OGC® DOCUMENT: 14-065R2

**External identifier of this OGC® document:** http://www.opengis.net/doc/IS/wps/2.0.2

OGC®
Making location count.

# OGC® WPS 2.0.2 INTERFACE STANDARD CORRIGENDUM 2

## STANDARD
## Implementation

**APPROVED**

**Version:** 2.0.2
**Submission Date:** 2014-11-25
**Approval Date:** 2015-01-27
**Publication Date:** 2015-03-05
**Editor:** Matthias Mueller, Benjamin Pross

OGC®
Making location count.

### License Agreement

Use of this document is subject to the license agreement at https://www.ogc.org/license

Suggested additions, changes and comments on this document are welcome and encouraged. Such suggestions may be submitted using the online change request form on OGC web site: http://ogc.standardstracker.org/

### Copyright notice



### Note

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

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

# CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# LIST OF RECOMMENDATIONS

# I ABSTRACT

In many cases geospatial or location data, including data from sensors, must be processed before the information can be used effectively. The OGC Web Processing Service (WPS) Interface Standard provides a standard interface that simplifies the task of making simple or complex computational processing services accessible via web services. Such services include well-known processes found in GIS software as well as specialized processes for spatio-temporal modeling and simulation. While the OGC WPS standard was designed with spatial processing in mind, it can also be used to readily insert non-spatial processing tasks into a web services environment.

The WPS standard provides a robust, interoperable, and versatile protocol for process execution on web services. It supports both immediate processing for computational tasks that take little time and asynchronous processing for more complex and time consuming tasks. Moreover, the WPS standard defines a general process model that is designed to provide an interoperable description of processing functions. It is intended to support process cataloguing and discovery in a distributed environment.

# II KEYWORDS

The following are keywords to be used by search engines and document catalogues.

TBD, geoprocessing, ogcdoc, OGC document, processes, processing, WPS

# III PREFACE

This standard is a continuation of WPS 1.0, a standard for web-based processing of geospatial data. It incorporates a range of change requests that have been submitted since the release of WPS 1.0 and further follows the OGC standard for modular specifications [OGC 08-131r3].

In contrast to the prior version, WPS 2.0 provides a core conceptual model that may be used to specify a WPS in different architectures such as REST or SOAP.

The WPS process model has been encapsulated into separate requirements and conformance classes, so it may be used independently from WPS servers in process catalogs and metadata records. The expressive power of process descriptions has been enhanced by permitting structured (or nested) inputs and outputs. The concept of process profiles has been clarified and extended to support process descriptions at different levels of abstraction.

Conversely, the process model itself has been largely decoupled from the WPS protocol, allowing the use of other domain-specific descriptions of processes, e.g. those defined in SensorML, and to execute them on a WPS server.

This specification also provides a Basic WPS conformance class that comprises the synchronous and asynchronous execution protocol, the WPS process model, and implements HTTP/POST +XML and HTTP/GET+KVP encodings.

Future work will target the definition of process interfaces for common processes based on the process model conformance class. Such profiles will encourage the development of well-defined, reliable, interoperable and exchangeable process implementations.

If OGC baseline and related specifications should further progress towards REST-oriented interfaces, the development of a REST-oriented WPS interface standard should be considered.

# IV SECURITY CONSIDERATIONS

No security considerations have been made for this document.

# V SUBMITTING ORGANIZATIONS

The following organizations submitted this Document to the Open Geospatial Consortium (OGC):

- TU Dresden
- 52°North
- Intergraph Corporation
- Image Matters LLC
- CRP Henri Tudor
- Airbus Defence and Space

# VI SUBMITTERS

| NAME | REPRESENTING | OGC MEMBER |
|---|---|---|
| Matthias Müller | TU Dresden | Yes |
| Benjamin Pross | 52°North | Yes |
| Stan Tillman | Intergraph Corporation | Yes |
| Jeff Yutzler | Image Matters LLC | Yes |
| Luís de Sousa | CRP Henri Tudor | Yes |
| Arnaud Cauchy | Airbus Defence & Space | Yes |

1

# SCOPE

# 1 SCOPE

This document specifies the interface to a general purpose Web Processing Service (WPS). A WPS is a web service that enables the execution of computing processes and the retrieval of metadata describing their purpose and functionality. Typically, these processes combine raster, vector, and/or coverage data with well-defined algorithms to produce new raster, vector, and/or coverage information.

The WPS protocol supports both synchronous and asynchronous execution of processes. Synchronous execution may be used in simple and quick computation scenarios, where the data processing takes little to almost no time. Asynchronous processing is particularly well suited for complex computation scenarios which may take significant time.

The specification uses a core and extensions model to organize its features:

1. A core conceptual model, defining basic requirements for a web based processing service,
2. A process model to support the description and discovery of processes on the web,
3. A basic data model that supports arbitrary (standard or non-standard) data formats for inputs and outputs,
4. A WPS service model and encoding based on OGC baseline standards, and
5. A Dismiss extension to allow clients to terminate asynchronous processing jobs.

2

# CONFORMANCE

# 2 CONFORMANCE

Conformance with this standard shall be checked using all the relevant tests specified in Annex A (normative) of this document. The framework, concepts, and methodology for testing, and the criteria to be achieved to claim conformance are specified in the OGC Compliance Testing Policies and Procedures and the OGC Compliance Testing web site[1].

**Table 1** — Conformance classes related to WPS 2.0

| CONFORMANCE CLASS | DESCRIPTION | CLAUSE |
|---|---|---|
| http://www.opengis.net/spec/WPS/2.0/conf/service/profile/**basic-wps** | Basic WPS service profile | Annex A.1 |
| http://www.opengis.net/spec/WPS/2.0/conf/service/**synchronous-wps** | Synchronous WPS | Annex A.2 |
| http://www.opengis.net/spec/WPS/2.0/conf/service/**asynchronous-wps** | Asynchronous WPS | Annex A.3 |
| http://www.opengis.net/spec/WPS/2.0/conf/**process-model-encoding** | WPS process model encoding | Annex A.4 |
| http://www.opengis.net/spec/WPS/2.0/conf/service/**dismiss-extension** | Dismiss extension | Annex A.6 |

[1] www.opengeospatial.org/cite

3

# NORMATIVE REFERENCES

# 3 NORMATIVE REFERENCES

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.


Arliss Whiteside Jim Greenwood: OGC 06-121r9, *OGC Web Service Common Implementation Specification*. Open Geospatial Consortium (2010).

Policy SWG: OGC 08-131r3, *The Specification Model — Standard for Modular specifications*. Open Geospatial Consortium (2009).

A. Phillips, M. Davis: IETF RFC 4646, *Tags for Identifying Languages*. RFC Publisher (2006). https://www.rfc-editor.org/info/rfc4646.

T. Berners-Lee, R. Fielding, L. Masinter: IETF RFC 3986, *Uniform Resource Identifier (URI): Generic Syntax*. RFC Publisher (2005). https://www.rfc-editor.org/info/rfc3986.

ISO: ISO 8601:2004, *Data elements and interchange formats — Information interchange — Representation of dates and times*. International Organization for Standardization, Geneva (2004). https://www.iso.org/standard/40874.html.

XML Schema Part 2: Datatypes Second Edition, W3C Recommendation 28 October 2004.

4

# TERMS AND DEFINITIONS

# 4 TERMS AND DEFINITIONS

This document uses the terms defined in OGC Policy Directive 49, 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 document and OGC documents do not use the equivalent phrases in the ISO/IEC Directives, Part 2.

This document also uses terms defined in the OGC Standard for Modular specifications (OGC 08-131r3), also known as the ‘ModSpec’. The definitions of terms such as standard, specification, requirement, and conformance test are provided in the ModSpec.

For the purposes of this document, the following additional terms and definitions apply.

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

For the purposes of this document, the following additional terms and definitions apply.

## 4.1. Process:

A **process** p is a function that for each input returns a corresponding output

$\mu : X \rightarrow Y$

where $X$ denotes the domain of arguments x and $Y$ denotes the co-domain of values y. Within this specification, process arguments are referred to as process inputs and result values are referred to as process outputs. Processes that have no process inputs represent value generators that deliver constant or random process outputs.

## 4.2. Process description:

A process description is an information model that specifies the interface of a process. A process description is used for a machine-readable description of the process itself but also provides some basic information about the process inputs and outputs.

## 4.3. Process input:

Process inputs are the arguments of a process and refer to data provided to a process. Each process input is an identifiable item.

## 4.4. Process output:

Process outputs are the results of a process and refer to data returned by a process. Each process output is an identifiable item.

## 4.5. Process profile:

A process profile is a description of a process on an interface level. Process profiles may have different levels of abstraction and cover several aspects. On a generic level, a process profile may only refer to the provided functionality of a process, i.e. by giving a verbal or formal definition how the outputs are derived from the inputs. On a concrete level a process profile may completely define inputs and outputs including data type definitions and formats.

## 4.6. WPS Server:

A WPS Server is a web server that provides access to simple or complex computational processing services.

## 4.7. Process offering:

A process offering is an identifiable process that may be executed on a particular service instance. A process offering contains a process description as well as service-specific information about the supported execution protocols (e.g. synchronous and asynchronous execution).

## 4.8. Process execution:

The execution of a process is an action that calculates the outputs of a given process for a given set of data inputs.

## 4.9. Job:

The (processing) job is a server-side object created by a processing service for a particular process execution. A job may be latent in the case of synchronous execution or explicit in the case of asynchronous execution. Since the client has only oblique access to a processing job, a Job ID is used to monitor and control a job.

## 4.10. Service profiles for WPS:

A service profile for WPS is a conformance class that defines the general capabilities of a WPS server, by (1) specifying the supported service operations, (2) the process model, (3) the supported process execution modes, (4) the supported operation binding(s).

6

# CONVENTIONS

# 6 CONVENTIONS

This section provides details and examples for any conventions used in the document. Examples of conventions are symbols, abbreviations, use of XML schema, or special notes regarding how to read the document.

## 6.1. Abbreviated terms

| | |
|---|---|
| **GML** | Geography Markup Language |
| **GRS** | Coordinate Reference System |
| **HTTP** | Hypertext Transfer Protocol |
| **ISO** | International Organization for Standardization |
| **KVP** | Keyword Value Pair |
| **MIME** | Multipurpose Internet Mail Extensions |
| **OGC** | Open Geospatial Consortium |
| **UML** | Unified Modeling Language |
| **URI** | Universal Resource Identifier |
| **URL** | Uniform Resource Locator |
| **WPS** | Web Processing Service |
| **XML** | Extensible Markup Language |

## 6.2. Use of the Term “Process”

The term process is one of the most used terms both in the information and geosciences domain. If not stated otherwise, this specification uses the term process as an umbrella term for any algorithm, calculation or model that either generates new data or transforms some input data into output data as defined in Clause 4.1.

## 6.3. UML Notation

Unified Modeling Language (UML) static structure diagrams appearing in this specification are used as described in section 5.2 of OGC06-121r9. Further, the following conventions hold:

- UML elements having a package name of “OWS Common” are those defined in the UML model of OWS Common [OGC 06-121r9].
- UML data type Any is used here as an equivalence to XML’s xsd:any.
- UML elements not qualified with a package name are those defined in this standard.

The UML model data dictionary is specified herein in a series of tables. The contents of the columns in these tables are described in section 5.5 of [OGC 06-121r9]. The contents of these data dictionary tables are normative, including any table footnotes.

## 6.4. Namespace Conventions

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

| PREFIX | NAMESPACE URI | DESCRIPTION |
|---|---|---|
| ows | http://www.opengis.net/ows/2.0 | OWS Common 2.0 XML Schema |
| xlink | http://www.w3.org/1999/xlink | Definitions for XLINK |
| xml | http://www.w3.org/XML/1998/namespace | XML (required for xml:lang) |
| xs | http://www.w3.org/2001/XMLSchema | XML Schema |

7

# WPS CONCEPTUAL MODEL

# 7 WPS CONCEPTUAL MODEL

The WPS service model defines basic properties of any WPS server. A WPS server is a web service that provides access to pre-defined processes and provides job control operations to instantiate, control and monitor processing jobs (Figure 1).

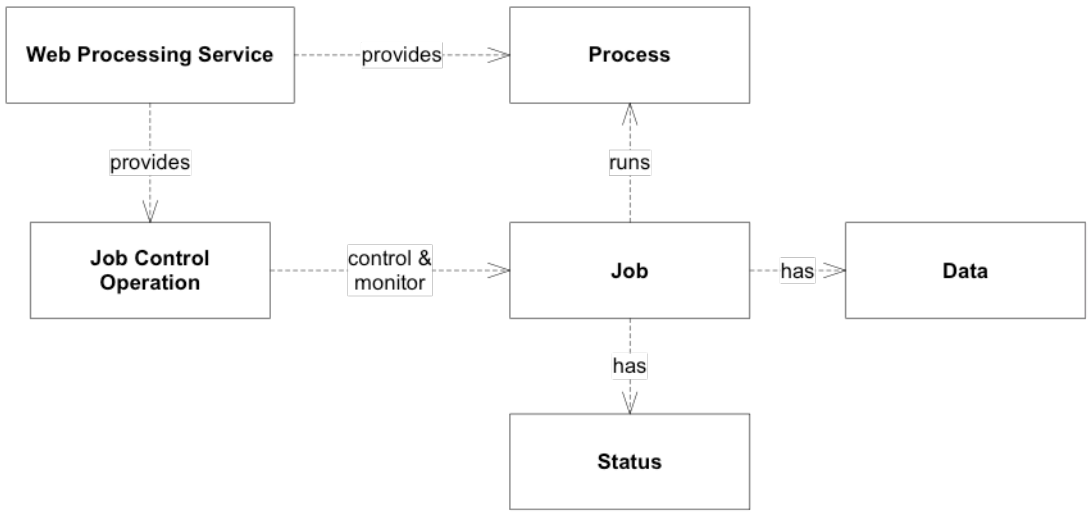


**Figure 1** — Artifacts of the WPS service model

| REQUIREMENTS CLASS 1 | |
|---|---|
| OBLIGATION | requirement |
| TARGET TYPE | Derived information model, encoding, and software implementation |
| NORMATIVE STATEMENTS | Requirement 1-1<br>Requirement 1-2<br>Recommendation 1-1<br>Requirement 1-3<br>Requirement 1-4<br>Requirement 1-5<br>Requirement 1-6 |

## 7.1. Service Discovery

Any WPS server shall be self-contained, i.e. provide an initial endpoint that can be used by a WPS client to determine the server's capabilities.

| REQUIREMENTS CLASS 2 | |
|---|---|
| **OBLIGATION** | requirement |
| **TARGET TYPE** | Derived information model, encoding, and software implementation |
| **NORMATIVE STATEMENTS** | Requirement 2-1<br>Requirement 2-2<br>Requirement 2-3 |

## 7.2. Service Capabilities

The basic capabilities of any WPS server fall into two categories: The first category comprises capabilities for process discovery and retrieval of process descriptions. The second category comprises capabilities to manage and monitor processing jobs.

Since the processes provided by a WPS server may have different degrees of complexity, the server shall indicate the allowed job control capabilities mode per process offering.

Further service capabilities, i.e. for secure communication and user authentication may be provided with the service but are neither covered nor restricted by this specification as long as they do not alter or change the semantics of other job control capabilities.

| REQUIREMENTS CLASS 3 | |
|---|---|
| **OBLIGATION** | requirement |
| **TARGET TYPE** | Derived information model, encoding, and software implementation |
| **PREREQUISITE** | http://www.opengis.net/spec/WPS/2.0/req/conceptual-model/process |
| **NORMATIVE STATEMENTS** | Requirement 3-1<br>Requirement 3-2<br>Requirement 3-3<br>Requirement 3-4 |

## 7.3. Abstract Process Model

The abstract process model specifies generic requirements for process offerings that can be used in conjunction with WPS. Processes, as well as their inputs and outputs, are elements

with identity. A process input may have arbitrarily defined value cardinality, i.e. for a given input, multiple datasets may be submitted for execution. A process output always has a value cardinality of one. Process inputs and outputs may also be nested. Identifiers for inputs and outputs shall be unique. For nested children it is sufficient to have a unique nesting path. Any input and output that does not have child elements shall have a defined data type so that a client is aware of the valid data formats for process execution.

The abstract process model provides many degrees of freedom for process descriptions. However, it does not enforce additional complexity for very simple processes.

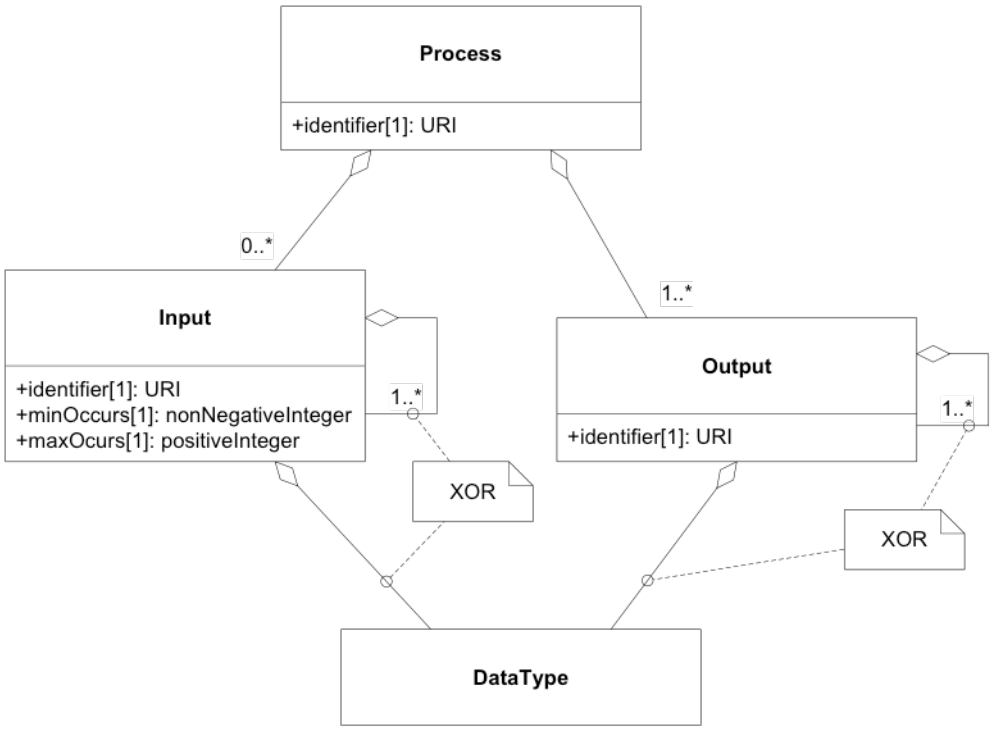


**Figure 2** — Abstract process model UML class diagram

| REQUIREMENTS CLASS 4 | |
|---|---|
| **OBLIGATION** | requirement |
| **TARGET TYPE** | Derived information model, encoding, and software implementation |
| **NORMATIVE STATEMENTS** | Requirement 4-1<br>Requirement 4-2<br>Requirement 4-3<br>Requirement 4-4<br>Requirement 4-5<br>Requirement 4-6<br>Requirement 4-7<br>Requirement 4-8<br>Requirement 4-9 |

| REQUIREMENTS CLASS 4 | |
|---|---|
| | Requirement 4-10<br>Requirement 4-11<br>Requirement 4-12<br>Requirement 4-13<br>Requirement 4-14<br>Requirement 4-15<br>Requirement 4-16 |

**Table 2** — Data encoding properties

| STATUS | DEFINITION |
|---|---|
| mimetype | Media type of the data. |
| encoding | Encoding procedure or character set used (e.g. raw, base64, or UTF-8). |
| schema | Identification of the data schema. |

## 7.4. Job Control

The execution capability, that permits WPS clients to instantiate and run processing jobs, is the most prominent job control capability. Furthermore, the ability to dismiss or delete a job is useful for long-running processes in order to release server resources.

| REQUIREMENTS CLASS 5 | |
|---|---|
| **OBLIGATION** | requirement |
| **TARGET TYPE** | Derived information model, encoding, and software implementation |
| **PREREQUISITE** | http://www.opengis.net/spec/WPS/2.0/req/conceptual-model/capabilities |
| **NORMATIVE STATEMENTS** | Requirement 5-1<br>Requirement 5-2<br>Requirement 5-3<br>Requirement 5-4 |

## 7.5. Process Execution

Process executions on a WPS server may be run either synchronously (Figure 3) or asynchronously (see Figure 4). Synchronous execution is a suitable approach for jobs that take a relatively short time[2] to complete. Asynchronous execution is preferable for jobs that may take a long time to complete.

In the synchronous case, a WPS client submits an execute request to the WPS server and keeps listening for a response until the processing job has completed and the processing result has been returned. This requires a persistent connection between client and server.

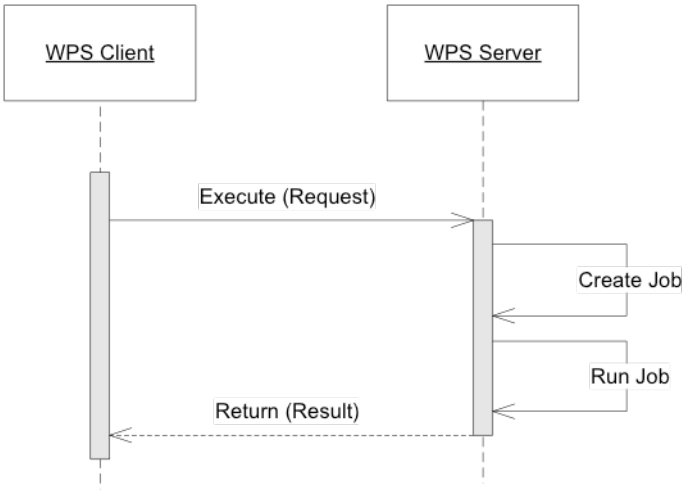


**Figure 3** — Synchronous process execution UML sequence diagram

In the asynchronous case, the client sends an execute request to the WPS server and immediately receives a status information response. This information confirms that the request was received and accepted by the server and that a processing job has been created and will be run in the future. The status information response also contains a processing job identifier that is used by the client when checking to see if execution has completed. (The client may also manage the processing job using available steering capabilities.) In addition, the status information response contains the result location, i.e. the URL where the processing result can be found after the processing job has completed.

[2]Short and long are relative terms and requires considerations on both client and server side and the application context. This specification does not give any guidelines regarding what short or long means. There may also be an architectural consideration that suggests either synchronous or asynchronous communication.

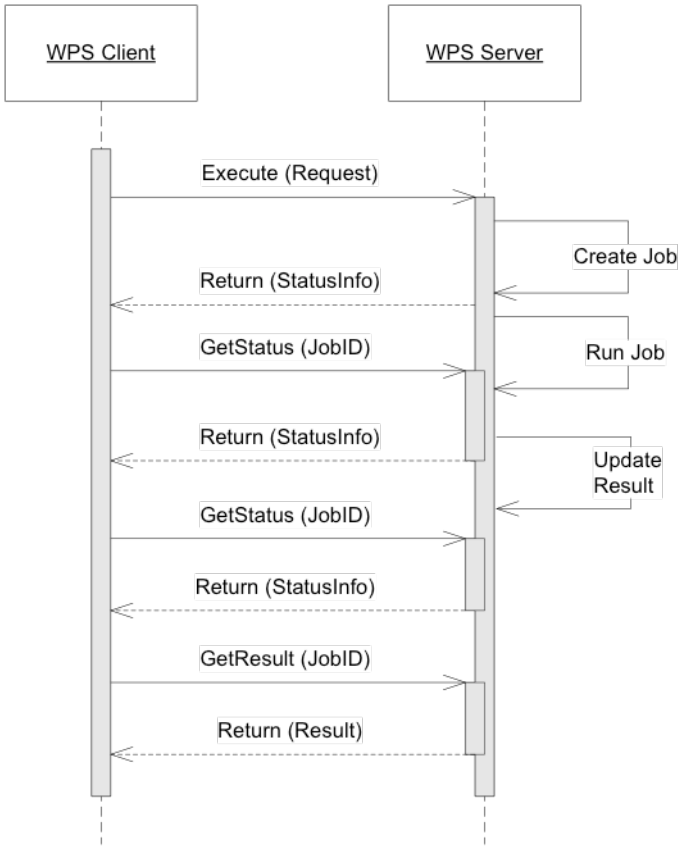


**Figure 4** — Asynchronous process execution UML sequence diagram

| REQUIREMENTS CLASS 6 | |
|---|---|
| **OBLIGATION** | requirement |
| **TARGET TYPE** | Derived information model, encoding, and software implementation |
| **PREREQUISITE** | http://www.opengis.net/spec/WPS/2.0/req/conceptual-model/job-control |
| **NORMATIVE STATEMENTS** | Requirement 6-1<br>Requirement 6-2<br>Requirement 6-3<br>Requirement 6-4<br>Requirement 6-5<br>Requirement 6-6<br>Requirement 6-7<br>Requirement 6-8<br>Requirement 6-9<br>Requirement 6-10<br>Requirement 6-11<br>Requirement 6-12<br>Requirement 6-13 |

Requirement 6-14

## 7.6. Data Transmission by Value and by Reference

Data exchange between WPS clients and servers requires an agreement on the general data exchange patterns and suitable communication protocols. This specification defines two general data transmission modes for data exchange between WPS client and server.

