Open Geospatial Consortium Inc.

Date:   2012-06-22

Reference number of this OGC® project document:    OGC 12-060r1

Version: 0.0.4

Category: OGC® Implementation Specification

Editor:   Clemens Portele, Satish Sankaran

GeoServices REST API — Part 7: Geoprocessing Service

**Copyright notice**

Copyright © 2012 Open Geospatial Consortium, Inc. All Rights Reserved.
To obtain additional rights of use, visit <http://www.opengeospatial.org/legal/>.

**Warning**

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

Document type:    OGC® Publicly Available Standard

Document subtype:    n/a

Document stage:    Draft

Document language:  English

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.

Draft copyright:

Copyright: Esri Inc.

The companies listed above have granted the Open Geospatial Consortium, Inc. (OGC) a nonexclusive, royalty-free, paid up, worldwide license to copy and distribute this document and to modify this document and distribute copies of the modified version.

Contents

1. Scope 1

2. Conformance 1

3. References 2

4. Terms and Definitions 2

5. Conventions 3

6. Geoprocessing Service overview 3

7. Geoprocessing Service Core 4

7.1 Overview 4

7.2 Geoprocessing Service Root 5

7.2.1 Geoprocessing Service Root URI 5

7.2.2 Geoprocessing Service Root resources 6

7.3 Task 7

7.3.1 Overview 7

7.3.2 Task URI 7

7.3.3 Task resources 8

7.4 Execute Task 10

7.4.1 Overview 10

7.4.2 Execute Task URI 10

7.4.3 Result resources 16

7.5 Submit Job 17

7.5.1 Overview 17

7.5.2 Submit Job URI 17

7.5.3 Job resources 18

7.6 Job 18

7.6.1 Overview 18

7.6.2 Job URI 19

7.6.3 Job resources 20

7.7 Result and Input Parameters 21

7.7.1 Overview 21

7.7.2 Result / Input Parameter URI 22

7.7.3 Result / Input Parameter resources 23

A.1 Conformance class: gpservice 30

A.1.1 Test: gpservice/root 30

A.1.2 Test: gpservice/task 30

A.1.3 Test: gpservice/executeTask 31

A.1.4 Test: gpservice/submitJob 31

A.1.5 Test: gpservice/job 31

Preface

The “Esri GeoServices REST Specification Version 1.0” was originally developed by Esri to provide interoperability between ArcGIS Server and the broader information technology community. The Esri specification had been widely implemented by Esri users and business partners over 4 years. In 2010 it was released as a non-proprietary open specification and has been implemented by developers outside of the Esri user community.

In 2011, Esri has offered the GeoServices REST API for consideration to become an OGC standard. An OGC Standards Working Group was formed to document the specification in conformance with the modular specification policy of the OGC and to address comments received from the OGC membership and during the public review.

This candidate standard is designed to be implemented without the use of Esri products.

Submitting organizations

The following organizations submitted this Implementation Specification to the Open Geospatial Consortium Inc.:

Esri Inc.

interactive instruments GmbH

Oracle USA

52°North

Submission contact points

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

|  |  |
| --- | --- |
| CONTACT | COMPANY |
| 1. Satish Sankaran
 | 1. Esri Inc.
 |
| 1. Keith Ryden
 | 1. Esri Inc.
 |
| 1. Clemens Portele
 | 1. interactive instruments GmbH
 |
| 1. John Herring
 | 1. Oracle USA
 |
| 1. Andreas Wytzisk
 | 1. 52° North
 |

Revision history

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Date | Release | Author | Paragraph modified | Description |
| 1. 23 Jan 2012
 | 1. DRAFT 1
 | 1. CP, SS
 | 1. All
 | 1. First internal draft
 |
| 1. 01 May 2012
 | 1. 0.0.2
 | 1. CP
 | 1. All
 | 1. Include changes identified in review process
 |
| 1. 31 May 2012
 | 1. 0.0.3
 | 1. CP
 |  | 1. RFC package for OAB review
 |
| 1. 22 Jun 2012
 | 1. 0.0.4
 | 1. CP
 |  | 1. References updated
2. Editorial changes
 |

Changes to the OGC® Abstract Specification

The OGC**®** Abstract Specification does not require changes to accommodate this OGC**®** standard.

Versioning Rules

See the “Versioning Rules” section in OGC document 12-054r1, GeoServices REST API – Part 1: Core.

Foreword

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. Open Geospatial Consortium Inc. shall not be held responsible for identifying any or all such patent rights. However, to date, no such rights have been claimed or identified.

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 specification set forth in this document, and to provide supporting documentation.

This document is part 7 of the GeoServices REST API series:

Part 1: Core

Part 2: Catalog

Part 3: Map Service

Part 4: Feature Service

Part 5: Geometry Service

Part 6: Image Service

Part 7: Geoprocessing Service

Part 8: Geocoding Service

The relationship with other parts of the OGC standards baseline is described in document 12-062r1.

GeoServices REST API — Part 7: Geoprocessing Service

# Scope

The GeoServices REST API provides a standard way for web clients to communicate with geographic information system (GIS) servers based on Representational State Transfer (REST) principles. Clients issue requests to the resources on the server identified by structured URLs. The server responds with map images, text-based geographic information, or other representations of resources that satisfy the request.

This document specifies the resources of the Geoprocessing Service in an implementation of the GeoServices REST API and extends the GeoServices REST API – Core standard.

# 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]](#footnote-1).

This Standard establishes one requirements class and the corresponding conformance class, extending the core conformance class of the GeoServices REST API series.

All requirements-classes and conformance-classes described in this document are owned by the standard identified as **http://www.opengis.net/spec/gsr-gps/1.0**. Requirements and conformance test URIs defined in this document are relative to this URI unless they start with "http://" and are absolute URIs.

Any implementation claiming conformance with a conformance class shall pass all the tests in the associated abstract test suite.

Table 1 summarizes the requirements and conformance tests associated per conformance class.

Table 1 – Conformance class summary

|  |  |  |
| --- | --- | --- |
| **gpservice** | **Title** | Geoprocessing Service Core |
| **Standardization target type** | Web service |
| **Dependencies** | **http://www.opengis.net/spec/gsr/1.0/conf/core****http://www.opengis.net/spec/gsr/1.0/conf/geometry** |
| **Requirements** | All requirements in Clause 7 |
| **Conformance tests** | Annex A.1 |

Figure 1 – Single conformance class

# 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.

GeoServices REST API – Core, Version 1.0 (2012), OGC document 12-054r1

# Terms and Definitions

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.

# Conventions

See Clause 5 in the GeoServices REST API – Core document.

# Geoprocessing Service overview

Geoprocessing (GP) is a fundamental part of enterprise GIS operations. Geoprocessing provides data analysis, management, and conversion tools that are needed by users of geographic information.

