OGC WPS 2.0 REST / JSON Binding Extension

Table of Contents

Open Geospatial Consortium
Submission Date: <yyyy-mm-dd>
Approval Date: <yyyy-mm-dd>
Publication Date: <yyyy-mm-dd>
External identifier of this OGC® document: http://www.opengis.net/doc/{doc-type}/{standard}/{m.n}
Internal reference number of this OGC® document: YY-nnnrx
Version: 0.1
Category: OGC® Implementation Specification
Editor: Benjamin Pross

OGC WPS REST/JSON Binding Extension

Warning

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

Recipients of this document are invited to submit, with their comments, notification of any relevant patent rights of which they are aware and to provide supporting documentation.

Document type: OGC® <Standard/Abstract Specification/Best Practice>
Document subtype: if applicable
Document stage: Draft
Document language: English

License Agreement

Permission is hereby granted by the Open Geospatial Consortium, ("Licensor"), free of charge and subject to the terms set forth below, to any person obtaining a copy of this Intellectual Property and any associated documentation, to deal in the Intellectual Property without restriction (except as set forth below), including without limitation the rights to implement, use, copy, modify, merge, publish, distribute, and/or sublicense copies of the Intellectual Property, and to permit persons to whom the Intellectual Property is furnished to do so, provided that all copyright notices on the intellectual property are retained intact and that each person to whom the Intellectual Property is furnished agrees to the terms of this Agreement.

If you modify the Intellectual Property, all copies of the modified Intellectual Property must include, in addition to the above copyright notice, a notice that the Intellectual Property includes modifications that have not been approved or adopted by LICENSOR.

THIS LICENSE IS A COPYRIGHT LICENSE ONLY, AND DOES NOT CONVEY ANY RIGHTS UNDER ANY PATENTS THAT MAY BE IN FORCE ANYWHERE IN THE WORLD.

THE INTELLECTUAL PROPERTY IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. THE COPYRIGHT HOLDER OR HOLDERS INCLUDED IN THIS NOTICE DO NOT WARRANT THAT THE FUNCTIONS CONTAINED IN THE INTELLECTUAL PROPERTY WILL MEET YOUR REQUIREMENTS OR THAT THE OPERATION OF THE INTELLECTUAL PROPERTY WILL BE UNINTERRUPTED OR ERROR FREE. ANY USE OF THE INTELLECTUAL PROPERTY SHALL BE MADE ENTIRELY AT THE USER'S OWN RISK. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR ANY CONTRIBUTOR OF INTELLECTUAL PROPERTY RIGHTS TO THE INTELLECTUAL PROPERTY BE LIABLE FOR ANY CLAIM, OR ANY DIRECT, SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM ANY ALLEGED INFRINGEMENT OR ANY LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR UNDER ANY OTHER LEGAL THEORY, ARISING OUT OF OR IN CONNECTION WITH THE IMPLEMENTATION, USE, COMMERCIALIZATION OR PERFORMANCE OF THIS INTELLECTUAL PROPERTY.

This license is effective until terminated. You may terminate it at any time by destroying the Intellectual Property together with all copies in any form. The license will also terminate if you fail to comply with any term or condition of this Agreement. Except as provided in the following sentence, no such termination of this license shall require the termination of any third party end-user sublicense to the Intellectual Property which is in force as of the date of notice of such termination. In addition, should the Intellectual

Property, or the operation of the Intellectual Property, infringe, or in LICENSOR's sole opinion be likely to infringe, any patent, copyright, trademark or other right of a third party, you agree that LICENSOR, in its sole discretion, may terminate this license without any compensation or liability to you, your licensees or any other party. You agree upon termination of any kind to destroy or cause to be destroyed the Intellectual Property together with all copies in any form, whether held by you or by any third party.

Except as contained in this notice, the name of LICENSOR or of any other holder of a copyright in all or part of the Intellectual Property shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Intellectual Property without prior written authorization of LICENSOR or such copyright holder. LICENSOR is and shall at all times be the sole entity that may authorize you or any third party to use certification marks, trademarks or other special designations to indicate compliance with any LICENSOR standards or specifications. This Agreement is governed by the laws of the Commonwealth of Massachusetts. The application to this Agreement of the United Nations Convention on Contracts for the International Sale of Goods is hereby expressly excluded. In the event any provision of this Agreement shall be deemed unenforceable, void or invalid, such provision shall be modified so as to make it valid and enforceable, and as so modified the entire Agreement shall remain in full force and effect. No decision, action or inaction by LICENSOR shall be construed to be a waiver of any rights or remedies available to it.