Clients may send input data to or receive output data from a process in two distinct ways: (1) by reference, and (2) by value (see Figure 5). For brevity, this figure only shows the transmission patterns in the pure form, i.e. the same pattern is used for all inputs and outputs. However, mixed patterns are possible. Typically, small or atomic data such as integers, doubles or short strings are submitted by value. Large data inputs (outputs) are usually supplied by reference.


**Figure 5** — Execute call and response "by value" and "by reference" UML sequence diagram.

This specification makes no assumptions about the encoding of the transmitted data. Due to the variety of data formats and supplementary encoding procedures, the receiver may not be able to automatically detect the data format. The set of encoding attributes that may be used by the receiver to decode inline or directly referenced data is defined by the data format description (see Table 2).

| REQUIREMENTS CLASS 7 | |
|---|---|
| **OBLIGATION** | requirement |
| **TARGET TYPE** | Derived information model, encoding, and software implementation |
| **PREREQUISITE** | http://www.opengis.net/spec/WPS/2.0/req/conceptual-model/process-execution |
| **NORMATIVE STATEMENTS** | Requirement 7-1<br>Requirement 7-2<br>Requirement 7-3<br>Requirement 7-4<br>Requirement 7-5<br>Requirement 7-6<br>Requirement 7-7 |

## 7.7. Job Monitoring

By definition, a processing job is a server-side object created by a processing service in response for a particular process execution. It is comprised of a process definition (i.e. one of the process offerings defined in the WPS server's capabilities), the input data provided or specified by the WPS client, and the outputs that are eventually delivered when the job has been completed.

Since the processing job is a server-side object, a WPS client has no means to inspect the status of a job on its own. Therefore, the server should provide a unique identifier for each job. For privacy, it is recommended to keep this identifier confidential between client and server.

For jobs running in asynchronous mode, the WPS server shall also provide monitoring information and it may also contain estimates on the completion time or additional elements related to status polling.

If a client finds a polling time in the status information, it shall respect it and behave accordingly. The service may rely on the client polling roughly around this time to obtain updated status information.

If a client finds an expiry date in the status information, it shall respect it and behave accordingly, i.e. ensure that the execution result is evaluated on time and the outputs are retrieved before the job is removed from the server. This requirement allows for robust WPS implementations and the timely re-allocation of server resources.

This section also defines a basic status set to communicate the status of a server-side job to the client. Extensions of this specification may introduce additional states for fine-grained monitoring or domain-specific purposes.

| REQUIREMENTS CLASS 8 | |
|---|---|
| OBLIGATION | requirement |
| TARGET TYPE | Derived information model, encoding, and software implementation |
| PREREQUISITES | http://www.opengis.net/spec/WPS/2.0/req/conceptual-model/job-control<br>http://www.opengis.net/spec/WPS/2.0/req/conceptual-model/process-execution |
| NORMATIVE STATEMENTS | Requirement 8-1<br>Requirement 8-2<br>Requirement 8-3<br>Requirement 8-4<br>Requirement 8-5<br>Requirement 8-6 |

**Table 3** — Basic status set for jobs

| STATUS | DEFINITION |
|---|---|
| Succeeded | The job has finished with no errors. |
| Failed | The job has finished with errors. |
| Accepted | The job is queued for execution.[a] |
| Running | The job is running.[a] |

[a] States are only defined for asynchronous execution.

8

# WPS NATIVE PROCESS MODEL

# 8 WPS NATIVE PROCESS MODEL

The WPS native process model is a realization of the abstract process model defined in Clause 7.3. The native process model breaks down into the following components:

1. A general structure for process interface descriptions (Clause 8.1, Clause 8.4) that support process discovery and cataloging
2. A set of data types to submit and receive data during process execution (Clause 8.2, Clause 8.3)
3. A framework for defining process profiles to improve interoperability between different process implementations (Clause 8.5).

The native process model is encapsulated in a separate conformance class (Annex A.4) and can be used independently from the rest of the WPS specification. It is designed to provide an interoperable description of processing functions, regardless whether they are offered by a web service for remote invocation, or reside in a Desktop GIS for local use. By abstracting from a concrete implementation this process model may be used to create metadata records in process catalogs or help comparing and aligning the processing functions in different systems.

This section describes the information model of requirements. The corresponding XML and plain text encodings are specified in Clause 9.

| REQUIREMENTS CLASS 9 | |
|---|---|
| **OBLIGATION** | requirement |
| **TARGET TYPE** | Derived encoding and software implementation |
| **PREREQUISITE** | http://www.opengis.net/spec/WPS/2.0/req/conceptual-model/process |
| **NORMATIVE STATEMENTS** | Requirement 9-1<br>Requirement 9-2<br>Requirement 9-3<br>Requirement 9-4<br>Requirement 9-5 |

## 8.1. Common Description Type

Descriptive elements of processes, inputs and outputs are derived from the BasicIdentificationType provided by OWS Common (Figure 6). Other descriptive information

shall be recorded in the Metadata element in the form of simple links with an appropriate role identifier.

| REQUIREMENTS CLASS 10 | |
|---|---|
| OBLIGATION | requirement |
| TARGET TYPE | Derived encoding and software implementation |
| PREREQUISITES | http://www.opengis.net/spec/WPS/2.0/req/conceptual-model/process<br>OWS Common 2.0 — BasicDescriptionType |
| NORMATIVE STATEMENTS | Requirement 10-1<br>Requirement 10-2<br>Requirement 10-3<br>Requirement 10-4<br>Requirement 10-5 |

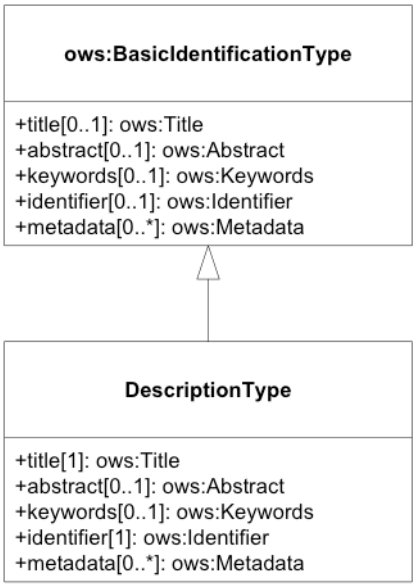


**Figure 6** — DescriptionType for processes, process inputs and process outputs UML class diagram

**Table 4** — Properties of the DescriptionType structure

| NAMES | DEFINITION | DATA TYPE AND VALUES | MULTIPLICITY AND USE |
|---|---|---|---|
| Title | Title of the process, input, and output. Normally available for display to a human. | ows:Title | One (mandatory) |
| Abstract | Brief narrative description of a process, input, and output. Normally available for display to a human. | ows:Abstract | Zero or one (optional) Include when available and useful. |

| NAMES | DEFINITION | DATA TYPE AND VALUES | MULTIPLICITY AND USE |
|---|---|---|---|
| Keywords | Keywords that characterize a process, its inputs, and outputs. | ows:Keywords | Zero or more (optional) Include when available and useful. |
| Identifier | Unambiguous identifier of a process, input, and output. | ows:Identifier Value is a URI or HTTP-URI[a] | One (mandatory) |
| Metadata | Reference to additional metadata about this item. | ows:Metadata Allowed values are specified in Table 5. | Zero or more (optional) |

[a] Additional content such as separate code space and version attributes in the Identifier element are not allowed.

**Table 5** — Properties of the Metadata structure

| NAMES | DEFINITION | DATA TYPE AND VALUES | MULTIPLICITY AND USE |
|---|---|---|---|
| Title | Title of the documentation. Normally available for display to a human. | Character String | One (mandatory) |
| Link type | Type of the xlink, fixed to simple. | Character String, fixed to "simple". | One (mandatory) |
| Role | Role identifier, indicating the role of the linked document. | HTTP-URI | One (mandatory) |
| href | Reference to a documentation site for a process, input, or output. | HTTP-URI | One (mandatory) |

## 8.2. Data Description Structure

The DataDescription structure contains basic properties for defining data inputs and outputs, including mimetype, encoding and schema. These properties specify supported formats for input and output data of computing processes. Any input or output item may support multiple formats, one of which is the default format. Processes may require that an input or output data set does not exceed a certain data volume.

| REQUIREMENTS CLASS 11 | |
|---|---|
| OBLIGATION | requirement |
| TARGET TYPE | Derived information model, encoding, and software implementation |

| REQUIREMENTS CLASS 11 | |
|---|---|
| PREREQUISITE | http://www.opengis.net/spec/WPS/2.0/req/conceptual-model/process |
| NORMATIVE STATEMENTS | Requirement 11-1<br>Requirement 11-2 |

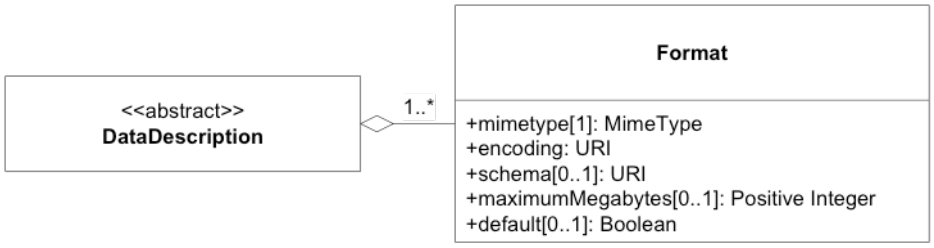


**Figure 7** — DataDescription and supported formats UML class diagram

**Table 6** — Format properties

| NAMES | DEFINITION | DATA TYPE AND VALUES | MULTIPLICITY AND USE |
|---|---|---|---|
| mimetype | Media type of the data. | Character String | One (mandatory) |
| encoding | Encoding procedure or character set of the data (e.g. raw or base64) | Character String, fixed to "simple". | One (mandatory) |
| schema | Identification of the data schema. | HTTP-URI | One (mandatory) |
| maximum Megabytes | The maximum size of the input data, in megabytes. | Integer | Zero or one (optional) |
| default | Indicates that this format is the default format.[a] | Boolean | Zero or one (conditional)[a,b] |

[a] Defaults to FALSE if omitted.

[b] One of the formats included in the DataDescription structure shall have the attribute "default" set to "true".

## 8.3. Data Types

This specification defines three common data types for process input and output data (Figure 8):

1. ComplexData, such as GML or geo-referenced imagery. This type is kept generic regarding the content and may be extended to provide more detailed domain-specific information.

2. LiteralData, defined as a value with an optional unit.

3. BoundingBoxData, defined as a minimum bounding rectangle in geographic coordinates.

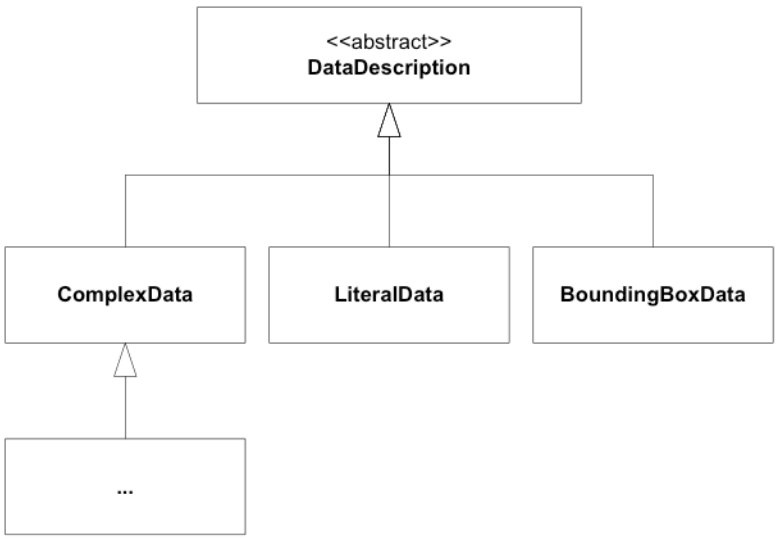


**Figure 8** — I/O data types overview

| REQUIREMENTS CLASS 12 | |
|---|---|
| OBLIGATION | requirement |
| TARGET TYPE | Derived encoding and software implementation |
| PREREQUISITES | http://www.opengis.net/spec/WPS/2.0/req/conceptual-model/process<br>http://www.opengis.net/spec/WPS/2.0/req/native-process/model/io-format<br>OWS Common 2.0 |
| NORMATIVE STATEMENTS | Requirement 12-1<br>Requirement 12-2<br>Requirement 12-3 |

### 8.3.1. Complex data

The ComplexData type does not describe the particular structure for value encoding. Instead, the passed values must comply with the given format and the extended information, if provided.

| REQUIREMENTS CLASS 13 | |
|---|---|
| OBLIGATION | requirement |

| REQUIREMENTS CLASS 13 | |
|---|---|
| TARGET TYPE | Derived encoding and software implementation |
| PREREQUISITES | http://www.opengis.net/spec/WPS/2.0/req/conceptual-model/process<br>OWS Common 2.0 |
| NORMATIVE STATEMENTS | Requirement 13-1<br>Requirement 13-2 |

### 8.3.1.1. ComplexData Description

The ComplexData structure is a direct realization of the abstract DataDescription element (Figure 7). It relies on the encoding attributes defined in Table 6 for a basic description of input and output data. In cases where these attributes do not sufficiently capture the structure and content of the required or produced data, the ComplexData element provides the ability to include any other descriptive elements next to the format specification. This hook may be used by process providers and consumers to communicate essential information and further constraints for input and output data items.

| REQUIREMENTS CLASS 14 | |
|---|---|
| OBLIGATION | requirement |
| TARGET TYPE | Derived encoding and software implementation |
| PREREQUISITES | http://www.opengis.net/spec/WPS/2.0/req/conceptual-model<br>http://www.opengis.net/spec/WPS/2.0/req/native-process/model/io-format<br>OWS Common 2.0 |
| NORMATIVE STATEMENT | Requirement 14-1 |

**Table 7** — ComplexData description properties

| NAMES | DEFINITION | DATA TYPE AND VALUES | MULTIPLICITY AND USE |
|---|---|---|---|
| Format | Identifies a valid format for an input or output. | Format properties, see Table 6. | One or more (mandatory) |
| Any | Placeholder for schema extensions to WPS complex data. | Any type. | Zero or more (optional) |

#### 8.3.1.2. ComplexData Values

A Complex data value is directly passed to (or returned by) a process. The generic nature of Complex data does not permit a particular structure for value encoding. Instead, this structure is defined by the ComplexData description and the passed values must comply with the given format and the extended information, if provided.

| REQUIREMENTS CLASS 15 | |
|---|---|
| **OBLIGATION** | requirement |
| **TARGET TYPE** | Derived encoding and software implementation |
| **PREREQUISITES** | http://www.opengis.net/spec/WPS/2.0/req/conceptual-model<br>http://www.opengis.net/spec/WPS/2.0/req/native-process/model/datatypes/complex-data/description |
| **NORMATIVE STATEMENT** | Requirement 15-1 |

### 8.3.2. Literal Data

The LiteralData type encodes atomic data such as scalars, linear units, or well-known names. Domains for LiteralData are a combination of data types (e.g. Double, Integer, String), a given value range, and an associated unit (e.g. meters, degrees Celsius).

| REQUIREMENTS CLASS 16 | |
|---|---|
| **OBLIGATION** | requirement |
| **TARGET TYPE** | Derived encoding and software implementation |
| **PREREQUISITES** | http://www.opengis.net/spec/WPS/2.0/req/conceptual-model/process<br>OWS Common 2.0 |
| **NORMATIVE STATEMENTS** | Requirement 16-1<br>Requirement 16-2 |

#### 8.3.2.1. LiteralData Description

The LiteralData description structure inherits essential elements from ows:DomainType allowing it to specify value domains including Units of Measure and Default Values. It restricts ows:DomainType by forbidding:

1. "NoValues" for a particular domain
2. the ability to specify further metadata on the values (since this information is already present at the level of input and output definitions in the DescriptionType element).

LiteralData data types should use the well-known types from XML Schema by their URI definition[3]. Table 11 lists the recommended URIs for the most common literal data types.

| REQUIREMENTS CLASS 17 | |
|---|---|
| **OBLIGATION** | requirement |
| **TARGET TYPE** | Derived encoding and software implementation |
| **PREREQUISITES** | http://www.opengis.net/spec/WPS/2.0/req/conceptual-model<br>http://www.opengis.net/spec/WPS/2.0/req/native-process/model/io-format<br>OWS Common 2.0 |
| **NORMATIVE STATEMENT** | Requirement 17-1 |

[3]see http://www.w3.org/TR/xmlschema-2/#built-in-datatypes

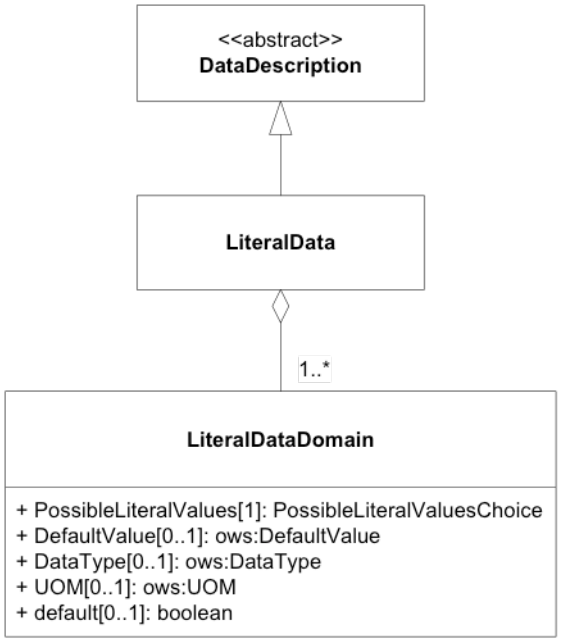


**Figure 9** — LiteralData UML class diagram

**Table 8** — The LiteralData structure

| NAMES | DEFINITION | DATA TYPE AND VALUES | MULTIPLICITY AND USE |
|---|---|---|---|
| Format | Identifies a valid format for an input or output. | Format properties, see Table 6. | One or more (mandatory) |
| LiteralData Domain | The valid domain for literal data | LiteralDataDomain type | One or more (mandatory) |

**Table 9** — Parts of the LiteralDataDomain structure

| NAMES | DEFINITION | DATA TYPE AND VALUES | MULTIPLICITY AND USE |
|---|---|---|---|
| PossibleLiteral Values | Identifies a valid format for an input or output. | PossibleLiteralValuesChoice, see Table 10. | One (mandatory) |
| DataType | Reference to the data type of this set of values | ows:DataType. The use of well-known data type URNs is highly recommended; see Table 11. | One (mandatory) |
| UOM | Indicates that this quantity has units and provides the unit of measurement. | ows:ValuesUnit | Zero or one (optional) Include when values have units or reference system. |

| NAMES | DEFINITION | DATA TYPE AND VALUES | MULTIPLICITY AND USE |
|---|---|---|---|
| DefaultValue | Default value for this quantity. | ows:DefaultValue | Zero or one (optional) Include if there is a default.[a] |
| default | Indicates that this is the default/native domain. | Boolean, defaults to false. | Zero or one (conditional)[b,c] |

[a] For outputs, the DefaultValue has no meaning and shall thus be omitted.

[b] Defaults to FALSE if omitted.

[c] One of the formats included in the LiteralData structure shall have the attribute “default” set to “true”.

**Table 10** — Parts of the PossibleLiteralValuesChoice structure

| NAMES | DEFINITION | DATA TYPE AND VALUES | MULTIPLICITY AND USE |
|---|---|---|---|
| AllowedValues | List of all valid values and/or ranges of values for this quantity. | ows:AllowedValues | Zero or one (conditional)[a] |
| AnyValue | Specifies that any value is allowed for this quantity. | ows:AnyValue | Zero or one (conditional)[a] |
| Values Reference | Reference to list of all valid values and/or ranges of values for this quantity. | ows:ValuesReference | Zero or one (conditional)[a] |

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

**Table 11** — Recommended data type URIs for literal data

| DATA TYPE | URI |
|---|---|
| String | http://www.w3.org/2001/XMLSchema#string |
| Integer | http://www.w3.org/2001/XMLSchema#integer |
| Decimal | http://www.w3.org/2001/XMLSchema#decimal |
| Boolean | http://www.w3.org/2001/XMLSchema#boolean |
| Double | http://www.w3.org/2001/XMLSchema#double |
| Float | http://www.w3.org/2001/XMLSchema#float |

### 8.3.2.2. LiteralData Values

LiteralData values represent values that correspond to a particular domain defined in the LiteralData structure. Figure 10 shows the mapping from the LiteralValue structure to the corresponding elements in the LiteralDataDescriptionType.

| REQUIREMENTS CLASS 18 | |
|---|---|
| OBLIGATION | requirement |
| TARGET TYPE | Derived encoding and software implementation |
| PREREQUISITES | http://www.opengis.net/spec/WPS/2.0/req/conceptual-model<br>http://www.opengis.net/spec/WPS/2.0/req/native-process/model/datatypes/literal-data/description<br>OWS Common 2.0 |
| NORMATIVE STATEMENT | Requirement 18-1 |

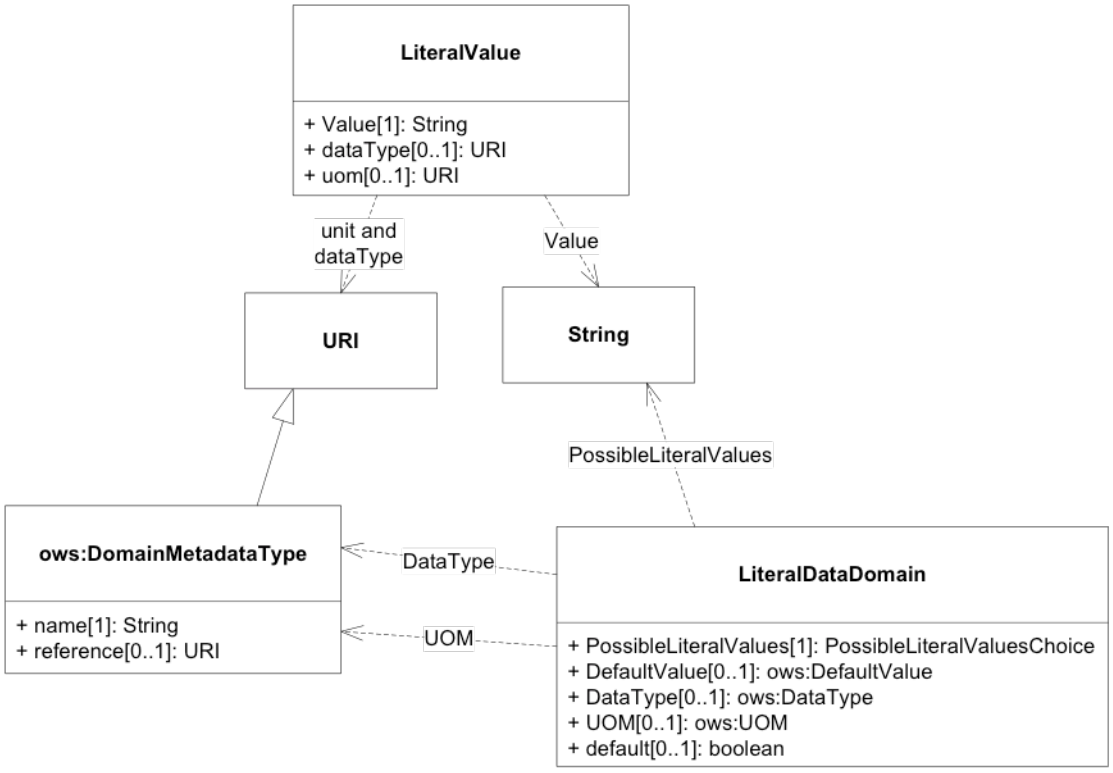


**Figure 10** — LiteralValue UML class diagram

**Table 12** — Parts of the LiteralValue structure

| NAMES | DEFINITION | DATA TYPE AND VALUES | MULTIPLICITY AND USE |
|---|---|---|---|
| Value | String representation of the actual value. | Character String | One (mandatory) |
| dataType | The data type of the Value. | URI | Zero or one (optional)[a] |
| uom | The unit of measurement of the value. | URI | Zero or one (optional)[a] |

[a] If not specified, the relevant defaults from the LiteralData description (see Clause 8.3.2.1) will be used.