The GeoServices REST API Geoprocessing Service provides a web service that performs geoprocessing. The root resource provides basic information associated with the service such as the service description, the tasks provided, the execution type, and, optionally, whether the geoprocessing service has been configured to return map images for use with a given map service (as denoted by the resultMapServerName property).

A GeoServices REST API Geoprocessing Service represents a collection of published tools that perform tasks necessary for manipulating and analyzing geographic information across a wide range of disciplines. Each tool performs one or more operations, such as projecting a dataset from one map projection to another, adding fields to a table, or creating buffer zones around features. A tool accepts input (such as feature sets, tables, and property values), executes operations using the input data, and generates output for presentation in a map or further processing by the client. Tools can be executed synchronously (meaning a user must wait for the results before proceeding) or asynchronously (meaning a user can do other things while awaiting notice that the task has completed).

A Geoprocessing service does the following:

* List available tools and their input/output properties.
* Execute a task synchronously.
* Submit a job to a task asynchronously.
* Get job details, including job status.
* Display results using a map service.
* Retrieve results for further processing by the client.

Many uses of GIS involve the repetition of work, and this creates the need for a framework to automate workflows. GP services answer this need by using a model to combine a series of operations in a sequence and exposing the model as a tool.

The GP Service resource has operations that return results after a task is successfully completed. The supported operations are

The Geoprocessing Service supports two operations via controller resources:

* Execute Task: Used when the execution type is synchronous
* Submit Job: Used when the execution type is asynchronous

The following figure provides an overview of the resources in a Geoprocessing Service. Resources in green color are controller resources, also called "operations", that a) either edit information in the server or b) process the operation parameters and the information in the server and create resources that are not persistently stored on the server and which are not made available with their own URI, but returned in the HTTP response to the operation, for example, as a result to a query. Controller resources of type b) could also be viewed simply as accessing existing resources on the server, shown in white colour in the figure, while in general these will be dynamically created by the controller resource. This is the case with synchronously executed tasks. Executing tasks asynchronously ("Submit Job") is a controller resource of type a) and creates a new Job resource and returns the link to the new resource.

Figure 2 – Resource overview

# Geoprocessing Service Core

## Overview

An implementation of the GeoServices REST API Geoprocessing Service Core provides capabilities that are needed by most applications using Geoprocessing services. Currently no additional capabilities are specified in additional conformance classes that depend on this core.

Table 2 – Geoprocessing Service Core overview

|  |  |  |
| --- | --- | --- |
| **Resource** | **Parameters** | **Resource representation** |
| Geoprocessing Service Root | f=json | JSON representation validAll JSON schema elements supported |
| Task | f=json | JSON representation validAll JSON schema elements supported |
| Execute Task | f=jsonenv:outSRenv:processSR*task input parameters* | JSON representation validAll JSON schema elements supported |
| Submit Job | same as "Execute Task" | JSON representation validAll JSON schema elements supported |
| Job | f=json | JSON representation validAll JSON schema elements supported |
| Parameter | f=json | JSON representation validAll JSON schema elements supported |

## Geoprocessing Service Root

### Geoprocessing Service Root URI

In the following URI templates, these variables are used:

* gpServiceRootURI: the URL of the service

If the Geoprocessing Service is referenced from a Catalog Service, the gpServiceRootURI variable is the same as

{+catServiceRootURI}/{gpServiceName}/GPServer

Table 3 – Geoprocessing Service Root reference

|  |  |
| --- | --- |
| **URI template** | {+gpServiceRootURI}{?f} |
| **HTTP methods** | GET |
| **Parent Resource Type** | - |
| **Child Resource Types**  | Task |

Table 4 – Geoprocessing Service Root parameters

|  |  |
| --- | --- |
| **Parameter** | **Details** |
| f | The response format.  |
| Required | Yes |
| Syntax | "json" |
| Example | f=json |

**Request Requirements**

|  |
| --- |
| * + 1. The Geoprocessing Service Root resource SHALL accept requests that conform to the URI template in Table 3 and use any HTTP method identified in the same table.

gpservice/request |

|  |
| --- |
| * + 1. The Geoprocessing Service Root resource SHALL support all parameters and values specified in Table 4.

gpservice/parameters |

### Geoprocessing Service Root resources

|  |
| --- |
| * + 1. The JSON representation of a Geoprocessing Service Root resource SHALL validate against the JSON Schema **http://schemas.opengis.net/gsr-gps/1.0/root.json** or in case of an exception against JSON Schema http://schemas.opengis.net/gsr/1.0/exception.json.

gpservice/valid |

|  |
| --- |
| * + 1. For each task listed in the response, a Task resource SHALL exist with the same execution type.

gpservice/tasksConsistent |

|  |
| --- |
| * + 1. If a result map server is listed, a GeoServices REST API Map Service SHALL exist at the URI "{+gpServiceRootURI}/../../{resultMapServerName}/MapServer".

gpservice/mapServerExists |

### Example

URL for a geoprocessing service:

http://example.com/rest/services/MyCustomTools/GPServer?f=json

**Request**

GET /rest/services/MyCustomTools/GPServer?f=json HTTP/1.1

Host: example.com

**Response**

HTTP/1.1 200 OK

Content-Type: application/json

Content-Length: nnn

{

"serviceDescription" : "Test Geoprocessing Service Description",

"tasks": [

"BufferPointsByRef",

"BufferLinesByRef",

"BufferPolygonsByRef"

],

"executionType": "ExecutionTypeAsynchronous",

"resultMapServerName": "BufferByRef"

}

## Task

### Overview

The Task resource represents a single geoprocessing task in a Geoprocessing Service. It provides basic information about the task, including its name and display name. It also provides detailed information about the various input and output parameters exposed by the task.

The GP task supports two operations:

* Execute Task: Used for synchronous tasks. Synchronous means that the application will wait while the tool executes on the server. Because the end user must wait, it should be determined if the wait time is acceptable for the type of application.
* Submit Job: Used for asynchronous tasks. Asynchronous means that the application does not wait for the task to finish execution, and the end user can continue using the application.

