Open Geospatial Consortium Inc.

Date:   2012-06-22

Reference number of this OGC® project document:    OGC 12-058r1

Version: 0.0.4

Category: OGC® Implementation Specification

Editor:   Clemens Portele, Satish Sankaran

GeoServices REST API — Part 5: Geometry 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 6

4 Terms and Definitions 7

5 Conventions 7

6 Geometry Service overview 7

7 Geometry Service Core 9

7.1 Overview 9

7.2 Geometry Service Root URI 10

7.3 Geometry Service Root resources 11

7.4 Example 11

8 Project 12

8.1 Overview 12

8.2 Project URI 12

8.3 Geometry Array resources 15

8.4 Example 15

9 Simplify 15

9.1 Overview 15

9.2 Simplify URI 16

9.3 Geometry Array resources 18

9.4 Example 18

10 Buffer 19

10.1 Overview 19

10.2 Buffer URI 19

10.3 Geometry Array resources 23

10.4 Example 24

11 Areas And Lengths 25

11.1 Overview 25

11.2 Areas and Lengths URI 25

11.3 Areas And Lengths resources 28

11.4 Example 28

12 Lengths 29

12.1 Lengths Overview 29

12.2 Lengths URI 29

12.3 Lengths resources 32

12.4 Example 32

13 Relation 33

13.1 Relation Overview 33

13.2 Relation URI 33

13.3 Relation resources 37

13.4 Example 37

14 Label Points 38

14.1 Label Points Overview 38

14.2 Label Points URI 38

14.3 Label Points resources 40

14.4 Example 41

15 Distance 41

15.1 Distance Overview 41

15.2 Distance URI 42

15.3 Distance resources 44

15.4 Example 44

16 Densify 45

16.1 Densify Overview 45

16.2 Densify URI 45

16.3 Geometry Array resources 48

16.4 Example 48

17 Generalize 49

17.1 Generalize Overview 49

17.2 Generalize URI 49

17.3 Geometry Array resources 51

17.4 Example 52

18 Convex Hull 53

18.1 Convex Hull Overview 53

18.2 Convex Hull URI 53

18.3 Polygon resources 55

18.4 Example 55

19 Offset 56

19.1 Offset Overview 56

19.2 Offset URI 56

19.3 Geometry Array resources 59

19.4 Example 59

20 Trim/Extend 61

20.1 Trim/Extend Overview 61

20.2 Trim/Extend URI 61

20.3 Geometry Array resources 64

20.4 Example 64

21 Auto Complete 65

21.1 Auto Complete Overview 65

21.2 AutoComplete URI 65

21.3 Geometry Array resources 67

21.4 Example 68

22 Cut 68

22.1 Cut Overview 68

22.2 Cut URI 69

22.3 Geometry Array resources 71

22.4 Example 71

23 Difference 72

23.1 Difference Overview 72

23.2 Difference URI 73

23.3 Geometry Array resources 75

23.4 Example 75

24 Intersect 76

24.1 Intersect Overview 76

24.2 Intersect URI 76

24.3 Geometry Array resources 78

24.4 Example 79

25 Reshape 80

25.1 Reshape Overview 80

25.2 Reshape URI 80

25.3 Single Geometry resources 82

25.4 Example 86

26 Union 87

26.1 Union Overview 87

26.2 Union URI 88

26.3 Single Geometry resources 89

26.4 Example 89

A.1 Conformance class: geometryservice 92

A.1.1 Test: geometryservice/root 92

A.2 Conformance class: project 92

A.2.1 Test: project/test 92

A.3 Conformance class: simplify 92

A.3.1 Test: simplify/test 92

A.4 Conformance class: buffer 93

A.4.1 Test: buffer/test 93

A.5 Conformance class: areasAndLengths 93

A.5.1 Test: areasAndLengths/test 93

A.6 Conformance class: lengths 93

A.6.1 Test: lengths/test 93

A.7 Conformance class: relation 94

A.7.1 Test: relation/test 94

A.8 Conformance class: labelPoints 94

A.8.1 Test: labelPoints/test 94

A.9 Conformance class: distance 94

A.9.1 Test: distance/test 94

A.10 Conformance class: densify 95

A.10.1 Test: densify/test 95

A.11 Conformance class: generalize 95

A.11.1 Test: generalize/test 95

A.12 Conformance class: convexHull 95

A.12.1 Test: convexHull/test 95

A.13 Conformance class: offset 95

A.13.1 Test: offset/test 95

A.14 Conformance class: trimExtend 96

A.14.1 Test: trimExtend/test 96

A.15 Conformance class: autoComplete 96

A.15.1 Test: autoComplete/test 96

A.16 Conformance class: cut 96

A.16.1 Test: cut/test 96

A.17 Conformance class: difference 97

A.17.1 Test: difference/test 97

A.18 Conformance class: intersect 97

A.18.1 Test: intersect/test 97

A.19 Conformance class: reshape 97

A.19.1 Test: reshape/test 97

A.20 Conformance class: union 98

A.20.1 Test: union/test 98

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. 13 Mar 2012
 | 1. 0.0.1
 | 1. CP, SS
 | 1. All
 | 1. First draft
 |
| 1. 27 May 2012
 | 1. 0.0.2
 | 1. CP
 | 1. All
 | 1. Include changes identified in review process, complete draft
 |
| 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
 |

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 5 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 5: Geometry 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 geometry service resources 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 20 requirements classes and corresponding conformance classes, 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-gs/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