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. The OGC WPS REST/JSON Binding extension builds on the WPS 2.0 standard and defines the processing standards to communicate over a RESTful protocol using JSON encodings. This binding definition will be a newer and more modern way of programming and interacting with resources over the web while allowing better integration into existing software packages.

ii. Keywords

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

geoprocessing, ogcdoc, OGC document, processes, WPS, REST, JSON

iii. Preface

This extension is a continuation of WPS 2.0, a standard for web-based processing of geospatial data. It defines how the interfaces for WPS 2.0 operations should be constructed and interpreted using a REST based protocol with JSON encoding. Within the current version of WPS 2.0, bindings are defined for HTTP/POST using XML encodings and HTTP/GET using KVP encodings. Also in the current WPS 2.0 standard, a core conceptual model is provided that may be used to specify a WPS in different architectures such as REST or SOAP. Therefore, this extension is a natural fit to what is already defined in the standard.

iv. Submitting organizations

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

52°North

Intergraph Corporation (Hexagon Geospatial)

v. Submitters

All questions regarding this submission should be directed to the editor or the submitters:

Name	Representing	OGC Member
Benjamin Pross	52°North	Yes
Stan Tillman	Intergraph Corporation (Hexagon Geospatial)	Yes

Chapter 1. Scope

This document specifies the interface to a general-purpose Web Processing Service (WPS) using a RESTful protocol transporting JSON encoded requests and responses. 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 document is an extension to the OGC WPS 2.0 Interface Standard [14-065]. It should be considered as another binding extension to HTTP/POST + XML and HTTP/GET + KVP as defined in Section 10 (Binding Extensions for WPS Operations) of the WPS 2.0 standard.

Chapter 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.

In order to conform to this OGC® interface standard, a software implementation shall choose to implement:

- Any one of the conformance levels specified in Annex B (normative).
- Any one of the Distributed Computing Platform profiles specified in Annexes TBD through TBD (normative).

All requirements-classes and conformance-classes described in this document are owned by the standard(s) identified.

Chapter 3. References

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

OGC 14-065, OGC WPS 2.0 Interface Standard, version 2.0.1

OGC 06-121r9, OGC Web Service Common Specification, version 2.0

OGC 08-131r3 – The Specification Model – A Standard for Modular Specifications

IETF RFC 4646: Tags for Identifying Languages

IETF RFC 3986: Uniform Resource Identifier (URI): Generic Syntax

ISO 8601:2004, Data elements and interchange formats – Information interchange – Representation of dates and times