### Task URI

In the following URI templates, these variables are used:

* gpServiceURI: URL of a Geoprocessing Service Root resource without any parameter
* taskName: a taskname returned as part of a Geoprocessing Service Root resource

Table 5 – Task reference

|  |  |
| --- | --- |
| **URI template** | {+gpServiceURI}/{taskName}{?f} |
| **HTTP methods** | GET |
| **Parent Resource** | Geoprocessing Service Root |
| **Child Resource** | Execute TaskSubmit JobJob |

Table 6 – Task parameters

|  |  |
| --- | --- |
| **Parameter** | **Details** |
| f | The response format.  |
| Required | Yes |
| Syntax | "json" |
| Example | f=json |

**Request Requirements**

|  |
| --- |
| * + 1. The Task resource SHALL accept requests that conform to the URI template in Table 5 and use any HTTP method identified in the same table.

gpservice/taskRequest |

|  |
| --- |
| * + 1. The Task resource SHALL support all parameters and values specified in Table 6.

gpservice/taskParameters |

### Task resources

|  |
| --- |
| * + 1. The JSON representation of a response to a request on a Task resource SHALL validate against the JSON Schema **http://schemas.opengis.net/gsr-gps/1.0/task.json** or in case of an exception against JSON Schema http://schemas.opengis.net/gsr/1.0/exception.json.

gpservice/taskValid |

### Example

URL for a geoprocessing task Viewshed:

http://example.com/rest/services/Elevation/GPServer/Viewshed?f=json

**Request**

GET /rest/services/Elevation/GPServer/Viewshed?f=json HTTP/1.1

Host: example.com

**Response**

HTTP/1.1 200 OK

Content-Type: application/json

Content-Length: nnn

{

 "name" : "Viewshed",

 "displayName" : "Viewshed",

 "category" : "",

 "helpUrl" : "http://example.com/output/Elevation/Viewshed.htm",

 "executionType" : "ExecutionTypeSynchronous",

 "parameters" : [

 {

 "name" : "Input\_Observation\_Point",

 "dataType" : "GPFeatureRecordSetLayer",

 "displayName" : "Input Observation Point",

 "direction" : "GPParameterDirectionInput",

 "defaultValue" : {

 "geometryType" : "GeometryPoint",

 "spatialReference" : {

 "wkid" : 54003

 },

 "fields" : [

 {

 "name" : "FID",

 "type" : "FieldTypeOID",

 "alias" : "FID"},

 {

 "name" : "Shape",

 "type" : "FieldTypeGeometry",

 "alias" : "Shape"},

 {

 "name" : "OffsetA",

 "type" : "FieldTypeDouble",

 "alias" : "OffsetA"}

 ]

 },

 "parameterType" : "GPParameterTypeRequired",

 "category" : "",

 "choiceList" : []

 },

 {

 "name" : "Viewshed\_Distance",

 "dataType" : "GPLinearUnit",

 "displayName" : "Viewshed Distance",

 "direction" : "GPParameterDirectionInput",

 "defaultValue" : {

 "distance" : 15000,

 "units" : "Meters"

 },

 "parameterType" : "GPParameterTypeRequired",

 "category" : "",

 "choiceList" : []

 },

 {

 "name" : "Viewshed\_Result",

 "dataType" : "GPFeatureRecordSetLayer",

 "displayName" : "Viewshed Result",

 "direction" : "GPParameterDirectionOutput",

 "defaultValue" : {

 },

 "parameterType" : "GPParameterTypeRequired",

 "category" : "",

 "choiceList" : []

 }

 ]

}

## Execute Task

### Overview

The Execute Task operation is performed on a controller resource of a task when the execution type is synchronous. The result of this operation is a Result resource that contains an array of result parameters and the geoprocessing task execution messages. Each result parameter provides information such as the parameter name; the data type; and, most importantly, the value for that parameter. The resource is not stored on the server, it is returned directly in the response to the request.

Users provide arguments to the operation as query parameters. These parameters include the input parameters accepted by this service and their corresponding values. The input values for the Execute Task operation are identical to the input values for the Submit Job operation. Additionally, environment parameters can be specified such as the output spatial reference and the process spatial reference.

The values have different structures based on the data types of the parameters. Details about values for every data type are included in sub-clause 7.7.3.

### Execute Task URI

In the following URI templates, these variables are used:

* gpServiceURI: URL of a Geoprocessing Service Root resource without any parameter
* taskName: name of the task

Table 7 – Execute Task reference

|  |  |
| --- | --- |
| **URI template** | {+gpServiceURI}/{taskName}/execute{?f,env:outSR,env:processSR} |
| **HTTP methods** | GETPOST (application/x-www-form-urlencoded) |
| **Parent Resource** | Task |

Table 8 – Execute Task parameters

|  |  |
| --- | --- |
| **Parameter** | **Details** |
| f | The response format.  |
| Required | Yes |
| Syntax | "json" |
| Example | f=json |
| env:outSR | The spatial reference of the output geometries. The spatial reference can be specified as either a well-known ID or a spatial reference JSON object.  |
| Required | No. Default: The output geometry is exported in the process spatial reference, if env:processSR is provided, otherwise the spatial reference of the input geometries. |
| Syntax | POSINT / JSON |
| Example | env:outSR=4326 |
| env:processSR | The spatial reference that the model will use to perform geometry operations. The spatial reference can be specified as either a well-known ID or a spatial reference JSON object. |
| Required | No. Default: The geometry operations will be performed in the spatial reference of the input geometries. |
| Syntax | POSINT / JSON |
| Example | env:processSR=3034 |
| *parameters listed in the parameters property of the Task resource* | The various input parameters accepted by the corresponding task. The valid values for the input parameters are dependent on the data type of the parameter specified in the task resource. |
| Required | Yes, if parameterType is GPParameterTypeRequired, direction is GPParameterDirectionInput and no defaultValue is specified. If a defaultValue is specified and no parameter value is provided, the default value is used. |
| Syntax | Depends on the data type.GPBoolean: BOOLEANGPDouble: NUMBERGPLong: INTEGERGPString: STRINGGPDate: POSINTGPLinearUnit: JSONGPFeatureRecordSetLayer, GPRecordSet: JSONGPDataFile: JSONGPRasterData, GPRasterDataLayer: JSONGPMultiValue:GPString: STRINGARRAY |
| Examples | Depends on the data type.GPBoolean:InputBoolean=trueGPDouble:InputDouble=123.45GPLong: InputLong=8462GPString: InputString=My textGPLinearUnit: InputLinearUnit={ "distance" : 345.678, "units" : "Miles" }GPFeatureRecordSetLayer: InputFeatureRecordSet={ "geometryType" : "GeometryPoint","spatialReference" : {"wkid" : 4326},"features" : [{ "geometry" : {"x" : -104.44, "y" : 34.83}, "attributes" : {"Id" : 43, "Name" : "Feature 1"}},{ "geometry" : {"x" : -100.65, "y" : 33.69}, "attributes" : {"Id" : 67, "Name" : "Feature 2"}}]}GPRecordSet: InputRecordSet={ "features" : [{ "attributes" : {"Id" : 43, "Name" : "Feature 1"}},{ "attributes" : {"Id" : 67, "Name" : "Feature 2"}}]}GPDate: InputDate=1199145600000 // 1 Jan 2008 00:00:00 GMTGPDataFile:InputDataFile={ "url" : "http://example.com/myfile" }GPRasterData, GPRasterDataLayer:InputRaster={ "url" : "http://example.com/lake.tif", "format" : "tif" }GPMultiValue:GPString: InputMulti=["Parcels", "Street Lights"] |

The majority of the parameters included in the request are dependent on the input parameter types of the geoprocessing task being executed.

NOTE The Execute Task and Submit Job operations share the same syntax for all input parameter types.