|  |  |  |
| --- | --- | --- |
| **geometryservice** | **Title** | Geometry Service Core |
| **Standardization target type** | Web service |
| **Dependencies** | **http://www.opengis.net/spec/gsr-gs/1.0/conf/core****http://www.opengis.net/spec/gsr-gs/1.0/conf/geometry** |
| **Requirements** | All requirements in Clause 7 |
| **Conformance tests** | Annex A.1 |
| **project** | **Title** | Project |
| **Standardization target type** | Web service |
| **Dependencies** | **conf/geometryservice** |
| **Requirements** | All requirements in Clause 8 |
| **Conformance tests** | Annex A.2 |
| **simplify** | **Title** | Simplify |
| **Standardization target type** | Web service |
| **Dependencies** | **conf/geometryservice** |
| **Requirements** | All requirements in Clause 9 |
| **Conformance tests** | Annex A.3 |
| **buffer** | **Title** | Buffer |
| **Standardization target type** | Web service |
| **Dependencies** | **conf/geometryservice** |
| **Requirements** | All requirements in Clause 10 |
| **Conformance tests** | Annex A.4 |
| **areasAndLengths** | **Title** | Areas and Lengths |
| **Standardization target type** | Web service |
| **Dependencies** | **conf/geometryservice** |
| **Requirements** | All requirements in Clause 11 |
| **Conformance tests** | Annex A.5 |
| **lengths** | **Title** | **Lengths** |
| **Standardization target type** | Web service |
| **Dependencies** | **conf/geometryservice** |
| **Requirements** | All requirements in Clause 12 |
| **Conformance tests** | Annex A.6 |
| **relation** | **Title** | **Relation** |
| **Standardization target type** | Web service |
| **Dependencies** | **conf/geometryservice** |
| **Requirements** | All requirements in Clause 13 |
| **Conformance tests** | Annex A.7 |
| **labelPoints** | **Title** | **Label Points** |
| **Standardization target type** | Web service |
| **Dependencies** | **conf/geometryservice** |
| **Requirements** | All requirements in Clause 14 |
| **Conformance tests** | Annex A.8 |
| **distance** | **Title** | **Distance** |
| **Standardization target type** | Web service |
| **Dependencies** | **conf/geometryservice** |
| **Requirements** | All requirements in Clause 15 |
| **Conformance tests** | Annex A.9 |
| **densify** | **Title** | **Densify** |
| **Standardization target type** | Web service |
| **Dependencies** | **conf/geometryservice** |
| **Requirements** | All requirements in Clause 16 |
| **Conformance tests** | Annex A.10 |
| **generalize** | **Title** | **Generalize** |
| **Standardization target type** | Web service |
| **Dependencies** | **conf/geometryservice** |
| **Requirements** | All requirements in Clause 17 |
| **Conformance tests** | Annex A.11 |
| **convexHull** | **Title** | **Convex Hull** |
| **Standardization target type** | Web service |
| **Dependencies** | **conf/geometryservice** |
| **Requirements** | All requirements in Clause 18 |
| **Conformance tests** | Annex A.12 |
| **offset** | **Title** | **Offset** |
| **Standardization target type** | Web service |
| **Dependencies** | **conf/geometryservice** |
| **Requirements** | All requirements in Clause 19 |
| **Conformance tests** | Annex A.13 |
| **trimExtend** | **Title** | **Trim/Extend** |
| **Standardization target type** | Web service |
| **Dependencies** | **conf/geometryservice** |
| **Requirements** | All requirements in Clause 20 |
| **Conformance tests** | Annex A.14 |
| **autoComplete** | **Title** | **AutoComplete** |
| **Standardization target type** | Web service |
| **Dependencies** | **conf/geometryservice** |
| **Requirements** | All requirements in Clause 21 |