## 8.3.3. BoundingBox Data

Bounding box data serves a variety of purposes in spatial data processing. Some simple applications are the definition of extents for a clipping operation or the definition of an analysis region. This specification inherits the bounding box specification from OWS Common.

| REQUIREMENTS CLASS 19 | |
|---|---|
| **OBLIGATION** | requirement |
| **TARGET TYPE** | Derived encoding and software implementation |
| **PREREQUISITES** | http://www.opengis.net/spec/WPS/2.0/req/conceptual-model/process<br>OWS Common 2.0 |
| **NORMATIVE STATEMENT** | Requirement 19-1 |

### 8.3.3.1. BoundingBox Description

The domain for bounding box data is described by a listing of supported CRSs.

| REQUIREMENTS CLASS 20 | |
|---|---|
| **OBLIGATION** | requirement |
| **TARGET TYPE** | Derived encoding and software implementation |
| **PREREQUISITES** | http://www.opengis.net/spec/WPS/2.0/req/conceptual-model<br>http://www.opengis.net/spec/WPS/2.0/req/native-process/model/io-format<br>OWS Common 2.0 |

| REQUIREMENTS CLASS 20 | |
|---|---|
| **NORMATIVE STATEMENT** | Requirement 20-1 |

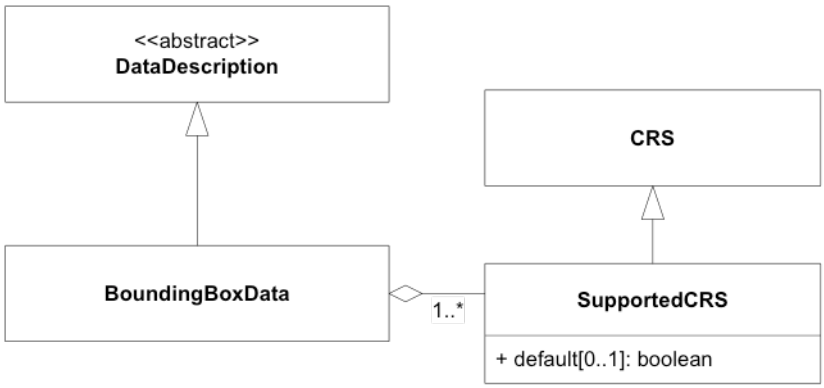


**Figure 11** — BoundingBoxData UML class diagram

**Table 13** — The BoundingBox structure

| NAMES | DEFINITION | DATA TYPE AND VALUES | MULTIPLICITY AND USE |
|---|---|---|---|
| Format | Identifies a valid format for an input or output. | Format properties, see Table 6. | One or more (mandatory) |
| Supported CRS | The supported CRS for BoundingBox data. | SupportedCRS type, see Table 14. | One or more (mandatory) |

**Table 14** — The SupportedCRS type structure

| NAMES | DEFINITION | DATA TYPE AND VALUES | MULTIPLICITY AND USE |
|---|---|---|---|
| CRS | Reference to a CRS definition. | URI | One or more (mandatory) |
| default | Indicates that this CRS is the default CRS. | Boolean, defaults to false. | Zero or one (conditional)[a,b] |

[a] Defaults to FALSE if omitted.

[b] One of the formats included in the BoundingBox structure shall have the attribute “default” set to “true”.

#### 8.3.3.2. BoundingBox Values

Values for bounding boxes are specified in the BoundingBox data type from OWS Common [OGC 06-121r9]. For consistency with the BoundingBoxData description, the specification of a CRS is mandatory.

| REQUIREMENTS CLASS 21 | |
|---|---|
| **OBLIGATION** | requirement |
| **TARGET TYPE** | Derived encoding and software implementation |
| **PREREQUISITES** | http://www.opengis.net/spec/WPS/2.0/req/conceptual-model<br>http://www.opengis.net/spec/WPS/2.0/req/native-process/model/datatypes/bounding-box-data/description<br>OWS Common 2.0 |
| **NORMATIVE STATEMENTS** | Requirement 21-1<br>Requirement 21-2 |

## 8.4. Process Description

This section defines information structures that describe a process. It includes elements that link to documentation resources on the behavior and mechanics of a process as well as descriptive elements about its inputs and outputs. The process description model realizes and extends the requirements defined in the abstract process model in Clause 7.3.

A process description is an extension of the DescriptionType (Figure 12). It shall be used to express identifier, title, and abstract and to link to associated metadata elements that provide additional or more detailed information about the process. An additional language attribute shall be used to indicate the language of human readable elements in the description of the process and its inputs and outputs.

The description structures for process inputs and outputs inherit common elements from the DescriptionType (Clause 8.1). These elements shall be used to express identifier, title, and abstract and to link to associated metadata elements that provide additional or more detailed information about the process inputs and outputs. The content of human readable elements in the description of inputs and outputs shall adhere to the language indicated in the process description.

Process inputs are arguments to a process. Process inputs have a cardinality in order to (1) pass multiple values with the same identifier to a process, or (2) declare process inputs as optional (cardinality “0”). Input elements may be simple (i.e. the input has no sub-inputs attached) or

aggregate (i.e. the input has one or more sub-input elements attached). A simple input includes a realization of the DataDescription element. An aggregate input contains one or more sub-inputs.

Outputs are the return values of a process. Outputs have a cardinality of one. Output elements may be simple (i.e. the output has no sub-outputs attached) or aggregate (i.e. the output has one or more sub-output elements attached). A simple output includes a realization of the DataDescription element. An aggregate output contains one or more sub-outputs.

| REQUIREMENTS CLASS 22 | |
|---|---|
| **OBLIGATION** | requirement |
| **TARGET TYPE** | Derived encoding and software implementation |
| **PREREQUISITES** | http://www.opengis.net/spec/WPS/2.0/req/conceptual-model/process<br>http://www.opengis.net/spec/WPS/2.0/req/native-process/model/description-type<br>IETF RFC 4646 |
| **NORMATIVE STATEMENTS** | Requirement 22-1<br>Requirement 22-2<br>Requirement 22-3<br>Requirement 22-4 |

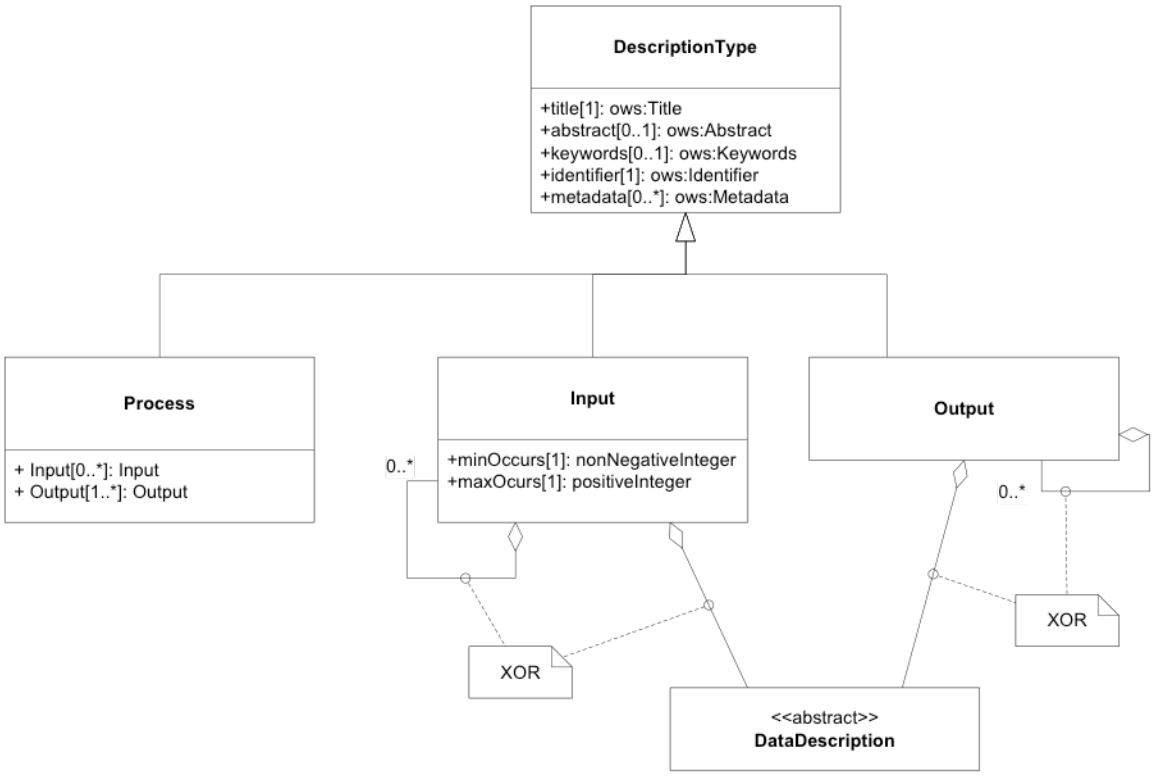


**Figure 12** — Process UML class diagram

**Table 15** — The Process structure

| NAMES | DEFINITION | DATA TYPE AND VALUES | MULTIPLICITY AND USE |
|---|---|---|---|
| Title | Inherited from Table 4 | | |
| Abstract | | | |
| Identifier | | | |
| Metadata | | | |
| Language | Language identifier for the human readable process description elements. | Character String. This language identifier shall be as specified in IETF RFC 4646. | One (mandatory) |
| Input | Input items (arguments) of a process. | Input structure, see Table 16. | Zero or more (optional) |
| Output | Output items (results) of a process | Output structure, see Table 17. | One or more (mandatory) |

**Table 16** — Parts of the Input structure

| NAMES | DEFINITION | DATA TYPE AND VALUES | MULTIPLICITY AND USE |
|---|---|---|---|
| Title | Inherited from Table 4 | | |
| Abstract | | | |
| Keywords | | | |
| Identifier | | | |
| Metadata | | | |
| minOccurs[a] | Minimum number of times that values for this parameter are required | Non-negative integer; defaults to "1", '0' means the input is optional. | Zero or on (optional) |
| maxOccurs[a] | Maximum number of times that this parameter may be present | Non-negative integer, defaults to "1". | Zero or one (optional) |
| Data Description | Data type and domain of this input. | A realization of DataDescription, i.e. ComplexData, LiteralData, Bounding BoxData. | Zero or one (conditional)[b] |

| NAMES | DEFINITION | DATA TYPE AND VALUES | MULTIPLICITY AND USE |
|---|---|---|---|
| Input | Nested Input.[c] | Input structure, Table 16 (this table). | Zero or more (conditional)[b] |

[a] The minOccurs and maxOccurs parameters have identical semantics to the like-named XML Schema occurrence constraints.

[b] The input shall either include one realization of DataDescription or an arbitrary number of sub-Inputs.

[c] It is recommended to keep the nesting level as low as possible.

**Table 17** — Parts of the Output structure

| NAMES | DEFINITION | DATA TYPE AND VALUES | MULTIPLICITY AND USE |
|---|---|---|---|
| Title | | | |
| Abstract | | | |
| Keywords | Inherited from Table 4 | | |
| Identifier | | | |
| Metadata | | | |
| Data Description | Data type and domain of this input. | A realization of DataDescription, i.e. ComplexData, LiteralData, BoundingBoxData. | Zero or one (conditional)[a] |
| Output | Nested Output.[b] | Output structure, Table 17 (this table). | Zero or more (conditional)[a] |

[a] The output shall either include either one realization of DataDescription or an arbitrary number of sub-Outputs.

[b] It is recommended to keep the nesting level as low as possible.

## 8.5. Process Profiles

Process profiles are blueprints for process implementations and are meant to harmonize process implementations to a certain degree. They serve as a reference for process implementations by providing a description of what the process actually does. While this specification does not attempt to enforce or suggest any particular process profiles, it provides a mechanism

to define common processing functionality within the scope of WPS, thus supporting basic process cataloguing and retrieval tasks for distributed processing infrastructures. Depending on the degree of harmonization, the definitions of process profiles may be used to foster a common understanding of widely used processing functions. However, they may also be used to harmonize the technical details of process interfaces and thus document particular interoperability arrangements between process providers and consumers.

| REQUIREMENTS CLASS 23 | |
|---|---|
| **OBLIGATION** | requirement |
| **TARGET TYPE** | Derived encoding and software implementation |
| **PREREQUISITE** | http://www.opengis.net/spec/WPS/2.0/req/conceptual-model/process |
| **NORMATIVE STATEMENTS** | Requirement 23-1<br>Requirement 23-2<br>Requirement 23-3<br>Requirement 23-4 |

## 8.5.1. Process Concept

A process concept is an object that provides high-level documentation about a general group of processes. It describes the purpose, methodology and properties of a process but not the specific input and output parameters. It is rather a documentation resource that may be referenced by refined process definitions to document their relation to a common principle.

**Example:** The concept “Buffer” may be used to describe all Buffer operations. More specific Buffer processes may be defined on raster or vector data models, be performed on a geoid or in CRS units, have further inputs, such as distance or tolerance and may even perform additional computations such as dissolve or line cap styling.

Due to the heterogeneity of process definitions and the variety of documentation requirements, there is no general information model for process concepts. Most of the time, concepts will be documented in HTML or similar multimedia formats. Formally, a concept consists of a unique identifier and a descriptive document.

| REQUIREMENTS CLASS 24 | |
|---|---|
| **OBLIGATION** | requirement |
| **TARGET TYPE** | Derived encoding and software implementation |
| **PREREQUISITE** | http://www.opengis.net/spec/WPS/2.0/req/conceptual-model/process |

| REQUIREMENTS CLASS 24 | |
|---|---|
| NORMATIVE STATEMENTS | Requirement 24-1<br>Requirement 24-2 |

## 8.5.2. Generic Process Profile

A generic profile is the abstract interface of a process. It provides a detailed description of the process mechanics and declares a signature for process inputs and outputs. The generic profile consists of a unique identifier and a generalized process description. This is similar to a process description as defined in Clause 8.4 but does not provide a definition of supported data exchange formats.

**Example:** A generic profile for a Buffer operation may be derived from the Buffer definition in the ISO standard for simple feature access [ISO 19125-1:2006]. The buffer method on simple features "Returns a geometric object that represents all points whose distance from this geometric object is less than or equal to distance. Calculations are in the spatial reference system of this geometric object" (ISO 19125-1:2006, subclause 6.1.2.4). This definition is specific about the conceptual data model details the behavior of the buffer method and the treatment of CRS units. In addition, it defines the name and data type of the distance parameter. This generic definition of a Simple features buffer process may be inherited by multiple implementations that use arbitrary encodings for input and output data. The important part here is the well-defined behavior of the process at a generic level and the standardization of parameter names.[4]

| REQUIREMENTS CLASS 25 | |
|---|---|
| OBLIGATION | requirement |
| TARGET TYPE | Derived encoding and software implementation |
| PREREQUISITES | http://www.opengis.net/spec/WPS/2.0/req/conceptual-model/process<br>http://www.opengis.net/spec/WPS/2.0/req/native-process/model/description-type<br>http://www.opengis.net/spec/WPS/2.0/req/native-process/model/description<br>IETF RFC 4646 |
| NORMATIVE STATEMENTS | Requirement 25-1<br>Requirement 25-2<br>Requirement 25-3 |

[4]NOTE: In terms of names, the Buffer definition in [ISO 19125-1:2006] only covers the distance parameter since it assumes a Geometry object providing that method. A complete generic profile would also have to define names for input and output geometries.

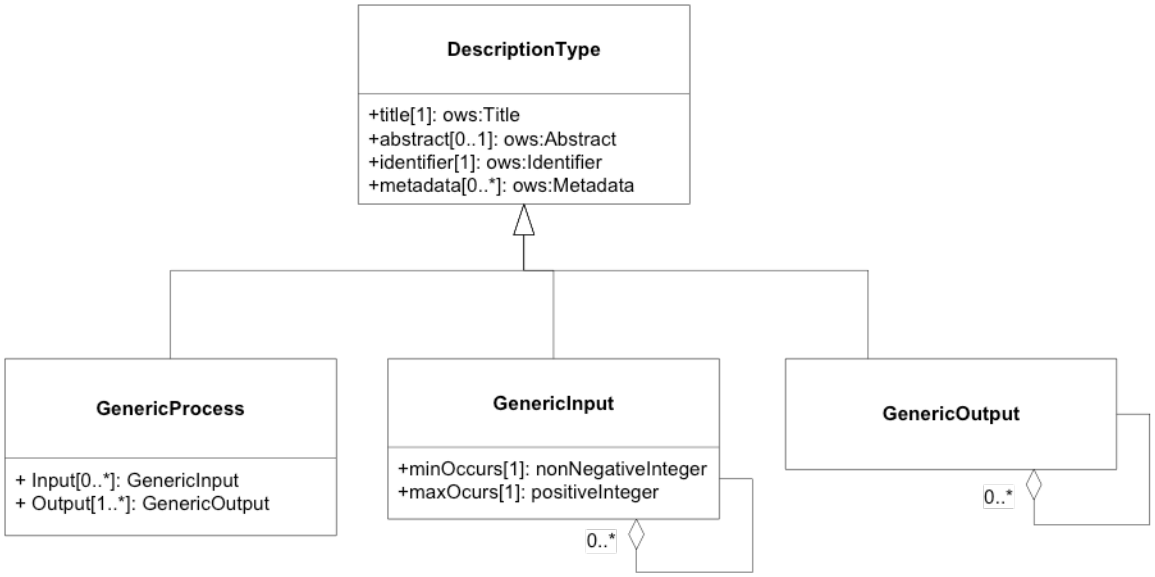


**Figure 13** — GenericProcess UML class diagram

**Table 18** — The GenericProcess structure

| NAMES | DEFINITION | DATA TYPE AND VALUES | MULTIPLICITY AND USE |
|---|---|---|---|
| Title | | | |
| Abstract | | | |
| Keywords | Inherited from Table 4 | | |
| Identifier | | | |
| Metadata | | | |
| Language | Language identifier for the human readable process description elements. | Character String. This language identifier shall be as specified in IETF RFC 4646. | One (mandatory) |
| Input | Input items (arguments) of a process. | GenericInput structure, see Table 19. | Zero or more (optional) |
| Output | Output items (results) of a process | GenericOutput structure, see Table 20. | One or more (mandatory) |

**Table 19** — Parts of the GenericInput structure

| NAMES | DEFINITION | DATA TYPE AND VALUES | MULTIPLICITY AND USE |
|---|---|---|---|
| Title | | | |
| Abstract | | | |
| Keywords | Inherited from Table 4 | | |
| Identifier | | | |
| Metadata | | | |
| min Occurs[a] | Minimum number of times that values for this parameter are required | Non-negative integer; defaults to "1", '0' means the input is optional. | Zero or one (optional) |
| max Occurs[a] | Maximum number of times that this parameter may be present | Non-negative integer, defaults to "1". | Zero or one (optional) |
| Input | Nested Input.[b] | GenericInput structure, Table 19 (this table). | Zero or more (optional) |

[a] The minOccurs and maxOccurs parameters have identical semantics to the like-named XML Schema occurrence constraints.

[b] It is recommended to keep the nesting level as low as possible.

**Table 20** — Parts of the GenericOutput structure

| NAMES | DEFINITION | DATA TYPE AND VALUES | MULTIPLICITY AND USE |
|---|---|---|---|
| Title | | | |
| Abstract | | | |
| Keywords | Inherited from Table 4 | | |
| Identifier | | | |
| Metadata | | | |
| Output | Nested Output.[a] | GenericOutput structure, Table 20 (this table). | Zero or more (conditional)[a] |

[a] It is recommended to keep the nesting level as low as possible.

### 8.5.3. Process Implementation Profile

Implementation profiles cover all descriptive elements of a process down to the supported data exchange formats. Technically they are process descriptions, but with the scope of a process profile, i.e., a harmonized and well-defined computing process that may be implemented by multiple service providers.

| REQUIREMENTS CLASS 26 | |
|---|---|
| **OBLIGATION** | requirement |
| **TARGET TYPE** | Derived encoding and software implementation |
| **PREREQUISITE** | http://www.opengis.net/spec/WPS/2.0/req/native-process/model/process-description |
| **NORMATIVE STATEMENT** | Requirement 26-1 |

### 8.5.4. Profile Inheritance

The hierarchical structure allows for inheritance between different types of profiles (see Figure 14). The definition and use of generic profiles for commonly used processing functions is generally recommended. However, there might be use cases for a product-specific harmonization of process interfaces. In this case, an implementation profile may be directly derived from a concept or defined in isolation.

If a profile in the hierarchy is derived from another profile at an equal or higher level, it must respect the inheritance rules given in Table 21. These rules ensure consistency in the use of identifiers, the specification of inputs and outputs as well as compliance to the behavior of the declared parent profiles.

Processes and process profiles may use metadata links to indicate compliance to a particular process profile. Links to particular profiles shall be embedded in the process' metadata elements. The reserved role identifiers for the different profile levels are listed in Table 22.

An example which illustrates the inheritance rules for process profiles is given in Annex B.10.

| REQUIREMENTS CLASS 27 | |
|---|---|
| **OBLIGATION** | requirement |
| **TARGET TYPE** | Derived encoding and software implementation |

| REQUIREMENTS CLASS 27 | |
|---|---|
| PREREQUISITES | http://www.opengis.net/spec/WPS/2.0/req/native-process/model/profile/concept<br>http://www.opengis.net/spec/WPS/2.0/req/native-process/model/profile/generic<br>http://www.opengis.net/spec/WPS/2.0/req/native-process/model/profile/implementation<br>http://www.opengis.net/spec/WPS/2.0/req/native-process/model/description |
| NORMATIVE STATEMENTS | Requirement 27-1<br>Requirement 27-2<br>Requirement 27-3<br>Requirement 27-4<br>Requirement 27-5 |

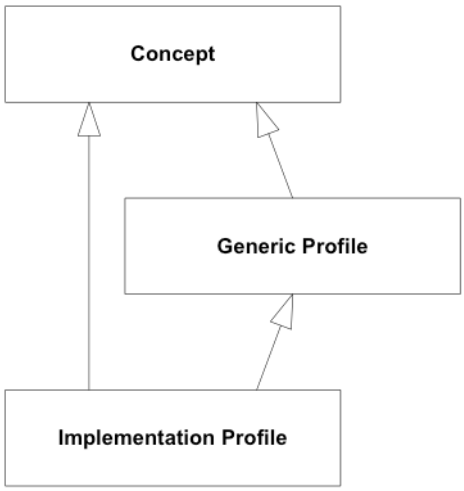


**Figure 14** — Inheritance hierarchy for process profiles UML class diagram

**Table 21** — Inheritance and override rules for process profiles

| PROPERTY | GENERIC PROFILE | IMPLEMENTATION PROFILE | IMPLEMENTATION (INSTANCE LEVEL) |
|---|---|---|---|
| Process | | | |
| Identifier | D | O | O |
| Title | D | O | O |
| Keywords | D | E | E |
| Abstract | D | O | O |
| Metadata | D | E/O[a] | E/O[a] |

| PROPERTY | GENERIC PROFILE | IMPLEMENTATION PROFILE | IMPLEMENTATION (INSTANCE LEVEL) |
|---|---|---|---|
| Input | | | E[b] |
| Identifier | D | I | I |
| Title | D | I | I |
| Keywords | D | E | E |
| Abstract | D | I | I |
| Metadata | D | E/O[a] | E/O[a] |
| Multiplicity | D | R[c] | E[d] |
| Data format | | D | E[d] |
| | | | |
| Output | | | E[b] |
| Identifier | D | I | I |
| Title | D | I | I |
| Keywords | D | E | E |
| Abstract | D | I | I |
| Metadata | D | O | O |
| Data format | | D | E[d] |

NOTE:

| | |
|---|---|
| D | Declare (Introduction of a new property) |
| I | Inherit (A property is inherited from a superior level AND its value remains unchanged.) |
| O | Override (A property is inherited from a superior level AND its value is overridden.) |
| E | Extend (A property is inherited from a superior level AND its value is extended.) |
| R | Restrict (A property is inherited from a superior level AND its value is restricted.) |

[a] The list of metadata references to superior process profiles shall be extended. Documentation metadata may be overridden, i.e. replaced by other documentation resources.

[b] Additional optional inputs or supplementary outputs may be added here.

[c] Implementation profiles may restrict the maximum cardinality of a superior generic profile (e.g. for theoretically infinite inputs). They shall not modify the minimum cardinality.

[d] Implementations may allow more or larger input datasets than the implementation profile, or support additional data exchange formats.

**Table 22** — Role identifiers for process profiles

| PROFILE LEVEL | URI |
|---|---|
| Process concept | http://www.opengis.net/spec/wps/2.0/def/process-profile/concept |
| Generic process profile | http://www.opengis.net/spec/wps/2.0/def/process-profile/generic |
| Process Implementation profile | http://www.opengis.net/spec/wps/2.0/def/process-profile/implementation |

9

# WPS NATIVE PROCESS MODEL ENCODING

# 9 WPS NATIVE PROCESS MODEL ENCODING

## 9.1. XML Schema Implementation

This section specifies the XML encoding of the elements of the WPS native process model. The referred XML schema elements are provided by the associated schema package delivered with this standard and located at http://schemas.opengis.net/wps/2.0/.

| REQUIREMENTS CLASS 28 | |
|---|---|
| OBLIGATION | requirement |
| TARGET TYPE | Derived encoding and software implementation |
| PREREQUISITES | http://www.opengis.net/spec/WPS/2.0/req/native-process/model<br>OWS Common 2.0 |
| NORMATIVE STATEMENTS | Requirement 28-1<br>Requirement 28-2<br>Requirement 28-3 |

### 9.1.1. Data Types

The XML encoding of data types defines encoding rules for ComplexData, LiteralData, BoundingBoxData as well as their values.

Examples for data type encodings are listed in Annex B.1.

| REQUIREMENTS CLASS 29 | |
|---|---|
| OBLIGATION | requirement |
| TARGET TYPE | Derived encoding and software implementation |
| PREREQUISITES | http://www.opengis.net/spec/WPS/2.0/req/native-process/model/datatypes/complex-data<br>http://www.opengis.net/spec/WPS/2.0/req/native-process/model/datatypes/literal-data<br>http://www.opengis.net/spec/WPS/2.0/req/native-process/model/datatypes/bounding-box-data |

| REQUIREMENTS CLASS 29 | |
|---|---|
| | OWS Common 2.0 |
| NORMATIVE STATEMENTS | Requirement 29-1<br>Requirement 29-2 |

### 9.1.2. Process Description

This clause specifies the XML encoding for the Process description.