**Request Requirements**

|  |
| --- |
| * + 1. The Execute Task resource SHALL accept requests that conform to the URI template in Table 7 and use any HTTP method identified in the same table.

gpservice/execRequest |

|  |
| --- |
| * + 1. The Execute Task resource SHALL support all parameters and values specified in Table 8.

gpservice/execParameters |

|  |
| --- |
| * + 1. A GPLinearUnit-typed parameter value SHALL validate against **http://schemas.opengis.net/gsr-gps/1.0/LinearUnit.json**.

gpservice/execLinearUnitValid |

The units property SHOULD be one of the following values:

|  |  |
| --- | --- |
|  |  |
| UnknownUnits | Unknown. |
| Inches | Inches. |
| Points | Points. |
| Feet | Feet. |
| Yards | Yards. |
| Miles | Miles. |
| NauticalMiles | Nautical miles. |
| Millimeters | Millimeters. |
| Centimeters | Centimeters. |
| Meters | Meters. |
| Kilometers | Kilometers. |
| DecimalDegrees | Decimal degrees. |
| Decimeters | Decimeters. |

|  |
| --- |
| * + 1. A GPFeatureRecordSetLayer-typed parameter value SHALL validate against **http://schemas.opengis.net/gsr/1.0/objectSet.json** and contain a geometry property for each feature in the set. If the task resource does not specify the geometry type and spatial reference for the parameter, geometryType and spatialReference SHALL be provided, too. Alternatively, the parameter value MAY also validate against **http://schemas.opengis.net/gsr-gps/1.0/url.json** and the referenced file validates against the objectSet.json schema.

gpservice/execFeatureRecordSetValid |

The geometryType property SHOULD be one of the following values:

|  |  |
| --- | --- |
|  |  |
| GeometryPoint | Point |
| GeometryPolyline | Polyline |
| GeometryPolygon | Polygon |

If the geometry type is not specified, it is assumed to be GeometryPoint. If the spatial reference is not specified, it defaults to an unknown coordinate system.

|  |
| --- |
| * + 1. A GPRecordSetLayer-typed parameter value SHALL validate against **http://schemas.opengis.net/gsr/1.0/objectSet.json** and contain no geometry properties. Alternatively, the parameter value MAY also validate against **http://schemas.opengis.net/gsr-gps/1.0/url.json** and the referenced file validates against the objectSet.json schema.

gpservice/execRecordSetValid |

|  |
| --- |
| * + 1. A GPDataFile-typed parameter value SHALL validate against **http://schemas.opengis.net/gsr-gps/1.0/url.json**.

gpservice/execDataFileValid |

|  |
| --- |
| * + 1. A GPRasterData- or GPRasterDataLayer-typed parameter value SHALL validate against **http://schemas.opengis.net/gsr-gps/1.0/url.json**.

gpservice/execRasterDataValid |

### Result resources

|  |
| --- |
| * + 1. The JSON representation of a response to a request on a Execute Task resource SHALL validate against the JSON Schema **http://schemas.opengis.net/gsr-gps/1.0/result.json** or in case of an exception against JSON Schema http://schemas.opengis.net/gsr/1.0/exception.json.

gpservice/execValid |

|  |
| --- |
| * + 1. The resource SHALL include a parameter value for each parameter of direction GPParameterDirectionOutput. The dataType and value SHALL be consistent with the information about the parameter in the task resource.

gpservice/execOutputParameters |

The messsage type property SHOULD be one of the following values:

|  |  |
| --- | --- |
|  |  |
| JobMessageTypeInformative | Informative. |
| JobMessageTypeWarning | Warning. |
| JobMessageTypeError | Error. |
| JobMessageTypeEmpty | Empty. |
| JobMessageTypeAbort | Abort. |
| JobMessageTypeProcessStart | ProcessStart. |
| JobMessageTypeProcessStop | ProcessStop. |
| JobMessageTypeProcessDefinition | ProcessDefinition. |

### Example

The task with the name "Viewshed" calculates the viewshed of a point given a user defined location (parameter Input\_Observation\_Point, here: 7.024° east, 49.918° north) and viewing distance (parameter Viewshed\_Distance, here 1000 meters). The coordinate reference system used during processing (parameter env:processSR) and for the result (parameter env:outSR) is WGS84 longitude/latitude.

This is a standard access to retrieve information. In this case, the resource does not exist on the server, but is compiled based on the query parameters.

NOTE This is inline with HTTP 1.1, section 9.3 says: "The GET method means retrieve whatever information (in the form of an entity) is identified by the Request-URI. If the Request-URI refers to a data-producing process, it is the produced data which shall be returned as the entity in the response and not the source text of the process, unless that text happens to be the output of the process."

The executed task will not change the state of the server and executing a task is safe and idempotent (unlike submitting a job for asynchronous execution, see 7.5).

URL for executing the geoprocessing task Viewshed:

http://example.com/rest/services/Elevation/GPServer/Viewshed/execute?Input\_Observation\_Point={"geometryType":"GeometryPoint","spatialReference":{"wkid":4326},"features":[{"geometry":{"x":7.024,"y":49.918},"attributes":{}}]}&Viewshed\_Distance={"distance":1000,"units":"Meters"}&env:outSR=4326&env:processSR=4326&f=json

**Request**

GET /rest/services/Elevation/GPServer/Viewshed/execute?Input\_Observation\_Point={"geometryType":"GeometryPoint","spatialReference":{"wkid":4326},"features":[{"geometry":{"x":7.024,"y":49.918},"attributes":{}}]}&Viewshed\_Distance={"distance":1000,"units":"Meters"}&env:outSR=4326&env:processSR=4326&f=json HTTP/1.1

Host: example.com

**Response**

HTTP/1.1 200 OK

Content-Type: application/json

Content-Length: nnn

{

 "results" : [

 {

 "paramName" : "Viewshed\_Result",

 "dataType" : "GPFeatureRecordSetLayer",

 "value" : {

 "geometryType" : "GeometryPolygon",

 "spatialReference" : {

 "wkid" : 4326

 },

 "features" : [

 {

 "attributes" : {

 "OBJECTID" : 1

 },

 "geometry" : {

 "rings" : [

 [

 [7.0291445880000651,49.921944675000077],

 [7.0271263390000627,49.921944675000077],

 [7.0271263390000627,49.924265866000042],

 [7.0281354630000692,49.924265866000042],

 [7.0281354630000692,49.923492142000043],

 [7.0291445880000651,49.923492142000043],

 [7.0291445880000651,49.921944675000077]

 ]

 ]

