 of that document.

Table 9 — Parameters in DescribeProcess operation request

Name [a]	Definition	Data type and value	Multiplicity and use
service	Service type identifier	Character String type Value is "WPS"	One (mandatory)
request	Operation name	Character String type Value is "GetCapabilities"	One (mandatory)
version	Specification version for operation	Character String type, not empty Value is specified by each WPS Implementation Specification and Schemas version	One (mandatory)
Identifier	Process identifier	Character String type, not empty Value is process Identifier defined in ProcessOfferings section of Capabilities document	One or more (mandatory) One for each desired Process, unordered list
a The name capitalization rules being used here are specified in Subclause 11.6.2 of [OGC 05-008].			

NOTE 2 The data type of many parameters is specified as “Character String type, not empty”. In the XML Schema Documents specified herein, these parameters are encoded with the xsd:string type, which does NOT require that these strings not be empty.

NOTE 3 The UML class diagrams contained in Subclause C.5 provides a graphical view of the contents of the DescribeProcess operation request listed in Table 9.

The “Multiplicity and use” column in Table 9 specifies the optionality of each parameter in the DescribeProcess operation request. Since all parameters are mandatory in this operation request, each parameter shall be implemented by all WPS clients, using a specified value(s). Similarly, each parameter shall be implemented by all WPS servers, checking that this request parameter is received with any allowed value(s).

9.2.2 DescribeProcess request KVP encoding (mandatory)

All WPS servers shall implement HTTP GET transfer of the DescribeProcess operation request, using KVP encoding. The KVP encoding of the DescribeProcess operation request shall use the parameters specified in Table 10. The parameters listed in Table 10 shall be as specified in Table 9 above.

Table 10 — DescribeProcess operation request URL parameters

Name and example [a]	Optionality	Definition and format
service=WPS	Mandatory	Service type identifier
request= DescribeProcess	Mandatory	Operation name
version=0.4.0	Mandatory	WPS specification and schema version for this operation
Identifier=intersection,union	Mandatory	List of one or more process identifiers as listed in Capabilities document, separated by commas

a All parameter names are here listed using mostly lower case letters. However, any parameter name capitalization shall be allowed in KVP encoding, see Subclause 11.5.2 of [OGC 05-008].

EXAMPLE An example DescribeProcess operation request KVP encoded for HTTP GET is:

http://foo.bar/foo?
Service=”WPS”&
Request=”DescribeProcess”&
Version=”0.4.0”&
Identifier=”intersection,union”

9.2.3 DescribeProcess request XML encoding (optional)

WPS servers may also implement HTTP POST transfer of the DescribeProcess operation request, using XML encoding only. The following schema fragment specifies the contents and structure of a DescribeProcess operation request encoded in XML:

```
   <element name="DescribeProcess">
      <annotation>
         <documentation>WPS DescribeProcess operation request.
</documentation>
      </annotation>
      <complexType>
         <complexContent>
            <extension base="wps:RequestBaseType">
```



```
                <sequence>
                   <element ref="ows:Identifier"
maxOccurs="unbounded">
                      <annotation>
                         <documentation>Unordered list of one or more
identifiers of the processes for which the client is requesting
detailed descriptions. This element shall be repeated for each process
for which a description is requested. These Identifiers are unordered,
but the WPS shall return the descriptions in the order in which they
were requested. </documentation>
                      </annotation>
                   </element>
                </sequence>
             </extension>
          </complexContent>
       </complexType>
   </element>
```

EXAMPLE An example DescribeProcess operation request XML encoded for HTTP POST is:

```
<?xml version="1.0" encoding="UTF-8"?>
<DescribeProcess service="WPS" version="0.4.0"
xmlns="http://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>
</DescribeProcess>
```

9.3 DescribeProcess operation response

9.3.1 DescribeProcess response parameters

The normal response to a valid DescribeProcess operation request shall be a ProcessDescriptons data structure, which contains one or more Process Descriptions for the requested process identifiers. Each Process Description includes the brief information returned in the ProcessOfferings section of the service metadata (Capabilities) document, plus descriptions of the input and output parameters. Each process can have any number of input and output parameters.

Each parameter is described by a data structure that specifies the allowable formats, encodings, and units of measure (when applicable). For each input parameter, the process can indicate that it needs one of the following:

a) “ComplexData” (such as XML or imagery), in one of the following allowable combinations of format, encoding, and schema. The value of this complex data structure can be either directly encoded in the Execute operation request

b) “LiteralData”, with a specified “DataType”, allowed values, “DefaultValue” and “SupportedUOMs” indicated.

c) Bounding Box information, using one of the supported coordinate reference systems.

For each output parameter, the process can indicate similar information about the corresponding forms of output parameters. Again, there are three types of process outputs: Complex, Literal, and BoundingBox.

More precisely, a response from the DescribeProcess operation shall be a ProcessDescriptons data structure that includes one or more ProcessDescription data structures, as listed in Table 11. The ProcessDescription data structure shall include the parts specified in Table 12 through Table 25. All these tables specify the UML model data type plus the multiplicity and use of each listed part in the DescribeProcess operation response.

Table 11 — Parts of ProcessDescriptions data structure

Name	Definition	Data type	Multiplicity and use
Process Description	Full description of process, including all input and output parameters	ProcessDescription data structure, see Table 12	One or more (mandatory) One for each Process identified in operation request

NOTE 1 The UML class diagrams contained in Subclause C.5 provide a useful graphical view of the contents of the ProcessDescriptons contents listed in Table 11 through Table 25.

NOTE 2 The first 6 parameters listed below (with partially grey background) are copied from Table 2 in Subclause 7.2 of this document.

Table 12 — Parts of ProcessDescription data structure

Name	Definition	Data type and values	Multiplicity and use
Identifier	Inherited from ProcessBrief data structure, see Table 2	ows:CodeType	One (mandatory)
Title		Character string type	One (mandatory)
Abstract		Character string type	Zero or one (optional)
Metadata		ows:Metadata	One (mandatory)
processVersion		ows:VersionType	Zero or one (optional)
ProcessInputs	List of the required and optional inputs to this process	ProcessInputs data structure, see Table 13	Zero or one (optional) Include if any inputs [a]
ProcessOutputs	List of the required and optional outputs from executing this process	ProcessOutputs data structure, see Table 14	One (mandatory)
storeSupported	Indicates if all complex data output(s) from this process can be stored by WPS server as web-accessible resources	Boolean type Values are: true and false Default is false (return directly in response)	Zero or one (optional) Include when storage of outputs supported [b]
statusSupported	Indicates if Execute operation response can be returned quickly with status information	Boolean type Values are: true and false Default is false [c]	Zero or one (optional) Include when return of status data supported [d]

a In almost all cases, at least one process input is required. However, no process inputs may be identified when all the inputs are predetermined fixed resources. In this case, those resources shall be identified in the Abstract parameter that describes the process.

b If "storeSupported" is "true", the Execute operation request may include "store" equals "true", directing that all complex data output(s) of the process be stored so that the client can retrieve them as required. The “storeSupported” parameter value “true” is recommended to make service chaining more efficient when the output(s) are large.

c By default, status information is not provided for this process, and the Execute operation response is not returned until process execution is complete.

d If "statusSupported" is "true", the Execute operation request may include "status" equals "true", directing that the Execute operation response be returned quickly with status information. The “statusSupported” parameter value “true” is recommended when a process takes a long time to execute.

Table 13 — Parts of ProcessInputs data structure

Name	Definition	Data type and values	Multiplicity and use
Input	Description of mandatory or optional input to this process	InputDescription data structure, see Table 15	One or more (mandatory) Include one for each possible process input, unordered

Table 14 — Parts of ProcessOutputs data structure

Name	Definition	Data type and values	Multiplicity and use
Output	Description of mandatory or optional output from executing this process	OutputDescription data structure, see Table 17	One or more (mandatory) Include one for each possible process output, unordered

NOTE 3 The first 3 parameters listed below (with partially grey background) are copied from Table 1 in Subclause 7.2 of this document.

Table 15 — Parts of InputDescription data structure

Name	Definition	Data type and values	Multiplicity and use
Identifier	Inherited from Description data structure, see Table 1, applied to an input	ows:CodeType	One (mandatory)
Title		Character string type	One (mandatory)
Abstract		Character string type	Zero or one (optional)
Minimum Occurs	Minimum number of times that values for this parameter are required	Integer type Values "0" or "1" Default is "1"	Zero or one (optional) Include when input is optional
InputForm Choice	Identifies the type of this input, and provides supporting information	InputFormChoice data structure, see Table 16	One (mandatory)

Table 16 — Parts of InputFormChoice data structure

Name	Definition	Data type	Multiplicity
ComplexData	Indicates that this input shall be a complex data structure (such as a GML fragment), and provides lists of formats, encodings, and schemas supported [b]	Supported-ComplexData data structure, see Table 19	Zero or one (conditional) [a]
LiteralData	Indicates that this input shall be a simple literal value (such as an integer) that is embedded in the execute request, and describes the possible values	LiteralInput data structure, see Table 24	Zero or one (conditional) [a]
BoundingBox Data	Indicates that this input shall be a BoundingBox data structure that is embedded in execute request, and provides a list of the CRSs supported in these Bounding Boxes	SupportedCRSs data structure, see Table 21	Zero or one (conditional) [a]

a One and only one of these three items shall be included.

b The value of this complex data structure can be input either embedded in the Execute request or remotely accessible to the server.

NOTE 4 The first 3 parameters listed below (with partially grey background) are copied from Table 1 in Subclause 7.2 of this document.

Table 17 — Parts of OutputDescription data structure

Name	Definition	Data type	Multiplicity and use
Identifier	Inherited from Description data structure, see Table 1, applied to an output	ows:CodeType	One (mandatory)
Title		Character string type	One (mandatory)
Abstract		Character string type	Zero or one (optional)
OutputFormChoice	Identifies the type of this output and provides supporting information	OutputFormChoice data structure, see Table 18	One (mandatory)

Table 18 — Parts of OutputFormChoice data structure

Name	Definition	Data type	Multiplicity
ComplexOutput	Indicates that this output shall be a complex data set (such as a GML fragment), and provides lists of formats, encodings, and schemas supported for this output [b, c]	Supported-ComplexData data structure, see Table 19	Zero or one (conditional) [a]
LiteralOutput	Indicates that this output shall be a simple literal value (such as an integer) that is embedded in execute response, and describes the possible values	LiteralOutput data structure, see Table 22	Zero or one (conditional) [a]
BoundingBox Output	Indicates that this output shall be a BoundingBox data structure that is embedded in execute response, and provides a list of the CRSs supported in these Bounding Boxes	Supported CRSs data structure, see Table 21	Zero or one (conditional) [a]

a One and only one of these three items shall be included.

b The client can select from among the identified formats, encodings, and schemas to specify the form of the output. This allows for complete specification of particular versions of GML, or image formats.

c The value of this complex data structure can be output either embedded in the execute operation response or remotely accessible to the client. 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 the execute operation response.

Table 19 — Parts of ComplexData data structure

Name	Definition	Data type and values	Multiplicity and use
Supported Complex Data	Combination of format, encoding, and/or schema supported by process input or output.	SupportedComplexData data structure, see Table 20	Zero or more (optional) Include when more than one [a]
default Format	Identification of default Format for process input or output [b]	Character String type, not empty ows:MimeType	Zero or one (optional) Include when Format other than text/XML [c]
default Encoding	Reference to default encoding for process input or output [d]	URI type	Zero or one (optional) Include when Encoding other than UTF-8 [e]
default Schema	Reference to default XML Schema Document for process input or output [f]	URI type	Zero or one (optional) Include when encoded using XML schema [g]

a This data structure should be included when this process supports more than one combination of format/encoding/schema for this Input/Output. This data structure shall be repeated for each combination of Format/Encoding/Schema that is supported for this Input/Output. This data structure shall not be included if there is only one (i.e., the default) Format/Encoding/Schema combination.

b The process shall expect input in or produce output in this Format unless the Execute request specifies another supported Format.

c This parameter shall be included when the default Format is other than text/XML. This parameter is optional if the Format is text/XML.

d The process shall expect input using or produce output using this encoding unless the Execute request specifies another supported encoding.

e This parameter shall be included when the default Encoding is other than the encoding of the XML response document (e.g. UTF-8). This parameter shall be omitted when there is no Encoding required for this input/output.

f The process shall expect input in or produce output conformant with this XML element or type unless the Execute request specifies another supported XML element or type.

g This parameter shall be omitted when there is no XML Schema associated with this input/output (e.g., a GIF file). This parameter shall be included when this input/output is XML encoded using an XML schema. When included, the input/output shall validate against the referenced XML Schema. Note: If the input/output uses a profile of a larger schema, the server administrator should provide that schema profile for validation purposes.

Table 20 — Parts of SupportedComplexData data structure

Name	Definition	Data type and values	Multiplicity and use
Format	Identification of Format supported by process input or output	Character String type, not empty ows:MimeType	Zero or more (optional) Include when Format other than default [a]
Encoding	Reference to encoding supported by process input or output	URI type	Zero or more (optional) Include when encoding other than default [b]
Schema	Reference to XML Schema Document supported by process input or output	URI type	Zero or more (optional) Include when XML schema other than default [c]

a This element shall be included when the format for this ComplexDataType differs from the defaultFormat for this Input/Output. This element shall not be included if there is only one (i.e., the default) format supported for this Input/Output, or Format does not apply to this Input/Output.

b This element shall be included when the encoding for this ComplexDataType differs from the defaultEncoding for this Input/Output. This element shall not be included if there is only one (i.e., the default) encoding supported for this Input/Output, or Encoding does not apply to this Input/Output.

c This element shall be included when the schema for this ComplexDataType differs from the defaultSchema for this Input/Output. This element shall not be included if there is only one (i.e., the default) XML Schema Document supported for this Input/Output, or Schema does not apply to this Input/Output. Each of these XML elements or types shall be defined in a separate XML Schema Document.

Table 21 — Parts of SupportedCRSs data structure

Name	Definition	Data type	Multiplicity and use
CRS	Reference to one coordinate reference system (CRS)	URI type	One or more (mandatory) Include for each CRS supported
defaultCRS	Reference to default coordinate reference system	URI type	One (mandatory)

Table 22 — Parts of LiteralOutput data structure

Name	Definition	Data type	Multiplicity and use
DataType	Data type of this output (or input)	DomainMetadata data structure, see Table D.7	Zero or one (optional) Include when data type not character string
Supported UOMs	List of units of measure supported of this numerical output (or input)	SupportedUOMs data structure, see Table 23	Zero or one (optional) Include when value(s) have a unit of measure

Table 23 — Parts of SupportedUOMs data structure

Name	Definition	Data type	Multiplicity and use
UOM	Unit of measure of this numerical output (or input)	DomainMetadata data structure, see Table D.7	One or more (mandatory) Include for each UOM supported
defaultUOM	Reference to default unit of measure	URI	Zero or one (optional) Include when default exists

NOTE 5 The first 2 parameters listed below (with partially grey background) are copied from Table 22 above.

Table 24 — Parts of LiteralInput data structure

Name	Definition	Data type	Multiplicity and use
DataType	Inherited from LiteralOutput data structure, see Table 22, applied to an input	DomainMetadata	Zero or one (optional)
Supported UOMs		SupportedUOMs	Zero or one (optional)
LiteralValues Choice	Identifies type of literal input and provides supporting information	LiteralValuesChoice data structure, see Table 25	One (mandatory)
DefaultValue	Default value of this input, encoded in character string	CharacterString, not empty	Zero or one (optional) Include when default exists

Table 25 — Parts of LiteralValuesChoice data structure

Name	Definition	Data type	Multiplicity
AllowedValues	Indicates that are finite set of values and ranges allowed for this input, and contains ordered list of all valid values and/or ranges	AllowedValues data structure, see Table D.5	Zero or one (conditional) [a]
AnyValue	Indicates that any value is allowed for this input	DomainMetadata data structure, see Table D.7	Zero or one (conditional) [a]
Values Reference	Indicates that there are a finite set of values and ranges allowed for this input, specified in referenced list	ValuesReference data structure, see Table D.8	Zero or one (conditional) [a]
a One and only one of these three items shall be included.			