| **Conformance tests** | Annex A.15 |
| **cut** | **Title** | **Cut** |
| **Standardization target type** | Web service |
| **Dependencies** | **conf/geometryservice** |
| **Requirements** | All requirements in Clause 22 |
| **Conformance tests** | Annex A.16 |
| **difference** | **Title** | **Difference** |
| **Standardization target type** | Web service |
| **Dependencies** | **conf/geometryservice** |
| **Requirements** | All requirements in Clause 23 |
| **Conformance tests** | Annex A.17 |
| **intersect** | **Title** | **Intersect** |
| **Standardization target type** | Web service |
| **Dependencies** | **conf/geometryservice** |
| **Requirements** | All requirements in Clause 24 |
| **Conformance tests** | Annex A.18 |
| **reshape** | **Title** | **Reshape** |
| **Standardization target type** | Web service |
| **Dependencies** | **conf/geometryservice** |
| **Requirements** | All requirements in Clause 25 |
| **Conformance tests** | Annex A.19 |
| **union** | **Title** | **Union** |
| **Standardization target type** | Web service |
| **Dependencies** | **conf/geometryservice** |
| **Requirements** | All requirements in Clause 26 |
| **Conformance tests** | Annex A.20 |

Figure 1 – Conformance class overview

# 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.

# Geometry Service overview

A geometry service contains utility methods that provide access to sophisticated and frequently used geometric operations. The GeoServices REST API Geometry Service resources are primarily processing and algorithmic resources that supports operations related to geometries. The Geometry Service resource has the following operations:

* Project: Returns an array of projected geometries
* Simplify: Returns an array of simplified geometries
* Buffer: Returns an array of polygons at the specified distances for the input geometry (An option is available to union buffer polygons at each distance.)
* Areas and Lengths: Calculates areas and perimeter lengths for each polygon specified in the input array
* Lengths: Calculates the lengths of each polyline specified in the input array
* Relation: Determines the pairs of geometries from the input geometry arrays that participate in the specified spatial relationship
* Label Points: Calculates an interior point for each polygon specified in the input array
* Distance: Reports the shortest distance between two points
* Densify: Densifies geometries by plotting intermediate points between existing vertices
* Generalize: Returns generalized (Douglas-Peucker) versions of the input geometries
* Convex Hull: Returns the convex hull of the input geometry
* Offset: Constructs the offset of the given input polyline based on an offset distance
* Trim/Extend: Trims or extends each polyline specified in the input array to meet user-specified guide polylines
* Auto Complete: Simplifies the process of constructing polygons that are adjacent to other polygons
* Cut: Splits the input polyline or polygon where it crosses a cutting polyline
* Difference: Constructs the set-theoretic difference between an array of geometries and another geometry
* Intersect: Constructs the set-theoretic intersection between an array of geometries and another geometry
* Reshape: Reshapes a polyline or part of a polygon using a reshaping line
* Union: Constructs the set-theoretic union of the input geometries

The above tasks could also optionally be accomplished through geoprocessing. The geometry service can be viewed as a lightweight alternative to a geoprocessing service, to be used for common operations.