 }

 },

 {

 "attributes" : {

 "OBJECTID" : 2

 },

 "geometry" : {

 "rings" : [

 [

 [7.0160259710000332,49.918075885000064],

 [7.0170350960000292,49.918075885000064],

 [7.0170350960000292,49.918849657000067],

 [7.0160259710000332,49.918849657000067],

 [7.0160259710000332,49.918075885000064],

 [7.0150168470000267,49.918075885000064],

 [7.0150168470000267,49.920397180000066],

 [7.0180442200000357,49.920397180000066],

 [7.0180442200000357,49.918849657000067],

 [7.0200624690000382,49.918849657000067],

 [7.0200624690000382,49.918075885000064],

 [7.0210715930000447,49.918075885000064],

 [7.0210715930000447,49.917302106000079],

 [7.0220807170000512,49.917302106000079],

 [7.0220807170000512,49.918849657000067],

 [7.0271263390000627,49.918849657000067],

 [7.0271263390000627,49.921170931000063],

 [7.0311628360000213,49.921170931000063],

 [7.0311628360000213,49.919623422000029],

 [7.0331810850000238,49.919623422000029],

 [7.0331810850000238,49.917302106000079],

 [7.0321719610000741,49.917302106000079],

 [7.0321719610000741,49.914206920000026],

 [7.0311628360000213,49.914206920000026],

 [7.0311628360000213,49.91343310600007],

 [7.0301537120000717,49.91343310600007],

 [7.0301537120000717,49.912659285000075],

 [7.0291445880000651,49.912659285000075],

 [7.0291445880000651,49.911885457000039],

 [7.0271263390000627,49.911885457000039],

 [7.0271263390000627,49.911111621000032],

 [7.0210715930000447,49.911111621000032],

 [7.0210715930000447,49.911885457000039],

 [7.0190533440000422,49.911885457000039],

 [7.0190533440000422,49.912659285000075],

 [7.0180442200000357,49.912659285000075],

 [7.0180442200000357,49.91343310600007],

 [7.0170350960000292,49.91343310600007],

 [7.0170350960000292,49.914206920000026],

 [7.0160259710000332,49.914206920000026],

 [7.0160259710000332,49.915754527000047],

 [7.0150168470000267,49.915754527000047],

 [7.0150168470000267,49.916528320000054],

 [7.0160259710000332,49.916528320000054],

 [7.0160259710000332,49.918075885000064]

 ],

 [

 [7.0180442200000357,49.915754527000047],

 [7.0180442200000357,49.917302106000079],

 [7.0170350960000292,49.917302106000079],

 [7.0170350960000292,49.915754527000047],

 [7.0180442200000357,49.915754527000047]

 ],

 [

 [7.0261172150000561,49.917302106000079],

 [7.0261172150000561,49.918075885000064],

 [7.0251080900000602,49.918075885000064],

 [7.0251080900000602,49.917302106000079],

 [7.0261172150000561,49.917302106000079]

 ],

 [

 [7.0291445880000651,49.917302106000079],

 [7.0291445880000651,49.918075885000064],

 [7.0271263390000627,49.918075885000064],

 [7.0271263390000627,49.917302106000079],

 [7.0291445880000651,49.917302106000079]

 ],

 [

 [7.0321719610000741,49.918075885000064],

 [7.0321719610000741,49.918849657000067],

 [7.0301537120000717,49.918849657000067],

 [7.0301537120000717,49.918075885000064],

 [7.0321719610000741,49.918075885000064]

 ]

 ]

 }

 }

 ]

 }

 }

 ],

 "messages" : []

}

## Submit Job

### Overview

The Submit Job operation is performed on a controller resource of a task when the execution type is asynchronous. The result of this operation is a Job resource. The resource is stored on the server.

Users provide arguments to the operation as query parameters. The input values for the operation are identical to the input values for the Execute Task operation.

### Submit Job URI

In the following URI templates, these variables are used:

* gpServiceURI: URL of a Geoprocessing Service Root resource without any parameter
* taskName: name of the task

Table 9 – Submit Job reference

|  |  |
| --- | --- |
| **URI template** | {+gpServiceURI}/{taskName}/submitJob{?f,env:outSR,env:processSR} |
| **HTTP methods** | POST (application/x-www-form-urlencoded) |
| **Parent Resource** | Task |

The majority of the parameters included in the request are dependent on the input parameter types of the geoprocessing task being executed.

NOTE The Execute Task and Submit Job operations share the same syntax for all input parameter types.

**Request Requirements**

|  |
| --- |
| * + 1. The Submit Job resource SHALL accept requests that conform to the URI template in Table 9 and use any HTTP method identified in the same table.

gpservice/submitRequest |

|  |
| --- |
| * + 1. The Submit Job resource SHALL support all parameters and values specified in Table 8 and meet requirements Req 11 to Req 15.

gpservice/submitParameters |

### Job resources

|  |
| --- |
| * + 1. The response to a request on a Submit Task resource SHALL be a response with HTTP status code 302 and redirect to a new Job resource (see 7.6) or in case of an exception against JSON Schema http://schemas.opengis.net/gsr/1.0/exception.json.

gpservice/submitValid |

### Example

Submit a job to a MailingList task for parcel ID 1N1E34CC-06600 within a search distance of 100 feet and request the output spatial reference to be Web Mercator (WKID 102113):

**Request**

POST /rest/services/MyCustomTools/GPServer/MailingList/submitJob HTTP/1.1

Host: example.com

Content-Length: nnn

Content-type: application/x-www-form-urlencoded

Parcel\_ID=1N1E34CC-06600&SearchDistance\_ft=100&env:outSR=102113&f=json

**Response**

HTTP/1.1 302 Found

Content-Type: text/plain

Content-Length: nnn

Location: /rest/services/MyCustomTools/GPServer/MailingList/jobs/j374235676?f=json

Object moved to http://example.com/rest/services/MyCustomTools/GPServer/MailingList/jobs/j374235676?f=json

The newly created Job resource can be accessed at the URL specified in the Location header.

NOTE HTTP 1.1 supports the use of 303 in the response to POST requests (see RFC 2616, 9.5). RFC 2616 further states in 10.3.4 that instead of 303 also 302 may be used for compatibility with older user agents.

## Job

### Overview

The Job resource represents a geoprocessing job submitted using the Submit Job operation. It provides basic information about the job such as the job ID, status, and messages. Additionally, if the job has been successfully completed, the resource provides information about the result parameters as well as input parameters.

All result values can be accessed via the Result resource. The JSON response specifies a relative URL to the Result resource with a paramUrl field. Similarly, all input parameter values are accessed via the GP Input resource. The JSON response specifies a relative URL to the Input resource with a paramUrl field as well.

Users can specify whether the job should return messages or not by using returnMessages.