The "Multiplicity and use" columns in Table 11 through Table 25 specify the optionality of each listed parameter and data structure in the DescribeProcess operation response. Each "mandatory" parameter and data structure shall be implemented by all OWS servers, using a specified value(s). Each "optional" parameter and data structure shall also be implemented by all OWS servers, using a specified value(s), for each implemented process for which that metadata is relevant and available.

9.3.2 DescribeProcess response XML encoding

The following schema fragment specifies the contents and structure of a DescribeProcess operation response, always encoded in XML. This schema fragment contains annotations that specify the meaning and use of each element and attribute.

```
   <element name="ProcessDescriptions">
      <annotation>
         <documentation>WPS DescribeProcess operation response.
</documentation>
      </annotation>
      <complexType>
         <sequence>
            <element name="ProcessDescription"
type="wps:ProcessDescriptionType" maxOccurs="unbounded">
               <annotation>
                  <documentation>Ordered list of one or more full
Process descriptions, listed in the order in which they were requested
in the DescribeProcess operation request. </documentation>
               </annotation>
            </element>
         </sequence>
      </complexType>
   </element>
   <!-- ========================================================== -->
   <complexType name="ProcessDescriptionType">
      <annotation>
         <documentation>Full description of a process.
</documentation>
      </annotation>
      <complexContent>
         <extension base="wps:ProcessBriefType">
            <sequence>
               <element name="DataInputs" minOccurs="0">
                  <annotation>
                     <documentation>List of the inputs to this
process. In almost all cases, at least one process input is required.
However, no process inputs may be identified when all the inputs are
predetermined fixed resources.  In this case, those resources shall be
identified in the ows:Abstract element that describes the
process.</documentation>
                  </annotation>
                  <complexType>
                     <sequence>
                        <element name="Input"
type="wps:InputDescriptionType" maxOccurs="unbounded">
                           <annotation>
                              <documentation>Unordered list of one
or more descriptions of the inputs that can be accepted by this
process, including all required and optional inputs.  Where an input is
optional because a default value exists, that default value must be
identified in the "ows:Abstract" element for that input, except in the
case of LiteralData, where the default must be indicated in the
corresponding ows:DefaultValue element. Where an input is optional
because it depends on the value(s) of other inputs, this must be
indicated in the ows:Abstract element for that input. </documentation>
                           </annotation>
```



```
                        </element>
                     </sequence>
                  </complexType>
               </element>
               <element name="ProcessOutputs">
                  <annotation>
                     <documentation>List of outputs which will or can
result from executing the process. </documentation>
                  </annotation>
                  <complexType>
                     <sequence>
                        <element name="Output"
type="wps:OutputDescriptionType" maxOccurs="unbounded">
                           <annotation>
                              <documentation>Unordered list of one
or more descriptions of all the outputs that can result from executing
this process. At least one output is required from each process.
</documentation>
                           </annotation>
                        </element>
                     </sequence>
                  </complexType>
               </element>
            </sequence>
            <attribute name="storeSupported" type="boolean"
use="optional" default="false">
               <annotation>
                  <documentation>Indicates if the ComplexData outputs
from this process can be stored by the WPS server as web-accessible
resources. If "storeSupported" is "true", the Execute operation request
may include "store" equals "true", directing that all ComplexData
outputs of the process be stored so that the client can retrieve them
as required. By default for this process, storage is not supported and
all outputs are returned encoded in the Execute response.
</documentation>
               </annotation>
            </attribute>
            <attribute name="statusSupported" type="boolean"
use="optional" default="false">
               <annotation>
                  <documentation>Indicates if the Execute operation
response can be returned quickly with status information, or will not
be returned until process execution is complete. If "statusSupported"
is "true", the Execute operation request may include "status" equals
"true", directing that the Execute operation response be returned
quickly with status information. By default, status information is not
provided for this process, and the Execute operation response is not
returned until process execution is complete. </documentation>
               </annotation>
            </attribute>
         </extension>
      </complexContent>
   </complexType>
   <!-- =================INPUT PARAMETERS========================== -->
   <!-- ============================================================ -->
   <complexType name="InputDescriptionType">
      <annotation>
```

```
          <documentation>Description of an input to a process.
</documentation>
       </annotation>
       <complexContent>
          <extension base="wps:DescriptionType">
             <annotation>
                <documentation>In this use, the DescriptionType shall
describe this process input. </documentation>
             </annotation>
             <sequence>
                <group ref="wps:InputFormChoice"/>
                <element name="MinimumOccurs" minOccurs="0">
                   <annotation>
                      <documentation>The minimum number of times that
values for this parameter are required. If MinimumOccurs is "0", this
data input is optional. If MinimumOccurs is "1" or if this element is
omitted, this process input is required. </documentation>
                   </annotation>
                   <simpleType>
                      <restriction base="nonNegativeInteger">
                         <enumeration value="0"/>
                         <enumeration value="1"/>
                      </restriction>
                   </simpleType>
                </element>
             </sequence>
          </extension>
       </complexContent>
    </complexType>
    <!-- ========================================================== -->
    <group name="InputFormChoice">
       <annotation>
          <documentation>Identifies the form of this output and
provides supporting information. </documentation>
       </annotation>
       <choice>
          <element name="ComplexData"
type="wps:SupportedComplexDataType">
             <annotation>
                <documentation>Indicates that this input shall be a
complex data structure (such as a GML document), and provides a list of
formats and encodings supported for this Input. The value of this
ComplexData structure can be input either embedded in the Execute
request or remotely accessible to the server.
                   This element also provides a list of formats,
encodings, and schemas supported for this output. The client can select
from among the identified combinations of formats, encodings, and
schemas to specify the form of the output. This allows for complete
specification of particular versions of GML, or image formats.
</documentation>
             </annotation>
          </element>
          <element name="LiteralData" type="wps:LiteralInputType">
             <annotation>
                <documentation>Indicates that this input shall be a
simple numeric value or character string that is embedded in the
execute request, and describes the possible values. </documentation>
             </annotation>
```



```
          </element>
          <element name="BoundingBoxData" type="wps:SupportedCRSsType">
             <annotation>
                <documentation>Indicates that this input shall be a
BoundingBox data structure that is embedded in the execute request, and
provides a list of the CRSs supported for this Bounding Box.
</documentation>
             </annotation>
          </element>
       </choice>
   </group>
   <!-- ========================================================== -->
   <complexType name="LiteralInputType">
      <annotation>
         <documentation>Description of a process input that consists
of a simple literal value (e.g., "2.1"). (Informative: This type is a
subset of the ows:UnNamedDomainType defined in owsDomaintype.xsd.)
</documentation>
      </annotation>
      <complexContent>
         <extension base="wps:LiteralOutputType">
            <sequence>
               <group ref="wps:LiteralValuesChoice"/>
               <element ref="ows:DefaultValue" minOccurs="0">
                  <annotation>
                     <documentation>Optional default value for this
quantity, which should be included when this quantity has a default
value. </documentation>
                  </annotation>
               </element>
            </sequence>
         </extension>
      </complexContent>
   </complexType>
   <!-- ========================================================== -->
   <group name="LiteralValuesChoice">
      <annotation>
         <documentation>Identifies the type of this literal input and
provides supporting information. </documentation>
      </annotation>
      <choice>
         <element ref="ows:AllowedValues">
            <annotation>
               <documentation>Indicates that there are a finite set of
values and ranges allowed for this input, and contains list of all the
valid values and/or ranges of values. Notice that these values and
ranges can be displayed to a human client. </documentation>
            </annotation>
         </element>
         <element ref="ows:AnyValue">
            <annotation>
               <documentation>Indicates that any value is allowed for
this input. This element shall be included when there are no
restrictions, except for data type, on the allowable value of this
input. </documentation>
            </annotation>
         </element>
         <element ref="ows:ValuesReference">
```



```
            <annotation>
               <documentation>Indicates that there are a finite set of
values and ranges allowed for this input, which are specified in the
referenced list. </documentation>
            </annotation>
         </element>
      </choice>
   </group>
   <!-- ================= INPUT AND OUTPUT TYPES ================== -->
   <!-- ============================================================ -->
   <complexType name="SupportedUOMsType">
      <annotation>
         <documentation>List of supported units of measure for a
process input or output. </documentation>
      </annotation>
      <sequence>
         <element ref="ows:UOM" minOccurs="0" maxOccurs="unbounded">
            <annotation>
               <documentation>Unordered list of references to the
Units of Measure supported for this input or output. This element shall
not be included if there is only one (i.e., the default) UOM supported.
</documentation>
            </annotation>
         </element>
      </sequence>
      <attribute name="defaultUOM" type="anyURI" use="optional">
         <annotation>
            <documentation>Reference to the default UOM supported for
this input or output, if any. The process shall expect input in or
produce output in this UOM unless the Execute request specifies another
supported UOM. </documentation>
         </annotation>
      </attribute>
   </complexType>
   <!-- ============================================================ -->
   <complexType name="SupportedCRSsType">
      <annotation>
         <documentation>List of supported Coordinate Reference
Systems. </documentation>
      </annotation>
      <sequence>
         <element name="CRS" type="anyURI" minOccurs="0"
maxOccurs="unbounded">
            <annotation>
               <documentation>Unordered list of references to the
coordinate reference systems supported. This element shall not be
included if there is only one (i.e., the default) CRS supported.
</documentation>
            </annotation>
         </element>
      </sequence>
      <attribute name="defaultCRS" type="anyURI" use="required">
         <annotation>
            <documentation>Reference to the CRS that will be used
unless the Execute operation request specifies another supported CRS.
</documentation>
         </annotation>
      </attribute>
```



```
   </complexType>
   <!-- ================================================================= -->
   <complexType name="SupportedComplexDataType">
      <annotation>
         <documentation>Formats, encodings, and schemas supported by a
process input or output. </documentation>
      </annotation>
      <sequence>
         <element name="SupportedComplexData"
type="wps:ComplexDataType" minOccurs="0" maxOccurs="unbounded">
            <annotation>
               <documentation>Unordered list of combinations of
format, encoding, and schema supported for this Input or Output (an
example of one such combination is format=text/XML, encoding=UTF-8,
schema=GML 2.1). This element should be included when this process
supports more than one combination of format/encoding/schema for this
Input/Output. This element shall be repeated for each combination of
Format/Encoding/Schema that is supported for this Input/Output. This
element shall not be included if there is only one (i.e., the default)
Format/Encoding/Schema combination. </documentation>
            </annotation>
         </element>
      </sequence>
      <attribute name="defaultFormat" type="ows:MimeType"
use="optional">
         <annotation>
            <documentation>Identifier of the default Format supported
for this input or output. The process shall expect input in or produce
output in this Format unless the Execute request specifies another
supported Format. This parameter shall be included when the default
Format is other than text/XML. This parameter is optional if the Format
is text/XML.
</documentation>
         </annotation>
      </attribute>
      <attribute name="defaultEncoding" type="anyURI" use="optional">
         <annotation>
            <documentation>Reference to the default encoding supported
for this input or output. The process will expect input using or
produce output using this encoding unless the Execute request specifies
another supported encoding. This parameter shall be included when the
default Encoding is other than the encoding of the XML response
document (e.g. UTF-8). This parameter shall be omitted when there is no
Encoding required for this input/output. </documentation>
         </annotation>
      </attribute>
      <attribute name="defaultSchema" type="anyURI" use="optional">
         <annotation>
            <documentation>Reference to the definition of the default
XML element or type supported for this input or output. This XML
element or type shall be defined in a separate XML Schema Document. The
process shall expect input in or produce output conformant with this
XML element or type unless the Execute request specifies another
supported XML element or type. This parameter shall be omitted when
there is no XML Schema associated with this input/output (e.g., a GIF
file). This parameter shall be included when this input/output is XML
encoded using an XML schema. When included, the input/output shall
validate against the referenced XML Schema. Note: If the input/output
```



```
uses a profile of a larger schema, the server administrator should
provide that schema profile for validation purposes. </documentation>
          </annotation>
       </attribute>
   </complexType>
   <!-- ========================================================== -->
   <complexType name="ComplexDataType">
      <annotation>
         <documentation>A combination of format, encoding, and/or
schema supported by a process input or output. </documentation>
      </annotation>
      <sequence>
         <element name="Format" type="ows:MimeType" minOccurs="0">
            <annotation>
               <documentation>Format supported for this input or
output (e.g., text/XML). This element shall be included when the format
for this ComplexDataType differs from the defaultFormat for this
Input/Output. This element shall not be included if there is only one
(i.e., the default) format supported for this Input/Output, or Format
does not apply to this Input/Output. </documentation>
            </annotation>
         </element>
         <element name="Encoding" type="anyURI" minOccurs="0">
            <annotation>
               <documentation>Reference to an encoding supported for
this input or output (e.g., UTF-8). This element shall be included when
the encoding for this ComplexDataType differs from the defaultEncoding
for this Input/Output. This element shall not be included if there is
only one (i.e., the default) encoding supported for this Input/Output,
or Encoding does not apply to this Input/Output. </documentation>
            </annotation>
         </element>
         <element name="Schema" type="anyURI" minOccurs="0">
            <annotation>
               <documentation>Reference to a definition of XML
elements or types supported for this Input or Output (e.g., GML 2.1
Application Schema). Each of these XML elements or types shall be
defined in a separate XML Schema Document. This element shall be
included when the schema for this ComplexDataType differs from the
defaultSchema for this Input/Output. This element shall not be included
if there is only one (i.e., the default) XML Schema Document supported
for this Input/Output, or Schema does not apply to this Input/Output.
</documentation>
            </annotation>
         </element>
      </sequence>
   </complexType>
   <!-- =================OUTPUT PARAMETERS====================== -->
   <!-- ========================================================== -->
   <complexType name="OutputDescriptionType">
      <annotation>
         <documentation>Description of a process Output.
</documentation>
      </annotation>
      <complexContent>
         <extension base="wps:DescriptionType">
            <annotation>
```



```
                <documentation>In this use, the DescriptionType shall
describe this process output. </documentation>
             </annotation>
             <group ref="wps:OutputFormChoice"/>
          </extension>
       </complexContent>
   </complexType>
   <!-- ========================================================== -->
   <group name="OutputFormChoice">
       <annotation>
          <documentation>Identifies the form of this output, and
provides supporting information. </documentation>
       </annotation>
       <choice>
          <element name="ComplexOutput"
type="wps:SupportedComplexDataType">
             <annotation>
                <documentation>Indicates that this Output shall be a
complex data structure (such as a GML fragment) that is returned by the
execute operation response. The value of this complex data structure
can be output either embedded in the execute operation response or
remotely accessible to the client. 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.
                This element also provides a list of format, encoding,
and schema combinations supported for this output. The client can
select from among the identified combinations of formats, encodings,
and schemas to specify the form of the output. This allows for complete
specification of particular versions of GML, or image formats.
</documentation>
             </annotation>
          </element>
          <element name="LiteralOutput" type="wps:LiteralOutputType">
             <annotation>
                <documentation>Indicates that this output shall be a
simple literal value (such as an integer) that is embedded in the
execute response, and describes that output. </documentation>
             </annotation>
          </element>
          <element name="BoundingBoxOutput"
type="wps:SupportedCRSsType">
             <annotation>
                <documentation>Indicates that this output shall be a
BoundingBox data structure, and provides a list of the CRSs supported
in these Bounding Boxes. This element shall be included when this
process output is an ows:BoundingBox element. </documentation>
             </annotation>
          </element>
       </choice>
   </group>
   <!-- ========================================================== -->
   <complexType name="LiteralOutputType">
       <annotation>
          <documentation>Description of a literal output (or input).
</documentation>
       </annotation>
       <sequence>
```



```
          <element ref="ows:DataType" minOccurs="0">
             <annotation>
                <documentation>Data type of this set of values (e.g.
integer, real, etc). This data type metadata should be included for
each quantity whose data type is not a string. </documentation>
             </annotation>
          </element>
          <element name="SupportedUOMs" type="wps:SupportedUOMsType"
minOccurs="0">
             <annotation>
                <documentation>List of supported units of measure for
this input or output. This element should be included when this literal
has a unit of measure (e.g., "meters", without a more complete
reference system). Not necessary for a count, which has no units.
</documentation>
             </annotation>
          </element>
       </sequence>
   </complexType>
```