A Process example is listed in Annex B.2.

| REQUIREMENTS CLASS 30 | |
|---|---|
| OBLIGATION | requirement |
| TARGET TYPE | Derived software implementation |
| PREREQUISITES | http://www.opengis.net/spec/WPS/2.0/req/native-process/model/description<br>OWS Common 2.0 |
| NORMATIVE STATEMENTS | Requirement 30-1<br>Requirement 30-2 |

### 9.1.3. Generic Process

This clause specifies the XML encoding for the GenericProcess.

A GenericProcess example is listed in Annex B.3.

| REQUIREMENTS CLASS 31 | |
|---|---|
| OBLIGATION | requirement |
| TARGET TYPE | Derived software implementation |
| PREREQUISITES | http://www.opengis.net/spec/WPS/2.0/req/native-process/profile/generic<br>OWS Common 2.0 |
| NORMATIVE STATEMENTS | Requirement 31-1<br>Requirement 31-2 |

## 9.2. Plain Text Encoding for LiteralData and BoundingBoxData Values

This clause specifies the plain text encoding of data types for literal and bounding box data values.

| REQUIREMENTS CLASS 32 | |
|---|---|
| **OBLIGATION** | requirement |
| **TARGET TYPE** | Derived encoding and software implementation |
| **PREREQUISITES** | http://www.opengis.net/spec/WPS/2.0/req/native-process/model/datatypes/complex-data<br>http://www.opengis.net/spec/WPS/2.0/req/native-process/model/datatypes/literal-data<br>http://www.opengis.net/spec/WPS/2.0/req/native-process/model/datatypes/bounding-box-data<br>OWS Common 2.0 |
| **NORMATIVE STATEMENTS** | Requirement 32-1<br>Requirement 32-2 |

```
Literal values - BNF schema:
literalvalue   = value *1("@datatype=" datatype) *1("@uom=" uom)
value          = 1*VCHAR
datatype  = URI
uom       = URI

Literal values - Example:
70@datatype=http://www.w3.org/2001/XMLSchema#integer@uom=meter

BoundingBox values - BNF schema[5]:
bbox      = lc_coords "," uc_coords ["," crs] ["," dimensions]
lc_coords = number ["," number]
uc_coords = number ["," number]
number         = 1*DIGIT["." 1*DIGIT]
crs       = 1*VCHAR
dimensions = 1*DIGIT

BoundingBoxData values - Examples:
51.9,7.0,53.0,8.0,EPSG:4326
```

[5] The dimensions attribute is included for compliance with the BoundingBox structure defined in OWS Common [OGC 06-121r9]. However, its use should be generally avoided since the number of dimensions is already part of the CRS definition and usually superfluous.

```
51.9,7.0,53.0,8.0,http://www.opengis.net/def/crs/EPSG/0/4258
```

10

# COMMON WPS SERVICE MODEL

# 10 COMMON WPS SERVICE MODEL

A Web Processing Service consists of processes and service operations. By definition, processes represent the computational functionality of a WPS, while service operations are used to interact with the WPS and in particular to use the service's process offerings.

| REQUIREMENTS CLASS 33 | |
|---|---|
| **OBLIGATION** | requirement |
| **TARGET TYPE** | Software implementation |
| **PREREQUISITES** | http://www.opengis.net/spec/WPS/2.0/req/conceptual-model<br>OWS Common 2.0 |
| **NORMATIVE STATEMENT** | Requirement 33-1 |
| **DESCRIPTION** | *Requirements class for WPS service handling.* |
| Example | |
| **NORMATIVE STATEMENTS** | Requirement 33-2<br>Requirement 33-3<br>Requirement 33-4<br>Requirement 33-5<br>Requirement 33-6<br>Requirement 33-7<br>Requirement 33-8<br>Requirement 33-9 |

## 10.1. Overview of WPS Core Operations

The WPS interface specified in this section supports retrieval and execution of processes for geospatial computation. For that purpose, the WPS service model specifies the following operations that may be invoked by a WPS client and performed by a WPS server[6]:

GetCapabilities — This operation allows a client to request information about the server's capabilities and processes offered (see Clause 10.7).

[6]NOTE: Future extensions of this specification may introduce additional operations.

DescribeProcess — This operation allows a client to request detailed metadata on selected processes offered by a server (see Clause 10.8).

Execute — This operation allows a client to execute a process comprised of a process identifier, the desired data inputs and the desired output formats (see Clause 10.9).

GetStatus — This operation allows a client to query status information of a processing job (conditional, see Clause 10.10).

GetResult — This operation allows a client to query the results of a processing job (conditional, see Clause 10.11).

During a sequence of WPS requests, a client should first issue a GetCapabilities request to the server to obtain an up-to date listing of available processes. Then, it may issue a DescribeProcess request to find out more details about particular processes offered, including the supported data formats. To run a process with the desired input data, a client will issue an Execute request[7] (Figure 15).

The operations GetStatus and GetResult are used in conjunction with asynchronous execution. If a WPS server offers synchronous process execution only, these operations may not be implemented. Detailed guidance is provided by the corresponding profiles (Clause 10.12 and Clause 10.13) and conformance classes.

[7]NOTE: A WPS server can change its offering at any time, in particular between a GetCapabilities, a subsequent DescribeProcess, and a subsequent Execute request. Any quality of service (QoS) guarantees are within the responsibilities of the service provider and not covered by this standard.

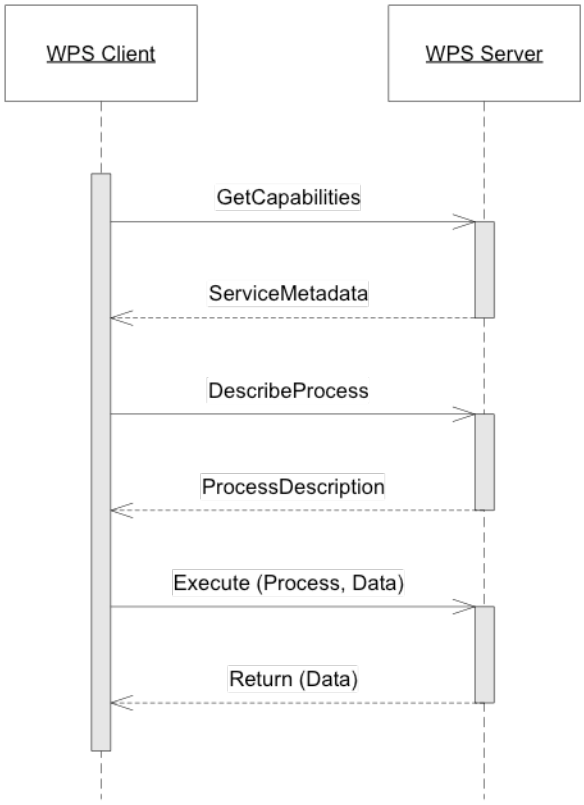


**Figure 15** — Common sequence of WPS operations UML sequence diagram

## 10.2. Data Transmission

Data exchange between WPS clients and servers requires an agreement on the general data exchange patterns and suitable communication protocols. Data may be sent to (received from) a WPS in two distinct ways: (1) by reference (using HTTP/GET or HTTP/POST), and (2) by value. Clients may send input data in either fashion. Output data may be requested in any fashion declared by the data transmission options defined for the process offering. Typically, large data inputs and outputs are delivered by reference.

| REQUIREMENTS CLASS 34 | |
|---|---|
| **OBLIGATION** | requirement |
| **TARGET TYPE** | Derived encoding and software implementation |
| **PREREQUISITE** | http://www.opengis.net/spec/WPS/2.0/req/conceptual-model |
| **NORMATIVE STATEMENT** | Requirement 34-1 |

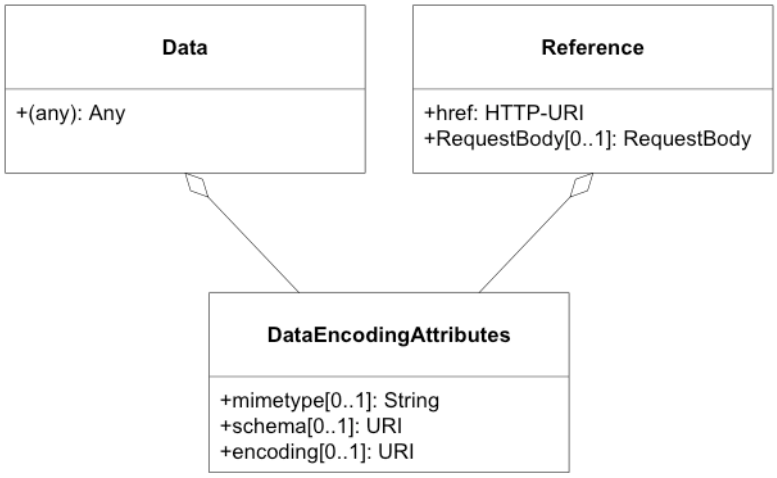


**Figure 16** — Input and output data transmission structures UML class diagram

**Table 23** — Parts of the inline Data structure

| NAMES | DEFINITION | DATA TYPE AND VALUES | MULTIPLICITY AND USE |
|---|---|---|---|
| mimetype | | | |
| encoding | See Table 24 — Properties of the DataEncodingAttributes structure. | | |
| schema | | | |
| (any)[a] | The actual input or output data. | Any type and value | One (mandatory) |

[a] The data is embedded here as part of the Data element, in the mimeType, encoding, and schema indicated by the first three parameters if they exist, or by the relevant defaults.

**Table 24** — Properties of the DataEncodingAttributes structure

| NAMES | DEFINITION | DATA TYPE AND VALUES | MULTIPLICITY AND USE |
|---|---|---|---|
| mimetype | MimeType of the data. | Character String | One (mandatory) |
| encoding | Well-known encoding or character set of the data. | URI "raw" shall be used for plain binary data "base64" shall be used for base64 encoded data Character set identifiers (e.g. "UTF-8") shall be used for text or CSV data. | Zero or one (conditional)[a] |
| schema | Identification of the data schema. | URI | Zero or one (conditional)[a] |

[a] This shall be provided if: 1) the process data item supports multiple encodings / schemas, and 2) the data is not of the default encoding / schema, and 3a) the schema / encoding cannot be retrieved from the data itself, or 3b) the

encoding / schema information is deeply buried inside the data (i.e. not part of some header) and requires significant parsing effort.

**Table 25** — Parts of the Reference structure

| NAMES | DEFINITION | DATA TYPE AND VALUES | MULTIPLICITY AND USE |
|---|---|---|---|
| mimetype | See Table 24 — Properties of the DataEncodingAttributes structure. | | |
| encoding | | | |
| schema | | | |
| href | HTTP URI that points to the remote resource where the data may be retrieved. | HTTP URI | One (mandatory) |
| Request Body | Request body element that is used for HTTP/POST requests to the above URL. If no request body is present, an HTTP/GET | Request should be used to retrieve the data. RequestBody structure, see Table 26. | Zero or one (optional) |

**Table 26** — Parts of the RequestBody structure

| NAMES | DEFINITION | DATA TYPE AND VALUES | MULTIPLICITY AND USE |
|---|---|---|---|
| Body | The contents of this element to be used as the body of the HTTP request message to be sent to the service identified in ../Reference/@href. For example, it could be an XML encoded WFS request using HTTP/POST. | Any type | Zero or one (conditional)[a] |
| Body Reference | Reference to a remote document to be used as the body of an HTTP/POST request message to the service identified in the href element in the Reference structure (Table 25). | BodyReference, see Table 27. | Zero or one (conditional)[a] |

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

**Table 27** — Parts of the BodyReference structure

| NAMES | DEFINITION | DATA TYPE AND VALUES | MULTIPLICITY AND USE |
|---|---|---|---|
| href | HTTP URI that points to the remote resource where the request body may be retrieved. | HTTP URI | One (mandatory) |

## 10.3. WPS Service Handling

The WPS service model seeks compliance with OWS Common and implements the baseline communication protocol for OWS services and the related information elements defined in [OGC 06-121r9], clause 9.2.1.

| REQUIREMENTS CLASS 35 | |
|---|---|
| **OBLIGATION** | requirement |
| **TARGET TYPE** | Derived encoding and software implementation |
| **PREREQUISITE** | http://www.opengis.net/spec/WPS/2.0/req/conceptual-model |
| **NORMATIVE STATEMENTS** | Requirement 35-1<br>Requirement 35-2<br>Requirement 35-3<br>Requirement 35-4 |

**Table 28** — Properties of the RequestBaseType

| NAMES | DEFINITION | DATA TYPE AND VALUES | MULTIPLICITY AND USE |
|---|---|---|---|
| service | Service type identifier | Character String, fixed to "WPS" | One (mandatory) |
| version | Specification version for operation | Character String, fixed to "2.0.0" | One or more (mandatory) |
| Extension | Any ancillary information to be sent from client to server. Placeholder for further request parameters defined by WPS extension standards. | Any type | Zero or more (optional) |

## 10.4. Process Offering

A process offering structure contains information about the processes that may be run on a WPS server. Furthermore, the process offerings structure contains properties that describe the available execution modes of a process, the allowed data transmission modes, its version, and the process type (if it deviates from the native process model).

| REQUIREMENTS CLASS 36 | |
|---|---|
| OBLIGATION | requirement |
| TARGET TYPE | Derived encoding and software implementation |
| PREREQUISITE | http://www.opengis.net/spec/WPS/2.0/req/conceptual-model |
| NORMATIVE STATEMENT | Requirement 36-1 |

**Table 29** — Parts of the ProcessOfferingPropertiesAttributes structure

| NAMES | DEFINITION | DATA TYPE AND VALUES | MULTIPLICITY AND USE |
|---|---|---|---|
| jobControlOptions | Job control options supported for this process | List of supported options for process control (see Table 30), extensions may introduce additional control options. | One or more (mandatory) |
| output Transmission | Supported transmission modes for output data (by value / by reference) | List of supported data transmission options (see Table 31). | One or more (mandatory) |
| processVersion | Release version of process (not of WPS specification). May be specified to reflect updates or changes in the process offering. | ows:VersionType | Zero or one (optional) Include when needed to identify process version.[a] |
| processModel | Type of the process description | HTTP-URI. Value is defined by the process description specification. Defaults to "native". | Zero or one (conditional) Include when using a different process model than the native process model.[b] |

[a] The processVersion is informative only. Version negotiation for processVersion is not available. Requests to Execute a process do not include a processVersion identifier.

[b] This is an extension hook to support processes that have been specified in other OGC Standards, such as SensorML. For those process models, compliance with the WPS abstract process model (Clause 7.3) has to be ensured.

**Table 30** — Basic job control options

| OPTION | DEFINITION |
|---|---|
| sync-execute | The process offering can/shall be executed synchronously. |

| OPTION | DEFINITION |
|---|---|
| async-execute | The process offering can/shall be executed asynchronously. |

**Table 31** — Data transmission options

| OPTION | DEFINITION |
|---|---|
| value | The data is delivered by value. |
| reference | The data is delivered by reference. |

## 10.5. StatusInfo Document

The StatusInfo document is used to provide identification and status information about jobs on a WPS server.

| REQUIREMENTS CLASS 37 | |
|---|---|
| **OBLIGATION** | requirement |
| **TARGET TYPE** | Derived encoding and software implementation |
| **PREREQUISITE** | http://www.opengis.net/spec/WPS/2.0/req/conceptual-model |
| **NORMATIVE STATEMENTS** | Requirement 37-1<br>Requirement 37-2<br>Requirement 37-3 |

**Table 32** — StatusInfo structure

| NAMES | DEFINITION | DATA TYPE AND VALUES | MULTIPLICITY AND USE |
|---|---|---|---|
| JobID | Unambiguously identifier of a job within a WPS instance. | Character String[a] | One (mandatory) |
| Status | Well-known identifier describing the status of the job. | Character String[b] | One (mandatory) |

| NAMES | DEFINITION | DATA TYPE AND VALUES | MULTIPLICITY AND USE |
|---|---|---|---|
| ExpirationDate | Date and time by which the job and its results will be no longer accessible.[c] | ISO-8601 date/time string in the form YYYY-MM-DDTHH:MM:SS.SSSZ with T separator character and Z suffix for coordinated universal time (UTC) | Zero or one (optional) Include if required. |
| Estimated Completion | Date and time by which the processing job will be finished. | ISO-8601 date/time string in the form YYYY-MM-DDTHH:MM:SS.SSSZ with T separator character and Z suffix for coordinated universal time (UTC) | Zero or one (optional) Include if available. |
| NextPoll | Date and time for the next suggested status polling. | ISO-8601 date/time string in the form YYYY-MM-DDTHH:MM:SS.SSSZ with T separator character and Z suffix for coordinated universal time (UTC) | Zero or one (optional) Include if required. |
| PercentCompleted | Percentage of process that has been completed. | Integer{0..100}[d] | Zero or one (optional) Include if available. |

[a] Particularly suitable JobIDs are UUIDs or monotonic identifiers such as unique timestamps. If the privacy of a Processing Job is imperative, the JobID should be non-guessable.

[b] The basic status set is defined in Table 3. Additional states may be defined by certain operations or extensions of this standard.

[c] This element will usually become available when the execution has finished (Status = “finished”).

[d] Zero (0) means the execution has just started, and 100 means the job is complete. This value is informative only without any accuracy guarantees.

## 10.6. Result Document

A Result document is a structure that contains the results of a process execution. It is a shared element between the Execute and GetResult operations.

| REQUIREMENTS CLASS 38 | |
|---|---|
| OBLIGATION | requirement |
| TARGET TYPE | Derived encoding and software implementation |
| PREREQUISITES | http://www.opengis.net/spec/WPS/2.0/req/conceptual-model<br>http://www.opengis.net/spec/WPS/2.0/req/service/model/data-transmission |

| REQUIREMENTS CLASS 38 | |
|---|---|
| **NORMATIVE STATEMENT** | Requirement 38-1 |

**Table 33** — Result structure

| NAMES | DEFINITION | DATA TYPE AND VALUES | MULTIPLICITY AND USE |
|---|---|---|---|
| JobID | Unambiguously identifier of a job within a WPS instance. | Character String[a] | Zero or one (conditional)[b] |
| Expiration Date | Date and time by which the results will be no longer accessible.[c] | ISO-8601 date/time string in the form YYYY-MM-DDTHH:MM:SS.SSSZ with T separator character and Z suffix for coordinated universal time (UTC) | Zero or one (conditional) Include if required, i.e. if the server will delete stored results at some point in time. |
| Output | Output item returned by a process execution. | DataOutputType structure, see Table 37. | One or more (mandatory) |

[a] Particularly suitable JobIDs are UUIDs or monotonic identifiers such as unique timestamps. If the privacy of a Processing Job is imperative, the JobID should be non-guessable. In asynchronous execution, the JobID would be shared among related StatusInfo and Result documents.

[b] Include if required, e.g. in a response to an asynchronous execution.

[c] For results delivered "by reference" this element may indicate when the Data Outputs will be deleted by the server.

**Table 34** — Parts of the DataOutputType structure

| NAMES | DEFINITION | DATA TYPE AND VALUES | MULTIPLICITY AND USE |
|---|---|---|---|
| id | Unambiguous identifier or name of an output item. | URI | One (mandatory) |
| Data | The data provided by this output item. | | Zero or one (conditional)[a] |
| Output | Nested output, child element. | DataOutputType structure, see Table 34 (this table) | Zero or one (conditional)[a] |

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

# 10.7. GetCapabilities Operation

Per OGC 06-121r9, a GetCapabilities operation is required for any OGC Web service. For WPS, this operation allows a client to retrieve service metadata, basic process offerings, and the available processes present on a WPS server.

| REQUIREMENTS CLASS 39 | |
|---|---|
| OBLIGATION | requirement |
| TARGET TYPE | Software implementation |
| PREREQUISITES | http://www.opengis.net/spec/WPS/2.0/req/conceptual-model<br>OWS Common 2.0 |
| NORMATIVE STATEMENTS | Requirement 39-1<br>Requirement 39-2<br>Requirement 39-3 |

## 10.7.1. GetCapabilities Request

The GetCapabilities request is mandatory for any OGC service. Figure 17 shows how the GetCapabilities request for a WPS relates to the generic GetCapabilitiesType defined by OWS Common. An Extension element provides a hook for further request parameters that may be defined by WPS extension specifications.

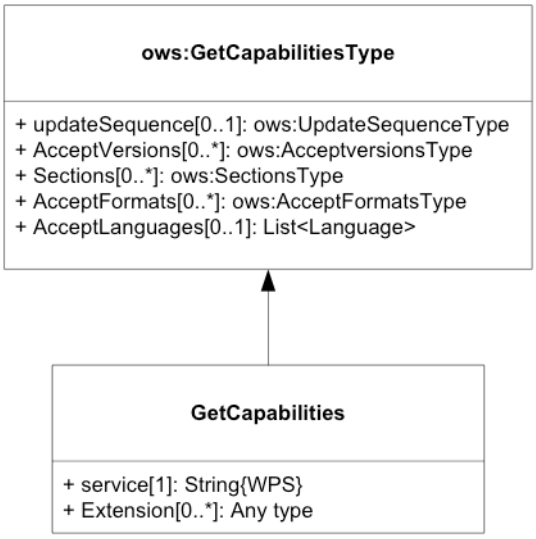


**Figure 17** — GetCapabilities request UML class diagram

| REQUIREMENTS CLASS 40 | |
|---|---|
| OBLIGATION | requirement |
| TARGET TYPE | Software implementation |
| PREREQUISITES | http://www.opengis.net/spec/WPS/2.0/req/conceptual-model<br>OWS Common 2.0 |
| NORMATIVE STATEMENTS | Requirement 40-1<br>Requirement 40-2<br>Requirement 40-3 |

**Table 35** — Additional properties in the GetCapabilities request

| NAMES | DEFINITION | DATA TYPE AND VALUES | MULTIPLICITY AND USE |
|---|---|---|---|
| Service | Service type identifier | Character string, fixed to “WPS” | One (mandatory) |
| Extension | Container for elements defined by extension specifications | Any type. Value is defined by the extension specification. | Zero or more (optional) |

## 10.7.2. GetCapabilities Response

The response to a GetCapabilities operation is a document describing the service’s capabilities. Figure 18 shows how the WPS Capabilities are derived from the CapabilitiesBaseType defined in [OGC 06-121r9]. The OperationsMetadata element lists the request types supported by a WPS server. The contents section delivers information about the process offerings of the server. An Extension element provides a hook for additional service capabilities that cannot be covered by other available elements.

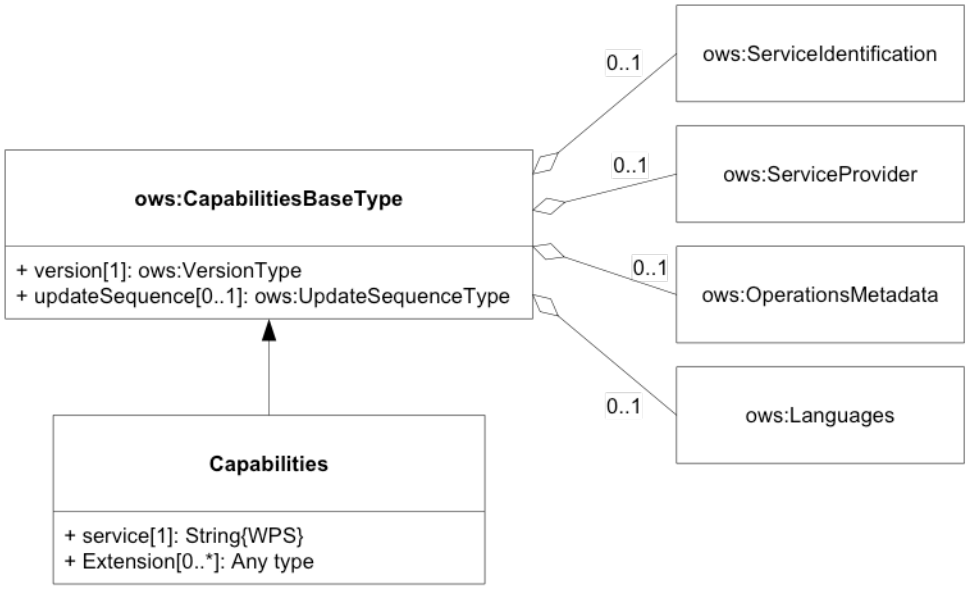


**Figure 18** — Capabilities document UML class diagram

| REQUIREMENTS CLASS 41 | |
|---|---|
| OBLIGATION | requirement |
| TARGET TYPE | Software implementation |
| PREREQUISITES | http://www.opengis.net/spec/WPS/2.0/req/conceptual-model<br>OWS Common 2.0 |
| NORMATIVE STATEMENTS | Requirement 41-1<br>Requirement 41-2<br>Requirement 41-3<br>Requirement 41-4<br>Requirement 41-5 |

**Table 36** — Additional properties in the Capabilities document

| NAMES | DEFINITION | DATA TYPE AND VALUES | MULTIPLICITY AND USE |
|---|---|---|---|
| service | Service type identifier | Character string, fixed to “WPS” | One (mandatory) |
| Contents | List of brief descriptions of the processes offered by this WPS server. | ProcessSummary, see Table 37 | One (mandatory) |
| Extension | container for elements defined by extension specifications | Any type. Value is defined by the extension specification. | Zero or more (optional) |

**Table 37** — Properties of the ProcessSummary

| NAMES | DEFINITION | DATA TYPE AND VALUES | MULTIPLICITY AND USE |
|---|---|---|---|
| Title | Title of a process, normally available for display to a human. | ows:Title | One (mandatory) |
| Abstract | Brief narrative description of a process, normally available for display to a human. | ows:Abstract | Zero or more (optional) |
| Keywords | Keywords that characterize a process, | ows:Keyword | Zero or more (optional) |
| Identifier | Unambiguous identifier or name of a process. | ows:Identifier[a] | One (mandatory) |
| Metadata | Reference to more metadata about this item. | ows:Metadata | Zero or more (optional) Include when available and useful |
| processModel | Inherited from Table 29. | | |
| jobControlOptions | | | |
| output Transmission | | | |

[a] Additional content such as separate code space and version attributes in the Identifier element are not allowed.