### Job URI

In the following URI templates, these variables are used:

* gpServiceURI: URL of a Geoprocessing Service Root resource without any parameter
* taskName: a taskname returned as part of a Geoprocessing Service Root resource
* jobid: job identifier

Table 10 – Job reference

|  |  |
| --- | --- |
| **URI template** | {+gpServiceURI}/{taskName}/jobs/{jobid}{?f,returnMessages} |
| **HTTP methods** | GET |
| **Parent Resource** | Task |
| **Child Resource** | Parameter |

Table 11 – Job parameters

|  |  |
| --- | --- |
| **Parameter** | **Details** |
| f | The response format.  |
| Required | Yes |
| Syntax | "json" |
| Example | f=json |
| returnMessages | If true, the job messages are included in the response. |
| Required | No. Default: true. |
| Syntax | Boolean |
| Example | returnMessages=false |

**Request Requirements**

|  |
| --- |
| * + 1. The Job resource SHALL accept requests that conform to the URI template in Table 10 and use any HTTP method identified in the same table.

gpservice/jobRequest |

|  |
| --- |
| * + 1. The Job resource SHALL support all parameters and values specified in Table 11.

gpservice/jobParameters |

### Job resources

|  |
| --- |
| * + 1. The JSON representation of a response to a request on a Job resource SHALL validate against the JSON Schema **http://schemas.opengis.net/gsr-gps/1.0/job.json** or in case of an exception against JSON Schema http://schemas.opengis.net/gsr/1.0/exception.json.

gpservice/jobValid |

The jobStatus property SHOULD be one of the following values:

|  |  |
| --- | --- |
|  |  |
| JobNew | The job is new. |
| JobSubmitted | The job is submitted for execution. |
| JobWaiting | The job is waiting for available job processor. |
| JobExecuting | The job is being executed by job processor. |
| JobSucceeded | The job has been completed successfully. |
| JobFailed | The job execution has failed. |
| JobTimedOut | The job execution has timed out. |
| JobCancelling | The job is being cancelled. |
| JobCancelled | The job has been cancelled. |
| JobDeleting | The job is being deleted. |
| JobDeleted | The job has been deleted. |

|  |
| --- |
| * + 1. For each result provided in the resource at "#/results/\*/paramUrl", a Result Parameter resource SHALL exist at the relative URI.

gpservice/resultsExist |

|  |
| --- |
| * + 1. For each input provided in the resource at "#/inputs/\*/paramUrl", an Input Parameter resource SHALL exist at the relative URI.

gpservice/inputsExist |

### Example

When submitting a job to the Mailing List task as shown in the Submit Job example, the user will be redirected to the Job resource page whose URL might be similar to the following (“j374235676” is an example job ID; job IDs might vary):

**Request**

GET /rest/services/MyCustomTools/GPServer/MailingList/jobs/j374235676?f=json HTTP/1.1

Host: example.com

**Response**

HTTP/1.1 200 OK

Content-Type: application/json

Content-Length: nnn

{
"jobId" : " j374235676",
"jobStatus" : "JobSucceeded",

"results" : {
"Output\_Long" : {"paramUrl" : "results/Output\_Long"},
"Output\_Feature\_Class" : {"paramUrl" : "results/Output\_Feature\_Class"}
},
"inputs" : {
"Input\_String" : {"paramUrl" : "inputs/Input\_String"},
"Input\_Feature\_Class" : {"paramUrl" : "inputs/Input\_Feature\_Class"}
},
"messages" : [
{"type" : "JobMessageTypeInformative", "description" : "Start Time: Thu Jul 05 16:36:25 2007"},
{"type" : "JobMessageTypeInformative", "description" : "Executed successfully."},
{"type" : "JobMessageTypeInformative", "description" : "End Time: Thu Jul 05 16:36:26 2007 (Elapsed Time: 1.00 seconds)"}
]
}

## Result and Input Parameters

### Overview

The Result and Input Parameter resource represents a result or an input parameter of a geoprocessing job. It provides information about the parameter such as its name, data type, and value. The value is the most important piece of information provided by this resource. Based on the data type of the parameter, the values provide different types of information. Given this fact, the value has different structures based on the data type as defined below.

### Result / Input Parameter URI

In the following URI templates, these variables are used:

* gpServiceURI: URL of a Geoprocessing Service Root resource without any parameter
* taskName: a taskname returned as part of a Geoprocessing Service Root resource
* jobid: job identifier
* param: relative URL provided in the paramUrl property of a Job resource

Table 12 – Result / Input Parameter reference

|  |  |
| --- | --- |
| **URI template** | {+gpServiceURI}/{taskName}/jobs/{jobid}/{+param}{?f,outSR,returnType} |
| **HTTP methods** | GET |
| **Parent Resource** | Job |
| **Child Resource** | - |

Table 13 – Result / Input Parameter parameters

|  |  |
| --- | --- |
| **Parameter** | **Details** |
| f | The response format.  |
| Required | Yes |
| Syntax | "json" |
| Example | f=json |
| outSR | The spatial reference of the output geometries. The spatial reference can be specified as either a well-known ID or a spatial reference JSON object. This parameter is applicable for result parameters that contain geometries. This parameter can be used to return the geometries in a spatial reference that is different from the spatial reference in which the outputs were initially created. |
| Required | No. Default: geometries are in original spatial reference. |
| Syntax | POSINT / JSON |
| Example | outSR=4326 |
| returnType | If the GP service is configured to return map images to be displayed with a certain map service (as described by the resultMapServerName property on the Geoprocessing Service Root resource), the default output for the GPRasterDataLayer and GPFeatureSetLayer parameters is a map image. However, users can explicitly request the raw data by using returnType and setting its value to "data". |
| Required | No.  |
| Syntax | "data" |
| Example | returnType=data |

**Request Requirements**

|  |
| --- |
| * + 1. The Result / Input Parameter resources SHALL accept requests that conform to the URI template in Table 12 and use any HTTP method identified in the same table.

gpservice/paramRequest |

|  |
| --- |
| * + 1. The Result / Input Parameter resources SHALL support all parameters and values specified in Table 13.

gpservice/paramParameters |

### Examples

For a Mailing List task job as described in the Job sub-clause, the user gets access to its Output\_Long result parameter with the following URL:

http://example.com/rest/services/MyCustomTools/GPServer/MailingList/jobs/j374235676/results/Output\_Long?f=json

**Request**

GET /rest/services/MyCustomTools/GPServer/MailingList/jobs/j374235676/results/Output\_Long?f=json HTTP/1.1

Host: example.com

**Response**

HTTP/1.1 200 OK

Content-Type: application/json

Content-Length: nnn

{

"paramName" : "Output\_Long",

"dataType" : "GPLong",

"value" : 123

}

On the same service, the user gets access to its SearchDistance\_ft input parameter with the following URL:

http://example.com/rest/services/MyCustomTools/GPServer/MailingList/jobs/j374235676/inputs/SearchDistance\_ft?f=json

**Request**

GET /rest/services/MyCustomTools/GPServer/MailingList/jobs/j374235676/inputs/SearchDistance\_ft?f=json HTTP/1.1

Host: example.com

**Response**

HTTP/1.1 200 OK

Content-Type: application/json

Content-Length: nnn

{

"paramName" : "SearchDistance\_ft",

"dataType" : "GPLong",

"value" : 100

}

### Result / Input Parameter resources

#### General rules

|  |
| --- |
| * + 1. The JSON representation of a response to a request on a Result / Input Parameter resource SHALL validate against the JSON Schema **http://schemas.opengis.net/gsr-gps/1.0/parameterValue.json** or in case of an exception against JSON Schema http://schemas.opengis.net/gsr/1.0/exception.json.

gpservice/paramValid |

The value field in the JSON response for a Result / Input Parameter resource varies based on the data type of the parameter. For certain data types, such as GPFeatureRecordSetLayer and GPRasterDataLayer, it can vary further based on whether the Geoprocessing service is configured to return map images for use with a given map service.