9.3.3 DescribeProcess response example

A DescribeProcess operation response for WPS can look like this encoded in XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<!--This example describes a buffer command that accepts polygon
coordinates in GML, and used a buffer distance in meters to produce a
buffered polygon feature, which is output in GML, in either UTF-8 or
base64 encoding.  The polygon can be returned directly as output, or
stored by the service as a web-accessible resource.  Ongoing processing
status reports are not available. -->
<ProcessDescriptions xmlns="http://www.opengeospatial.net/wps"
xmlns:wps="http://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">
   <ProcessDescription processVersion="2" storeSupported="true"
statusSupported="false">
      <ows:Identifier>Buffer</ows:Identifier>
      <ows:Title>Create a buffer around a polygon.</ows:Title>
      <ows:Abstract>Create a buffer around a single polygon. Accepts
the polygon as GML and provides GML output for the buffered feature.
</ows:Abstract>
      <ows:Metadata xlink:title="spatial" />
      <ows:Metadata xlink:title="geometry" />
      <ows:Metadata xlink:title="buffer" />
      <ows:Metadata xlink:title="GML" />
      <DataInputs>
         <Input>
            <ows:Identifier>InputPolygon</ows:Identifier>
            <ows:Title>Polygon to be buffered</ows:Title>
            <ows:Abstract>URI to a set of GML that describes the
polygon.</ows:Abstract>
```



```
            <ComplexData defaultFormat="text/XML"
defaultEncoding="base64"
defaultSchema="http://foo.bar/gml/3.1.0/polygon.xsd">
                <SupportedComplexData>
                   <Format>text/XML</Format>
                   <Encoding>UTF-8</Encoding>

   <Schema>http://foo.bar/gml/3.1.0/polygon.xsd</Schema>
                </SupportedComplexData>
            </ComplexData>
            <MinimumOccurs>1</MinimumOccurs>
         </Input>
         <Input>
            <ows:Identifier>BufferDistance</ows:Identifier>
            <ows:Title>Buffer Distance</ows:Title>
            <ows:Abstract>URI to a GML resource file</ows:Abstract>
            <LiteralData>
               <SupportedUOMs defaultUOM="meters"/>
               <ows:AnyValue/>
            </LiteralData>
            <MinimumOccurs>1</MinimumOccurs>
         </Input>
      </DataInputs>
      <ProcessOutputs>
         <Output>
            <ows:Identifier>BufferedPolygon</ows:Identifier>
            <ows:Title>Buffered Polygon</ows:Title>
            <ows:Abstract>GML stream describing the buffered polygon
feature.</ows:Abstract>
            <ComplexOutput defaultFormat="text/XML"
defaultEncoding="base64"
defaultSchema="http://foo.bar/gml/3.1.0/polygon.xsd">
                <SupportedComplexData>
                   <Format>text/XML</Format>
                   <Encoding>UTF-8</Encoding>

   <Schema>http://foo.bar/gml/3.1.0/polygon.xsd</Schema>
                </SupportedComplexData>
            </ComplexOutput>
         </Output>
      </ProcessOutputs>
   </ProcessDescription>
</ProcessDescriptions>
```

9.3.4 DescribeProcess exceptions

When a WPS server encounters an error while performing a DescribeProcess operation, it shall return an exception report message as specified in Subclause 7.4 of [OGC 05-008]. The allowed standard exception codes shall include those listed in Table 26. For each listed exceptionCode, the contents of the “locator” parameter value shall be as specified in the right column of Table 26.

NOTE To reduce the need for readers to refer to other documents, all the values listed below are copied from Table 20 in Subclause 8.3 of [OGC 05-008].

Table 26 — Exception codes for DescribeProcess operation

exceptionCode value	Meaning of code	"locator" value
MissingParameterValue	Operation request does not include a parameter value, and this server did not declare a default value for that parameter	Name of missing parameter
InvalidParameterValue	Operation request contains an invalid parameter value	Name of parameter with invalid value
NoApplicableCode	No other exceptionCode specified by this service and server applies to this exception	None, omit "locator" parameter

10 Execute operation (mandatory)

10.1 Introduction

The mandatory Execute operation allows WPS clients to run a specified process implemented by a server, using input parameter values provided and returning the output value(s) produced. The outputs can be returned as a direct response to the request. Alternatively, the server can be directed to store the result(s) as web accessible resources. If the results are stored, the Execute response will consist of a XML document that includes a URL for each stored output, which the client can use to retrieve those outputs.

The Execute response can be returned after process execution is completed. Alternately, the Execute response can be returned immediately following acceptance by the server of this process execution. In that case, the Execute response includes information about the status of the process, which indicates whether or not the process has completed, as well as a status URL. The status URL must return an updated Execute response. This status URL allows the client to poll the server if the process takes a substantial amount of time to execute. The updated Execute response indicates the completion status of the process and a measure of the amount of processing remaining if the process is not complete. An example of how this works is shown in the UML activity diagram shown in Figure 2.

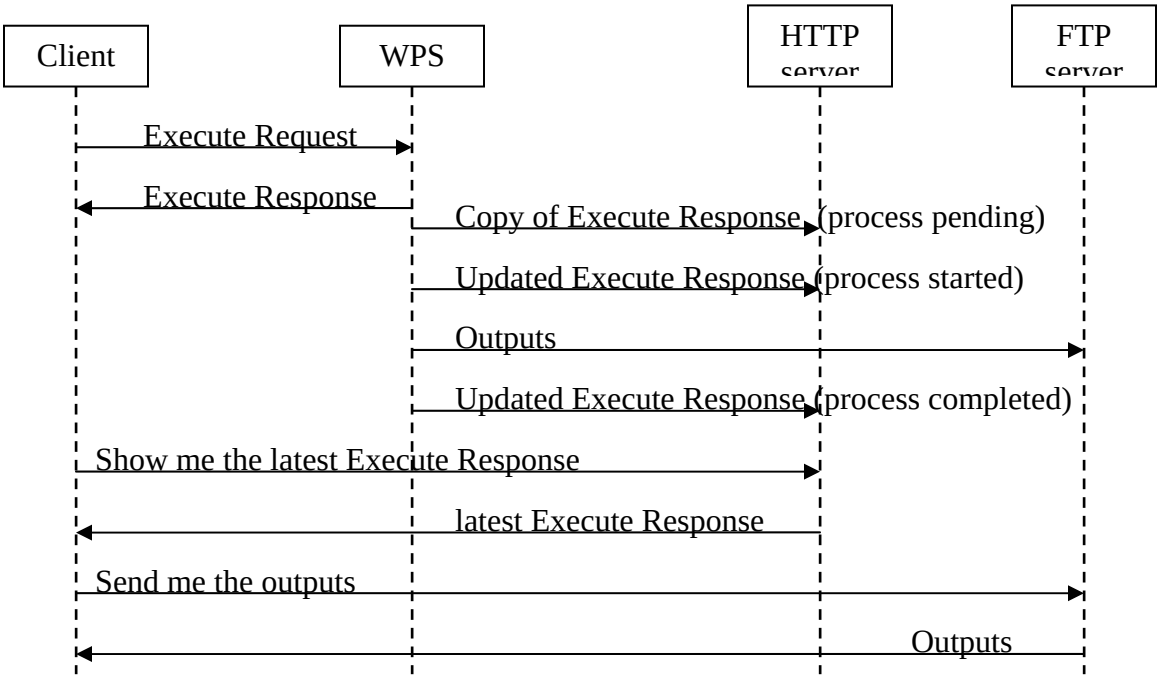

Figure 2 — Activity diagram when client requests storage of results

10.2 Execute operation request

10.2.1 Execute request parameters

A request to perform the Execute operation shall include the parameters listed and defined in Table 27 through Table 36. These tables specify the UML model data type, source of values, and multiplicity of each listed parameter in the operation request, plus the meaning to servers when each optional parameter is included. Although some values listed in the “Name” columns appear to contain spaces, they shall not contain spaces.

NOTE 1 To reduce the need for readers to refer to other documents, the first three parameters listed below are largely copied from Table 21 in Subclause 9.2.1 of [OGC 05-008]. The Identifier parameter is largely copied from Table 1 in Subclause 7.2 of this document.

Table 27 — Parts of Execute operation request

Name	Definition	Data type and value	Multiplicity and use
service	Service type identifier	Character String type, not empty Value is OWS type abbreviation, namely "WPS"	One (mandatory)
request	Operation name	Character String type, not empty Value is operation name, namely "Execute"	One (mandatory)
version	Specification version for operation	Character String type, not empty Value is specified by each Implementation Specification and Schemas version	One (mandatory)
Identifier	Unambiguous identifier or name of a process	ows:CodeType, as adaptation of MD_Identifier in ISO 19115 Value is process Identifier used in Capabilities document.	One (mandatory)
DataInputs	List of inputs provided to this process execution	DataInputs data structure, see Table 28	Zero or one (optional) Include if any input [a]
Output Definitions	List of definitions of outputs desired from executing this process	OutputDefinitions data structure, see Table 29	Zero or one (optional) Include [b]
store	Specifies if all complex valued output(s) of this process should be stored by process as web-accessible resources	Boolean type Values are: true and false [c] Default is false (return directly in response)	Zero or one (optional) Include when direct return not desired [d]
status	Specifies if Execute operation response shall be returned quickly with status information	Boolean type Values are: true and false [e] Default is false	Zero or one (optional) Include when status information desired [f]

a It is possible to have no inputs provided only when all the inputs are predetermined fixed resources. In all other cases, at least one input is required.

b These outputs are not normally identified, unless the client is specifically requesting a limited subset of outputs, and/or is requesting output formats and/or schemas and/or encodings different from the defaults and selected from the alternatives identified in the process description, or wishes to customize the descriptive information about the output.

c If "store" is "true", the server shall store all the complex valued output(s) of the process so that the client can retrieve them as required. If "store" is "false", all the complex valued output(s) shall be encoded in the Execute operation response.

d This "store" parameter shall not be included unless the "storeSupported" parameter is included and is "true" in the ProcessDescription for this process. The "store" parameter value "true" is recommended to make service chaining more efficient when the output(s) are large.

e If “status” is “true”, the server shall return the Execute operation response as soon as it accepts execution of this process. If “status” is “false”, the server shall not return the Execute operation response until process execution is complete.

f This “status” parameter shall not be included unless the "statusSupported" parameter is included and is "true" in the ProcessDescription for this process. The “status” parameter value “true” is recommended when a process takes a long time to execute.

NOTE 2 The data type of some parameters is specified as “Character String type, not empty”. In the XML Schema Documents specified herein, these parameters are encoded with the xsd:string type, which does NOT require that these strings not be empty.

NOTE 3 The UML class diagrams contained in Subclause C.6 provides a useful graphical view of the contents of the Execute operation request listed in Table 27 through Table 36.

The operation request provides support for multiple inputs. Each input refers to one of the forms of input that may be required for a single Execute request. The normal way to provide inputs to a WPS is through providing of one or more URIs (usually URLs) of input values, unless the inputs are simple scalar values. This is not intended to be used to facilitate batch processing (e.g., multiple images to be processed through a single algorithm). If a process is to be run multiple times (probably using different inputs each time), each run shall be submitted as a separate Execute operation request.

Table 28 — Parts of DataInputs data structure

Name	Definition	Data type	Multiplicity and use
Input	Value of input to this process execution	IOValue data structure, see Table 32	One or more (mandatory) Include one for each input, unordered

Table 29 — Parts of OutputDefinitions data structure

Name	Definition	Data type	Multiplicity and use
Output	Definition of format, encoding, and schema for output to be returned from this process	OutputDefinition data structure, see Table 30	One or more (mandatory) Include [a]

a This OutputDefinition shall be repeated for each Output that offers a choice of format, and the client wishes to use one that is not identified as the default, and/or for each Output that the client wishes to customize the descriptive information about the output.

NOTE 4 The first 3 parameters listed below (with partially grey background) are copied from Table 1 in Subclause 7.2 of this document. The second 3 parameters (with partially grey background) are copied from Table 31 below.

Table 30 — Parts of OutputDefinition data structure

Name	Definition	Data type	Multiplicity and use
Identifier	Inherited from Description data structure, see Table 1, applied to an input	ows:CodeType	One (mandatory)
Title		Character string type	One (mandatory)
Abstract		Character string type	Zero or one (optional)
format	Uses ComplexValueEncoding data structure, see Table 31	Character String type	Zero or one (optional)
encoding		URI type	Zero or one (optional)
schema		URL type	Zero or one (optional)
uom	Identifier of unit of measure requested for this output	URI type	Zero or one (optional) Include [a]
a A uom can be referenced when a client chooses to specify one of the uoms supported for this numerical output. This uom shall be a unit of measure referenced for this output in the Process full description.			

Table 31 — Parts of ComplexValueEncoding data structure

Name	Definition	Data type and values	Multiplicity and use
format	Identification of format of this input or requested for this output parameter's value	Character String type, not empty ows:MimeType	Zero or one (optional) Include when format not in http header
encoding	Reference to encoding of this input or requested for this output	URI type	Zero or one (optional) Include when not default encoding
schema	Reference to XML Schema Document that specifies content model of input or output parameter's value	URL type	Zero or one (optional) Include when XML encoded resource

NOTE 5 The first 3 parameters listed below (with partially grey background) are copied from Table 1 in Subclause 7.2 of this document.

Table 32 — Parts of IOValue data structure

Name	Definition	Data type	Multiplicity and use
Identifier	Inherited from Description data structure, see Table 1, applied to an input or output	ows:CodeType	One (mandatory)
Title		Character string type	One (mandatory)
Abstract		Character string type	Zero or one (optional)
ValueForm Choice	Identifies the type of this input or output value, and provides supporting information	ValueFormChoice data structure, see Table 33	One (mandatory)

Table 33 — Parts of ValueFormChoice data structure

Name	Definition	Data type	Multiplicity
ComplexValue Reference	Identifies this input or output value as a web accessible resource, and references that resource	ValueReference data structure, see Table 34 [b]	Zero or one (conditional) [a]
ComplexValue	Identifies this input or output value as a complex value data structure, and provides that value	ComplexValue data structure, see Table 35 [c]	Zero or one (conditional) [a]
LiteralValue	Identifies this input or output value as a literal value of a simple quantity, and provides that value	LiteralValue data structure, see Table 36	Zero or one (conditional) [a]
BoundingBox Value	Identifies this input or output value as a BoundingBox data structure, and provides that value	ows:BoundingBox data structure, see Subclause 10.2 of [OGC 05-008]	Zero or one (conditional) [a]

a One and only one of these four items shall be included.

b For an input, this data structure may be used by a client for any process input coded as ComplexData in the ProcessDescription. For an output, this element shall be used by a server when "store" in the Execute request is "true".

c For an input, this element may be used by a client for any process input coded as ComplexData in the ProcessDescription. For an output, this element shall be used by a server when either "storeSupported" in the ProcessDescription or "store" in the Execute request is "true".

NOTE 4 The first 3 parameters (with partially grey background) below are copied from Table 31 above.

Table 34 — Parts of ValueReference data structure

Name	Definition	Data type	Multiplicity
format	Uses ComplexValueEncoding data structure, see Table 31	Character String type	Zero or one (optional)
encoding		URI type	Zero or one (optional)
schema		URL type	Zero or one (optional)
reference	Reference to web-accessible resource to be used as input, or provided by process as output	URL type	One (mandatory)

NOTE 5 The first 3 parameters (with partially grey background) below are copied from Table 31 above.

Table 35 — Parts of ComplexValue data structure

Name	Definition	Data type	Multiplicity
format	Uses ComplexValueEncoding data structure, see Table 31	Character String type	Zero or one (optional)
encoding		URI type	Zero or one (optional)
schema		URL type	Zero or one (optional)
Value [a]	Complex value to be used as input to process	Any type	One (mandatory)

a The complex value is embedded here, in the format/encoding, and according to the schema indicated by the first three parameters if they exist, or by the relevant defaults.

Table 36 — Parts of LiteralValue data structure

Name	Definition	Data type	Multiplicity
value	Literal value, encoded in a character string	Character String type, not empty	One (mandatory)
dataType	Identifier of data type of this literal value	URI type	Zero or one (optional) [a]
uom	Identifier of unit of measure of this literal numerical value	URI type	Zero or one (optional) [b]

a This dataType should be included for each quantity whose value is not a simple string.

b This unit of measure should be referenced for any numerical value that has units (e.g., "meters", but not a more complete reference system). Shall be a UOM identified in the Process description for this input or output.

The “Multiplicity and use” columns in Table 27 through Table 36 specify the optionality of each listed parameter and data structure in the Execute operation request. Each “mandatory” parameter and data structure shall be implemented by all WPS clients, using a specified value(s). Similarly, each “mandatory” parameter and data structure shall be implemented by all WPS servers, checking that each request parameter and data structure is received with any allowed value(s).

Each “optional” parameter and data structure, in the Execute operation request, should be implemented by all WPS clients using a specified value(s), for each implemented process to which that parameter or data structure applies. Similarly, each “optional” parameter and data structure shall be implemented by all WPS servers, for each implemented process to which that parameter or data structure applies.

10.2.2 Execute request KVP encoding (optional)

Servers may implement HTTP GET transfer of the Execute operation request, using KVP encoding. The KVP encoding of the Execute operation request shall use the parameters specified in Table 37. The parameters listed in Table 37 shall be as specified in Table 27 above. The full functionality of WPS is not available using KVP encoding. More specifically, KVP encoding is suitable for simple Execute requests that do not include embedded ComplexValues or BoundingBoxValues. There are no output parameter capabilities with Execute request KVP encoding.

Table 37 — Execute operation request URL parameters

Name and example [a]	Optionality and use	Definition and format
service=WPS	Mandatory	Service type identifier
request= Execute	Mandatory	Operation name
version=0.4.0	Mandatory	Specification and schema version for this operation
Identifier=Buffer	Mandatory	Process identifier
store=true	Optional, include when direct return not desired [b]	Boolean value that specifies if output data to be stored as remote resource (or returned directly in response message)
DataInputs= Buffer,http://foo.bar/foo 1,BufferDistance,100	Optional, include when one or more inputs provided	List of titles and URL references to values of inputs to this process execution, comma separated [c]

a All parameter names are here listed using mostly lower case letters. However, any parameter name capitalization shall be allowed in KVP encoding, see Subclause 11.5.2 of [OGC 05-008].

b The "store" parameter value "true" is recommended when a process takes a long time to execute. Also recommended to make service chaining more efficient when the output(s) are large. However, this parameter should not be included unless the corresponding parameter is included and is "true" in the ProcessDescription for this process.

c Each input shall be recorded as a pair of DataInputs values, with the first value being the input Identifier, and the second value being either the input value, or the URL of that (complex) input value.

EXAMPLE An example Execute operation request using KVP encoding is:

```
http://foo.bar/foo?
    request="Execute"&
    service="WPS"&
    version="0.4.0"&
    store="true"&
    Identifier="Buffer"&
    DataInputs= Buffer,http://foo.bar/foo1,BufferDistance,100
```

10.2.3 Execute request XML encoding (mandatory)

All WPS servers shall implement HTTP POST transfer of the Execute operation request, using XML encoding only. The following schema fragment specifies the contents and structure of an Execute operation request encoded in XML:

```
    <element name="Execute">
        <annotation>
            <documentation>WPS Execute operation request, to execute one
identified Process. If a process is to be run multiple times, each run
shall be submitted as a separate Execute request. </documentation>
        </annotation>
        <complexType>
            <complexContent>
                <extension base="wps:RequestBaseType">
                    <sequence>
                        <element ref="ows:Identifier">
                            <annotation>
                                <documentation>Identifier of the Process to
be executed. This Process identifier shall be as listed in the
ProcessOfferings section of the WPS Capabilities document.
</documentation>
```

```
                        </annotation>
                    </element>
                    <element name="DataInputs"
type="wps:DataInputsType" minOccurs="0">
                        <annotation>
                            <documentation>List of input (or parameter)
values provided to the process, including each of the Inputs needed to
execute the process. It is possible to have no inputs provided only
when all the inputs are predetermined fixed resources. In all other
cases, at least one input is required. </documentation>
                        </annotation>
                    </element>
                    <element name="OutputDefinitions"
type="wps:OutputDefinitionsType" minOccurs="0">
                        <annotation>
                            <documentation>List of definitions of the
outputs (or parameters) requested from the process. These outputs are
not normally identified, unless the client is specifically requesting a
limited subset of outputs, and/or is requesting output formats and/or
schemas and/or encodings different from the defaults and selected from
the alternatives identified in the process description, or wishes to
customize the descriptive information about the output.
</documentation>
                        </annotation>
                    </element>
                </sequence>
                <attribute name="store" type="boolean" use="optional"
default="false">
                    <annotation>
                        <documentation>Specifies if the complex valued
output(s) of this process should be stored by the process as web-
accessible resources. If store is "true", the server shall store all
the complex valued output(s) of the process so that the client can
retrieve them as required. If store is "false", all the complex valued
output(s) shall be encoded in the Execute operation response. This
parameter shall not be included unless the corresponding
"storeSupported" parameter is included and is "true" in the
ProcessDescription for this process. </documentation>
                    </annotation>
                </attribute>
                <attribute name="status" type="boolean" use="optional"
default="false">
                    <annotation>
                        <documentation>Specifies if the Execute
operation response shall be returned quickly with status information,
or not returned until process execution is complete. This parameter
shall not be included unless the corresponding "statusSupported"
parameter is included and is "true" in the ProcessDescription for this
process. </documentation>
                    </annotation>
                </attribute>
            </extension>
        </complexContent>
    </complexType>
</element>
<!-- ========================================================== -->
<complexType name="DataInputsType">
    <annotation>
```

```
            <documentation>List of the Inputs provided as part of the
Execute Request. </documentation>
        </annotation>
        <sequence>
            <element name="Input" type="wps:IOValueType"
maxOccurs="unbounded">
                <annotation>
                    <documentation>Unordered list of one or more inputs to
be used by the process, including each of the Inputs needed to execute
the process. </documentation>
                </annotation>
            </element>
        </sequence>
    </complexType>
    <!-- ========================================================== -->
    <complexType name="OutputDefinitionsType">
        <annotation>
            <documentation>List of definitions of the outputs (or
parameters) requested from the process. </documentation>
        </annotation>
        <sequence>
            <element name="Output" type="wps:OutputDefinitionType"
maxOccurs="unbounded">
                <annotation>
                    <documentation>Unordered list of one or more
definitions of the outputs requested. This element shall be repeated
for each Output that offers a choice of format, and the client wishes
to use one that is not identified as the default, and/or for each
Output that the client wishes to customize the descriptive information
about the output. </documentation>
                </annotation>
            </element>
        </sequence>
    </complexType>
    <!-- ========================================================== -->
    <complexType name="OutputDefinitionType">
        <annotation>
            <documentation>Definition of a format, encoding, schema, and
unit-of-measure for an output to be returned from a process.
</documentation>
        </annotation>
        <sequence>
            <element ref="ows:Identifier">
                <annotation>
                    <documentation>Unambiguous identifier or name of an
output, unique for this process. </documentation>
                </annotation>
            </element>
            <element ref="ows:Title" minOccurs="0">
                <annotation>
                    <documentation>Title of the process output, normally
available for display to a human. This element should be used if the
client wishes to customize the Title in the execute response. This
element should not be used if the Title provided for this output in the
ProcessDescription is adequate. </documentation>
                </annotation>
            </element>
            <element ref="ows:Abstract" minOccurs="0">
```

```
            <annotation>
               <documentation>Brief narrative description of a process
output, normally available for display to a human. This element should
be used if the client wishes to customize the Abstract in the execute
response. This element should not be used if the Abstract provided for
this output in the ProcessDescription is adequate. </documentation>
            </annotation>
         </element>
      </sequence>
      <attribute name="uom" type="anyURI" use="optional">
         <annotation>
            <documentation>Reference to the unit of measure (if any)
requested for this output. A uom can be referenced when a client wants
to specify one of the units of measure supported for this output. This
uom shall be a unit of measure referenced for this output of this
process in the Process full description. </documentation>
         </annotation>
      </attribute>
      <attributeGroup ref="wps:ComplexValueEncoding"/>
   </complexType>
   <!-- ========================================================== -->
   <attributeGroup name="ComplexValueEncoding">
      <annotation>
         <documentation>References the XML schema, format, and
encoding of a complex value. </documentation>
      </annotation>
      <attribute name="format" type="ows:MimeType" use="optional">
         <annotation>
            <documentation>The Format of this input or requested for
this output (e.g., text/XML). This element shall be omitted when the
Format is indicated in the http header of the output. When included,
this format shall be one published for this output or input in the
Process full description. </documentation>
         </annotation>
      </attribute>
      <attribute name="encoding" type="anyURI" use="optional">
         <annotation>
            <documentation>The encoding of this input or requested for
this output (e.g., UTF-8). This "encoding" shall be included whenever
the encoding required is not the default encoding indicated in the
Process full description. When included, this encoding shall be one
published for this output or input in the Process full description.
</documentation>
         </annotation>
      </attribute>
      <attribute name="schema" type="anyURI" use="optional">
         <annotation>
            <documentation>Web-accessible XML Schema Document that
defines the content model of this complex resource (e.g., encoded using
GML 2.2 Application Schema). This reference should be included for XML
encoded complex resources to facilitate validation. </documentation>
         </annotation>
      </attribute>
   </attributeGroup>
   <!-- ==============INPUT AND OUTPUT VALUES ====================== --
>
   <!-- ========================================================== -->
   <complexType name="IOValueType">
```

```
      <annotation>
         <documentation>Value of one input to a process or one output
from a process. </documentation>
      </annotation>
      <complexContent>
         <extension base="wps:DescriptionType">
            <annotation>
               <documentation>In this use, the DescriptionType shall
describe this process input or output. </documentation>
            </annotation>
            <group ref="wps:ValueFormChoice"/>
         </extension>
      </complexContent>
   </complexType>
   <!-- ========================================================== -->
   <group name="ValueFormChoice">
      <annotation>
         <documentation>Identifies the form of this input or output
value, and provides supporting information. </documentation>
      </annotation>
      <choice>
         <element name="ComplexValueReference">
            <annotation>
               <documentation>Identifies this input or output value as
a web accessible resource, and references that resource. For an input,
this element may be used by a client for any process input coded as
ComplexData in the ProcessDescription. For an output, this element
shall be used by a server when "store" in the Execute request is
"true". </documentation>
            </annotation>
            <complexType>
               <attributeGroup ref="wps:ValueReference"/>
            </complexType>
         </element>
         <element name="ComplexValue" type="wps:ComplexValueType">
            <annotation>
               <documentation>Identifies this input or output value as
a complex value data structure encoded in XML (e.g., using GML), and
provides that complex value data structure. For an input, this element
may be used by a client for any process input coded as ComplexData in
the ProcessDescription. For an output, this element shall be used by a
server when "store" in the Execute request is "false". </documentation>
            </annotation>
         </element>
         <element name="LiteralValue" type="wps:LiteralValueType">
            <annotation>
               <documentation>Identifies this input or output value as
a literal value of a simple quantity (e.g., one number), and provides
that value. </documentation>
            </annotation>
         </element>
         <element name="BoundingBoxValue" type="ows:BoundingBoxType">
            <annotation>
               <documentation>Identifies this input or output value as
an ows:BoundingBox data structure, and provides that ows:BoundingBox
data structure. </documentation>
            </annotation>
         </element>
```

```
      </choice>
   </group>
   <!-- ========================================================== -->
   <attributeGroup name="ValueReference">
      <annotation>
         <documentation>Reference to an input or output value that is
a web accessible resource. </documentation>
      </annotation>
      <attribute ref="ows:reference" use="required">
         <annotation>
            <documentation>Reference to a web-accessible resource that
can be used as input, or is provided by the process as output. This
attribute shall contain a URL from which this input/output can be
electronically retrieved. </documentation>
         </annotation>
      </attribute>
      <attributeGroup ref="wps:ComplexValueEncoding"/>
   </attributeGroup>
   <!-- ========================================================== -->
   <complexType name="ComplexValueType">
      <annotation>
         <documentation>One complex value (such as an image),
including a definition of the complex value data structure (i.e.,
schema, format, and encoding). </documentation>
      </annotation>
      <complexContent>
         <extension base="anyType">
            <attributeGroup ref="wps:ComplexValueEncoding"/>
         </extension>
      </complexContent>
   </complexType>
   <!-- ========================================================== -->
   <complexType name="LiteralValueType">
      <annotation>
         <documentation>One simple literal value (such as an integer
or real number) that is embedded in the Execute operation request or
response. </documentation>
      </annotation>
      <simpleContent>
         <extension base="string">
            <annotation>
               <documentation>String containing the Literal value
(e.g., "49").</documentation>
            </annotation>
            <attribute name="dataType" type="anyURI" use="optional">
               <annotation>
                  <documentation>Identifies the data type of this
literal input or output. This dataType should be included for each
quantity whose value is not a simple string. </documentation>
               </annotation>
            </attribute>
            <attribute name="uom" type="anyURI" use="optional">
               <annotation>
                  <documentation>Identifies the unit of measure of
this literal input or output. This unit of measure should be referenced
for any numerical value that has units (e.g., "meters", but not a more
complete reference system). Shall be a UOM identified in the Process
description for this input or output. </documentation>
```

```
                </annotation>
            </attribute>
        </extension>
    </simpleContent>
</complexType>
```

EXAMPLE An example Execute operation request using XML encoding is:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Execute service="WPS" version="0.4.0" store="true" status="false"
xmlns="http://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
..\wpsExecute.xsd">
    <ows:Identifier>Buffer</ows:Identifier>
    <DataInputs>
        <Input>
            <ows:Identifier>InputPolygon</ows:Identifier>
            <ows:Title>Playground area</ows:Title>
            <ComplexValueReference
ows:reference="http://foo.bar/some_WFS_request.xml"
schema="http://foo.bar/gml_polygon_schema.xsd" />
        </Input>
        <Input>
            <ows:Identifier>BufferDistance</ows:Identifier>
            <ows:Title>Distance which people will walk to get to a
playground</ows:Title>
            <LiteralValue uom="meters">400</LiteralValue>
        </Input>
    </DataInputs>
    <OutputDefinitions>
        <Output>
            <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>
        </Output>
    </OutputDefinitions>
</Execute>
```

10.3 Execute operation response

10.3.1 Execute response parameters

The response to an Execute operation request depends on the value of the “store” parameter, the number of outputs produced by the process, and the type of output when a single output is produced. A complete description of possible Execute operation responses is provided in Subclause 10.3.2.

In the simplest case, when the “store” parameter is “false”, process execution is successful, only one output is produced, and that output is a ComplexValue, then the Execute operation response is that one complex output, from the process directly to the client. For example, if a WPS process creates one GML document as its output, that GML document will be returned to the client as a direct response to the Execute request.