Geometry input and output, where required, is always packaged as geometry arrays, see GeoServices REST API – Core, subclause 9.8.

The following figure provides an overview of the resources in a Geometry Service. Resources in green color are controller resources (also called "operations") that process the input geometries and transiently create and return the output geometries. The created geometry arrays are not persistently stored on the server and made available with their own URI, but returned in the response from the controller resource. These resources are shown in white color.

Figure 2 – Resource overview

# Geometry Service Core

## Overview

The Geometry Service Core provides basic information about the service and the geometry operations supported.

Table 2 – Geometry Service Core overview

|  |  |  |
| --- | --- | --- |
| **Resource** | **Parameters** | **Resource representation** |
| Geometry Service Root | f=json | JSON representation validAll JSON schema elements supported |

## Geometry Service Root URI

In the following URI templates, these variables are used:

* geometryServiceRootURI: the URL of the geometry service

If the Geometry Service is referenced from a Catalog Service, geometryServiceRootURI is the same as

{+catServiceRootURI}/{geometryServiceName}/GeometryServer

where geometryServiceName is the name of the geometry service referenced in the catalogue.

Table 3 – Geometry Service Root reference

|  |  |
| --- | --- |
| **URI template** | {+geometryServiceRootURI}{?f} |
| **HTTP methods** | GET |
| **Parent Resource Type** | n/a |
| **Child Resource Types**  | Project, Simplify, Buffer, Areas and Lengths, Lengths, Relation, Label Points, Auto Complete, Convex Hull, Cut, Densify, Difference, Generalize, Intersect, Offset, Reshape, Trim/Extend, Union |

Table 4 – Geometry Service Root parameters

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

**Request Requirements**

|  |
| --- |
| * + 1. A read request on a Geometry Service Root resource SHALL conform to the URI template in Table 3 and be accessed using a HTTP method identified in the same table.

geometryservice/request |

|  |
| --- |
| * + 1. A read request on a Geometry Service Root resource SHALL support all parameters and values specified in Table 4.

geometryservice/parameters |

## Geometry Service Root resources

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

geometryservice/valid |

|  |
| --- |
| * + 1. If the operations property in a Geometry Service Root resource is empty, the service SHALL support all conformance classes specified in this standard. If it contains a list of names then the Geometry Service SHALL conform to all conformance classes that are included in this list. A Geometry Service SHALL support at least one operation.

geometryservice/supportedOperations |

## Example

URL for a geometry service on example.com:

http://example.com/rest/services/Geometry/GeometryServer?f=json

**Request**

GET /rest/services/Geometry/GeometryServer?f=json HTTP/1.1

Host: example.com

**Response (**a Geometry Service supporting all conformance classes**)**

HTTP/1.1 200 OK

Content-Type: application/json

Content-Length: nnn

{

"serviceDescription" : "Test Geometry Service Description"

}

**Response (**a Geometry Service supporting the "project" and "simplify" conformance classes**)**

HTTP/1.1 200 OK

Content-Type: application/json

Content-Length: nnn

{

"serviceDescription" : "Test Geometry Service Description",

"operations" : [ "Project", "Simplify" ]

}

# Project

## Overview

The Project operation is performed on a controller resource of the Geometry Service. The result of this operation is an array of projected geometries. The result is not stored on the server and is returned in the response to the request.

This operation projects an array of input geometries from an input spatial reference to an output spatial reference. Users can provide arguments to the Project operation as parameters.