#### GPBoolean

|  |
| --- |
| * + 1. A GPBoolean-typed parameter value SHALL be either true or false.

gpservice/paramBooleanValid |

#### GPDouble

|  |
| --- |
| * + 1. A GPDouble-typed parameter value SHALL be a number.

gpservice/paramDoubleValid |

**JSON Response Example**

{

"paramName" : "Output\_Double",

"dataType" : "GPDouble",

"value" : 1234.56

}

#### GPLong

|  |
| --- |
| * + 1. A GPLong-typed parameter value SHALL be an integer.

gpservice/paramLongValid |

#### GPString

|  |
| --- |
| * + 1. A GPString-typed parameter value SHALL be a string.

gpservice/paramStringValid |

**JSON Response Example**

{
"paramName" : "Output\_String",
"dataType" : "GPString",
"value" : "TestString"
}

#### GPDate

|  |
| --- |
| * + 1. A GPDate-typed parameter value SHALL be a number that represents the number of milliseconds since epoch (January 1, 1970) in UTC.

gpservice/paramDateValid |

**JSON Response Example**

{

 "paramName": "Output\_Date",

 "dataType": "GPDate",

 "value": 1199145600000 // 1 Jan 2008 00:00:00 GMT

}

#### GPLinearUnit

|  |
| --- |
| * + 1. A GPLinearUnit-typed parameter value SHALL validate against http://schemas.opengis.net/gsr/1.0/LinearUnit.json.

gpservice/paramLinearUnitValid |

**JSON Response Example**

{

"paramName" : "Output\_Linear\_Unit",

"dataType" : "GPLinearUnit",

"value" : { "distance" : 1234.56, "units" : "Miles" }

}

#### GPFeatureRecordSetLayer

|  |
| --- |
| * + 1. If the resultMapServerName property in the Geoprocessing Service Root resource is empty or missing or if the request included the parameter "returnType" with a value "data": A GPFeatureRecordSetLayer-typed parameter value SHALL validate against http://schemas.opengis.net/gsr/1.0/objectSet.json and contain a geometry property for each feature in the set. If the task resource does not specify the geometry type and spatial reference for the parameter, geometryType and spatialReference SHALL be provided, too.

gpservice/paramFeatureRecordSetValid |

**JSON Response Example**

{

"paramName" : "Output\_Features",

"dataType" : "GPFeatureRecordSetLayer",

"value" :

 {

 "geometryType" : "GeometryPoint",

 "spatialReference" : {"wkid" : 4326},

 "features" : [

 {

 "geometry" : {"x" : -104.36, "y" : 34.657},

 "attributes" :

 {

 "TextField" : "a",

 "IntField" : 1234,

 "DoubleField" : 1234.56,

 "DateField" : "Sun Apr 10 16:00:00 PST 1977"

 }

 },

 {

 "geometry" : {"x" : -114.749, "y" : 31.439},

 "attributes" :

 {

 "TextField" : "b",

 "IntField" : 5678,

 "DoubleField" : 5678.91,

 "DateField" : "Mon Apr 11 16:00:00 PST 1977"

 }

 }

 ]

 }

}

#### GPRecordSetLayer

|  |
| --- |
| * + 1. A GPRecordSetLayer-typed parameter value SHALL validate against http://schemas.opengis.net/gsr/1.0/RecordSet.json and contain no geometry property for each feature in the set.

gpservice/paramRecordSetValid |

The features field is an array of features. Each feature in turn contains an attributes field. Attributes are key-value pairs where the key is a field name in the list of fields of the record set and the value is the value of the corresponding field.

**JSON Response Example**

{

"paramName" : "Output\_Record\_Set",

"dataType" : "GPRecordSet",

"value" :

 {

 "features" : [

 {

 "attributes" :

 {

 "TextField" : "a",

 "IntField" : 1234,

 "DoubleField" : 1234.56,

 "DateField" : "Sun Apr 10 16:00:00 PST 1977"

 }

 },

 {

 "attributes" :

 {

 "TextField" : "b",

 "IntField" : 5678,

 "DoubleField" : 5678.91,

 "DateField" : "Mon Apr 11 16:00:00 PST 1977"

 }

 }

 ],

"exceededTransferLimit" : false

 }

}

NOTE The exceededTransferLimit property is a sample extension that indicates, if the number of records exceeds the maximum number configured by the server administrator. Otherwise, it is false.

#### GPDataFile

|  |
| --- |
| * + 1. A GPDataFile-typed parameter value SHALL validate against http://schemas.opengis.net/gsr/1.0/DataFile.json.

gpservice/paramDataFileValid |

The value of the url field is a URL to the location of the data file.

**JSON Response Example**

{

"paramName" : "Output\_File",

"dataType" : "GPDataFile",

"value" :

 {

 "url" : "http://example.com/jobs/ByValTools\_GPServer/J1E7A1738AC054CDCBFC4A413DD9033CE/scratch/output.txt"

 }

}

#### GPRasterData or GPRasterDataLayer

|  |
| --- |
| * + 1. A GPRasterData-typed parameter value SHALL validate against http://schemas.opengis.net/gsr/1.0/RasterData.json.

gpservice/paramRasterDataValid |

|  |
| --- |
| * + 1. If the resultMapServerName property in the Geoprocessing Service Root resource is empty or missing or if the request included the parameter "returnType" with a value "data": A GPRasterDataLayer-typed parameter value SHALL validate against http://schemas.opengis.net/gsr/1.0/RasterData.json.

gpservice/paramRasterDataLayerValid |

The parameter value is a JSON object with the following properties:

* url: A URL to the location of the raster data file
* format: A string representing the format of the raster

**JSON Response Examples**

{

"paramName" : "Output\_Raster",

"dataType" : "GPRasterData",

"value" :

 {

 "url" : "http://example.com/jobs/ByValTools\_GPServer/JD613584CA6AC462AB8229A9A27B3DA79/scratch/slpgrd.tif",

 "format" : "tif"

 }

}

{

"paramName" : "Output\_Raster\_Layer",

"dataType" : "GPRasterDataLayer",

"value" :

 {

 "url" : "http://example.com/jobs/ByRefTools\_GPServer/J3D1737BA4584441FACBD5563AD1A47D5/scratch/outrast.tif",

 "format" : "tif"

 }

}

#### GPMultiValue:GPString

|  |
| --- |
| * + 1. A GPMultiValue:GPString-typed parameter value SHALL be a JSON array of strings.

gpservice/paramMultiValueStringValid |

**JSON Response Example**

{

 "paramName": "Output\_Layers",

 "dataType": "GPMultiValue:GPString",

 "value": ["Parcels", "Street Lights"]

}

#### Map Images as Geoprocessing Results

Geoprocessing services MAY be configured to return map images that are associated with a particular Map Service resource as described in the Geoprocessing Service Root resource's resultMapServerName property. If a Geoprocessing service is configured to return map images from a given map service, the results of GPFeatureRecordSetLayer and GPRasterDataLayer data types can be drawn by that map service and hence can be provided to the client as a map image.