In all other cases, the response to a valid Execute operation request is an ExecuteResponse XML document. The contents of this ExecuteResponse depend on the value of the “store” parameter, the number of outputs produced, and the types of those outputs, as specified in Subclause 10.3.2. If storage of the outputs has not been requested, the ExecuteResponse document will contain all of the outputs. ComplexValues such as images will be encoded. If storage of the outputs is requested, the ExecuteResponse will reference the web-accessible resource at which the outputs can be retrieved.

An ExecuteResponse document returned by the Execute operation shall include the parts listed in

Table 38 through Table 42. These tables also specify the UML model data type plus the multiplicity and use of each listed part in the Execute operation response.

NOTE 1 To reduce the need for readers to refer to other documents, the first parameter listed below is largely copied from Table 21 in Subclause 9.2.1 of [OGC 05-008]. The Identifier parameter listed is largely copied from Table 1 in Subclause 7.2 of this document. The DataInput and OutputDefinition data structures are copied from Table 27 in Subclause 10.2.1 of this document.

Table 38 — Parts of ExecuteResponse data structure

Name	Definition	Data type and values	Multiplicity and use
version	Specification version for operation	Character String type, not empty Value is specified by each Implementation Specification and Schemas version	One (mandatory)
Identifier	Unambiguous identifier or name of a process	ows:CodeType, as adaptation of MD_Identifier in ISO 19115 Value is process Identifier used in Capabilities document.	One (mandatory)
DataInputs	List of inputs provided to this process execution	DataInputs data structure, see Table 28	Zero or one (optional) Include if any input [a]
Output Definitions	List of definitions of outputs desired from executing this process	OutputDefinitions data structure, see Table 29	Zero or one (optional) Include if input [b]
Process Outputs	List of values of outputs from process execution	ProcessOutputs data structure, see Table 39	Zero or one (optional) Include when process execution succeeded
Status	Execution status of this process	Status data structure, see Table 40	One (mandatory)
status Location	Reference to location where current ExecuteResponse document is stored	URL type	Zero or one (optional) Include when “store” is true in request

a This DataInputs data structure can be omitted as an implementation decision by the WPS server. However, it is often advisable to have the response include this information, so the client can confirm that the request was received correctly, and to provide a source of metadata if the client wishes to store the result for future reference.

b This OutputDefinitions data structure can be omitted as an implementation decision by the WPS server. However, it is often advisable to have the response include this information, so the client can confirm that the request was received correctly, and to provide a source of metadata if the client wishes to store the result for future reference.

NOTE The UML class diagrams contained in Subclause C.6 provide a useful graphical view of the contents of the ExecuteResponse listed in Table 38 through Table 42.

Table 39 — Parts of ProcessOutputs data structure

Name	Definition	Data type	Multiplicity and use
Output	Value of output from process execution	IOValue data structure, see Table 32	One or more (mandatory) Include one for each output, unordered

Table 40 — Parts of Status data structure

Name	Definition	Data type and values	Multiplicity
Process Accepted	Indicates that process has been accepted by server, but is in a queue and has not yet started to execute	Character string type, not empty [b]	Zero or one (conditional) [a]
Process Started	Indicates that process has been accepted by server, and processing has begun	ProcessStarted data structure, see Table 41	Zero or one (conditional) [a]
Process Succeeded	Indicates that process has successfully completed execution	Character string type, not empty [c]	Zero or one (conditional) [a]
Process Failed	Indicates that execution of this process has failed, and includes error information [d]	ProcessFailed data structure, see Table 42	Zero or one (conditional) [a]

a One and only one of these four elements can be present

b The contents of this human-readable text string is left open to definition by each server, but is expected to include any messages the server wishes to let the clients know. Such information could include how long the queue is, or any warning conditions that may have been encountered. The client may display this text to a human user.

c The contents of this human-readable text string is left open to definition by each server, but is expected to include any messages the server wished to let the clients know, such as how long the process took to execute, or any warning conditions that may have been encountered. The client may display this text string to a human user. The client should use the presence of this parameter to trigger automated or manual access to the results of process execution. If manual access is intended, the client should use the presence of this parameter to present the results as downloadable links to the user.

d The reason(s) for failure is given in the exception report.

Table 41 — Parts of ProcessStarted data structure

Name	Definition	Data type and values	Multiplicity and use
message	Human-readable text string expected to include any messages server may wish to let clients know	Character string type [a]	One (mandatory)
percent Completed	Percentage of process that has been completed, where 0 means process has just started, and 100 means process is complete, expected to be accurate to within ten percent	Integer type Values from 0 to 100, inclusive	Zero or one (optional) Include when process expected to execute for long time [b]

a A human-readable text string whose contents are left open to definition by each WPS server, but is expected to include any messages the server may wish to let the clients know. Such information could include how much longer the process may take to execute, or any warning conditions that may have been encountered to date. The client may display this text to a human user.

b Recommended for use when processes take more than a minute to complete.

Table 42 — Parts of ProcessFailed data structure

Name	Definition	Data type	Multiplicity
Exception Report	Exception report containing one or more Exception data structures, each signalling detection of an independent error	ExceptionReport data structure, see Table 18 in OGC 05-008	One (mandatory)

Once a process has completed successfully, the Status data structure shall include the ProcessSucceeded parameter, and the ProcessOutputs data structure shall be fully populated. Typically, it is expected that the Outputs will be stored as web-accessible resources, pointed to using the "reference" element, as shown in this example XML fragment:

```
<ComplexResultReference xlink:href="http://foo.bar/execute_results_MapExtent.xml"/>
```

The information available from the status URL shall be updated with an updated ExecuteResponse document as soon as the results are available. If the URL included in a ComplexValueReference is accessed before the process has had time to complete and populate the online resource(s), the server shall return an Error 403 (Not Found).

The ExecuteResponse document normally contains the input that was provided by the client, in the DataInputs and OutputDefinitions data structures. This information includes any URIs provided in the execute request. If an input was embedded in the request, then the server may generate and populate a URL for the payload and reference it, instead of returning a copy of the original payload in the body of the ExecuteResponse. This is recommended in the case of large files.

The WPS is not specifically designed to store outputs for a long time, but it does not preclude such storage either. Clients may wish to download the outputs to some other web-accessible location if long term storage is required.

The "Multiplicity and use" columns in Table 28 through Table 42 specify the optionality of each listed parameter and data structure in the Execute operation response. Each "mandatory" parameter and data structure shall be implemented by all OWS servers, using an allowed value(s). Each "optional" parameter and data structure shall also be implemented by all OWS servers, using an allowed value(s), for each implemented process to which that data is relevant and available.

10.3.2 Control of Execute response and storage of outputs

As stated above, the response to an Execute operation request depends on the value of the "store" parameter, the number of outputs produced by the process, and the type of output when a single output produced. In the simplest case, when the "store" parameter is "false", process execution is successful, only one output is produced, and that output is a ComplexValue, then the Execute operation response is that one complex output, from the process directly to the client. No ExecuteResponse XML document is returned or stored in this case.

In all other cases, the response to a valid Execute operation request is an ExecuteResponse XML document. The contents of this ExecuteResponse depend on the value of the "store" parameter and the forms of the output(s). The behaviour of a WPS for each form of output is shown in Table 43. Notice that when a WPS returns multiple forms of output, they shall be returned within a single ExecuteResponse document. The behaviours of a WPS for the two values of the "store" parameter are shown in Table 44.

Table 43 — Server behaviour for output forms

Form of output	Server behaviour
ComplexValueReference	Store Complex Value at web-accessible location, and store that URL in ExecuteResponse document
ComplexValue	Return Complex Value in ExecuteResponse document [a]
LiteralValue	Return Literal Value in ExecuteResponse document
BoundingBoxValue	Return BoundingBox value in ExecuteResponse document
a If the "store" parameter is "false", process execution was successful, there is only one output, and that output has a ComplexValue, then this ComplexValue shall be returned to the client outside of any ExecuteResponse document.	

Table 44 — Server behaviour depending on "store" parameter

When store = "false" (default)	When store = "true"
Determine all requested outputs, storing each output URL or value in ExecuteResponse document [a] Return ExecuteResponse document [a] Do not store ExecuteResponse document.	Store initial ExecuteResponse document at web-accessible location Return initial ExecuteResponse document Determine all requested outputs, storing each output value either in the ExecuteResponse document or at a separate URL, and updating the PercentCompleted element on a regular basis Determine all requested outputs, storing each output URL or value in ExecuteResponse document Store final ExecuteResponse document at same web-accessible location
a If the "store" parameter is "false", process execution was successful, there is only one output, and that output has a ComplexValue, then this ComplexValue shall be returned to the client outside of any ExecuteResponse document.	

10.3.3 Execute response XML encoding

The following schema fragment specifies the contents and structure of an Execute operation response, always encoded in XML:

```
   <element name="ExecuteResponse" type="wps:ExecuteResponseType">
      <annotation>
         <documentation>WPS Execute operation response. By default,
this XML document is delivered to the client in response to an Execute
request. If "status" is "false" in the Execute operation request, this
document is normally returned when process execution has been
completed.
         If "status" in the Execute request is "true", this response
shall be returned as soon as the Execute request has been accepted for
processing. In this case, the same XML document is also made available
as a web-accessible resource from the URL identified in the
statusLocation, and the WPS server shall repopulate it once the process
has completed. It may repopulate it on an ongoing basis while the
process is executing.
         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. </documentation>
      </annotation>
   </element>
   <!-- ========================================================== -->
   <complexType name="ExecuteResponseType">
      <annotation>
         <documentation>Response to an Execute operation request.
</documentation>
      </annotation>
      <sequence>
         <element ref="ows:Identifier">
            <annotation>
               <documentation>Identifier of the Process requested to
be executed. This Process identifier shall be as listed in the
ProcessOfferings section of the WPS Capabilities document.
</documentation>
            </annotation>
         </element>
         <element name="Status" type="wps:StatusType">
            <annotation>
               <documentation>Execution status of this process.
</documentation>
            </annotation>
         </element>
         <element name="DataInputs" type="wps:DataInputsType"
minOccurs="0">
            <annotation>
               <documentation>Inputs that were provided as part of the
execute request. This element can be omitted as an implementation
decision by the WPS server. However, it is often advisable to have the
response include this information, so the client can confirm that the
request was received correctly, and to provide a source of metadata if
the client wishes to store the result for future reference.
</documentation>
            </annotation>
         </element>
         <element name="OutputDefinitions"
type="wps:OutputDefinitionsType" minOccurs="0">
            <annotation>
               <documentation>Complete list of Output data types that
were requested as part of the Execute request. This element can be
omitted as an implementation decision by the WPS server. However, it is
often advisable to have the response include this information, so the
client can confirm that the request was received correctly, and to
provide a source of metadata if the client wishes to store the result
for future reference. </documentation>
            </annotation>
         </element>
         <element name="ProcessOutputs" minOccurs="0">
            <annotation>
               <documentation>List of values of the Process output
parameters. Normally there would be at least one output when the
process has completed successfully. If the process has not finished
```

```
executing, the implementer can choose to include whatever final results
are ready at the time the Execute response is provided. If the
reference locations of outputs are known in advance, these URLs may be
provided before they are populated. </documentation>
            </annotation>
            <complexType>
               <sequence>
                  <element name="Output" type="wps:IOValueType"
maxOccurs="unbounded">
                     <annotation>
                        <documentation>Unordered list of values of
all the outputs produced by this process. It is not necessary to
include an output until the Status is ProcessSucceeded.
</documentation>
                     </annotation>
                  </element>
               </sequence>
            </complexType>
         </element>
      </sequence>
      <attribute name="statusLocation" type="anyURI" use="optional">
         <annotation>
            <documentation>The URL referencing the location from which
the ExecuteResponse can be retrieved. If "status" is "true" in the
Execute request, the ExecuteResponse should also be found here as soon
as the process returns the initial response to the client. It should
persist at this location as long as the outputs are accessible from the
server. The outputs may be stored for as long as the implementer of the
server decides. If the process takes a long time, this URL can be
repopulated on an ongoing basis in order to keep the client updated on
progress. Before the process has succeeded, the ExecuteResponse
contains information about the status of the process, including whether
or not processing has started, and the percentage completed. It may
also optionally contain the inputs and any ProcessStartedType interim
results. When the process has succeeded, the ExecuteResponse found at
this URL shall contain the output values or references to them.
</documentation>
         </annotation>
      </attribute>
      <attribute name="version" type="ows:VersionType" use="required">
         <annotation>
            <documentation>Version of the WPS interface specification
implemented by the server. </documentation>
         </annotation>
      </attribute>
   </complexType>
   <!-- =========== STATUS ======================================== -->
   <!-- ============================================================ -->
   <complexType name="StatusType">
      <annotation>
         <documentation>Description of the status of process
execution. </documentation>
      </annotation>
      <choice>
         <element name="ProcessAccepted" type="string">
            <annotation>
               <documentation>Indicates that this process has been
accepted by the server, but is in a queue and has not yet started to
```



```
execute. The contents of this human-readable text string is left open
to definition by each server implementation, but is expected to include
any messages the server may wish to let the clients know. Such
information could include how long the queue is, or any warning
conditions that may have been encountered. The client may display this
text to a human user. </documentation>
            </annotation>
         </element>
         <element name="ProcessStarted" type="wps:ProcessStartedType">
            <annotation>
               <documentation>Indicates that this process has been has
been accepted by the server, and processing has begun. </documentation>
            </annotation>
         </element>
         <element name="ProcessSucceeded" type="string">
            <annotation>
               <documentation>Indicates that this process has
successfully completed execution. The contents of this human-readable
text string is left open to definition by each server, but is expected
to include any messages the server may wish to let the clients know,
such as how long the process took to execute, or any warning conditions
that may have been encountered. The client may display this text string
to a human user. The client should make use of the presence of this
element to trigger automated or manual access to the results of the
process. If manual access is intended, the client should use the
presence of this element to present the results as downloadable links
to the user. </documentation>
            </annotation>
         </element>
         <element name="ProcessFailed" type="wps:ProcessFailedType">
            <annotation>
               <documentation>Indicates that execution of this process
has failed, and includes error information. </documentation>
            </annotation>
         </element>
      </choice>
   </complexType>
   <!-- ========================================================== -->
   <complexType name="ProcessStartedType">
      <annotation>
         <documentation>Indicates that this process has been has been
accepted by the server, and processing has begun. </documentation>
      </annotation>
      <simpleContent>
         <extension base="string">
            <annotation>
               <documentation>A human-readable text string whose
contents are left open to definition by each WPS server, but is
expected to include any messages the server may wish to let the clients
know. Such information could include how much longer the process may
take to execute, or any warning conditions that may have been
encountered to date. The client may display this text to a human user.
</documentation>
            </annotation>
            <attribute name="PercentCompleted" use="optional">
               <annotation>
                  <documentation>Percentage of the process that has
been completed, where 0 means the process has just started, and 100
```

```
means the process is complete. This attribute should be included if the
process is expected to execute for a long time (i.e. more than a few
minutes). This percentage is expected to be accurate to within ten
percent. </documentation>
                </annotation>
                <simpleType>
                    <restriction base="integer">
                        <minInclusive value="0"/>
                        <maxInclusive value="100"/>
                    </restriction>
                </simpleType>
            </attribute>
        </extension>
    </simpleContent>
</complexType>
<!-- ========================================================== -->
<complexType name="ProcessFailedType">
    <annotation>
        <documentation>Indicator that the process has failed to
execute successfully. The reason for failure is given in the exception
report. </documentation>
    </annotation>
    <sequence>
        <element ref="ows:ExceptionReport"/>
    </sequence>
</complexType>
```

This schema fragment uses parts of the schema fragment for the Execute operation request, listed in Subclause 10.2.3.