## 10.7.3. GetCapabilities Exceptions

If a WPS server encounters an error while performing a GetCapabilities operation, it shall return an exception report as specified in Clause 7.4 of [OGC 06-121r9].

| REQUIREMENTS CLASS 42 | |
|---|---|
| **OBLIGATION** | requirement |
| **TARGET TYPE** | Software implementation |
| **PREREQUISITES** | http://www.opengis.net/spec/WPS/2.0/req/conceptual-model<br>OWS Common 2.0 |
| **NORMATIVE STATEMENT** | Requirement 42-1 |

## 10.8. DescribeProcess Operation

The DescribeProcess operation allows WPS clients to query detailed process descriptions for the process offerings.

| REQUIREMENTS CLASS 43 | |
|---|---|
| **OBLIGATION** | requirement |
| **TARGET TYPE** | Software implementation |
| **PREREQUISITES** | http://www.opengis.net/spec/WPS/2.0/req/conceptual-model<br>OWS Common 2.0 |
| **NORMATIVE STATEMENTS** | Requirement 43-1<br>Requirement 43-2<br>Requirement 43-3 |

### 10.8.1. DescribeProcess Request

The DescribeProcess request inherits basic properties from the RequestBaseType. An Identifier element shall contain a list of the process identifiers for which the process descriptions shall be obtained. If the service supports multilingual process descriptions, the desired language of the free-text elements in the process description may be queried with a language parameter.

| REQUIREMENTS CLASS 44 | |
|---|---|
| **OBLIGATION** | requirement |
| **TARGET TYPE** | Software implementation |
| **PREREQUISITES** | http://www.opengis.net/spec/WPS/2.0/req/conceptual-model<br>OWS Common 2.0<br>IETF RFC 4646<br>http://www.opengis.net/spec/WPS/2.0/req/service/model/handling |
| **NORMATIVE STATEMENTS** | Requirement 44-1<br>Requirement 44-2 |

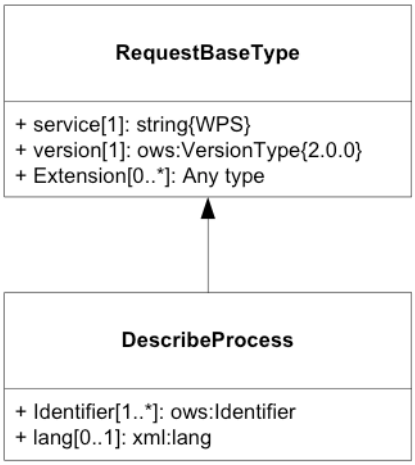


**Figure 19** — DescribeProcess request UML class diagram

**Table 38** — Additional properties in the DescribeProcess request

| NAMES | DEFINITION | DATA TYPE AND VALUES | MULTIPLICITY AND USE |
|---|---|---|---|
| Identifier | One or more process identifiers for which the detailed description shall be obtained. | ows:Identifier Value shall be one of the process identifiers listed in the ProcessSummary elements in the Capabilities document. The fixed value "ALL" indicates that the description of all process offerings shall be returned. | One or more (mandatory) |
| lang | Desired language of the process description. | xml:lang IETF RFC 4646 language code of the human-readable text elements in the process description (e.g. "en"). This shall be one of the languages defined in the Capabilities document. | Zero or one (optional) |

## 10.8.2. DescribeProcess Response

The response to a DescribeProcess operation is a ProcessOfferings document. This document contains a ProcessOfferings section for each available process on the server. In contrast to the ProcessSummary in the server's capabilities, the processes are described in their declared description format.

| REQUIREMENTS CLASS 45 | |
|---|---|
| OBLIGATION | requirement |
| TARGET TYPE | Software implementation |
| PREREQUISITES | http://www.opengis.net/spec/WPS/2.0/req/conceptual-model<br>IETF RFC 4646 |

| REQUIREMENTS CLASS 45 | |
|---|---|
| NORMATIVE STATEMENTS | Requirement 45-1<br>Requirement 45-2 |

**Table 39** — Properties in the ProcessOfferings document

| NAMES | DEFINITION | DATA TYPE AND VALUES | MULTIPLICITY AND USE |
|---|---|---|---|
| lang | Language in which the process offerings are described. | xml:lang IETF RFC 4646 language code of the human-readable text elements in the process description (e.g. "en"). This shall be the language identified in the DescribeProcess request. | Zero or one (optional) |
| Process Offerings | List of ProcessOfferings. | ProcessOffering, defined in Table 40. | One or more (optional) |

**Table 40** — ProcessOffering properties

| NAMES | DEFINITION | DATA TYPE AND VALUES | MULTIPLICITY AND USE |
|---|---|---|---|
| processModel | | | |
| jobControlOptions | Inherited from Table 29 | | |
| output Transmission | | | |
| Process | Native Process description. | Process type, as defined in the native process model. | Zero or one (conditional)[a] |
| Any | Any other well-defined process description, identified by the processType. | Any type. Must conform to requirements associated with the declared processType. | Zero or one (conditional)[a] |

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

## 10.8.3. DescribeProcess Exceptions

If a WPS server encounters an error while performing a DescribeProcess operation, it shall return an exception report as specified in Clause 8 of [OGC 06-121r9]. If the error was encountered due to an invalid process identifier, the server shall respond with the exception code defined in Table 41.

| REQUIREMENTS CLASS 46 | |
|---|---|
| OBLIGATION | requirement |
| TARGET TYPE | Software implementation |
| PREREQUISITES | http://www.opengis.net/spec/WPS/2.0/req/conceptual-model<br>OWS Common 2.0 |
| NORMATIVE STATEMENTS | Requirement 46-1<br>Requirement 46-2 |

**Table 41** — Additional exception codes for the DescribeProcess operation

| EXCEPTIONCODE VALUE | EXCEPTIONTEXT | LOCATOR | HTTP STATUS CODE |
|---|---|---|---|
| NoSuchProcess | One of the identifiers passed does not match with any of the processes offered by this server. | List of violating process identifiers. | 400 (Bad request) |

## 10.9. Execute Operation

The Execute operation allows WPS clients to run a specified process implemented by a server, using the input parameter values provided and returning the output values produced. Inputs may be included directly in the Execute request (by value), or reference web accessible resources (by reference). The outputs may be returned in the form of an XML response document, either embedded within the response document or stored as web accessible resources. Alternatively, for a single output, the server may be directed to return that output in its raw form without being wrapped in an XML response document. This is illustrated in Figure 20.

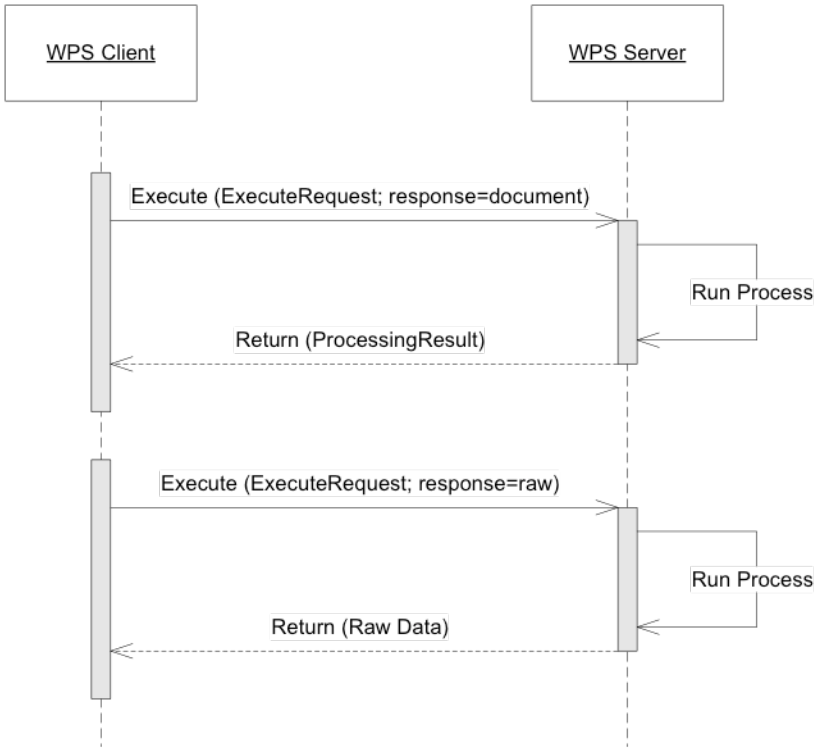


**Figure 20** — Execute response document and raw data UML sequence diagram

| REQUIREMENTS CLASS 47 | |
|---|---|
| OBLIGATION | requirement |
| TARGET TYPE | Software implementation |
| PREREQUISITES | http://www.opengis.net/spec/WPS/2.0/req/conceptual-model<br>OWS Common 2.0 |
| NORMATIVE STATEMENTS | Requirement 47-1<br>Requirement 47-2<br>Requirement 47-3 |

## 10.9.1. Execute Request

The Execute request is a common structure for synchronous and asynchronous execution. It inherits basic properties from the RequestBaseType and contains additional elements that identify the process that shall be executed, the data inputs and outputs, and the response type of the service.

| REQUIREMENTS CLASS 48 | |
|---|---|
| OBLIGATION | requirement |
| TARGET TYPE | Software implementation |
| PREREQUISITES | http://www.opengis.net/spec/WPS/2.0/req/conceptual-model<br>OWS Common 2.0<br>http://www.opengis.net/spec/WPS/2.0/req/service/model/handling |
| NORMATIVE STATEMENTS | Requirement 48-1<br>Requirement 48-2 |

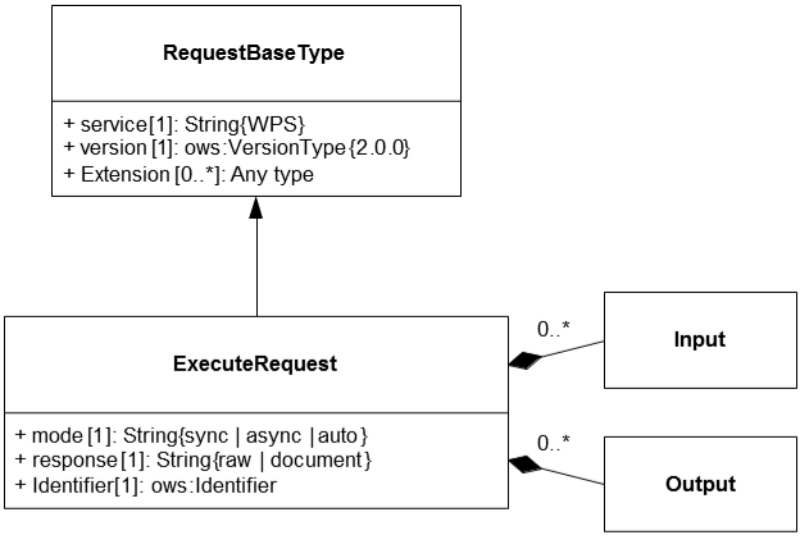


**Figure 21** — Execute request UML class diagram

**Table 42** — Additional properties in the Execute request

| NAMES | DEFINITION | DATA TYPE AND VALUES | MULTIPLICITY AND USE |
|---|---|---|---|
| response | Desired response format, i.e. a response document or raw data. | String {raw[a] \| document} | One (mandatory) |
| mode | Desired execution mode. | String{sync \| async \| auto[b]} Valid values are to be derived from the jobControlOptions property of each ProcessOffering. "auto" delegates the choice of execution mode to the server. | One (mandatory) |
| Identifier | Unambiguous identifier of the process that shall be executed. | ows:Identifier Value shall be one of the process identifiers listed in the ProcessSummary elements in the Capabilities document. | One (mandatory) |
| Input | Data inputs provided to this process execution. | DataInputType structure, see Table 43. | Zero or more (conditional)[c] |

| NAMES | DEFINITION | DATA TYPE AND VALUES | MULTIPLICITY AND USE |
|---|---|---|---|
| Output | Specification of outputs expected from the process execution, including the desired format and transmission mode for each output. | OutputDefinitionType structure, see Table 44. | Zero or more (conditional)[d] |

[a] Raw output shall only be requested if the execution would return a single output.

[b] In the case of auto, the server shall respond quickly to avoid connection timeouts.

[c] If a process does not have any inputs, or if, for all required inputs the process description contains default values, this element may be omitted.

[d] If a process shall return all outputs in their default format, this element may be omitted.

**Table 43** — Properties of the DataInputType

| NAMES | DEFINITION | DATA TYPE AND VALUES | MULTIPLICITY AND USE |
|---|---|---|---|
| id | Identifier of a particular input, as defined in the process description. | URI | One (mandatory) |
| Data | The data provided for this input item. | Data structure, Table 23 | Zero or one (conditional)[a] |
| Reference | HTTP resource that provides the input data. | Reference structure, Table 25 | Zero or one (conditional)[a] |
| Input | Nested input, child element. | DataInputType structure, Table 43 (this table) | Zero or more (conditional)[a] |

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

**Table 44** — Properties of the OutputDefinitionType

| NAMES | DEFINITION | DATA TYPE AND VALUES | MULTIPLICITY AND USE |
|---|---|---|---|
| mimetype | | | |
| encoding | See Table 24 — Properties of the DataEncodingAttributes structure. | | |
| schema | | | |

| NAMES | DEFINITION | DATA TYPE AND VALUES | MULTIPLICITY AND USE |
|---|---|---|---|
| transmission | Code that indicates the desired data transmission mode for this output. | Character string. Valid values are listed in the outputTransmission property of each Process Offering. | Zero or one (conditional)[a] |
| id | Identifier of a particular output, as defined in the process description. | URI | One (mandatory) |
| Output[b] | Nested output, child element. | OutputDefinitionType structure, Table 44 (this table). | Zero or more[c] (conditional)[d] |

[a] This element may be omitted for (1) raw data output, (2) output elements that serve as nesting parents.

[b] Depending on the process model, a client may provide a basic specification of the desired output format. This information shall ensure a successful decoding of the process outputs. Use and interpretation of this element shall be specified within a WPS service profile.

[c] See Figure 2

[d] Include only for nested outputs.

## 10.9.2. Execute Response

Depending on the desired execution mode and the response type declared in the execute request, the execute response may take one of three different forms: A response document, a StatusInfo document, or raw data.

| REQUIREMENTS CLASS 49 | |
|---|---|
| OBLIGATION | requirement |
| TARGET TYPE | Software implementation |
| PREREQUISITES | http://www.opengis.net/spec/WPS/2.0/req/conceptual-model<br>http://www.opengis.net/spec/WPS/2.0/req/service/model/status-info<br>http://www.opengis.net/spec/WPS/2.0/req/service/model/result |
| NORMATIVE STATEMENTS | Requirement 49-1<br>Requirement 49-2<br>Requirement 49-3<br>Requirement 49-4<br>Requirement 49-5 |

**Table 45** — Possible responses to an execute request

| | | EXECUTION MODE | | |
|---|---|---|---|---|
| | | SYNC | ASYNC | AUTO |
| Response format | Raw | Raw data | StatusInfo document | Raw data or StatusInfo document (choice is made by server)[a] |
| | document | Response document | StatusInfo document | Response document or StatusInfo document (choice is made by server)[a] |
| | n/a | Response document (default) | StatusInfo document | |

[a] The client identifies the server's choice by inspection of the response type (Table 45).

## 10.9.3. Execute Exceptions

When a WPS server encounters an error while performing an Execute operation, it shall return an exception report as specified in clause 8 of [OGC 06-121r9]. If appropriate, the server shall use additional exception codes as defined in this section.

For synchronous execution, an exception is returned instead of a result. For asynchronous execution, it is recommended to return the exception at the earliest possible time. In the case of a syntactically wrong request (e.g. due to the use of wrong identifiers and data formats), the exception report message may be returned instead of a StatusInfo document. If the exception occurs later during execution, the StatusInfo shall be set to "Failed" and the GetResult operation shall be used to deliver the exception report.

| REQUIREMENTS CLASS 50 | |
|---|---|
| **OBLIGATION** | requirement |
| **TARGET TYPE** | Software implementation |
| **PREREQUISITES** | http://www.opengis.net/spec/WPS/2.0/req/conceptual-model<br>OWS Common 2.0 |
| **NORMATIVE STATEMENTS** | Requirement 50-1<br>Requirement 50-2<br>Requirement 50-3 |

**Table 46** — Additional exception codes for the Execute operation

| EXCEPTIONCODE VALUE | EXCEPTIONTEXT | LOCATOR | HTTP STATUS CODE |
|---|---|---|---|
| NoSuchProcess | One of the identifiers passed does not match with any of the processes offered by this server. | List of violating process identifiers. | 400 (Bad request) |
| NoSuchMode | The process does not permit the desired execution mode. | Violating mode value. | 400 (Bad request) |
| NoSuchInput | One or more of the input identifiers passed does not match with any of the input identifiers of this process. | List of violating input identifiers | 400 (Bad request) |
| NoSuchOutput | One or more of the output identifiers passed does not match with any of the input identifiers of this process. | List of violating input identifiers | 400 (Bad request) |
| DataNotAccessible | One of the referenced input data sets was inaccessible. | List of violating input identifiers | 400 (Bad request) |
| SizeExceeded | The size of one of the input parameters was too large for this process to handle. | List of violating input identifiers | 400 (Bad request) |
| TooManyInputs | Too many input items have been specified. | List of violating input identifiers | 400 (Bad request) |
| TooManyOutputs | Too many output items have been specified.[a] | List of violating input identifiers | 400 (Bad request) |
| ServerBusy | The server is too busy to accept and queue the request at this time. | None (omit locator parameter) | 503 (Service Unavailable) |
| StorageNotSupported | Execute operation request included transmission= ”reference” for one of the outputs, but storage is not offered by this server. | None (omit locator parameter) | 400 (Bad request) |
| NoSuchFormat | One or more of the input or output formats specified in the request did not match with any of the formats defined for that particular input or output. | List of violating input and / or output identifiers | 400 (Bad request) |
| WrongInputData | One or more of inputs for which the service was able to retrieve the data but could not read it. | List of violating input identifiers | 400 (Bad request) |
| InternalServerError | None (omit exception text) | None (omit locator parameter) | 500 (Internal Server Error)[b] |

[a] This shall be used in conjunction with raw data, where the execute request must specify only one output.

[b] If the cause could not be determined or is not covered by the above exception codes, this exception shall be used.

# 10.10. GetStatus Operation

The GetStatus operation allows WPS clients to query the status of an asynchronously executed job.

| REQUIREMENTS CLASS 51 | |
|---|---|
| **OBLIGATION** | requirement |
| **TARGET TYPE** | Software implementation |
| **PREREQUISITES** | http://www.opengis.net/spec/WPS/2.0/req/conceptual-model<br>OWS Common 2.0 |
| **NORMATIVE STATEMENTS** | Requirement 51-1<br>Requirement 51-2<br>Requirement 51-3 |

## 10.10.1. GetStatus Request

The GetStatus request inherits basic properties from the RequestBaseType. It contains an additional element that identifies the JobID of the processing job, of which the status shall be returned.

| REQUIREMENTS CLASS 52 | |
|---|---|
| **OBLIGATION** | requirement |
| **TARGET TYPE** | Software implementation |
| **PREREQUISITES** | http://www.opengis.net/spec/WPS/2.0/req/conceptual-model<br>OWS Common 2.0<br>http://www.opengis.net/spec/WPS/2.0/req/service/model/handling |
| **NORMATIVE STATEMENTS** | Requirement 52-1<br>Requirement 52-2 |

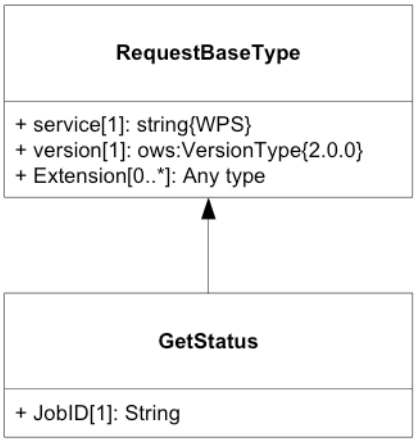


**Figure 22** — GetStatus request UML class diagram

**Table 47** — Additional properties in the GetStatus request

| NAMES | DEFINITION | DATA TYPE AND VALUES | MULTIPLICITY AND USE |
|---|---|---|---|
| JobID | Job identifier. | Character String. This shall be a JobID the client has received during process execution. | One (mandatory) |

## 10.10.2. GetStatus Response

The response to a GetStatus request is a StatusInfo document as defined in Clause 10.5.

| REQUIREMENTS CLASS 53 | |
|---|---|
| OBLIGATION | requirement |
| TARGET TYPE | Software implementation |
| PREREQUISITES | http://www.opengis.net/spec/WPS/2.0/req/conceptual-model<br>http://www.opengis.net/spec/WPS/2.0/req/service/model/status-info |
| NORMATIVE STATEMENT | Requirement 53-1 |

## 10.10.3. GetStatus Exceptions

If a WPS server encounters an error while performing a GetStatus operation, it shall return an exception report as specified in Clause 8 of [OGC 06-121r9]. If the error was encountered due to an invalid process identifier, the server shall respond with the exception code defined in Table 48.

| REQUIREMENTS CLASS 54 | |
|---|---|
| OBLIGATION | requirement |
| TARGET TYPE | Software implementation |
| PREREQUISITES | http://www.opengis.net/spec/WPS/2.0/req/conceptual-model<br>OWS Common 2.0 |
| NORMATIVE STATEMENTS | Requirement 54-1<br>Requirement 54-2 |

**Table 48** — Additional exception codes for the GetStatus operation

| NAMES | DEFINITION | DATA TYPE AND VALUES | MULTIPLICITY AND USE |
|---|---|---|---|
| NoSuch Job | The JobID from the request does not match any of the Jobs running on this server | Violating JobID | 400 (Bad request) |

## 10.11. GetResult Operation

The GetResult operation allows WPS clients to query the result of a finished processing job. It is used in conjunction with asynchronous execution.

| REQUIREMENTS CLASS 55 | |
|---|---|
| OBLIGATION | requirement |
| TARGET TYPE | Derived encoding and software implementation |
| PREREQUISITES | http://www.opengis.net/spec/WPS/2.0/req/conceptual-model<br>OWS Common 2.0 |
| NORMATIVE STATEMENTS | Requirement 55-1<br>Requirement 55-2<br>Requirement 55-3 |

### 10.11.1. GetResult Request

The GetResult request inherits basic properties from the RequestBaseType. It contains an additional element that identifies the JobID of the processing job, of which the result shall be returned.

| REQUIREMENTS CLASS 56 | |
|---|---|
| OBLIGATION | requirement |
| TARGET TYPE | Software implementation |
| PREREQUISITES | http://www.opengis.net/spec/WPS/2.0/req/conceptual-model<br>OWS Common 2.0<br>http://www.opengis.net/spec/WPS/2.0/req/service/model/handling |
| NORMATIVE STATEMENTS | Requirement 56-1<br>Requirement 56-2 |

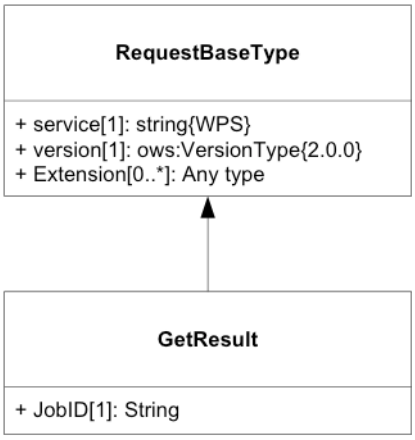


**Figure 23** — GetResult request UML class diagram

**Table 49** — Additional properties in the GetResult request

| NAMES | DEFINITION | DATA TYPE AND VALUES | MULTIPLICITY AND USE |
|---|---|---|---|
| JobID | Job identifier. | Character String. This shall be a JobID the client has received during process execution. | One (mandatory) |

## 10.11.2. GetResult Response

The response to a GetResult request is a Processing Result document as defined in Clause 10.6.

| REQUIREMENTS CLASS 57 | |
|---|---|
| **OBLIGATION** | requirement |
| **TARGET TYPE** | Software implementation |
| **PREREQUISITES** | http://www.opengis.net/spec/WPS/2.0/req/conceptual-model<br>http://www.opengis.net/spec/WPS/2.0/req/service/model/processing-result |
| **NORMATIVE STATEMENT** | Requirement 57-1 |

## 10.11.3. GetResult Exceptions

When a WPS server encounters an error while performing a GetResult operation, it shall return an exception report as specified in Clause 8 of [OGC 06-121r9]. If the error was encountered due to an invalid JobID, the server shall respond with the exception code defined in Table 48 (NoSuchJob). If, for some reason, GetResult was invoked too early and the results have not been computed yet, the server shall respond with the exception defined in Table 50 (ResultNotReady).