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

Chapter 4. Terms and Definitions

This document uses the terms defined in Sub-clause 5.3 of [OGC 06-121r8], 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

```
p: X → 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).

4.11. REST or RESTful

Representational state transfer. REST-compliant Web services allow requesting systems to access and manipulate textual representations of Web resources using a uniform and predefined set of stateless operations.

4.12. JSON

JavaScript Object Notation is a lightweight data-interchange format. It is easy for humans to read and write and it is easy for machines to parse and generate.

Chapter 5. Conventions

This sections 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.

5.1. Identifiers

The normative provisions in this specification are denoted by the URI

http://www.opengis.net/spec/{standard}/{m.n}

All requirements and conformance tests that appear in this document are denoted by partial URIs which are relative to this base.

5.2. Abbreviated Terms

Abbreviated Term	Meaning
CRS	Coordinate Reference System
GML	Geography Markup Language
HTTP	Hypertext Transfer Protocol
ISO	International Organization for Standardization
KVP	Keyword Value Pair
MIME	Multipurpose Internet Mail Extensions
OGC	Open Geospatial Consortium
URI	Universal Resource Identifier
URL	Uniform Resource Locator
WPS	Web Processing Service
XML	Extensible Markup Language
REST	Representational State Transfer
JSON	JavaScript Object Notation

Table 1. Abbreviated Terms

5.3. 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 section 4.1 of the WPS 2.0 standard.

5.4. Namespace Conventions

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

Table 2. Namespace Conventions

Chapter 6. REST/JSON Binding

In the following sections, the resources and endpoint-URLs of the WPS 2.0 REST/JSON binding are described.

6.1. Resources to be provided by WPS

The resources that are provided by a WPS REST server are listed in Table 3 below and include the capabilities document of the server, the list of processes available (ProcessCollection and Process), jobs (running processes) and outputs of processes.

Resource Class	Description	Acccess Path
Capabilities	The complete service metadata document.	{WpsRESTBaseURL}
ProcessCollection	List of processes available	{WpsRESTBaseURL}/processes
Process	Detailed process description of a single process	{WpsRESTBaseURL}/processes/{process-id}
JobCollection	List of jobs of a process	{WpsRESTBaseURL}/processes/{processID}/jobs
Job	Representation of a job (execution of a process) containing status information	{WpsRESTBaseURL}/processes/{processID}/jobs/{job-id}
Process Output Data	Resource containing the different process outputs inline or as reference.	{WpsRESTBaseURL}/processes/{processID}/jobs/{job-id}/outputs
Notations: {WpsRESTBaseURL}: The base URL of the WPS REST endpoint. {process-id}: identifier of a process. {job-id}: identifier to a job.		

Table 3. Resources provided by the WPS REST server

6.2. Operations on WPS resources

In general, the HTTP GET operation is used to provide access to the resources described above. However, in order to create a new job, the HTTP POST method is used to post a new job by sending a new job resource represented by an execute request to the server. This results in the operations listed in Table 4 below.

HTTP Operation	Acccess Path	Parameters	Description
GET	{WpsRESTBaseURL}	NA	Retrieval of Capabilities resource
GET	{WpsRESTBaseURL}/processes	NA	Retrieval of ProcessCollection resource
GET	{WpsRESTBaseURL}/processes/{process-id}	NA	Retrieval of a Process resource (process description)
GET	{WpsRESTBaseURL}/processes/{processID}/jobs	NA	Retrieval of list of jobs of a specific process (JobCollection)
GET	{WpsRESTBaseURL}/processes/{processID}/jobs/{job-id}	NA	Retrieval of a single Job resource
POST	{WpsRESTBaseURL}/processes/{processID}/jobs	Execute request (contained in body)	Execution of a process
Notations: {WpsRESTBaseURL}: The base URL of the WPS REST endpoint. {process-id}: identifier of a process. {job-id}: identifier to a job.			

Table 4. Operations on resources provided by the WPS REST server

6.3. Associations between WPS resources

As stated above in the listing of resources (Table 3), the basic resources managed by the WPS REST server are processes, jobs, outputs and its corresponding collections. An overview on the associations between the resources is given in Figure 1.

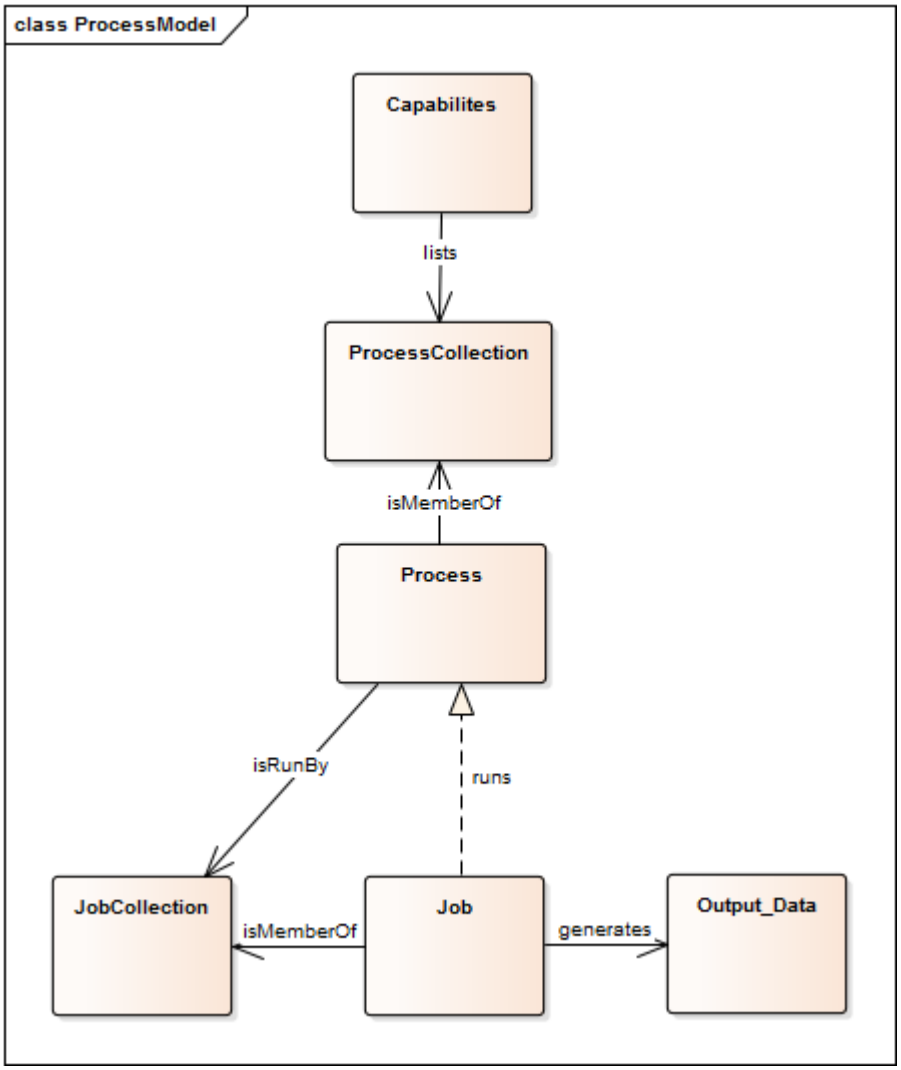

Figure 1. Resource model of the WPS REST server

The Capabilities has an association to the ProcessCollection that aggregates the single Processes offered by the Web Processing Server. The process has an association to the collection of jobs (JobCollection) that aggregates single jobs, i.e. instances that run a certain process.

6.4. JSON Encoding

6.4.1. JSON Information Model

In this document, the choice was made to simplify the WPS 2.0 information model in the JSON representation. Namespaces are not regarded. Bi-directional conversion between XML and JSON can lead to loss of information.

In the following, example requests and responses for the different operations and resources will be shown.

Requirements Class
http://www.opengis.net/spec/WPS/2.0/req/service/binding/rest-json

Requirements Class	
Target type	Software implementation
Depedency	TODO
Requirement	http://www.opengis.net/spec/WPS/2.0/req/service/binding/rest-json/get-capabilities Requirements class for the GetCapabilities operation
Requirement	http://www.opengis.net/spec/WPS/2.0/req/service/binding/rest-json/TODO

Table 5. General requirements classes for the WPS 2.0 REST/JSON Binding Extension

6.4.2. GetCapabilities

This clause specifies the JSON encoding for the GetCapabilities operation.

A GetCapabilities example is listed in Annex TODO.

Requirements Class	
http://www.opengis.net/spec/WPS/2.0/req/service/binding/rest-json/get-capabilities	
Target type	Software implementation
Depedency	TODO
Requirement	http://www.opengis.net/spec/WPS/2.0/req/service/binding/rest-json/get-capabilities/request A GetCapabilities request using REST/JSON shall be a valid JSON document of the type capabilities.
Requirement	http://www.opengis.net/spec/WPS/2.0/req/service/binding/rest-json/TODO

Table 6. Requirements classes for the WPS 2.0 REST/JSON Binding Extension GetCapabilities operation

Example of HTTP GET request for retrieving JSON WPS Capabilities.