EXAMPLE An example response to the Execute operation request example given in Subclause 10.2.3 is:

```
<?xml version="1.0" encoding="UTF-8"?>
<ExecuteResponse statusLocation="http://foo.bar/execute_response_url.xml"
version="0.4.0" xmlns="http://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 ..\wpsExecute.xsd">
    <ows:Identifier>Buffer</ows:Identifier>
    <Status>
        <ProcessSucceeded/>
    </Status>
    <DataInputs>
        <Input>
            <ows:Identifier>InputPolygon</ows:Identifier>
            <ows:Title>Playground area</ows:Title>
            <ComplexValueReference
ows:reference="http://foo.bar/some_WFS_request.xml"
schema="http://foo.bar/gml_polygon_schema.xsd" />
        </Input>
        <Input>
            <ows:Identifier>BufferDistance</ows:Identifier>
            <ows:Title>Distance which people will walk to get to a
playground</ows:Title>
            <LiteralValue uom="meters">400</LiteralValue>
        </Input>
    </DataInputs>
    <OutputDefinitions>
        <Output>
```



```
            <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>
        </Output>
    </OutputDefinitions>
    <ProcessOutputs>
        <Output>
            <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>
            <ComplexValueReference
ows:reference="http://foo.bar/buffered_polygon.xml"/>
        </Output>
    </ProcessOutputs>
</ExecuteResponse>
```

This example response includes the statusLocation as an attribute of the <ExecuteResponse>. This attribute contains a URL that will return an ExecuteResponse document, which contains the latest status information about the Execute request, and, if the process has completed, the URL(s) at which the output(s) may be retrieved. If the process has not completed by the time the response is sent, the location(s) of the output(s) will not necessarily be identified. In this example, because the execute request specified status="false", the URL at which the status information is located will be populated when the process has completed. Therefore, if the WPS delays creation of the Execute response until after the process has completed, that URL will end up with contents identical to that of the execute response.

10.3.4 Execute exceptions

When a WPS server encounters an error while performing an Execute operation, it shall return an exception report message as specified in Subclause 7.4 of [OGC 05-008]. The allowed standard exception codes shall include those listed in Table 45. For each listed exceptionCode, the contents of the "locator" parameter value shall be as specified in the right column of Table 45.

NOTE To reduce the need for readers to refer to other documents, the first three values listed below are copied from Table 20 in Subclause 8.3 of [OGC 05-008].

Table 45 — Exception codes for Execute operation

exceptionCode value	Meaning of code	"locator" value
MissingParameterValue	Operation request does not include a parameter value, and this server did not declare a default value for that parameter	Name of missing parameter
InvalidParameterValue	Operation request contains an invalid parameter value	Name of parameter with invalid value
NoApplicableCode	No other exceptionCode specified by this service and server applies to this exception	None, omit "locator" parameter
ServerBusy	The server is too busy to accept and queue the request at this time.	None, omit "locator" parameter
FileSizeExceeded	The file size of one of the input parameters was too large for this process to handle.	None, omit "locator" parameter

Annex A
(normative)

Abstract test suite

An abstract test suite is not provided in this version of this Implementation Specification.

Annex B
(normative)

XML Schema Documents

In addition to this document, this specification includes several normative XML Schema Documents. These XML Schema Documents are bundled in a zip file with the present document.

The WPS abilities now specified in this document use five new XML Schema Documents, all included in the zip file with this document. These XML Schema Documents combine the XML schema fragments listed in various subclauses of this document, eliminating duplications. These XML Schema Documents roughly match the five UML packages described in Annex C, and are named:

wpsCommon.xsd (implements WPS Service package)

wpsDescribeProcess.xsd

wpsExecute.xsd

wpsGetCapabilities.xsd

owsDomaintype.xsd

NOTE The owsDomaintype.xsd is a proposed addition to OWS Common [OGC 05-008], and is thus placed in the “ows” XML namespace. The draft expanded version of OWS Common, OGC 05-008r1, included a draft owsDomainType.xsd. However that draft owsDomainType.xsd has been simplified for use by this WPS.

These XML Schema Documents use and build on the OWS Common XML Schema Documents specified in [OGC 05-008] and named:

ows19115subset.xsd (slightly modified per change request OGC 05-059r1)

owsCommon.xsd (slightly modified per change request OGC 05-059r1)

owsDataIdentification.xsd (slightly modified per change request OGC 05-059r1)

owsExceptionReport.xsd

owsGetCapabilities.xsd

owsOperationsMetadata.xsd (slightly modified per Subclause 8.3.2)

owsServiceIdentification.xsd

owsServiceProvider.xsd

Since four of these OWS Common XML Schema Documents are slightly modified, all these documents are also included in the zip file with this document.

All these XML Schema Documents contain documentation of the meaning of each element and attribute, and this documentation shall be considered normative as specified in Subclause 11.6.3 of [OGC 05-008].

Annex C
(informative)

UML model

C.1 Introduction

This annex provides a UML model of the WPS interface, using the OGC/ISO profile of UML summarized in Subclause 5.3 of [OGC 05-008].

Figure C.1 is a simple UML diagram summarizing the WPS interface. This class diagram shows that the WPService class inherits the getCapabilities operation from the OGCWebService interface class, and adds the "describeProcess" and "execute" operations. (This capitalization of names uses the OGC/ISO profile of UML.)

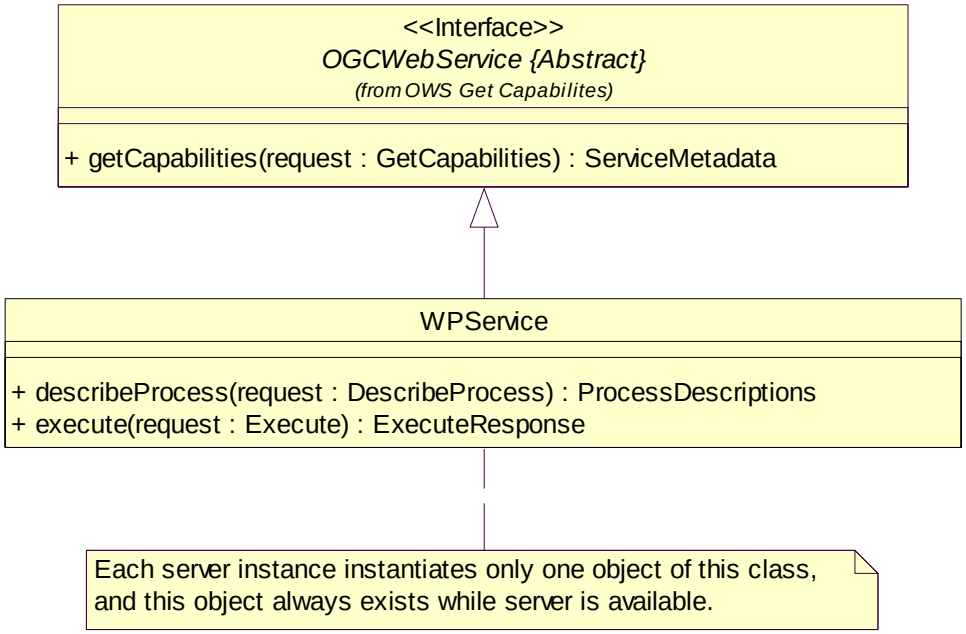

Figure C.1 — WPS interface UML diagram

Each of the three operations uses a request and a response data type, each of which is defined by one or more additional UML classes. The following subclauses provide a more complete UML model of the WPS interface, adding UML classes defining the operation request and response data types.

C.2 UML packages

The WPS interface UML model is organized in four packages, as shown in the package diagram in Figure C.2. These four WPS-specific packages make direct use of four OWS Common packages, named OWS Get Capabilities, ISO 19115 Subset, OWS Common, and OWS Domain. (The OWS Get Capabilities package makes use of the OWS Operations Metadata, OWS Service Identification, and OWS Service Provider packages, not shown in this diagram). This package diagram shows the dependencies among the various packages shown.

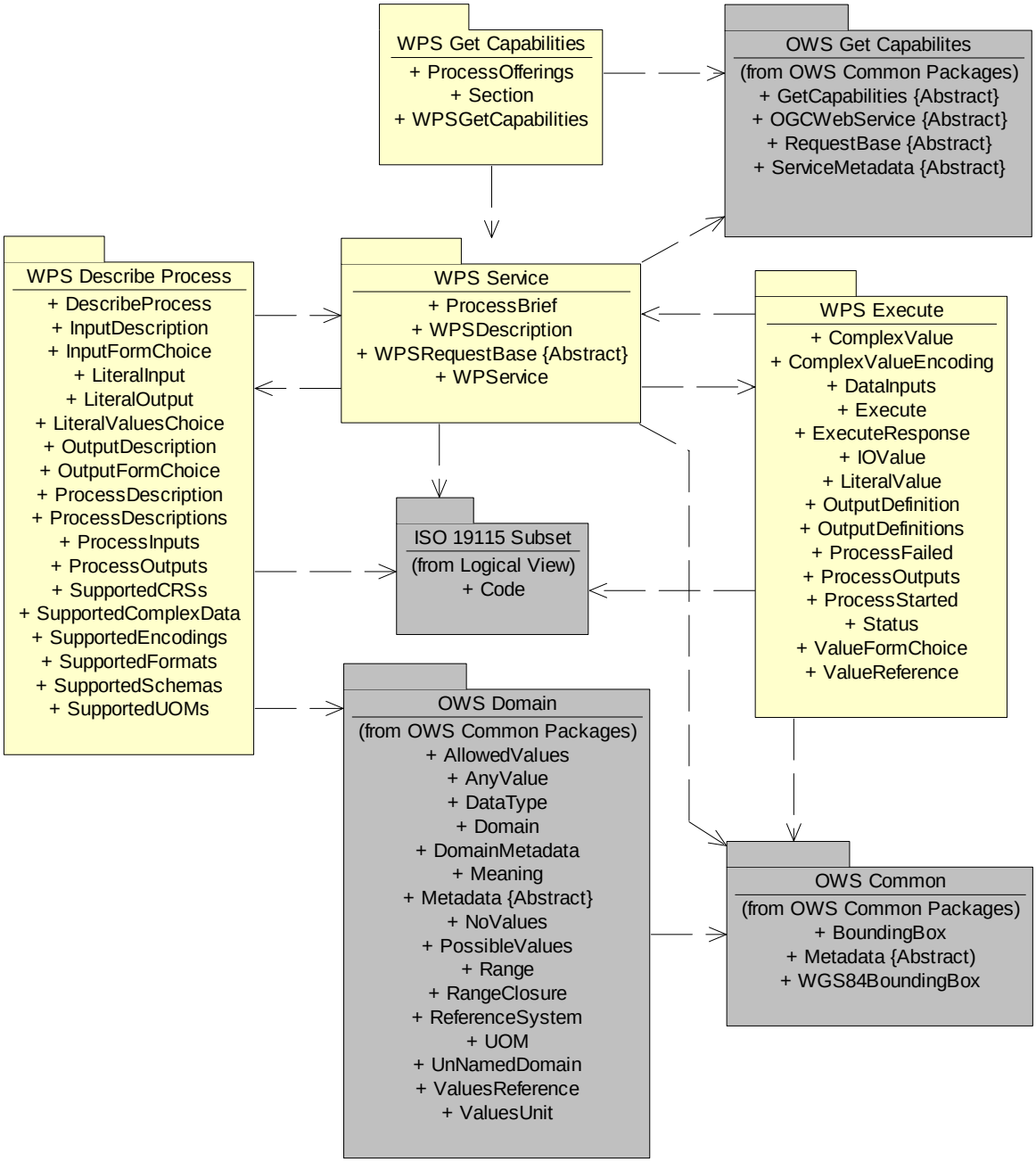

Figure C.2 — WPS interface package diagram

Each of the four WPS-specific packages shown in Figure C.2 is described in the following subclauses. The OWS Get Capabilities, OWS Common, OWS Operations Metadata, OWS Service Identification, OWS Service Provider, and ISO 19115 Subset packages are described in Annex B of [OGC 05-008]. The OWS Domain package is a proposed addition to OWS Common [OGC 05-008], and is described in Subclause C.7 and Annex D of this document.

C.3 WPS Service package

The WPS Service package is shown in the class diagram in Figure C.3. This diagram does not show the classes used by the WPS operation requests and responses, which are shown (with part of this package) in the WPS Get Capabilities, Describe Process, and Execute packages. This diagram also shows four used classes from other packages. The WPSDescription and ProcessBrief classes introduced by this package are further defined by Table 1 and Table 2 in this document.

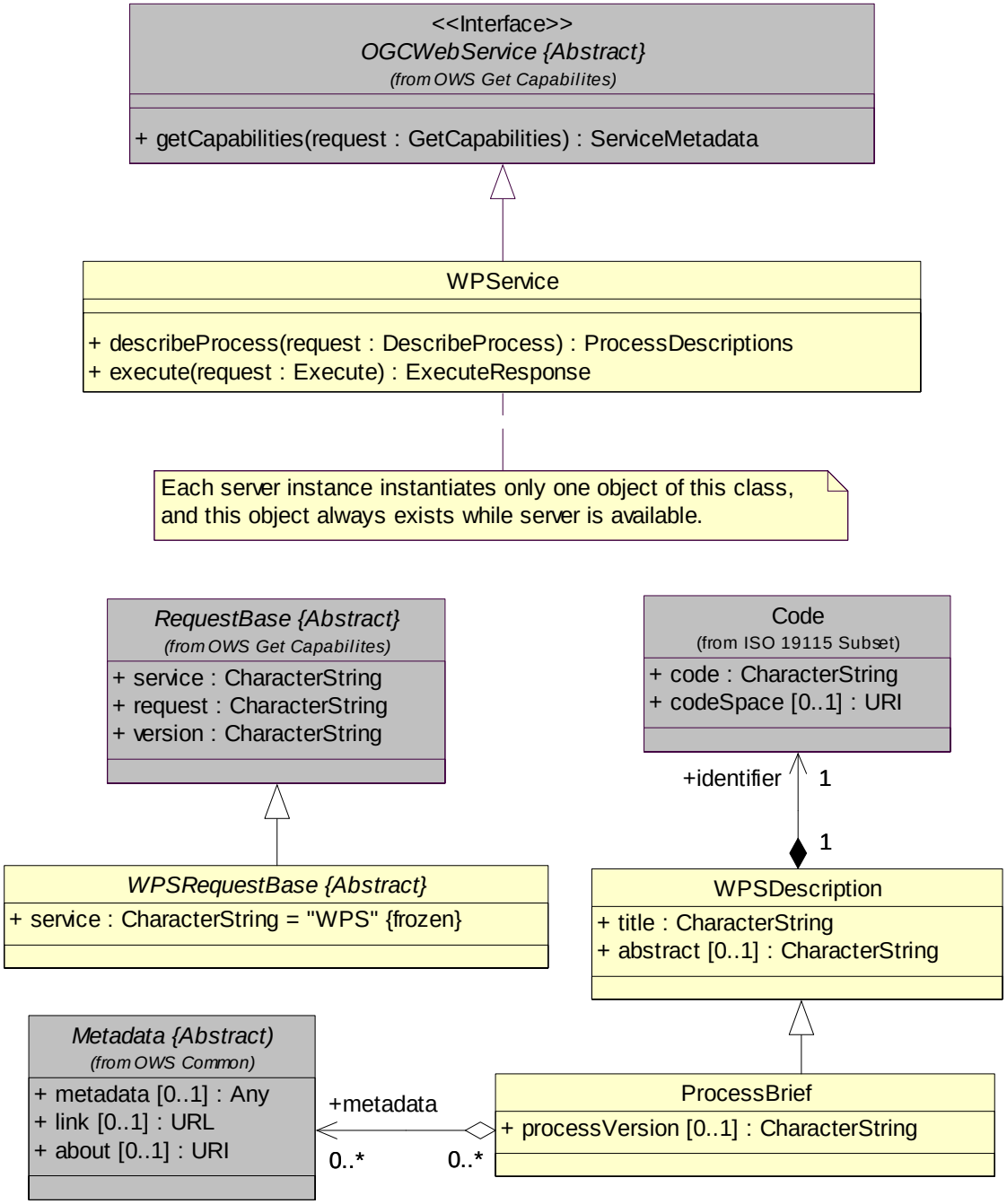

Figure C.3 — WPS Service package class diagram

C.4 WPS Get Capabilities package

The WPS Get Capabilities package is shown in the class diagram in Figure C.4. This diagram also shows many classes from the other packages, with grey fill. The classes introduced by this package are further defined by Table 4 through Table 7 in this document.