If a job finishes with status “failed”, GetResult will provide the corresponding exception report. In this case, the appropriate exception codes defined for the execute operation shall be used.

| REQUIREMENTS CLASS 58 | |
|---|---|
| **OBLIGATION** | requirement |
| **TARGET TYPE** | Software implementation |
| **PREREQUISITES** | http://www.opengis.net/spec/WPS/2.0/req/conceptual-model<br>OWS Common 2.0 |
| **NORMATIVE STATEMENTS** | Requirement 58-1<br>Requirement 58-2<br>Requirement 58-3<br>Requirement 58-4 |

**Table 50** — Additional exception codes for the GetResult operation

| NAMES | DEFINITION | DATA TYPE AND VALUES | MULTIPLICITY AND USE |
|---|---|---|---|
| Result NotReady[a] | The result for the requested JobID has not yet been generated. | Violating JobID | 400 (Bad request) |

[a] Typically, this exception is thrown if a client violates the asynchronous protocol (Figure 4) and calls GetResult before the job execution has completed.

## 10.12. Synchronous WPS

The synchronous WPS is a requirements class profile that indicates the general availability of synchronous execution capabilities on a WPS server.

| REQUIREMENTS CLASS 59 | |
|---|---|
| OBLIGATION | requirement |
| TARGET TYPE | Derived encoding and software implementation |
| PREREQUISITES | http://www.opengis.net/spec/WPS/2.0/req/service/model/get-capabilities<br>http://www.opengis.net/spec/WPS/2.0/req/service/model/describe-process<br>http://www.opengis.net/spec/WPS/2.0/req/service/model/execute |
| NORMATIVE STATEMENTS | Requirement 59-1<br>Requirement 59-2 |

## 10.13. Asynchronous WPS

The asynchronous WPS is a requirements class that indicates the general availability of asynchronous execution capabilities on a WPS server.

| REQUIREMENTS CLASS 60 | |
|---|---|
| OBLIGATION | requirement |
| TARGET TYPE | Derived encoding and software implementation |

| REQUIREMENTS CLASS 60 | |
|---|---|
| **PREREQUISITES** | http://www.opengis.net/spec/WPS/2.0/req/service/model/get-capabilities<br>http://www.opengis.net/spec/WPS/2.0/req/service/model/describe-process<br>http://www.opengis.net/spec/WPS/2.0/req/service/model/execute<br>http://www.opengis.net/spec/WPS/2.0/req/service/model/get-status<br>http://www.opengis.net/spec/WPS/2.0/req/service/model/get-result |
| **NORMATIVE STATEMENTS** | Requirement 60-1<br>Requirement 60-2<br>Requirement 60-3 |

11

# BINDING EXTENSIONS FOR WPS OPERATIONS

# 11 BINDING EXTENSIONS FOR WPS OPERATIONS

## 11.1. HTTP/POST + XML Binding

This section specifies the encoding of WPS requests using HTTP/POST and XML. The referred XML schema elements are provided by the associated schema package delivered with this standard and located at http://schemas.opengis.net/wps/2.0/.

| REQUIREMENTS CLASS 61 | |
|---|---|
| OBLIGATION | requirement |
| TARGET TYPE | Software implementation |
| PREREQUISITES | http://www.opengis.net/spec/WPS/2.0/req/service/model<br>OWS Common 2.0 |
| NORMATIVE STATEMENTS | Requirement 61-1<br>Requirement 61-2<br>Requirement 61-3<br>Requirement 61-4<br>Requirement 61-5<br>Requirement 61-6<br>Requirement 61-7 |

### 11.1.1. GetCapabilities

This clause specifies the XML encoding for the GetCapabilities operation.

A GetCapabilities example is listed in Annex B.4.

| REQUIREMENTS CLASS 62 | |
|---|---|
| OBLIGATION | requirement |
| TARGET TYPE | Software implementation |

| REQUIREMENTS CLASS 62 | |
|---|---|
| PREREQUISITES | http://www.opengis.net/spec/WPS/2.0/req/service/model/get-capabilities<br>OWS Common 2.0 |
| NORMATIVE STATEMENTS | Requirement 62-1<br>Requirement 62-2 |

### 11.1.2. DescribeProcess

This clause specifies the XML encoding for the DescribeProcess operation.

A DescribeProcess example is listed in Annex B.5.

| REQUIREMENTS CLASS 63 | |
|---|---|
| OBLIGATION | requirement |
| TARGET TYPE | Software implementation |
| PREREQUISITES | http://www.opengis.net/spec/WPS/2.0/req/service/model/describe-process<br>OWS Common 2.0 |
| NORMATIVE STATEMENTS | Requirement 63-1<br>Requirement 63-2 |

### 11.1.3. Execute

This clause specifies the XML encoding for the Execute operation.

An Execute example is listed in Annex B.6.

| REQUIREMENTS CLASS 64 | |
|---|---|
| OBLIGATION | requirement |
| TARGET TYPE | Software implementation |
| PREREQUISITES | http://www.opengis.net/spec/WPS/2.0/req/service/model/execute<br>OWS Common 2.0 |
| NORMATIVE STATEMENTS | Requirement 64-1<br>Requirement 64-2 |

### 11.1.4. GetStatus

This clause specifies the XML encoding for the GetStatus operation.

A GetStatus example is listed in Annex B.7.

| REQUIREMENTS CLASS 65 | |
|---|---|
| OBLIGATION | requirement |
| TARGET TYPE | Software implementation |
| PREREQUISITES | http://www.opengis.net/spec/WPS/2.0/req/service/model/get-status<br>OWS Common 2.0 |
| NORMATIVE STATEMENTS | Requirement 65-1<br>Requirement 65-2 |

### 11.1.5. GetResult

This clause specifies the XML encoding for the GetResult operation.

A GetResult example is listed in Annex B.8.

| REQUIREMENTS CLASS 66 | |
|---|---|
| OBLIGATION | requirement |
| TARGET TYPE | Software implementation |
| PREREQUISITES | http://www.opengis.net/spec/WPS/2.0/req/service/model/get-result<br>OWS Common 2.0 |
| NORMATIVE STATEMENTS | Requirement 66-1<br>Requirement 66-2 |

## 11.2. HTTP/GET + KVP Binding

This section specifies the encoding of WPS requests using HTTP/GET and KVP. Since different HTTP client and server libraries may restrict the size of the URL this binding should only be

used if the resulting HTTP/GET URL is reasonably short. The HTTP/GET + KVP binding is only defined for the operations GetCapabilities, DescribeProcess, GetStatus, and GetResult.[8].

Any WPS request in the HTTP/GET + KVP binding consists of a URL with KVP parameters. The response is always an XML document defined for the HTTP/POST + XML binding.

| REQUIREMENTS CLASS 67 | |
|---|---|
| **OBLIGATION** | requirement |
| **TARGET TYPE** | Software implementation |
| **PREREQUISITES** | http://www.opengis.net/spec/WPS/2.0/req/service/model<br>http://www.opengis.net/spec/WPS/2.0/req/service/binding/post-xml<br>OWS Common 2.0 |
| **NORMATIVE STATEMENTS** | Requirement 67-1<br>Requirement 67-2<br>Requirement 67-3<br>Requirement 67-4<br>Requirement 67-5<br>Requirement 67-6 |

### 11.2.1. GetCapabilities

This clause specifies the KVP encoding for the GetCapabilities operation. A minimalist GetCapabilities request might look like this:

http://hostname:port/path?service=WPS&request=GetCapabilities

| REQUIREMENTS CLASS 68 | |
|---|---|
| **OBLIGATION** | requirement |
| **TARGET TYPE** | Software implementation |
| **PREREQUISITES** | http://www.opengis.net/spec/WPS/2.0/req/service/model/get-capabilities<br>OWS Common 2.0 |
| **NORMATIVE STATEMENTS** | Requirement 68-1<br>Requirement 68-2 |

[8]There is no HTTP/GET + KVP binding for execute requests due to the complex and nested structure of the request.

## 11.2.2. DescribeProcess

This clause specifies the KVP encoding for the DescribeProcess operation. Possible DescribeProcess requests might look like these:

http://hostname:port/path?
service=WPS&version=2.0.0&request=DescribeProcess&identifier=buffer,viewshed

http://hostname:port/path?
service=WPS&version=2.0.0&request=DescribeProcess&identifier=ALL

| REQUIREMENTS CLASS 69 | |
|---|---|
| **OBLIGATION** | requirement |
| **TARGET TYPE** | Software implementation |
| **PREREQUISITES** | http://www.opengis.net/spec/WPS/2.0/req/service/model/describe-process<br>OWS Common 2.0 |
| **NORMATIVE STATEMENTS** | Requirement 69-1<br>Requirement 69-2 |

**Table 51** — DescribeProcess request KVP encoding

| NAMES | DEFINITION | DATA TYPE AND VALUES | MULTIPLICITY AND USE |
|---|---|---|---|
| service | Identifier of the OGC service. | String, fixed to “WPS” | One (mandatory) |
| version | Request protocol version. | String, 2.0.0 | One (mandatory) |
| request | Request name. | String, fixed to “DescribeProcess”, case insensitive. | One (mandatory) |
| lang | Desired language of the process description. | IETF RFC 4646 language code of the human-readable text elements in the process description (e.g. “en”). This shall be one of the languages defined in the Capabilities document. If the client specifies more than one language, the server may create a response in any of these languages. | Zero or one (optional) |
| identifier | List of one or more process identifiers. | Comma-separated list of URI escaped character strings.[a] These shall be one or more of the process identifiers listed in the Process Summary elements in the Capabilities document. The fixed | One (mandatory) |

| NAMES | DEFINITION | DATA TYPE AND VALUES | MULTIPLICITY AND USE |
|---|---|---|---|
| | | case insensitive value “ALL” indicates that the description of all processes shall be returned. | |

[a] The general syntax for URIs and URI escaping are defined in IETF RFC 3986.

### 11.2.3. GetStatus

This clause specifies the KVP encoding for the GetStatus operation. A possible GetStatus request might look like this:

http://hostname:port/path?
service=WPS&version=2.0.0&request=GetStatus&jobid=FB6DD4B0-A2BB-11E3-A5E2-0800200C9A66

| REQUIREMENTS CLASS 70 | |
|---|---|
| OBLIGATION | requirement |
| TARGET TYPE | Software implementation |
| PREREQUISITES | http://www.opengis.net/spec/WPS/2.0/req/service/model/get-status<br>OWS Common 2.0 |
| NORMATIVE STATEMENTS | Requirement 70-1<br>Requirement 70-2 |

**Table 52** — GetStatus request KVP encoding

| NAMES | DEFINITION | DATA TYPE AND VALUES | MULTIPLICITY AND USE |
|---|---|---|---|
| service | Identifier of the OGC service. | String, fixed to “WPS” | One (mandatory) |
| version | Request protocol version. | String, 2.0.0 | One (mandatory) |
| request | Request name. | String, fixed to “GetStatus”, case insensitive. | One (mandatory) |
| jobid | Job identifier. | Character String. This shall be a JobID the client has received during process execution. | One (mandatory) |

### 11.2.4. GetResult

This clause specifies the KVP encoding for the GetResult operation. A possible GetResult request might look like this:

http://hostname:port/path?service=WPS&version=2.0.0&request=GetResult&jobid=FB6DD4B0-A2BB-11E3-A5E2-0800200C9A66

| REQUIREMENTS CLASS 71 | |
|---|---|
| OBLIGATION | requirement |
| TARGET TYPE | Software implementation |
| PREREQUISITES | http://www.opengis.net/spec/WPS/2.0/req/service/model/get-result<br>OWS Common 2.0 |
| NORMATIVE STATEMENTS | Requirement 71-1<br>Requirement 71-2 |

**Table 53** — GetResult request KVP encoding

| NAMES | DEFINITION | DATA TYPE AND VALUES | MULTIPLICITY AND USE |
|---|---|---|---|
| service | Identifier of the OGC service. | String, fixed to “WPS” | One (mandatory) |
| version | Request protocol version. | String, 2.0.0 | One (mandatory) |
| request | Request name. | String, fixed to “GetResult”, case insensitive. | One (mandatory) |
| jobid | Job identifier. | Character String. This shall be a JobID the client has received during process execution. | One (mandatory) |

12

# SERVICE PROFILES

# 12 SERVICE PROFILES

## 12.1. Basic WPS

The basic WPS is an aggregate requirements class that implements the WPS conceptual model based on the following requirements classes:

1. The WPS Service Model,
2. The binding extensions defined in Clause 10.12, and
3. The native WPS process model

| REQUIREMENTS CLASS 72 | |
|---|---|
| **OBLIGATION** | requirement |
| **TARGET TYPE** | Derived encoding and software implementation |
| **PREREQUISITES** | http://www.opengis.net/spec/WPS/2.0/req/service/model<br>http://www.opengis.net/spec/WPS/2.0/req/native-process/model<br>http://www.opengis.net/spec/WPS/2.0/req/service/binding/post-xml<br>http://www.opengis.net/spec/WPS/2.0/req/service/binding/get-kvp |
| **NORMATIVE STATEMENTS** | Requirement 72-1<br>Requirement 72-2<br>Requirement 72-3 |

13

# DISMISS EXTENSION

# 13 DISMISS EXTENSION

The dismiss extension for WPS allows a client to communicate to the server that he is no longer interested in the results of a job. In this case, the server may free all associated resources and "forget" the JobID (Figure 24). For jobs that are still running, the server may cancel the execution at any time. For jobs that were already finished, the associated status information and the stored results may be deleted without further notice, regardless of the expiration time given in the last status report.

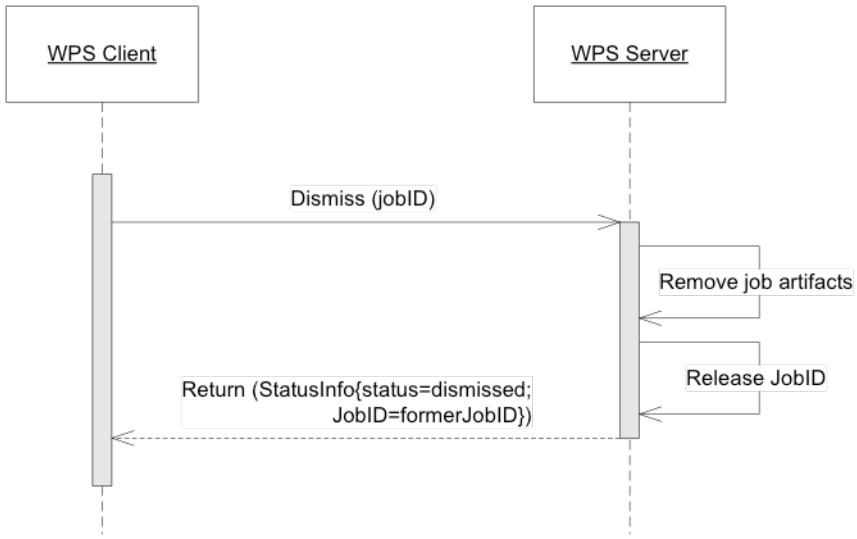


**Figure 24** — Dismiss operation — UML sequence diagram

## 13.1. Dismiss Operation

The dismiss operation is a job control operation. Its availability is indicated per process using the jobControlOptions in the ProcessSummary and the ProcessOffering structures.

| REQUIREMENTS CLASS 73 | |
|---|---|
| OBLIGATION | requirement |
| TARGET TYPE | Derived encoding and software implementation |
| PREREQUISITES | http://www.opengis.net/spec/WPS/2.0/req/conceptual-model<br>OWS Common 2.0 |
| NORMATIVE STATEMENTS | Requirement 73-1<br>Requirement 73-2<br>Requirement 73-3 |

| REQUIREMENTS CLASS 73 | |
|---|---|
| | Requirement 73-4<br>Requirement 73-5<br>Requirement 73-6<br>Requirement 73-7 |

### 13.1.1. Dismiss Request

The Dismiss request inherits basic properties from the RequestBaseType. It contains an additional element that identifies the JobID of the processing job to be dismissed.

| REQUIREMENTS CLASS 74 | |
|---|---|
| OBLIGATION | requirement |
| TARGET TYPE | Derived encoding and software implementation |
| PREREQUISITES | http://www.opengis.net/spec/WPS/2.0/req/conceptual-model<br>OWS Common 2.0<br>http://www.opengis.net/spec/WPS/2.0/req/service/model/handling |
| NORMATIVE STATEMENTS | Requirement 74-1<br>Requirement 74-2 |

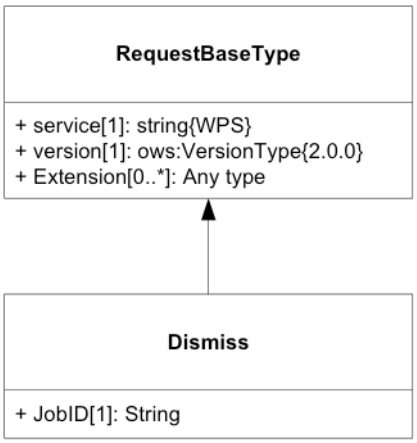


**Figure 25** — Dismiss request UML class diagram

**Table 54** — Additional properties in the Dismiss request

| NAMES | DEFINITION | DATA TYPE AND VALUES | MULTIPLICITY AND USE |
|---|---|---|---|
| JobID | Job identifier. | Character String. This shall be a JobID the client has received during process execution. | One (mandatory) |

### 13.1.2. Dismiss Response

The response to a Dismiss request is a StatusInfo document as defined in Clause 10.5. The status of the job shall be set to “Dismissed”. Subsequent requests to the service using the dismissed jobID shall result in a NoSuchJob exception.

| REQUIREMENTS CLASS 75 | |
|---|---|
| **OBLIGATION** | requirement |
| **TARGET TYPE** | Derived encoding and software implementation |
| **PREREQUISITE** | http://www.opengis.net/spec/WPS/2.0/req/conceptual-model |
| **NORMATIVE STATEMENTS** | Requirement 75-1<br>Requirement 75-2<br>Requirement 75-3 |

### 13.1.3. Dismiss Exceptions

If a WPS server encounters an error while performing a Dismiss operation, it shall return an exception report as specified in Clause 8 of [OGC 06-121r9]. If the error was encountered due to an invalid JobID, the server shall respond with the exception code defined in Table 48 (NoSuchJob).

| REQUIREMENTS CLASS 76 | |
|---|---|
| **OBLIGATION** | requirement |
| **TARGET TYPE** | Derived encoding and software implementation |
| **PREREQUISITES** | http://www.opengis.net/spec/WPS/2.0/req/conceptual-model<br>OWS Common 2.0 |
| **NORMATIVE STATEMENTS** | Requirement 76-1<br>Requirement 76-2 |

# 13.2. Binding Extensions for the Dismiss Operation

## 13.2.1. HTTP/POST + XML Binding

This clause specifies the XML encoding for the Dismiss operation.

A Dismiss example is listed in Annex B.9.

| REQUIREMENTS CLASS 77 | |
|---|---|
| OBLIGATION | requirement |
| TARGET TYPE | Software implementation |
| PREREQUISITES | http://www.opengis.net/spec/WPS/2.0/req/service/model/dismiss<br>http://www.opengis.net/spec/WPS/2.0/req/service/binding/post-xml |
| NORMATIVE STATEMENTS | Requirement 77-1<br>Requirement 77-2<br>Requirement 77-3 |

## 13.2.2. HTTP/GET + KVP Binding

This clause specifies the KVP encoding for the Dismiss operation. A possible Dismiss request might look like this:

http://hostname:port/path?service=WPS&version=2.0.0&request=dismiss&jobid= FB6DD4B0-A2BB-11E3-A5E2-0800200C9A66

| REQUIREMENTS CLASS 78 | |
|---|---|
| OBLIGATION | requirement |
| TARGET TYPE | Software implementation |
| PREREQUISITES | http://www.opengis.net/spec/WPS/2.0/req/service/model/dismiss<br>http://www.opengis.net/spec/WPS/2.0/req/service/binding/get-kvp |
| NORMATIVE STATEMENTS | Requirement 78-1<br>Requirement 78-2<br>Requirement 78-3 |

**Table 55** — Dismiss request KVP encoding

| NAMES | DEFINITION | DATA TYPE AND VALUES | MULTIPLICITY AND USE |
|---|---|---|---|
| service | Identifier of the OGC service. | String, fixed to “WPS” | One (mandatory) |
| version | Request protocol version. | String, 2.0.0 | One (mandatory) |
| request | Request name. | String, fixed to “Dismiss”, case insensitive. | One (mandatory) |
| jobid | Job identifier. | Character String. This shall be a JobID the client has received during process execution. | One (mandatory) |

A

# ANNEX A (NORMATIVE) ABSTRACT TEST SUITE

A

# ANNEX A (NORMATIVE) ABSTRACT TEST SUITE

Tests and requirement identifiers below are relative to **http://www.opengis.net/spec/WPS/2.0**

## A.1. Basic WPS (Conformance Class)

The OGC URI identifier of this conformance class is: **http://www.opengis.net/spec/WPS/2.0/conf/service/profile/basic-wps**

**Example**

| REQUIREMENT A.1 | |
|---|---|
| STATEMENT | conf/service/profile/basic-wps |

| RECOMMENDATION A.1 | |
|---|---|
| STATEMENT | Verify that the server implements the Basic WPS conformance class. |

| REQUIREMENT A.2 | |
|---|---|
| STATEMENT | Verify that the server implements the Synchronous WPS and/or the Asynchronous WPS conformance class. Verify that the requests and responses to a supported operation are syntactically correct. Verify that the service supports the Synchronous WPS Conformance class, the Asynchronous WPS Conformance class or both. Verify that all process offerings implement the native process model. Verify the following list of conformance tests:<br>• Annex A.4.1, Annex A.4.3<br>• Annex A.2 and/or Annex A.3 |

# A.2. Synchronous WPS (Conformance Class)

The OGC URI identifier of this conformance class is: **http://www.opengis.net/spec/WPS/2.0/conf/service/synchronous-wps**

**Example**

| REQUIREMENT A.3 | |
|---|---|
| STATEMENT | conf/service/synchronous-wps |

| REQUIREMENT A.4 | |
|---|---|
| STATEMENT | req/service/model/synchronous-wps/sync-execute |

| REQUIREMENT A.5 | |
|---|---|
| STATEMENT | Verify that the server correctly advertises synchronous execution capabilities. Verify that the server correctly implements all operations that are mandatory for synchronous execution. |

| REQUIREMENT A.6 | |
|---|---|
| STATEMENT | Verify that the GetCapabilities, DescribeProcess and Execute operations appear in the ows: OperationsMetadata element.<br>Verify that the server offers at least one wps:ProcessSummary element whose “jobControlOptions” attribute contains “sync-execute”.<br>Verify that the service supports at least one binding (e.g. HTTP GET/POST) for each supported operation. Verify the following list of conformance tests:<br>• Annex A.5.1, Annex A.5.2, Annex A.5.3, Annex A.5.4, Annex A.5.5, Annex A.5.6, Annex A.5.7, Annex A.5.8<br>• Annex A.5.9 and/or Annex A.5.16<br>• Annex A.5.10 and/or Annex A.5.17<br>• Annex A.5.11, Annex A.5.13. |

# A.3. Asynchronous WPS (Conformance Class)

The OGC URI identifier of this conformance class is: **http://www.opengis.net/spec/WPS/2.0/conf/service/asynchronous-wps**

**Example**

| REQUIREMENT A.7 | |
|---|---|
| STATEMENT | conf/service/asynchronous-wps |

| REQUIREMENT A.8 | |
|---|---|
| STATEMENT | req/service/model/asynchronous-wps |

| REQUIREMENT A.9 | |
|---|---|
| STATEMENT | Verify that the server correctly advertises asynchronous execution capabilities. Verify that the server correctly implements all operations that are mandatory for asynchronous execution. |

| REQUIREMENT A.10 | |
|---|---|
| STATEMENT | Verify that the GetCapabilities, DescribeProcess, Execute, GetStatus and GetResult operations appear in the ows:OperationsMetadata element.<br>Verify that the server offers at least one wps:ProcessSummary element whose “jobControlOptions” attribute contains “async-execute”.<br>Verify that the service supports at least one binding (e.g. HTTP GET/POST) for each supported operation. Verify the following list of conformance tests:<br>• Annex A.5.1, Annex A.5.2, Annex A.5.3, Annex A.5.4, Annex A.5.5, Annex A.5.6, Annex A.5.7, Annex A.5.8<br>• Annex A.5.9 and/or Annex A.5.16<br>• Annex A.5.10 and/or Annex A.5.17<br>• Annex A.5.12, Annex A.5.13<br>• Annex A.5.14 and/or Annex A.5.18<br>• Annex A.5.15 and / or Annex A.5.19 |

# A.4. WPS Process Model Encoding (Conformance Class)

The OGC URI identifier of this conformance class is: **http://www.opengis.net/spec/WPS/2.0/conf/process-model-encoding**

## A.4.1. Process XML Encoding

**Example**

| REQUIREMENT A.11 | |
|---|---|
| **STATEMENT** | conf/process-model-encoding/xml-encoding/process |

| REQUIREMENT A.12 | |
|---|---|
| **STATEMENT** | req/native-process/xml-encoding/process |

| REQUIREMENT A.13 | |
|---|---|
| **STATEMENT** | Verify that a given process description is in compliance with the Process XML encoding. |

| REQUIREMENT A.14 | |
|---|---|
| **STATEMENT** | Verify that the tested document fulfils all requirements listed in req/native-process/xml-encoding/process. |