```
http://hostname.org/wps-rest
```

Example of WPS Capabilities encoded as JSON.

```
{
    "Capabilities": {
        "ServiceIdentification": {
            "Title": "RESTful WPS ",
            "Abstract": "Service implementing the WPS 2.0 REST/JSON binding
extension",
            "ServiceType": "WPS",
            "ServiceTypeVersion": ["1.0.0",
            "2.0.0"],
            "Fees": "NONE",
            "AccessConstraints": "NONE"
        },
        "ServiceProvider": {
            "ProviderName": "OGC",
            "ProviderSite": {
                "HRef": "http://www.opengeospatial.org/"
            },
            "ServiceContact": {
                "IndividualName": "Your name",
                "ContactInfo": {

                }
            }
        },
        "Contents": {
            "ProcessSummaries": [{
                "identifier": "testbed12.fo.DouglasPeuckerAlgorithm",
                "title": "testbed12.fo.DouglasPeuckerAlgorithm",
                "processVersion": "1.0.0",
                "jobControlOptions": "sync-execute async-execute",
                "outputTransmission": "reference value",
                "processDescription": "http://hostname.org/wps-
rest/processes/ConvexHullAlgorithm"
            },
            ...]
        },
        "service": "WPS",
        "version": "2.0.0"
    }
}
```

The process summaries in the contents-section contain links to the process description of the respective process. The standalone list of processes can be requested as follows:

Example of HTTP GET request for retrieving the list of offered processes encoded as JSON.