In such cases where the result is a map image, the value field for the GP parameter is a JSON structure with a mapImage field. The structure of the mapImage field is a JSON object whose structure is the same as that of the JSON response of the Map Service Export Map operation. Further, most of the query parameters available for the Export Map operation are available for GP map image results as well. The only exceptions are the layers and the transparent parameters. The layers parameter is not available for GP map image results because it exports the map only for the layer corresponding to the GP parameter. The transparent parameter is available; however, the default value is true for GP map image results, whereas the default value is false for the Export Map operation.

|  |
| --- |
| * + 1. If the resultMapServerName property in the Geoprocessing Service Root resource is not empty and if the request did not include a parameter "returnType" with a value "data": A GPFeatureRecordSetLayer- or GPRasterDataLayer-typed parameter value SHALL validate against http://schemas.opengis.net/gsr-ms/1.0/MapImage.json.

gpservice/paramMapImageValid |

**JSON Response Example**

{

"paramName" : "Output\_Raster\_Layer",

"dataType" : "GPRasterDataLayer",

"value" :

 {

 "mapImage" :

 {

 "href" : "http://example.com/output/map40a7f57f31474933a94b5c672b7205f0.png",

 "width" : 400,

 "height" : 400,

 "extent" : {

 "xmin" : -109.55, "ymin" : 25.76, "xmax" : -86.39, "ymax" : 49.94,

 "spatialReference" : {"wkid" : 4326}

 },

 "scale" : 2.53E7

 }

 }

}

Annex A
(normative)

Abstract Test Suite

Conformance class: gpservice

* 1. Test: gpservice/root

|  |  |
| --- | --- |
| Requirements | **gpservice/request, gpservice/parameters, gpservice/valid, gpservice/tasksConsistent, gpservice/mapServerExists** |
| Test purpose | Verify that the Geoprocessing Service Root resource supports the request and response requirements. |
| Test method | Set up a test service that covers both execution types and all parameter types. Construct valid requests for the Geoprocessing Service Root resource.Inspect the responses and validate them against the JSON Schema http://schemas.opengis.net/gsr-gps/1.0/root.json or for exceptions against http://schemas.opengis.net/gsr/1.0/exception.json. Verify that * for each task listed in the response, a Task resource exists in the service, that the task has the same execution type and that it passes the test gpservice/task.
* if a result map server is listed that the Map Service exists.
 |
| Test type | Capability |

* 1. Test: gpservice/task

|  |  |
| --- | --- |
| Requirements | **gpservice/taskRequest, gpservice/taskParameters, gpservice/taskValid** |
| Test purpose | Verify that the Task resource supports the request and response requirements. |
| Test method | For each task listed in the root resource, construct valid requests for the resource.Inspect the responses and validate them against the JSON Schema http://schemas.opengis.net/gsr-gps/1.0/task.json or for exceptions against http://schemas.opengis.net/gsr/1.0/exception.json.  |
| Test type | Capability |

* 1. Test: gpservice/executeTask

|  |  |
| --- | --- |
| Requirements | **gpservice/execRequest, gpservice/execParameters, gpservice/execValid, gpservice/execLinearUnitValid, gpservice/execFeatureRecordSetValid, gpservice/execDataFileValid, gpservice/execRasterDataValid, gpservice/execOutputParameters** |
| Test purpose | Verify that the Execute Task resource supports the request and response requirements. |
| Test method | For each synchronous task listed in the root resource, construct valid requests for the Execute Task resource.Inspect the responses and validate them against the JSON Schema http://schemas.opengis.net/gsr-gps/1.0/result.json or for exceptions against http://schemas.opengis.net/gsr/1.0/exception.json. Verify that the response includes a parameter value for each parameter of direction GPParameterDirectionOutput and that the dataType and value is consistent with the information about the parameter in the task resource. |
| Test type | Capability |

* 1. Test: gpservice/submitJob

|  |  |
| --- | --- |
| Requirements | **gpservice/submitRequest, gpservice/submitParameters, gpservice/submitValid** |
| Test purpose | Verify that the Execute Task resource supports the request and response requirements. |
| Test method | For each asynchronous task listed in the root resource, construct valid requests for the Submit Job resource.Inspect the responses and verify that it is a response with a HTTP status code 302 and redirect to a Job resource that validates against the JSON Schema http://schemas.opengis.net/gsr-gps/1.0/job.json. For an exception verify that it validates against http://schemas.opengis.net/gsr/1.0/exception.json.  |
| Test type | Capability |

* 1. Test: gpservice/job

|  |  |
| --- | --- |
| Requirements | **gpservice/jobRequest, gpservice/jobParameters, gpservice/jobValid, gpservice/resultsExist, gpservice/inputsExist, gpservice/paramRequest, gpservice/paramParameters, gpservice/paramValid, gpservice/paramBooleanValid, gpservice/paramDoubleValid, gpservice/paramLongValid, gpservice/paramStringValid, gpservice/paramDateValid, gpservice/paramLinearUnitValid, gpservice/paramFeatureRecordSetValid, gpservice/paramRecordSetValid, gpservice/paramDataFileValid, gpservice/paramRasterDataValid, gpservice/paramRasterDataLayerValid, gpservice/paramMultiValueStringValid, gpservice/paramMapImageValid** |
| Test purpose | Verify that the Job resource supports the request and response requirements. |
| Test method | For each job created by a Submit Job operation, construct valid requests for the resource at multiple times until the job is complete.Inspect the responses and validate them against the JSON Schema http://schemas.opengis.net/gsr-gps/1.0/job.json or for exceptions against http://schemas.opengis.net/gsr/1.0/exception.json. For each result and input parameter referenced in the job resource, verify that the resource exists at the relative URI. For each parameter, construct valid requests for the resource and inspect the responses and validate them against the JSON Schema http://schemas.opengis.net/gsr-gps/1.0/parameterValue.json and the parameter-type-specific schema or for exceptions against http://schemas.opengis.net/gsr/1.0/exception.json. |
| Test type | Capability |

1. [www.opengeospatial.org/cite](http://www.opengeospatial.org/cite) [↑](#footnote-ref-1)