Table 5 – Project overview

|  |  |  |
| --- | --- | --- |
| **Resource** | **Parameters** | **Resource representation** |
| Project | f=jsongeometriesinSRoutSR | JSON representation validAll JSON schema elements supported |

## Project URI

In the following URI templates, these variables are used:

* geometryServiceRootURI: the URL of the geometry service without any parameter

Table 6 – Project reference

|  |  |
| --- | --- |
| **URI template** | {+geometryServiceRootURI}/project{?f,geometries,inSR,outSR} |
| **HTTP methods** | GETPOST (application/x-www-form-urlencoded) |
| **Parent Resource** | Geometry Service Root |

Table 7 – Project parameters

|  |  |
| --- | --- |
| **Parameter** | **Details** |
| f | The response format.  |
| Required | Yes |
| Syntax | "json" |
| Example | f=json |
| geometries | The array of geometries to be projected. Three representations are supported: JSON, a simplified syntax for point geometries, and a remote reference.**JSON:**An array of input geometries. All geometries in the array are of the type defined by the geometryType property.See GeoServices REST API – Core, subclause 9.8 for the schema of the geometry array.**Remote reference (URL-based):**For a large set of geometries, a URL may be provided to the input geometries stored in a JSON representation (see above) in a file on a public server. The URL is the value of a property "url" in a JSON object.**Simple syntax for point geometries:**When using points, the geometries can alternatively be specified with a simpler comma-separated syntax. |
| Required | Yes |
| Syntax | JSON / X "," Y \*("," X "," Y)  |
| Examples | **JSON:**geometries={"geometryType":"GeometryPoint","geometries":[{"x":-104.53,"y":34.74},{"x":-63.53,"y":10.23}]}**Remote reference (URL-based):**geometries={"url":"http://example.com/mygeometries/afile.json"}**Simple syntax for point geometries:**geometries=-104.53,34.74,-63.53,10.23 |
| inSR | The spatial reference for the input geometries.The spatial reference is specified as either a well-known ID (WKID) or a spatial reference JSON object. See the Geoservices REST API Core standard for the requirements related to spatial references. |
| Required | Yes |
| Syntax | POSINT / JSON |
| Example | inSR=4326 |
| outSR | The spatial reference for the output geometries.The spatial reference is specified as either a well-known ID (WKID) or a spatial reference JSON object. See the Geoservices REST API Core standard for the requirements related to spatial references. |
| Required | Yes |
| Syntax | POSINT / JSON |
| Example | outSR=3857 |

**Request Requirements**

|  |
| --- |
| * + 1. A request on a Project resource SHALL conform to the URI template in Table 6 and be accessed using a HTTP method identified in the same table.

project/Request |

|  |
| --- |
| * + 1. A request on a Project resource SHALL support all parameters and values specified in Table 7.

project/ParametersBasic |

|  |
| --- |
| * + 1. If the geometries parameter uses a JSON syntax, it SHALL either validate against the JSON Schema **http://schemas.opengis.net/gsr-gs/1.0/url.json** or conform to the requirements for geometry arrays (see requirements class **http://www.opengis.net/spec/gsr-gs/1.0/req/geometry**) and contain a "geometryType" property but no "spatialReference" property.

project/ParametersGeometries |

|  |
| --- |
| * + 1. The values of the inSR and outSR parameters SHALL either be a well-known ID (WKID) or a spatial reference JSON object. WKID values SHALL be a wkid property value of an object in http://schemas.opengis.net/gsr/1.0/sr.json.

project/ParametersSR |

## Geometry Array resources

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

project/ResponseValid |

## Example

Project the point [-117, 34] from WGS84 (WKID 4326) to Web Mercator (WKID 3857)::

http://example.com/rest/services/Geometry/GeometryServer/project?inSR=4326&outSR=3857&geometries={"geometryType":"GeometryPoint","geometries":[{"x":-117,"y":34}]}&f=json