```
http://hostname.org/wps-rest/processes
```

Example of Process list encoded as JSON.

```
{
  "ProcessSummaries": [
    {
      "identifier": "ConvexHullAlgorithm",
      "title": "Convex Hull Algorithm",
      "processVersion": "1.0.0",
      "jobControlOptions": "sync-execute async-execute",
      "processDescription": "http://hostname.org/wps-
rest/processes/ConvexHullAlgorithm"
    },...
    ]
}
```

6.4.3. DescribeProcess

Example of HTTP GET request for retrieving the process description of a process encoded in JSON. (NOTE: Request has been line-wrapped for easier reading).

```
http://hostname.org/wps-rest/processes/ConvexHullAlgorithm
```

Example of WPS DescribeProcess encoded as JSON.

```
{
  "ProcessOffering": {
    "Process": {
      "Title": "Hootenanny Conflation Process",
      "Identifier": "testbed12.lsa.HootenannyConflation",
      "Input": [
        {
          "Title": "INPUT1",
          "Identifier": "INPUT1",
          "ComplexData": {
            "Format": [
              {
                "default": "true",
                "mimeType": "application/x-zipped-shp"
              },...
            ]
```

```
        },
        "minOccurs": "1",
        "maxOccurs": "1"
      },
      {
        "Title": "INPUT1_TRANSLATION",
        "Identifier": "INPUT1_TRANSLATION",
        "ComplexData": {
          "Format": [
            {
              "default": "true",
              "mimeType": "text/x-script.phyton"
            },
            {
              "default": "false",
              "mimeType": "text/plain"
            }
          ]
        },
        "minOccurs": "0",
        "maxOccurs": "1"
      },
      {
        "Title": "INPUT2",
        "Identifier": "INPUT2",
        "ComplexData": {
          "Format": [
            {
              "default": "true",
              "mimeType": "application/x-openstreetmap+xml"
            }...
          ]
        },
        "minOccurs": "1",
        "maxOccurs": "1"
      },
      {
        "Title": "CONFLATION_TYPE",
        "Identifier": "CONFLATION_TYPE",
        "LiteralData": {
          "Format": [
            {
              "default": "true",
              "mimeType": "text/plain"
            },
            {
              "default": "false",
```

```
                "mimeType": "text/xml"
              }
            ],
            "LiteralDataDomain": [
              {
                "AnyValue": null,
                "DataType": {
                  "reference": "xs:string"
                }
              }
            ]
          },
          "minOccurs": "0",
          "maxOccurs": "1"
        },...
      ],
      "Output": [
        {
          "Title": "CONFLATION_OUTPUT",
          "Identifier": "CONFLATION_OUTPUT",
          "ComplexData": {
            "Format": [
              {
                "default": "true",
                "mimeType": "application/x-zipped-shp"
              },...
            ]
          }
        },
        {
          "Title": "CONFLATION_REPORT",
          "Identifier": "CONFLATION_REPORT",
          "ComplexData": {
            "Format": [
              {
                "default": "true",
                "mimeType": "text/plain"
              }
            ]
          }
        }
      ]
    },
    "processVersion": "1.0.0",
    "jobControlOptions": "sync-execute async-execute",
    "executeEndpoint": "http://hostname.org/wps-
rest/processes/ConvexHullAlgorithm/jobs"
```

```
  }
}
```

Example of HTTP GET request for getting a list of jobs of a process.

```
http://hostname.org/wps-rest/processes/ConvexHullAlgorithm/jobs
```

Example of a list of jobs for a process encoded as JSON.

```
{
  "Jobs": [
    "1317c058-cb4d-4ab4-ad21-b78e51229a17",
    "1319d2fc-cac8-4e8d-8039-2c511f55a9d3"
  ]
}
```

6.4.4. Execute

Example of HTTP POST request for executing a process.

```
http://hostname.org/wps-rest/processes/ConvexHullAlgorithm/jobs
```

By default, the proces will be executed asynchronously. If the process supports synchronous execution, this can be achieved by appending the following URL-parameter:

Example of HTTP POST request for synchronously executing a process.

```
http://hostname.org/wps-rest/processes/ConvexHullAlgorithm/jobs?sync-
execute=true
```

Example of WPS Execute request encoded as JSON.