## A.4.2. Generic Process XML Encoding

**Example**

| REQUIREMENT A.15 | |
|---|---|
| **STATEMENT** | conf/process-model-encoding/xml-encoding/generic-process |

| REQUIREMENT A.16 | |
|---|---|
| STATEMENT | req/native-process/xml-encoding/generic-process |

| REQUIREMENT A.17 | |
|---|---|
| STATEMENT | Verify that a given generic process description is in compliance with the generic process XML encoding. |

| REQUIREMENT A.18 | |
|---|---|
| STATEMENT | Verify that the tested document fulfils all requirements listed in req/native-process/xml-encoding/generic-process. |

## A.4.3. Process data types XML Encoding

### Example

| REQUIREMENT A.19 | |
|---|---|
| STATEMENT | conf/process-model-encoding/xml-encoding/datatypes |

| REQUIREMENT A.20 | |
|---|---|
| STATEMENT | req/native-process/xml-encoding/datatypes |

| REQUIREMENT A.21 | |
|---|---|
| STATEMENT | Verify that any XML data type description and values that are used in conjunction with the native process model are encoded in compliance with the process model XML encoding. |

| REQUIREMENT A.22 | |
|---|---|
| STATEMENT | For **ComplexData** descriptions: Test passes if the tested XML fragment validates against *wps:ComplexData*. For LiteralData descriptions: Test passes if the tested XML fragment validates against wps:LiteralData. For **BoundingBoxData** descriptions: Test passes if the tested XML fragment validates against *wps:BoundingBoxData*. For **ComplexData values**: No general test available; the correctness of complex data values must be tested against a particular data type specification given by mime |

| REQUIREMENT A.22 | |
|---|---|
| | type, encoding and schema. For **LiteralData values**: Test passes if the tested XML fragment validates against wps:LiteralValue. For **BoundingBoxData values**: Test passes if the tested XML fragment validates against *ows:BoundingBox*. |

# A.5. Basic tests

## A.5.1. Request service name

**Example**

| REQUIREMENT A.23 | |
|---|---|
| STATEMENT | **conf/common-wps/handling/service** |

| REQUIREMENT A.24 | |
|---|---|
| STATEMENT | **req/service/model/handling/service** |

| REQUIREMENT A.25 | |
|---|---|
| STATEMENT | Verify that the correctly handles the service name parameter. |

| REQUIREMENT A.26 | |
|---|---|
| STATEMENT | For each request type, send valid requests to server under test. Modulate service parameter:<br>• Parameter value equal to what is required. Verify that request succeeds.<br>• Parameter value not equal to what is required. Verify that request fails. Overall test passes if all individual tests pass. |

## A.5.2. Request version number

**Example**

| REQUIREMENT A.27 | |
|---|---|
| STATEMENT | conf/common-wps/handling/version |

| REQUIREMENT A.28 | |
|---|---|
| STATEMENT | req/service/model/handling/version |

| REQUIREMENT A.29 | |
|---|---|
| STATEMENT | Verify that the correctly handles the service version parameter. |

| REQUIREMENT A.30 | |
|---|---|
| STATEMENT | For each request type, send valid requests to server under test. Modulate the version parameter:<br>• Parameter value equal to what is required. Verify that request succeeds.<br>• Parameter value not equal to what is required. Verify that request fails. Overall test passes if all individual tests pass. |

## A.5.3. Input data transmission by value

### Example

| REQUIREMENT A.31 | |
|---|---|
| STATEMENT | conf/common-wps/data-transmission/input-by-value |

| REQUIREMENT A.32 | |
|---|---|
| STATEMENT | req/conceptual-model/data-transmission/input-by-value |

| REQUIREMENT A.33 | |
|---|---|
| STATEMENT | Verify that the server correctly handles input data transmission by value. |

| REQUIREMENT A.34 | |
|---|---|
| STATEMENT | Send Execute requests to the server under test with valid inputs passed by value. Test passed if the execution finishes successfully. |

## A.5.4. Input data transmission by reference

**Example**

| REQUIREMENT A.35 | |
|---|---|
| STATEMENT | conf/common-wps/input-by-reference |

| REQUIREMENT A.36 | |
|---|---|
| STATEMENT | req/conceptual-model/data-transmission/input-by-reference |

| REQUIREMENT A.37 | |
|---|---|
| STATEMENT | Verify that the server correctly handles input data transmission by reference. |

| REQUIREMENT A.38 | |
|---|---|
| STATEMENT | Send Execute requests to the server under test with valid inputs passed by reference. Test passed if the execution finishes successfully. |

## A.5.5. Output data transmission by value

**Example**

| REQUIREMENT A.39 | |
|---|---|
| STATEMENT | conf/common-wps/data-transmission/output-by-value |

| REQUIREMENT A.40 | |
|---|---|
| STATEMENT | req/conceptual-model/data-transmission/output-by-value |

| REQUIREMENT A.41 | |
|---|---|
| STATEMENT | Verify that the server correctly handles output data transmission by value. |

| REQUIREMENT A.42 | |
|---|---|
| STATEMENT | Check the available process offerings for outputs that can be retrieved by value. If there is an output that can be retrieved by value, send an Execute request to the server requesting the output by value. Test passes if a valid Execute response is returned containing the requested output. Skip this test if no output can be retrieved by value. |

## A.5.6. Output data transmission by reference

### Example

| REQUIREMENT A.43 | |
|---|---|
| STATEMENT | conf/common-wps/data-transmission/output-by-reference |

| REQUIREMENT A.44 | |
|---|---|
| STATEMENT | req/conceptual-model/data-transmission/output-by-reference |

| REQUIREMENT A.45 | |
|---|---|
| STATEMENT | Verify that the server correctly handles output data transmission by reference. |

| REQUIREMENT A.46 | |
|---|---|
| STATEMENT | Check the available process offerings for outputs that can be retrieved by reference. If there is an output that can be retrieved by reference, send an Execute request to the server requesting the |

| REQUIREMENT A.46 | |
|---|---|
| | output by reference. Test passes if a valid Execute response is returned containing a reference to the requested output. Skip this test if no output can be retrieved by reference. |

## A.5.7. Unique process identifier

### Example

| REQUIREMENT A.47 | |
|---|---|
| STATEMENT | conf/common-wps/identifier |

| REQUIREMENT A.48 | |
|---|---|
| STATEMENT | req/conceptual-model/process/identifier |

| REQUIREMENT A.49 | |
|---|---|
| STATEMENT | Verify that each process the server offers has a unique identifier. |

| REQUIREMENT A.50 | |
|---|---|
| STATEMENT | Get all available processes from the server under test. Test passes if all processes have a unique identifier. |

## A.5.8. Unique job identifier

### Example

| REQUIREMENT A.51 | |
|---|---|
| STATEMENT | conf/common-wps/job/identifier |

| REQUIREMENT A.52 | |
|---|---|
| STATEMENT | **req/conceptual-model/job/identifier** |

| REQUIREMENT A.53 | |
|---|---|
| STATEMENT | Verify that the server creates a unique jobID for each job. |

| REQUIREMENT A.54 | |
|---|---|
| STATEMENT | Send more than one asynchronous Execute requests to the server under test. Test passes if the retrieved JobIDs differ from each other. |

## A.5.9. GetCapabilities POST/XML encoding request/response

### Example

| REQUIREMENT A.55 | |
|---|---|
| STATEMENT | **conf/service/binding/post-xml/get-capabilities/request-response** |

| REQUIREMENT A.56 | |
|---|---|
| STATEMENT | **req/service/binding/post-xml/get-capabilities/request req/service/binding/post-xml/get-capabilities/response** |

| REQUIREMENT A.57 | |
|---|---|
| STATEMENT | Verify that the server can handle GetCapabilities requests via POST/XML. |

| REQUIREMENT A.58 | |
|---|---|
| STATEMENT | Send a valid GetCapabilities request to the server under test. Test passes if a valid document of the type *wps:Capabilities* is returned. |

## A.5.10. DescribeProcess POST/XML encoding request/response

**Example**

| REQUIREMENT A.59 | |
|---|---|
| **STATEMENT** | conf/service/binding/post-xml/describe-process/**request-response** |

| REQUIREMENT A.60 | |
|---|---|
| **STATEMENT** | **req/service/binding/post-xml/describe-process/request req/service/binding/post-xml/describe-process/response** |

| REQUIREMENT A.61 | |
|---|---|
| **STATEMENT** | Verify that the server can handle DescribeProcess requests via POST/XML. |

| REQUIREMENT A.62 | |
|---|---|
| **STATEMENT** | Send a valid DescribeProcess request to the server under test. Test passes if a valid document of the type *wps:ProcessOfferings* is returned. |

## A.5.11. Synchronous Execute POST/XML encoding request/response

**Example**

| REQUIREMENT A.63 | |
|---|---|
| **STATEMENT** | conf/service/binding/post-xml/execute-sync/request-response |

| REQUIREMENT A.64 | |
|---|---|
| **STATEMENT** | req/service/binding/post-xml/execute/request req/service/binding/post-xml/execute/response |

**REQUIREMENT A.65**

| | |
|---|---|
| **STATEMENT** | Verify that the server can handle synchronous Execute requests via POST/XML. |

**REQUIREMENT A.66**

| | |
|---|---|
| **STATEMENT** | Send a valid XML Execute request to the server under test, setting the “mode” attribute to “sync”. Modulate the “response” parameter:<br>• Parameter value equal “document”. Verify that a valid Execute *wps:Result* is returned.<br>• Parameter equal to “raw”. Verify that is returned. Overall test passes if all individual tests pass. |

## A.5.12. Asynchronous Execute POST/XML encoding request/response

**Example**

**REQUIREMENT A.67**

| | |
|---|---|
| **STATEMENT** | conf/service/binding/post-xml/execute-async/request-response |

**REQUIREMENT A.68**

| | |
|---|---|
| **STATEMENT** | req/service/binding/post-xml/execute/request req/service/binding/post-xml/execute/response |

**REQUIREMENT A.69**

| | |
|---|---|
| **STATEMENT** | Verify that the server can handle asynchronous Execute requests via POST/XML. |

**REQUIREMENT A.70**

| | |
|---|---|
| **STATEMENT** | Send a valid XML Execute request to the server under test, setting the “mode” attribute to “async”. Test passes if a valid Execute *wps:StatusInfo* document is returned. |

## A.5.13. Auto Execute POST/XML encoding request/response

**Example**

**REQUIREMENT A.71**

| | |
|---|---|
| **STATEMENT** | **conf/service/binding/post-xml/execute-auto/request-response** |

**REQUIREMENT A.72**

| | |
|---|---|
| **STATEMENT** | **req/service/binding/post-xml/execute/request req/service/binding/post-xml/execute/response** |

**REQUIREMENT A.73**

| | |
|---|---|
| **STATEMENT** | Verify that the server can handle the execution mode “auto” requested via POST/XML. |

**REQUIREMENT A.74**

| | |
|---|---|
| **STATEMENT** | Send a valid XML Execute request to the server under test, setting the “mode” attribute to “auto”.<br>Modulate the “response” parameter.<br>1. If the process offering supports document output set “response” parameter value equal “document”. Check the execute response according to the following cases:<br>a) If the process offering supports “sync-execute” and not “async-execute”: Verify that a valid Execute wps:Result document is returned.<br>b) If the process offering supports “async-execute” and not “sync-execute”: Verify that a valid Execute wps:StatusInfo document is returned.<br>c) If the process offering supports “sync-execute” and “async-execute”: Verify that a valid Execute wps:Result document or a valid wps:StatusInfo document is returned.<br>2. If the process offering supports raw output set “response” parameter equal to “raw”. Check the execute response according to the following cases:<br>a) If the process offering supports “sync-execute” and not “async-execute”: Verify that valid that raw data is returned.<br>b) If the process offering supports “async-execute” and not “sync-execute”: Verify that a valid Execute wps:StatusInfo document is returned.<br>c) If the process offering supports “sync-execute” and “async-execute”: Verify that raw data or a valid wps:StatusInfo document is returned. Overall test passes if all individual tests pass. |

## A.5.14. GetStatus POST/XML encoding request/response

### Example

| REQUIREMENT A.75 | |
|---|---|
| STATEMENT | conf/service/binding/post-xml/get-status/request-response |

| REQUIREMENT A.76 | |
|---|---|
| STATEMENT | req/service/binding/post-xml/get-status/request req/service/binding/post-xml/get-status/response |

| REQUIREMENT A.77 | |
|---|---|
| STATEMENT | Verify that the server can handle GetStatus requests via POST/XML. |

| REQUIREMENT A.78 | |
|---|---|
| STATEMENT | Send a valid XML Execute request to the server under test, setting the “mode” attribute to “async”. Verify that a valid *wps:StatusInfo* document is returned. Extract the *wps:JobID*. Send a valid XML Get Status request to the server under test using the extracted JobID. Test passes if a valid *wps:StatusInfo* document is returned. |

## A.5.15. GetResult POST/XML encoding request/response

**Example**

| REQUIREMENT A.79 | |
|---|---|
| STATEMENT | conf/service/binding/get-kvp/describe-process/request-response |

| REQUIREMENT A.80 | |
|---|---|
| STATEMENT | req/service/binding/get-kvp/describe-process/request req/service/binding/get-kvp/describe-process/response |

| REQUIREMENT A.81 | |
|---|---|
| STATEMENT | Verify that the server can handle DescribeProcess requests via GET/KVP. |

| REQUIREMENT A.82 | |
|---|---|
| STATEMENT | Send a valid KVP DescribeProcess request to the server under test, modulating upper and lower case of the parameter names. Test passes if a valid document of the type *wps:ProcessOfferings* is returned. |

## A.5.16. GetCapabilities GET/KVP encoding request/response

**Example**

| REQUIREMENT A.83 | |
|---|---|
| STATEMENT | conf/service/binding/get-kvp/get-capabilities/request-response |

| REQUIREMENT A.84 | |
|---|---|
| STATEMENT | req/service/binding/get-kvp/get-capabilities/request req/service/binding/get-kvp/get-capabilities/response |

| REQUIREMENT A.85 | |
|---|---|
| STATEMENT | Verify that the server can handle GetCapabilities requests via GET/KVP. |

| REQUIREMENT A.86 | |
|---|---|
| STATEMENT | Send a valid KVP GetCapabilities request to the server under test, modulating upper and lower case of the parameter names. Test passes if a valid document of the type *wps:Capabilities* is returned. |

## A.5.17. DescribeProcess GET/KVP encoding request/response

**Example**

| REQUIREMENT A.87 | |
|---|---|
| STATEMENT | conf/service/binding/get-kvp/describe-process/request-response |

| REQUIREMENT A.88 | |
|---|---|
| STATEMENT | req/service/binding/get-kvp/describe-process/request req/service/binding/get-kvp/describe-process/response |

| REQUIREMENT A.89 | |
|---|---|
| STATEMENT | Verify that the server can handle DescribeProcess requests via GET/KVP. |

| REQUIREMENT A.90 | |
|---|---|
| STATEMENT | Send a valid KVP DescribeProcess request to the server under test, modulating upper and lower case of the parameter names. Test passes if a valid document of the type *wps:ProcessOfferings* is returned. |

## A.5.18. GetStatus GET/KVP encoding request/response

### Example

| REQUIREMENT A.91 | |
|---|---|
| STATEMENT | **conf/service/binding/get-kvp/get-status/request-response** |

| REQUIREMENT A.92 | |
|---|---|
| STATEMENT | **req/service/binding/get-kvp/get-status/request req/service/binding/get-kvp/get-status/response** |

| REQUIREMENT A.93 | |
|---|---|
| STATEMENT | Verify that the server can handle GetStatus requests via GET/KVP. |

| REQUIREMENT A.94 | |
|---|---|
| STATEMENT | Send a valid XML Execute request to the server under test, setting the “mode” attribute to “async”. Verify that a valid *wps:StatusInfo* document is returned. Extract the *wps:JobID*. Send a valid KVP Get Status request to the server under test, using the extracted JobID and modulating upper and lower case of the parameter names. Test passes if a valid document of the type *wps:StatusInfo* is returned. |

### A.5.19. GetResult GET/KVP encoding request/response

**Example**

| REQUIREMENT A.95 | |
|---|---|
| STATEMENT | conf/service/binding/get-kvp/get-result/request-response |

| REQUIREMENT A.96 | |
|---|---|
| STATEMENT | req/service/binding/get-kvp/get-result/request req/service/binding/get-kvp/get-result/response |

| REQUIREMENT A.97 | |
|---|---|
| STATEMENT | Verify that the server can handle GetResult requests via GET/KVP. |

| REQUIREMENT A.98 | |
|---|---|
| STATEMENT | Send a valid XML Execute request to the server under test, setting the "mode" attribute to "async". Modulate the "response" parameter. Verify that a valid *wps:StatusInfo* document is returned. Extract the *wps:JobID*. Check the status of the job. If the job succeeded, send a valid KVP GetResult request to the server under test using the extracted JobID and modulating upper and lower case of the parameter names. Depending on the value of the "response" parameter of the above Execute request:<br>• Parameter value equal "document". Verify that a valid Execute *wps:Result* document is returned.<br>• Parameter equal to "raw". Verify that raw is returned. Overall test passes if all individual tests pass. |

## A.6. Dismiss Extension (Conformance Class)

The OGC URI identifier of this conformance class is: **http://www.opengis.net/spec/WPS/2.0/conf/service/dismiss-extension**

### A.6.1. Dismiss POST/XML encoding request/response

**Example**

| REQUIREMENT A.99 | |
|---|---|
| **STATEMENT** | conf/service/binding/post-xml/dismiss/request-response |

| REQUIREMENT A.100 | |
|---|---|
| **STATEMENT** | req/service/binding/post-xml/dismiss/request req/service/binding/post-xml/dismiss/response |

| REQUIREMENT A.101 | |
|---|---|
| **STATEMENT** | Verify that the server can handle Dismiss requests via POST/XML. |

| REQUIREMENT A.102 | |
|---|---|
| **STATEMENT** | Precondition: The process offering used for testing must have “dismiss” listed among its job control options. Send a valid XML Execute request to the server under test, setting the “mode” attribute to “async”. Verify that a valid *wps:StatusInfo* document is returned. Extract the *wps:JobID*. Send a valid XML Dismiss request to the server under test using the extracted JobID. Test passes if a valid *wps:StatusInfo* document is returned containing a *wps:Status* element with value “Dismissed” (case insensitive). |

## A.6.2. Dismiss GET/KVP encoding request/response

### Example

| REQUIREMENT A.103 | |
|---|---|
| **STATEMENT** | conf/service/binding/get-kvp/dismiss/request-response |

| REQUIREMENT A.104 | |
|---|---|
| **STATEMENT** | req/service/binding/get-kvp/dismiss/request req/service/binding/get-kvp/dismiss/response |

## REQUIREMENT A.105

| | |
|---|---|
| **STATEMENT** | Verify that the server can handle Dismiss requests via GET/KVP. |

## REQUIREMENT A.106

| | |
|---|---|
| **STATEMENT** | Precondition: The process offering used for testing must have “dismiss” listed among its job control options. Send a valid XML Execute request to the server under test, setting the “mode” attribute to “async”. Verify that a valid *wps:StatusInfo* document is returned. Extract the *wps:JobID*. Send a valid KVP Dismiss request to the server under test using the extracted JobID and modulating upper and lower case of the parameter names. Test passes if a valid document of the type *wps:StatusInfo* document is returned containing a *wps:Status* element with value “Dismissed” (case insensitive). |

B

# ANNEX B (INFORMATIVE) XML EXAMPLES

B

# ANNEX B (INFORMATIVE) XML EXAMPLES

## B.1. Data Types

### B.1.1. Complex Data Description

```
<wps:ComplexData>
  <wps:Format mimeType="application/geotiff" encoding="raw"
default="true"/>
  <wps:Format mimeType="application/geotiff" encoding="base64"/>
</wps:ComplexData>
```

### B.1.2. Literal Data Description

```
<wps:LiteralData>
  <wps:Format mimeType="text/plain" default="true"/>
  <wps:Format mimeType="text/xml"/>
  <LiteralDataDomain default="true">
    <ows:AllowedValues>
      <ows:Range>
        <ows:MinimumValue>1</ows:MinimumValue>
        <ows:MaximumValue>1000</ows:MaximumValue>
      </ows:Range>
    </ows:AllowedValues>
    <ows:DataType
ows:reference="http://www.w3.org/2001/XMLSchema#float">float
</ows:DataType>
    <ows:UOM>meters</ows:UOM>
    <ows:DefaultValue>100</ows:DefaultValue>
  </LiteralDataDomain>
  <LiteralDataDomain>
    <ows:AllowedValues>
      <ows:Range>
        <ows:MinimumValue>1</ows:MinimumValue>
        <ows:MaximumValue>3000</ows:MaximumValue>
      </ows:Range>
    </ows:AllowedValues>
    <ows:DataType
ows:reference="http://www.w3.org/2001/XMLSchema#float">float
</ows:DataType>
    <ows:UOM>feet</ows:UOM>
  </LiteralDataDomain>
```

```
</wps:LiteralData>
```

### B.1.3. Literal data values

```
<LiteralValue
  dataType=http://www.w3.org/2001/XMLSchema#double
  uom="meter">
    42.1
</LiteralValue>

<LiteralValue
  dataType="http://www.w3.org/2001/XMLSchema#string">
    ArableLand
</LiteralValue>
```

### B.1.4. BoundingBox Data Description

```
<wps:BoundingBoxData>
  <wps:Format mimeType="text/plain" default="true"/>
  <wps:Format mimeType="text/xml"/>
  <wps:SupportedCRS default="true">EPSG:4326</wps:SupportedCRS>
  <wps:SupportedCRS>
    http://www.opengis.net/def/crs/EPSG/0/4258
  </wps:SupportedCRS>
</wps:BoundingBoxData>
```

### B.1.5. BoundingBox Data Values

```
<ows:BoundingBox crs="EPSG:4326">
  <ows:LowerCorner>51.9 7.0</ows:LowerCorner>
  <ows:UpperCorner>53.0 8.0</ows:UpperCorner>
</ows:BoundingBox>

<ows:BoundingBox
   crs="http://www.opengis.net/def/crs/EPSG/0/4258">
  <ows:LowerCorner>51.9 7.0</ows:LowerCorner>
  <ows:UpperCorner>53.0 8.0</ows:UpperCorner>
</ows:BoundingBox>
```

## B.2. Process Description

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.

```
<wps:Process
  xmlns:wps="http://www.opengis.net/wps/2.0"
  xmlns:ows="http://www.opengis.net/ows/2.0"
  xmlns:xlink="http://www.w3.org/1999/xlink"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation=
"http://www.opengis.net/wps/2.0 ../../wps.xsd">
```

```
  <ows:Title>Planar Buffer operation for GML features</ows:Title>
  <ows:Abstract>
  Create a buffer around a GML feature. Accepts any valid GML
  feature and computes the joint buffer.</ows:Abstract>
  <ows:Identifier>
    http://some.host/profileregistry/implementation/Planar-GML-
Buffer
  </ows:Identifier>
  <ows:Metadata
    xlink:role="http://www.opengis.net/spec/wps/2.0/def/
process-profile/concept"
    xlink:href="http://some.host/profileregistry/concept/
buffer"/>
  <ows:Metadata
    xlink:role="http://www.opengis.net/spec/wps/2.0/def/
process-profile/concept"
    xlink:href="http://some.host/profileregistry/concept/
planarbuffer"/>
  <ows:Metadata
    xlink:role="http://www.opengis.net/spec/wps/2.0/def/
process-profile/generic"
    xlink:href="http://some.host/profileregistry/generic/
SF-Buffer"/>
  <ows:Metadata
    xlink:role="http://www.opengis.net/spec/wps/2.0/def/
process/description/documentation"
    xlink:href="http://some.host/profileregistry/implementation/
Planar-GML-Buffer.html"/>
  <wps:Input>
    <ows:Title>Geometry to be buffered</ows:Title>
    <ows:Abstract>Geometry input in GML</ows:Abstract>
    <ows:Identifier>INPUT_GEOMETRY</ows:Identifier>
    <ows:Metadata
      xlink:role="http://www.opengis.net/spec/wps/2.0/def/
process/description/documentation"
      xlink:href="http://some.host/profileregistry/
implementation/Planar-GML-Buffer.html#input_geometry"/>
    <wps:ComplexData>
      <wps:Format mimeType="text/xml" encoding="UTF-8"
        schema="http://schemas.opengis.net/gml/3.2.1/feature.xsd"
        default="true"/>
    </wps:ComplexData>
  </wps:Input>
  <wps:Input>
    <ows:Title>Distance</ows:Title>
    <ows:Abstract>
      Distance to be used to calculate buffer.
    </ows:Abstract>
    <ows:Identifier>DISTANCE</ows:Identifier>
    <ows:Metadata
      xlink:role="http://www.opengis.net/spec/wps/2.0/def/
process/description/documentation"
      xlink:href="http://some.host/profileregistry/
implementation/Planar-GML-Buffer.html#distance"/>
    <wps:LiteralData>
      <wps:Format mimeType="text/plain" default="true"/>
      <wps:Format mimeType="text/xml"/>
      <LiteralDataDomain default="true">
        <ows:AllowedValues>
          <ows:Range>
            <ows:MinimumValue>-INF</ows:MinimumValue>
            <ows:MaximumValue>INF</ows:MaximumValue>
```

```
          </ows:Range>
        </ows:AllowedValues>
        <ows:DataType
          ows:reference="http://www.w3.org/2001/
XMLSchema#double">Double</ows:DataType>
      </LiteralDataDomain>
    </wps:LiteralData>
  </wps:Input>
  <wps:Output>
    <ows:Title>Buffered Geometry</ows:Title>
    <ows:Abstract>
      GML stream describing the buffered Geometry.</ows:Abstract>
    <ows:Identifier>BUFFERED_GEOMETRY</ows:Identifier>
    <ows:Metadata
      xlink:role="http://www.opengis.net/spec/wps/2.0/def/
process/description/documentation"
      xlink:href="http://some.host/profileregistry/
implementation/Planar-GML-Buffer.html#buffered_geometry"/>
    <wps:ComplexData>
      <wps:Format mimeType="text/xml" encoding="UTF-8"
        schema="http://schemas.opengis.net/gml/3.2.1/feature.xsd"
        default="true"/>
    </wps:ComplexData>
  </wps:Output>
</wps:Process>
```