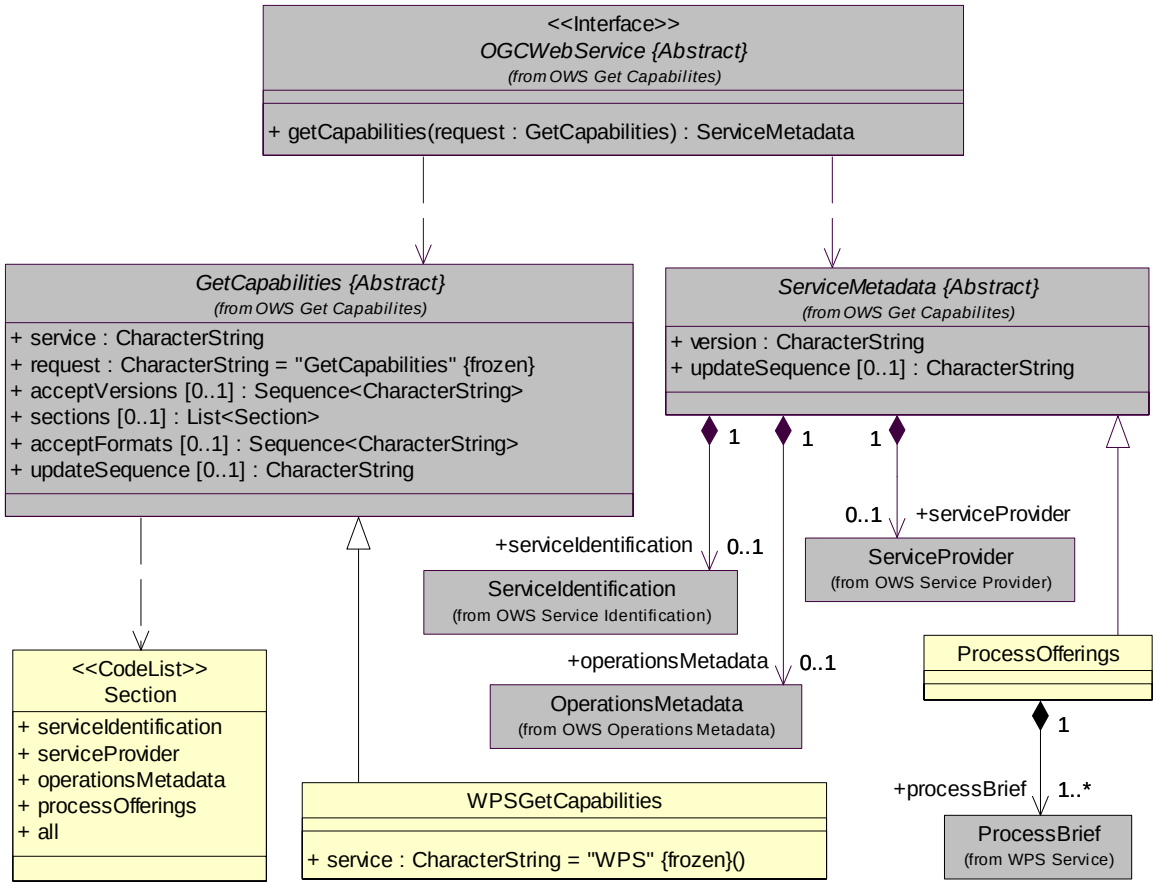

Figure C.4 — WPS Get Capabilities package class diagram

C.5 Describe Process package

The Describe Process package is shown in the class diagrams in Figures C.5 and C.6. These diagrams also show many classes from the other packages, with grey fill. The classes introduced by this package are further defined by Table 9 through Table 25 in this document.

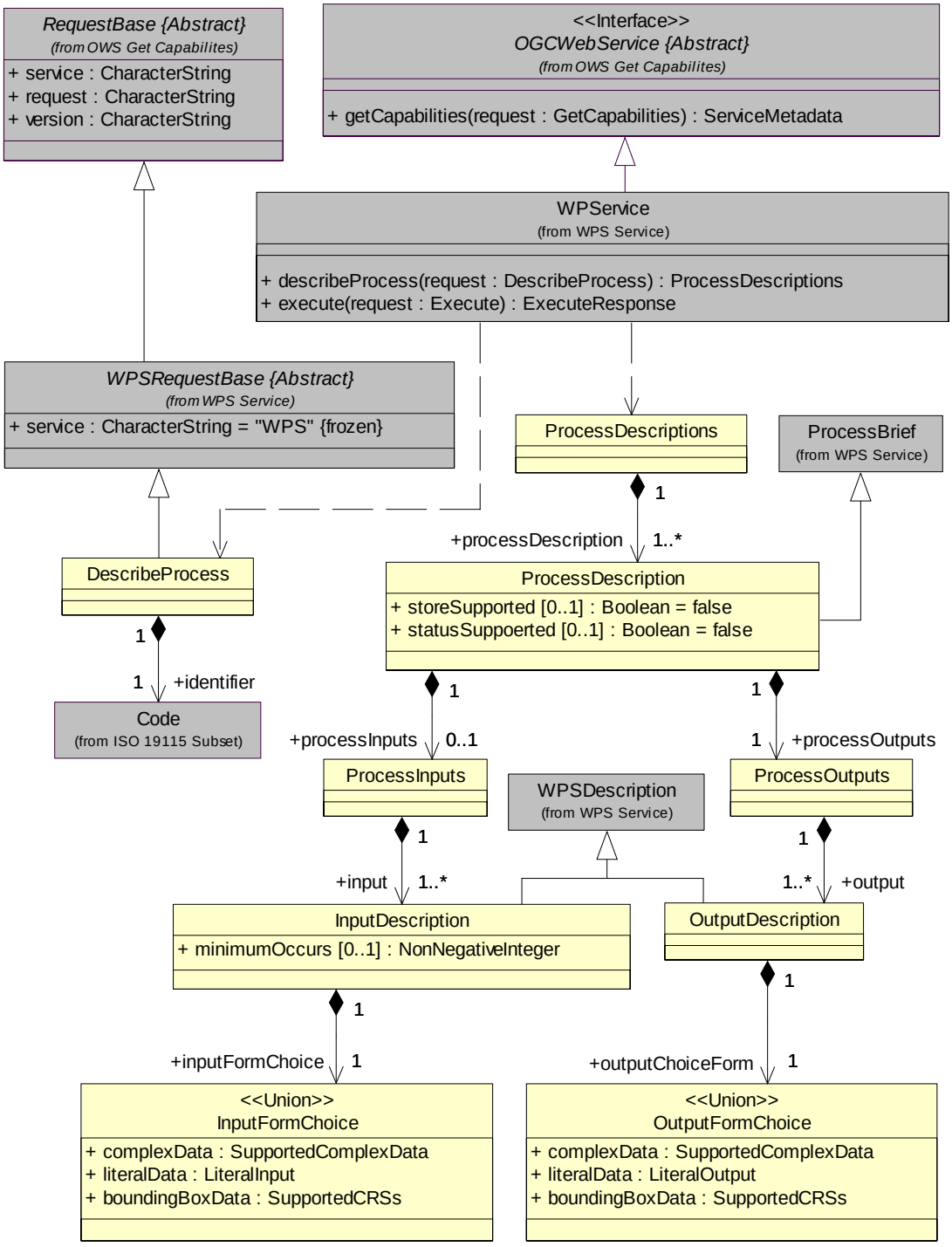

Figure C.5 — Describe Process package class diagram, part 1

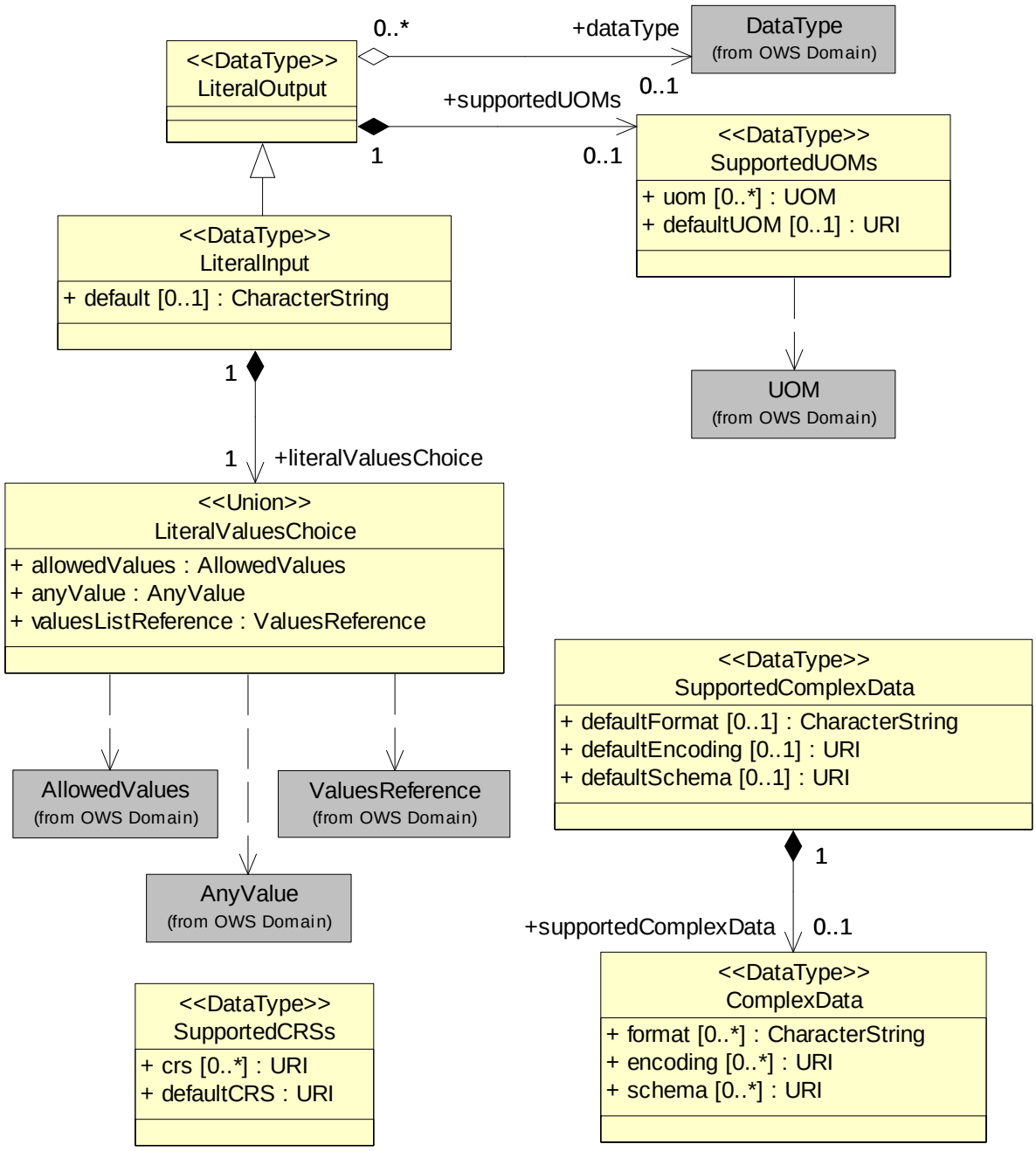

Figure C.6 — Describe Process package class diagram, part 2

C.6 Execute package

The Execute package is shown in the class diagrams in Figures C.7 and C.8. These diagrams also show many classes from the other packages, with grey fill. The classes introduced by this package are further defined by Table 27 through Table 42 in this document.

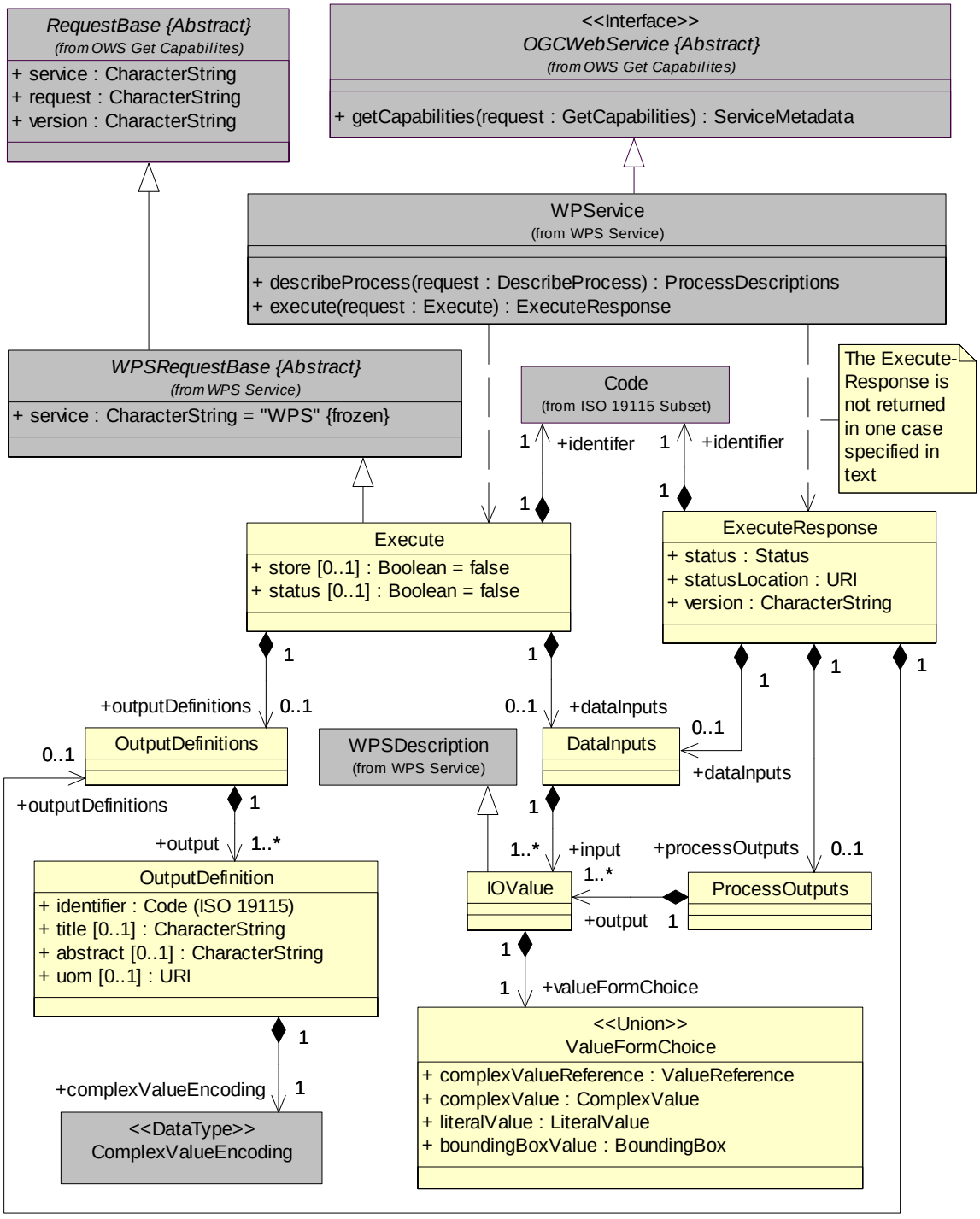

Figure C.7 — Execute package class diagram, part 1

Notice that the ComplexValueEncoding class shown in Figure C.7 is detailed in Figure C.8.