```
{
    "Execute": {
        "Input": [
            {
            "Reference": {
                    "mimeType": "application/x-zipped-shp",
                    "href":
"http://geoprocessing.demo.52north.org:8080/data/Trans_RoadSegment-aoi.zip"
            },
            "id": "INPUT1"
            },
            {
            "Reference": {
                    "mimeType": "text/x-script.phyton",
                    "href":
"http://geoprocessing.demo.52north.org:8080/data/TNM_Roads.py"
            },
            "id": "INPUT1_TRANSLATION"
            },
            {
            "Reference": {
                    "mimeType": "application/x-openstreetmap+xml",
                    "href":
"http://geoprocessing.demo.52north.org:8080/data/sf_only_roads-aoi.osm"
            },
            "id": "INPUT2"
            }
        ],
        "output":[{
            "mimeType": "application/x-zipped-shp",
            "id": "CONFLATION_OUTPUT"
        },{
            "mimeType": "text/plain",
            "id": "CONFLATION_REPORT"
        }]
    }
}
```

The direct response to a asynchronously executed process is HTTP status code 201 (created) and the URL to obtain status information and finally the result. The URL will be returned in a HTTP header named *Location*. For synchronous execution, the result document will be returned after the process has finished.

6.4.5. GetStatus

Example of HTTP GET request for retrieving status information about a asynchonously executed process (NOTE: Request has been line-wrapped for easier reading).

```
http://hostname.org/wps-rest/processes/
testbed12.fo.DouglasPeuckerAlgorithm/jobs/
c731d14b-1de6-499c-9317-20224e056012
```

Example of WPS StatusInfo response encoded as JSON. The process is still running.

```
{
  "StatusInfo": {
    "JobID": "c731d14b-1de6-499c-9317-20224e056012",
    "Status": "Running",
    "Progress": 0
  }
}
```

After the process has finished, the progress element will be replaced by the URL to obtain the outputs

Example of WPS StatusInfo response encoded as JSON. The process has finished.

```
{
  "StatusInfo": {
    "JobID": "c731d14b-1de6-499c-9317-20224e056012",
    "Status": "Succeeded",
    "Output": "http://hostname.org/wps-
rest/processes/testbed12.lsa.HootenannyConflation/jobs/c731d14b-1de6-499c-
9317-20224e056012/outputs"
  }
}
```

6.4.6. GetResult

Example of WPS Result response encoded as JSON.


```
{
  "Result": {
    "JobID": "c731d14b-1de6-499c-9317-20224e056012",
    "Output": [
      {
        "ID": "CONFLATION_OUTPUT",
        "Reference": {
          "mimeType": "application/x-zipped-shp",
          "href": "http://hostname.org/wps/RetrieveResultServlet?id=c731d14b-
1de6-499c-9317-20224e056012CONFLATION_OUTPUT.b1172b1c-c9aa-495a-aa8b-
62220ac93605"
        }
      },
      {
        "ID": "CONFLATION_REPORT",
        "Reference": {
          "mimeType": "text/plain",
          "href": "http://hostname.org/wps/RetrieveResultServlet?id=c731d14b-
1de6-499c-9317-20224e056012****CONFLATION_REPORT.220391a6-4357-44e2-b5f3-
c0a0983cedae"
        }
      }
    ]
  }
}
```

Annex A: Conformance Class Abstract Test Suite (Normative)

NOTE	Ensure that there is a conformance class for each requirements class and a test for each requirement (identified by requirement name and number)

A.1. Conformance Class A

A.1.1. Requirement 1

Test id:	/conf/conf-class-a/req-name-1
Requirement:	/req/req-class-a/req-name-1
Test purpose:	Verify that…
Test method:	Inspect…

A.1.2. Requirement 2

Annex B: Revision History

Date	Release	Editor	Primary clauses modified	Description
2017-03-09	0.1	Benjamin Pross	all	initial version
2017-xx-xx	0.2	Benjamin Pross	6	Update REST/JSON section
2017-10-16	0.3	Stan Tillman	1-5	Update section 1-5

Annex C: Bibliography

Example Bibliography (Delete this note).

The TC has approved Springer LNCS as the official document citation type.

Springer LNCS is widely used in technical and computer science journals and other publications

NOTE

- For citations in the text please use square brackets and consecutive numbers: [1], [2], [3]

– Actual References:

[n] Journal: Author Surname, A.: Title. Publication Title. Volume number, Issue number, Pages Used (Year Published)

[n] Web: Author Surname, A.: Title, http://Website-Url

[1] OGC: OGC Testbed 12 Annex B: Architecture. (2015).