## B.3. Generic Process

This example describes a generic profile for a simple features buffer. It returns a geometry that represents all points whose distance from this Geometry is less than or equal to distance. Calculations are in the Spatial Reference System of this Geometry.

```
<wps:GenericProcess
  xmlns:ows="http://www.opengis.net/ows/2.0"
  xmlns:wps="http://www.opengis.net/wps/2.0"
  xmlns:xlink="http://www.w3.org/1999/xlink"
  xmlns:xml="http://www.w3.org/XML/1998/namespace"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.opengis.net/wps/2.0 ../../wps.xsd">
  <ows:Title>Simple Features Buffer</ows:Title>
  <ows:Identifier>http://some.host/profileregistry/generic/
    SF-Buffer</ows:Identifier>
  <ows:Metadata
    xlink:role="http://www.opengis.net/spec/wps/2.0/def/
process-profile/concept"
    xlink:href="http://some.host/profileregistry/
concept/buffer"/>
  <ows:Metadata
    xlink:role="http://www.opengis.net/spec/wps/2.0/def/
process-profile/concept"
    xlink:href="http://some.host/profileregistry/
concept/planarbuffer"/>

  <ows:Metadata
    xlink:role="http://www.opengis.net/spec/wps/2.0/def/
process/description/documentation"
    xlink:href="http://some.host/profileregistry/
```

```
generic/SF-Buffer.html"/>
  <wps:Input>
    <ows:Title>Input Geometry</ows:Title>
    <ows:Identifier>INPUT_GEOMETRY</ows:Identifier>
    <ows:Metadata
      xlink:role="http://www.opengis.net/spec/wps/2.0/def/
process/description/documentation"
      xlink:href="http://some.host/profileregistry/
generic/SF-Buffer.html#input_geometry"/>
  </wps:Input>
  <wps:Input>
    <ows:Title>Distance</ows:Title>
    <ows:Identifier>DISTANCE</ows:Identifier>
    <ows:Metadata
      xlink:role="http://www.opengis.net/spec/wps/2.0/def/
process/description/documentation"
      xlink:href="http://some.host/profileregistry/
generic/SF-Buffer.html#distance"/>
  </wps:Input>
  <wps:Output>
    <ows:Title>Buffered Geometry</ows:Title>
    <ows:Identifier>BUFFERED_GEOMETRY</ows:Identifier>
    <ows:Metadata
      xlink:role="http://www.opengis.net/spec/wps/2.0/def/
process/description/documentation"
      xlink:href="http://some.host/profileregistry/
generic/SF-Buffer.html#buffered_geometry"/>
  </wps:Output>
</wps:GenericProcess>
```

## B.4. GetCapabilities

### B.4.1. GetCapabilities Request

```
<wps:GetCapabilities service="WPS"
  xmlns:ows="http://www.opengis.net/ows/2.0"
  xmlns:wps="http://www.opengis.net/wps/2.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.opengis.net/wps/2.0 ../wps.xsd">
</wps:GetCapabilities>
```

### B.4.2. GetCapabilities Response

```
<wps:Capabilities service="WPS" version="2.0.0"
  xmlns:ows="http://www.opengis.net/ows/2.0"
  xmlns:wps="http://www.opengis.net/wps/2.0"
  xmlns:xlink="http://www.w3.org/1999/xlink"
  xmlns:xml="http://www.w3.org/XML/1998/namespace"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.opengis.net/wps/2.0 ../wps.xsd ">
  <ows:ServiceIdentification>
    <ows:Title>MyWebProcessingService</ows:Title>
    <ows:Abstract>
```

```
      A Web Processing Service offering typical GIS distance
      transform processes.
    </ows:Abstract>
    <ows:Keywords>
      <ows:Keyword>Geoprocessing</ows:Keyword>
      <ows:Keyword>Toolbox</ows:Keyword>
      <ows:Keyword>Distance transform</ows:Keyword>
    </ows:Keywords>
    <ows:ServiceType>WPS</ows:ServiceType>
    <ows:ServiceTypeVersion>2.0.0</ows:ServiceTypeVersion>
    <ows:Fees>NONE</ows:Fees>
    <ows:AccessConstraints>NONE</ows:AccessConstraints>
  </ows:ServiceIdentification>
  <ows:ServiceProvider>
    <ows:ProviderName>TU Dresden</ows:ProviderName>
    <ows:ProviderSite
xlink:href="http://tu-dresden.de/geo/gis" />
    <ows:ServiceContact>
      <ows:IndividualName>Matthias Mueller</ows:IndividualName>
      <ows:ContactInfo>
        <ows:Address>
          <ows:ElectronicMailAddress>
            matthias_mueller@tu-dresden.de
          </ows:ElectronicMailAddress>
        </ows:Address>
      </ows:ContactInfo>
    </ows:ServiceContact>
  </ows:ServiceProvider>
  <ows:OperationsMetadata>
    <ows:Operation name="GetCapabilities">
      <ows:DCP>
        <ows:HTTP>
          <ows:Get
          xlink:href="http://wps1.gis.geo.tu-dresden.de/wps"/>
        </ows:HTTP>
      </ows:DCP>
    </ows:Operation>
    <ows:Operation name="DescribeProcess">
      <ows:DCP>
        <ows:HTTP>
          <ows:Get
          xlink:href="http://wps1.gis.geo.tu-dresden.de/wps"/>
          <ows:Post
          xlink:href="http://wps1.gis.geo.tu-dresden.de/wps"/>
        </ows:HTTP>
      </ows:DCP>
    </ows:Operation>
    <ows:Operation name="Execute">
      <ows:DCP>
        <ows:HTTP>
          <ows:Post
          xlink:href="http://wps1.gis.geo.tu-dresden.de/wps"/>
        </ows:HTTP>
      </ows:DCP>
    </ows:Operation>
    <ows:Operation name="GetStatus">
      <ows:DCP>
        <ows:HTTP>
          <ows:Get
          xlink:href="http://wps1.gis.geo.tu-dresden.de/wps"/>
          <ows:Post
          xlink:href="http://wps1.gis.geo.tu-dresden.de/wps"/>
        </ows:HTTP>
```

```
      </ows:DCP>
    </ows:Operation>
    <ows:Operation name="GetResult">
      <ows:DCP>
        <ows:HTTP>
          <ows:Get
          xlink:href="http://wps1.gis.geo.tu-dresden.de/wps"/>
          <ows:Post
          xlink:href="http://wps1.gis.geo.tu-dresden.de/wps"/>
        </ows:HTTP>
      </ows:DCP>
    </ows:Operation>
    <ows:Operation name="Dismiss">
      <ows:DCP>
        <ows:HTTP>
          <ows:Get
          xlink:href="http://wps1.gis.geo.tu-dresden.de/wps"/>
          <ows:Post
          xlink:href="http://wps1.gis.geo.tu-dresden.de/wps"/>
        </ows:HTTP>
      </ows:DCP>
    </ows:Operation>
  </ows:OperationsMetadata>
  <wps:Contents>
    <wps:ProcessSummary
jobControlOptions="sync-execute async-execute dismiss">
      <ows:Title>Euclidean Distance</ows:Title>
      <ows:Identifier>
        http://my.site/distance-transform/euclidean-distance
      </ows:Identifier>
    </wps:ProcessSummary>
    <wps:ProcessSummary
jobControlOptions="async-execute dismiss">
      processVersion="1.4.0">
      <ows:Title>Cost Distance</ows:Title>
      <ows:Identifier>
        http://my.site/distance-transform/cost-distance
      </ows:Identifier>
    </wps:ProcessSummary>
  </wps:Contents>
</wps:Capabilities>
```

**Figure B.1**

## B.5. DescribeProcess

### B.5.1. DescribeProcess Request

```
<wps:DescribeProcess service="WPS" version="2.0.0"
  xmlns:ows="http://www.opengis.net/ows/2.0"
  xmlns:wps="http://www.opengis.net/wps/2.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.opengis.net/wps/2.0 ../wps.xsd">
  <ows:Identifier>Buffer</ows:Identifier>
  <ows:Identifier>Viewshed</ows:Identifier>
```

```
</wps:DescribeProcess>
```

### B.5.2. DescribeProcess Response

```
<wps:ProcessOfferings xmlns:wps="http://www.opengis.net/wps/2.0.0"
  xmlns:ows=http://www.opengis.net/ows/2.0
  xmlns:xlink="http://www.w3.org/1999/xlink"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.opengis.net/wps/2.0 ../wps.xsd">
  <wps:ProcessOffering
jobControlOptions="sync-execute async-execute dismiss"
outputTransmission="value reference">
    <wps:Process>
      <ows:Title>
        Planar Buffer operation for Simple Features
      </ows:Title>
      <ows:Abstract>
        Create a buffer around Simple Feature geometries. Accepts
        any valid simple features input in GML or GeoJson and
        computes their joint buffer geometry.
      </ows:Abstract>
      <ows:Identifier>
        http://my.wps.server/processes/proximity/Planar-Buffer
      </ows:Identifier>
      <ows:Metadata
        xlink:role="http://www.opengis.net/spec/wps/2.0/
def/process-profile/concept"
        xlink:href="http://some.host/profileregistry/
concept/buffer"/>
      <ows:Metadata
        xlink:role="http://www.opengis.net/spec/wps/2.0/
def/process-profile/concept"
        xlink:href="http://some.host/profileregistry/
concept/planarbuffer"/>
      <ows:Metadata
        xlink:role="http://www.opengis.net/spec/wps/2.0/
def/process-profile/generic"
        xlink:href="http://some.host/profileregistry/
generic/SF-Buffer"/>
      <ows:Metadata
        xlink:role="http://www.opengis.net/spec/wps/2.0/
def/process/description/documentation"
        xlink:href="http://my.wps.server/processes/
proximity/Planar-Buffer.html"/>
      <wps:Input>
        <ows:Title>Geometry to be buffered</ows:Title>
        <ows:Abstract>
          Simple Features geometry input in GML or GeoJson
        </ows:Abstract>
        <ows:Identifier>INPUT_GEOMETRY</ows:Identifier>
        <ows:Metadata
          xlink:role="http://www.opengis.net/spec/wps/2.0/
def/process/description/documentation"
          xlink:href="http://my.wps.server/processes/
proximity/Planar-Buffer.html#input_geometry"/>
        <wps:ComplexData>
          <wps:Format mimeType="text/xml" encoding="UTF-8"
            schema="http://schemas.opengis.net/gml/
3.2.1/feature.xsd" default="true"/>
          <wps:Format mimeType="application/json"
encoding="UTF-8"/>
```

```
        </wps:ComplexData>
      </wps:Input>
      <wps:Input>
        <ows:Title>Distance</ows:Title>
        <ows:Abstract>
           Distance to be used to calculate buffer.
        </ows:Abstract>
        <ows:Identifier>DISTANCE</ows:Identifier>
        <ows:Metadata
          xlink:role="http://www.opengis.net/spec/wps/2.0/
def/process/description/documentation"
          xlink:href="http://my.wps.server/processes/
proximity/Planar-Buffer.html#distance"/>
        <wps:LiteralData>
          <wps:Format mimeType="text/plain" default="true"/>
          <wps:Format mimeType="text/xml"/>
          <LiteralDataDomain default="true">
            <ows:AllowedValues>
              <ows:Range>
                <ows:MinimumValue>-INF</ows:MinimumValue>
                <ows:MaximumValue>INF</ows:MaximumValue>
              </ows:Range>
            </ows:AllowedValues>
            <ows:DataType
              ows:reference="http://www.w3.org/2001/
XMLSchema#double">
              Double
            </ows:DataType>
          </LiteralDataDomain>
        </wps:LiteralData>
      </wps:Input>
      <wps:Output>
        <ows:Title>Buffered Geometry</ows:Title>
        <ows:Abstract>
          Output Geometry in GML or GeoJson
        </ows:Abstract>
        <ows:Identifier>BUFFERED_GEOMETRY</ows:Identifier>
        <ows:Metadata
          xlink:role="http://www.opengis.net/spec/wps/2.0/
def/process/description/documentation"
          xlink:href="http://my.wps.server/processes/
proximity/Planar-Buffer.html#buffered_geometry"/>
        <wps:ComplexData>
          <wps:Format mimeType="text/xml" encoding="UTF-8"
            schema="http://schemas.opengis.net/gml/
3.2.1/feature.xsd" default="true"/>
          <wps:Format mimeType="application/json"
encoding="UTF-8"/>
        </wps:ComplexData>
      </wps:Output>
    </wps:Process>
  </wps:ProcessOffering>
</wps:ProcessOfferings>
```

**Figure B.2**

# B.6. Execute

## B.6.1. Execute Request (asynchronous, result document)

```
<wps:Execute
  xmlns:wps="http://www.opengis.net/wps/2.0"
  xmlns:ows="http://www.opengis.net/ows/2.0"
  xmlns:xlink="http://www.w3.org/1999/xlink"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.opengis.net/wps/2.0 ../wps.xsd"
  service="WPS" version="2.0.0" response="document" mode="async">
  <ows:Identifier>
    http://my.wps.server/processes/proximity/Planar-Buffer
  </ows:Identifier>
  <wps:Input id="INPUT_GEOMETRY">
    <wps:Reference xlink:href="http://some.data.server/
mygmldata.xml"/>
  </wps:Input>
  <wps:Input id="DISTANCE">
    <wps:Data>10</wps:Data>
  </wps:Input>
  <!– Uses default output format –>
  <wps:Output id="BUFFERED_GEOMETRY"
wps:dataTransmissionMode="reference">
  </wps:Output>
</wps:Execute>
```

## B.6.2. Execute Response (StatusInfo)

```
<wps:StatusInfo xmlns:ows="http://www.opengis.net/ows/2.0"
  xmlns:wps="http://www.opengis.net/wps/2.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.opengis.net/wps/2.0 ../wps.xsd">
  <wps:JobID>FB6DD4B0-A2BB-11E3-A5E2-0800200C9A66</wps:JobID>
  <wps:Status>Accepted</wps:Status>
  <wps:NextPoll>2014-12-24T16:00:00Z</wps:NextPoll>
</wps:StatusInfo>
```

## B.6.3. Execute Response (Result)

```
<wps:Result
  xmlns:ows="http://www.opengis.net/ows/2.0"
  xmlns:wps="http://www.opengis.net/wps/2.0"
  xmlns:xlink="http://www.w3.org/1999/xlink"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.opengis.net/wps/2.0 ../wps.xsd ">
  <wps:JobID>FB6DD4B0-A2BB-11E3-A5E2-0800200C9A66</wps:JobID>
  <wps:ExpirationDate>2014-12-24T24:00:00Z</wps:ExpirationDate>
  <wps:Output id="BUFFERED_GEOMETRY">
    <wps:Reference
      xlink:href="http://result.data.server/
FB6DD4B0-A2BB-11E3-A5E2-0800200C9A66/
BUFFERED_GEOMETRY.xml"/>
```

```
  </wps:Output>
</wps:Result>
```

# B.7. GetStatus

## B.7.1. GetStatus Request

```
<wps:GetStatus service="WPS" version="2.0.0"
  xmlns:wps="http://www.opengis.net/wps/2.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.opengis.net/wps/2.0 ../wps.xsd ">
  <wps:JobID>FB6DD4B0-A2BB-11E3-A5E2-0800200C9A66</wps:JobID>
</wps:GetStatus>
```

## B.7.2. GetStatus Response

```
<wps:StatusInfo xmlns:ows="http://www.opengis.net/ows/2.0"
  xmlns:wps="http://www.opengis.net/wps/2.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.opengis.net/wps/2.0 ../wps.xsd">
  <wps:JobID>FB6DD4B0-A2BB-11E3-A5E2-0800200C9A66</wps:JobID>
  <wps:Status>Running</wps:Status>
  <wps:NextPoll>2014-12-24T16:00:00Z</wps:NextPoll>
</wps:StatusInfo>
```

# B.8. GetResult

## B.8.1. GetResult Request

```
<wps:GetResult service="WPS" version="2.0.0"
  xmlns:wps="http://www.opengis.net/wps/2.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.opengis.net/wps/2.0 ../wps.xsd ">
  <wps:JobID>FB6DD4B0-A2BB-11E3-A5E2-0800200C9A66</wps:JobID>
</wps:GetResult>
```

## B.8.2. GetResult Response

See Annex B.6.3.

# B.9. Dismiss

## B.9.1. Dismiss Request

```
<wps:Dismiss service="WPS" version="2.0.0"
  xmlns:wps="http://www.opengis.net/wps/2.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.opengis.net/wps/2.0 ../wps.xsd ">
  <wps:JobID>FB6DD4B0-A2BB-11E3-A5E2-0800200C9A66</wps:JobID>
</wps:Dismiss>
```

## B.9.2. Dismiss Response

```
<wps:StatusInfo xmlns:ows="http://www.opengis.net/ows/2.0"
  xmlns:wps="http://www.opengis.net/wps/2.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.opengis.net/wps/2.0 ../wps.xsd">
  <wps:JobID>FB6DD4B0-A2BB-11E3-A5E2-0800200C9A66</wps:JobID>
  <wps:Status>Dismissed</wps:Status>
</wps:StatusInfo>
```

# B.10. Profile inheritance example

Taking the example of a simple Mosaic process, Figure B.3 and Figure B.4 illustrate the inheritance rules for process profiles provided in Clause 8.5.4. In its most simple form, a mosaic process transforms a set if tiles (Coverages or georeferenced imagery) into a single output dataset. Starting from the conceptual level, the process definition is incrementally refined until the specificity of an implementation profile is reached. At the level of a profile implementation, e.g. on a particular WPS instance, the implementer may still extend the contract of the implementation profile (e.g. by allowing more or larger input data).

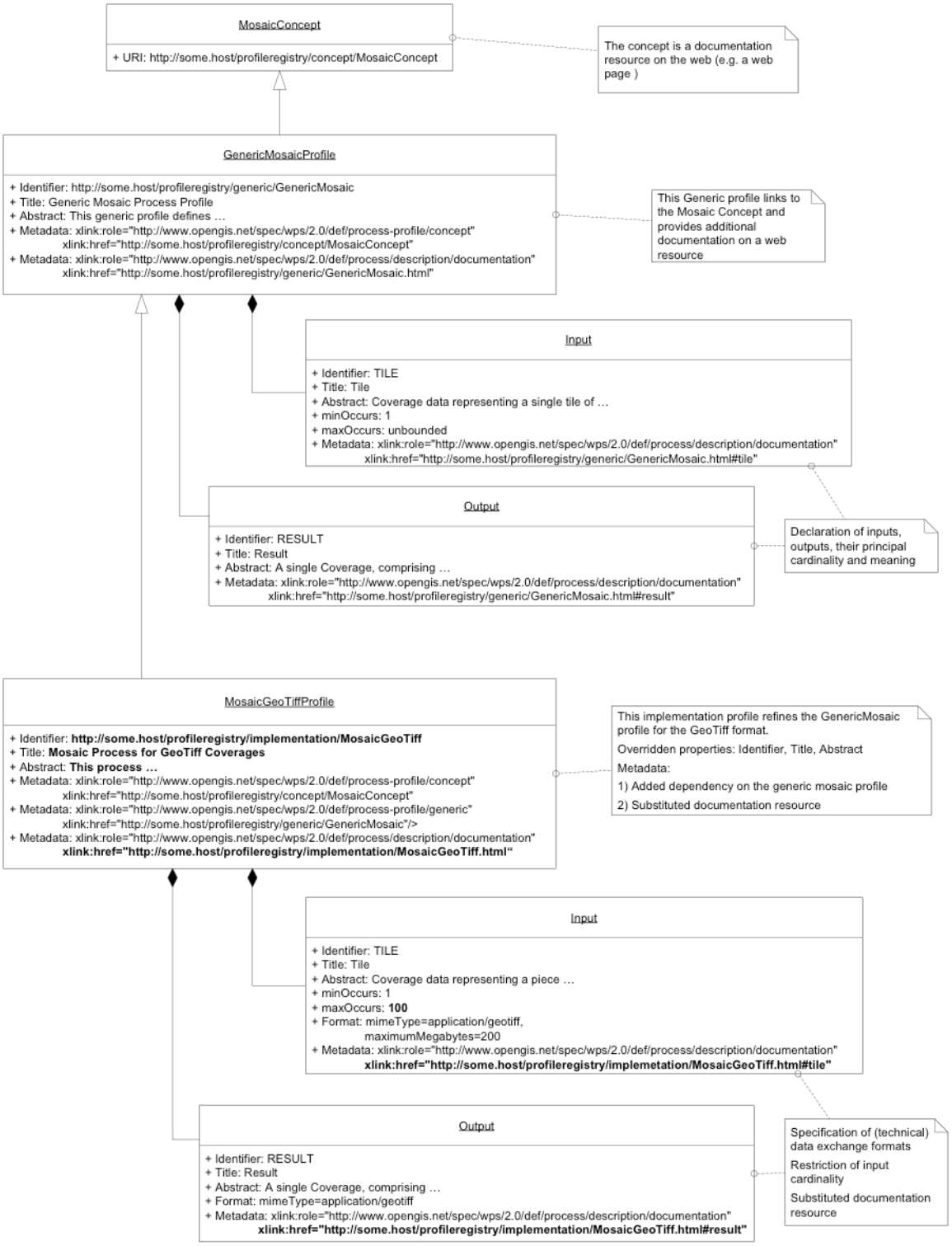


**Figure B.3** — Profile inheritance example for a mosaic process

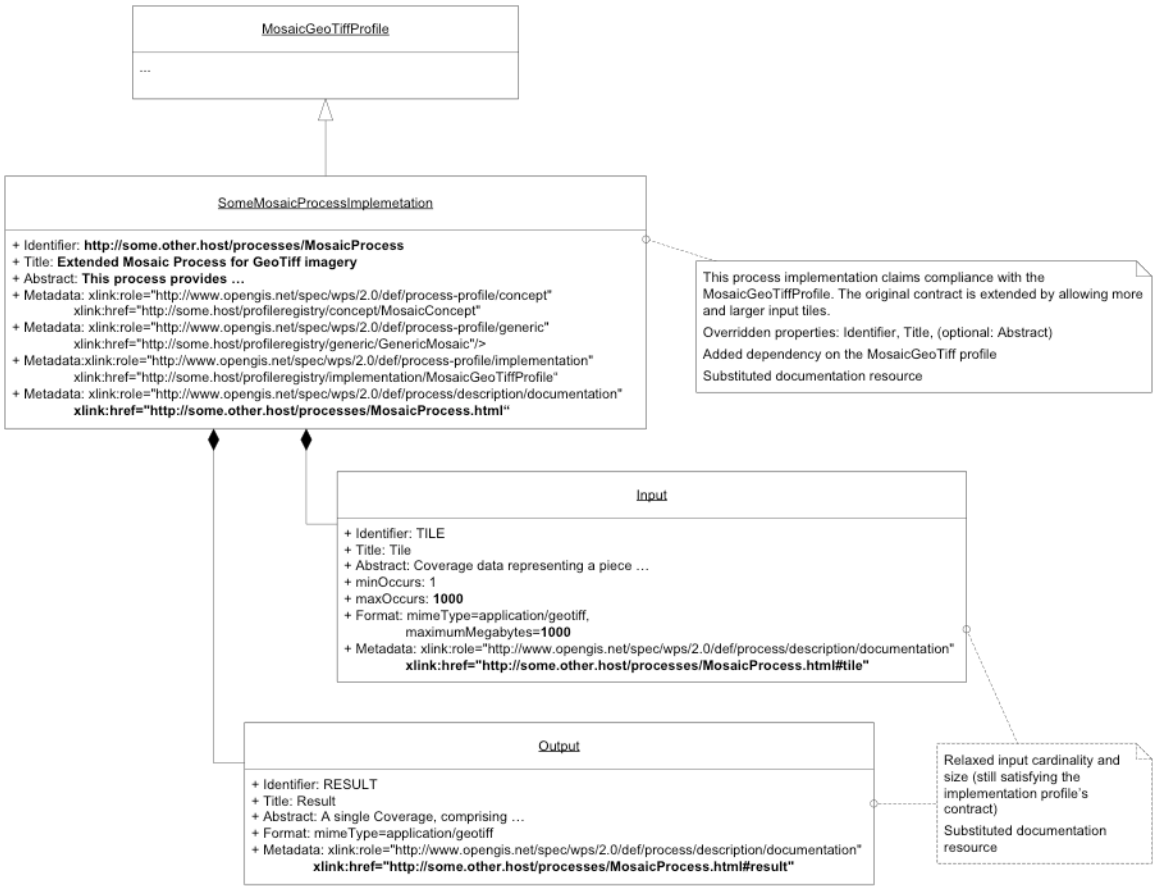


**Figure B.4** — Profile inheritance example for a mosaic process, extension by an implementation