As indicated by the note in Figure C.7, the response to an Execute operation request is not the ExecuteResponse document in one special case. When the “store” parameter is “false”, process execution is successful, only one output is produced, and that output is a ComplexValue, then the Execute operation response is that one complex output, from the process directly to the client. In this case, no ExecuteResponse XML document is returned or stored.

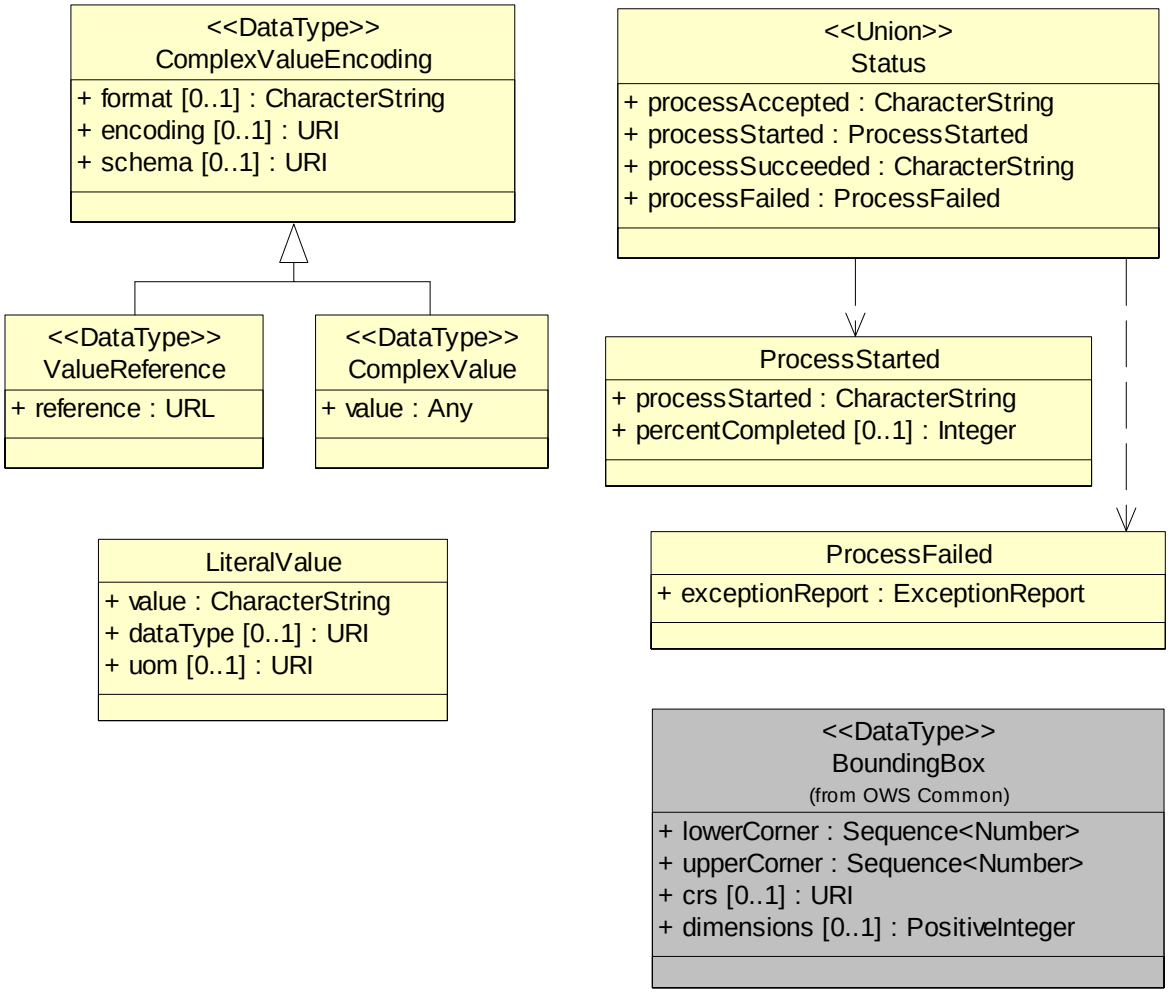

Figure C.8 — Execute package class diagram, part 2

C.7 Domain package

The Domain package is shown in the class diagram in Figure C.9. The classes introduced by this package are further defined by Tables D.1 through D.8 in this document.

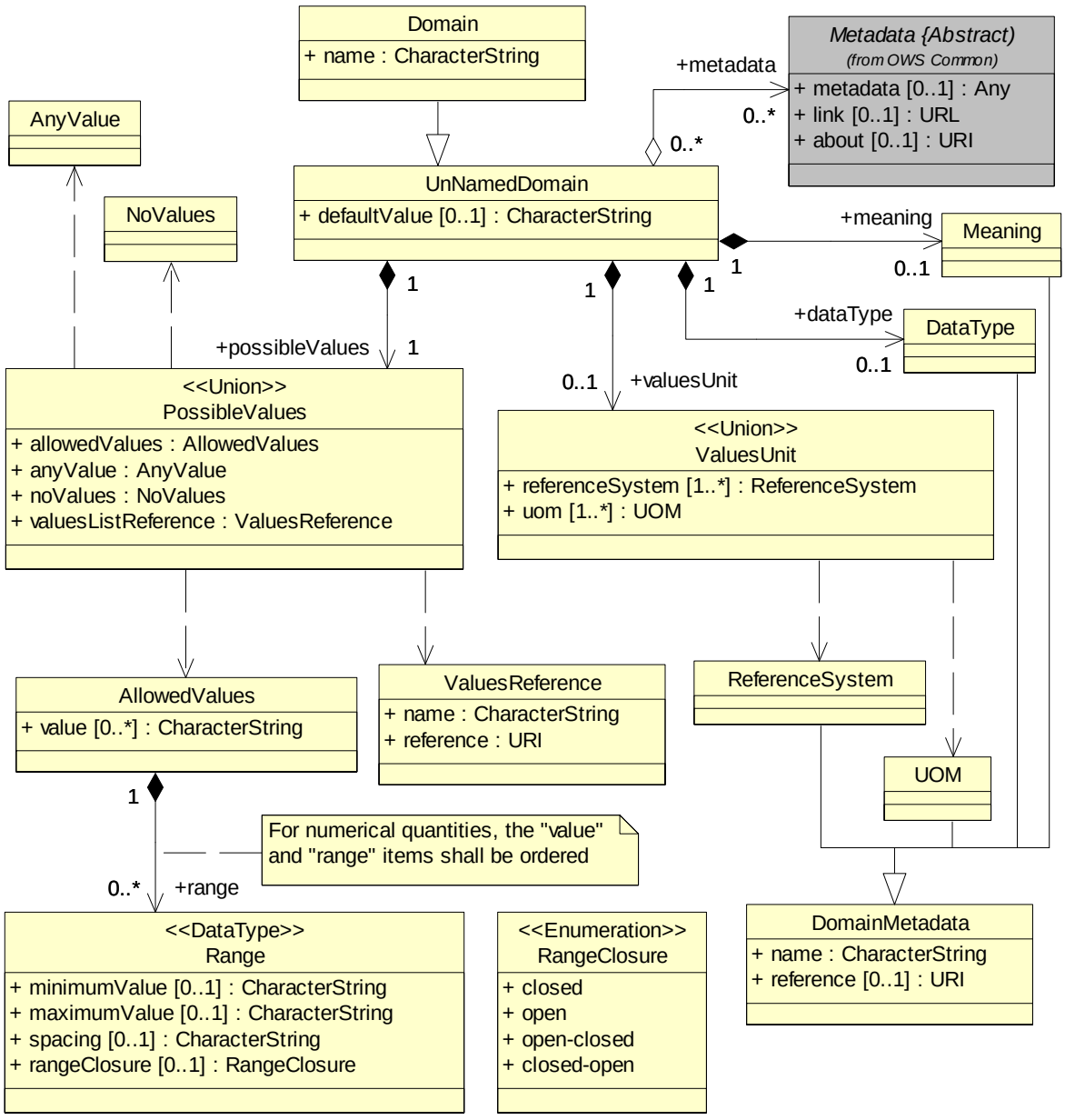

Figure C.9 — Domain package class diagram

Annex D
(normative)

DomainType data structure

D.1 Overview

This annex specifies the DomainType data structure that is used by this WPS.

NOTE The owsDomainType.xsd is a proposed addition to OWS common [OGC 05-008], and is thus in the "ows" XML namespace. The draft expanded version of OWS Common OGC 05-008r1 included a draft owsDomainType.xsd. However that draft owsDomainType.xsd has been simplified for use by this WPS.

Some data structures can specify the allowed values and metadata for a parameter or other quantity. For all such quantities, the DomainType data structure specified in Table D.1 shall be used or adapted as required.

NOTE 1 The first 6 parameters listed below (with partially grey background) are copied from Table D.2 below.

Table D.1 — Parts of DomainType data structure

Name	Definition	Data type	Multiplicity
PossibleValues	Inherited from UnNamed-DomainType data structure, see Table D.2	PossibleValues data structure	One (mandatory)
DefaultValue		Character string type, not empty	Zero or one (optional)
Meaning		ows:DomainMetadata data structure	Zero or one (optional)
DataType		ows:DomainMetadata data structure	Zero or one (optional)
ValuesUnit		ValuesUnit data structure	Zero or one (optional)
Metadata		ows:DomainMetadata data structure	Zero or more (optional)
name	Name or identifier of this quantity	Character string type, not empty	One (mandatory)

Table D.2 — Parts of UnNamedDomainType data structure

Name	Definition	Data type	Multiplicity and use
Possible Values	Specifies the possible values of this quantity [a]	PossibleValues data structure, see Table D.3	One (mandatory)
Default Value	Default value for this quantity	Character string type, not empty [b]	Zero or one (optional) Include when is a default
Meaning	Reference to meaning or semantics of this value or set of values	DomainMetadata data structure, see Table D.7	Zero or one (optional) Include when useful
DataType	Reference to the data type of this set of values	DomainMetadata data structure, see Table D.7	Zero or one (optional) Include when useful [c]
Values Unit	Indicates that this quantity has units or reference system, and provides the value used [d]	ValuesUnit data structure, see Table D.4	Zero or one (optional) Include when values have units or reference system
Metadata	Additional metadata about domain of this quantity	ows:Metadata, see Table 23 of OGC 05-008	Zero or more (optional) One for each such metadata object [e]

a For quantities that contain a list or sequence of values, these values shall be for individual values in the list.

b Default value shall be string encoding of any value of another data type.

c This metadata should be referenced or included unless this information is clearly specified elsewhere.

d Provides the identifier of the units or reference system used by the AllowedValues or ValuesListReference.

e These metadata objects can be included in any order. A list of the required and/or optional metadata objects for each quantity should be specified in the Implementation Specification for a specific OWS service.

Table D.3 — Parts of PossibleValues data structure

Name	Definition	Data type	Multiplicity
Allowed Values	List of all valid values and/or ranges of values for this quantity	AllowedValues data structure, see Table D.5	Zero or one (conditional) [a]
AnyValue	Specifies that any value is allowed for this quantity	Empty data structure	Zero or one (conditional) [a]
NoValues	Specifies that no values are allowed for this quantity	Empty data structure	Zero or one (conditional) [a]
Values Reference	Reference to list of all valid values and/or ranges of values for this quantity	ValuesReference data structure, see Table D.8	Zero or one (conditional) [a]

a One and only one of these four items shall be included.

Table D.4 — Parts of ValuesUnit data structure

Name	Definition	Data type	Multiplicity
UOM	Identifier of unit of measure of this set of values	DomainMetadata data structure, see Table D.7	Zero or one (conditional) [a]
Reference System	Identifier of reference system used by this set of values	DomainMetadata data structure, see Table D.7	Zero or one (conditional) [a]
a One and only one of these items shall be included.			

Table D.5 — Parts of AllowedValues data structure

Name	Definition	Data type	Multiplicity and use
Value	Value for this quantity	Character string type, not empty [a]	Zero or more (optional) One for each separate value [b]
Range	Range of values of numeric parameter [c]	Range data structure, see Table D.6	Zero or more (optional) One for each separate range [b]
a Default value shall be string encoding of any value of another data type. b For numeric parameters, signed values shall be ordered from negative infinity to positive infinity. c This range can be continuous or discrete, defined by a fixed spacing between adjacent valid values. If the MinimumValue or MaximumValue is not included, there is no value limit in that direction. Inclusion of the specified minimum and maximum values in the range shall be defined by the rangeClosure.			

Table D.6 — Parameters in Range data structure

Name	Definition	Data type and values	Multiplicity and use
Minimum Value	Minimum value of this range of this numeric parameter	Character String, not empty [a] Default is negative infinity	Zero or one (optional) Include when not default
Maximum Value	Maximum value of this range of this numeric parameter	Character String, not empty [a] Default is positive infinity	Zero or one (optional) Include when not default
Spacing	Regular distance or spacing between allowed values in this range [b]	Character String, not empty [a]	Zero or one (optional) Include when range is not continuous
range Closure	Specifies which of minimum and maximum values are included in this range	Enumeration type, either: ”closed” ”open” ”open-closed” ”closed-open”	Zero or one (optional) Include when not default of “closed”
a Parameter value shall be string encoding of any value of another data type. b This range may be continuous or discrete, defined by this fixed spacing between adjacent valid values.			

Table D.7 — Parameters in DomainMetadata data structure

Name	Definition	Data type	Multiplicity and use
Name	Human-readable name of domain metadata described by associated referenced document	Character String, not empty [a]	One (mandatory)
reference	Reference to metadata about this domain	URI	Zero or one (optional) Include when available
a Reference to metadata recorded elsewhere, either external to this XML document or within it. Whenever practical, this attribute with type anyURI should be a URL from which this metadata can be electronically retrieved. Alternately, this attribute can reference a URN for well-known metadata. For example, such a URN could be a URN defined in the "ogc" URN namespace.			

NOTE Possible URNs in the "ogc" URN namespace for data types are proposed in change request OGC 05-060.

Table D.8 — Parameters in ValuesReference data structure

Name	Definition	Data type	Multiplicity and use
Name	Human-readable name of domain metadata described by associated referenced document	Character String, not empty	One (mandatory)
reference	Reference to metadata about this domain	URI [a]	Zero or one (optional) Include when available
a Reference to metadata recorded elsewhere, either external to this XML document or within it. Whenever practical, this attribute with type anyURI should be a URL from which this metadata can be electronically retrieved. Alternately, this attribute can reference a URN for well-known metadata. For example, such a URN could be a URN defined in the "ogc" URN namespace.			

D.2 Domain typed parameter encoding

No KVP encoding of these domain typed parameters is now specified, since KVP encoding is probably impractical.

The XML Schema fragment for encoding domain type metadata shall be the attached file owsDomainType.xsd. That Schema uses the owsCommon.xsd schema, also attached.

EXAMPLE A XML document fragment using the ows:DomainType in the ows:Parameter element specified in Subclause 7.4.5 of [OGC 05-008] is:

```
<Parameter name="Length">
   <AllowedValues>
      <Value>1.0</Value>
      <Range>
         <MinimumValue>4.0</MinimumValue>
         <MaximumValue>7.0</MaximumValue>
      </Range>
      <Value>10.0</Value>
      <Range>
         <MinimumValue>15.0</MinimumValue>
         <MaximumValue>17.0</MaximumValue>
      </Range>
```



```
       </AllowedValues>
       <Meaning>TBD definition of parameter. </Meaning>
       <DataType
ows:reference="urn:ogc:def:dataType:OGC:0.0:Double">Double</DataType>
       <UOM ows:reference="urn:ogc:def:uom:OGC:1.0:metre">metre</UOM>
       <Metadata xlink:href="urn:ogc:def:crs:EPSG:6.6:4326"></Metadata>
   </Parameter>
```

Bibliography

[1] CGDI architecture pages at http://www.geoconnections.org/architecture/

[2] OGC 05-060, URNs 05-010 change request – Specify data type URNs