**Request**

GET /rest/services/Geometry/GeometryServer/project?inSR=4326&outSR=3857&geometries={"geometryType":"GeometryPoint","geometries":[{"x":-117,"y":34}]}&f=json HTTP/1.1

Host: example.com

**Response**

HTTP/1.1 200 OK

Content-Type: application/json

Content-Length: nnn

{

 "geometryType" : "GeometryPoint",

 "geometries" :

 [

 {

 "x" : -13024380.422813,

 "y" : 4028802.02613441

 }

 ]

}

# Simplify

## Overview

The Simplify operation is performed on a Geometry Service resource. Simplify permanently alters the input geometry so that the geometry becomes topologically consistent. This includes detecting and repairing polygons that have overlapping rings and polygons that self-intersect. Users can provide arguments to the Simplify operation as query parameters.

Table 8 – Simplify overview

|  |  |  |
| --- | --- | --- |
| **Resource** | **Parameters** | **Resource representation** |
| Simplify | f=jsongeometriesSR | JSON representation validAll JSON schema elements supported |

## Simplify URI

In the following URI templates, these variables are used:

* geometryServiceRootURI: the URL of the geometry service without any parameter

Table 9 – Simplify reference

|  |  |
| --- | --- |
| **URI template** | {+geometryServiceRootURI}/simplify{?f,geometries,sr} |
| **HTTP methods** | GETPOST (application/x-www-form-urlencoded) |
| **Parent Resource** | Geometry Service Root |

Table 10 – Simplify parameters

|  |  |
| --- | --- |
| **Parameter** | **Details** |
| f | The response format.  |
| Required | Yes |
| Syntax | "json"  |
| Example | f=json |
| geometries | The array of geometries to be simplified. Three representations are supported: JSON, a simplified syntax for point geometries, and a remote reference.**JSON:**An array of input geometries. All geometries in the array are of the type defined by the geometryType property.See GeoServices REST API – Core, subclause 9.8 for the schema of the geometry array.**Remote reference (URL-based):**For a large set of geometries, a URL may be provided to the input geometries stored in a JSON representation (see above) in a file on a public server. The URL is the value of a property "url" in a JSON object.**Simple syntax for point geometries:**When using points, the geometries can alternatively be specified with a simpler comma-separated syntax. |
| Required | Yes |
| Syntax | JSON / X "," Y \*("," X "," Y)  |
| Examples | **JSON:**geometries={"geometryType":"GeometryPoint","geometries":[{"x":-104.53,"y":34.74},{"x":-63.53,"y":10.23}]}**Remote reference (URL-based):**geometries={"url":"http://example.com/mygeometries/afile.json"}**Simple syntax for point geometries:**geometries=-104.53,34.74,-63.53,10.23 |
| sr | The spatial reference for the input and ouput geometries.The spatial reference is specified as either a well-known ID (WKID) or a spatial reference JSON object. See the Geoservices REST API Core standard for the requirements related to spatial references. |
| Required | Yes |
| Syntax | POSINT / JSON |
| Example | SR=4326 |

**Request Requirements**

|  |
| --- |
| * + 1. A request on a Simplify resource SHALL conform to the URI template in Table 9 and be accessed using a HTTP method identified in the same table.

simplify/Request |

|  |
| --- |
| * + 1. A request on a Simplify resource SHALL support all parameters and values specified in Table 10.

simplify/ParametersBasic |

|  |
| --- |
| * + 1. If the geometries parameter uses a JSON syntax, it SHALL either validate against the JSON Schema **http://schemas.opengis.net/gsr-gs/1.0/url.json** or conform to the requirements for geometry arrays (see requirements class **http://www.opengis.net/spec/gsr-gs/1.0/req/geometry**) and contain a "geometryType" property but no "spatialReference" property.

simplify/ParametersGeometries |

|  |
| --- |
| * + 1. The values of the sr parameter SHALL either be a well-known ID (WKID) or a spatial reference JSON object. A WKID value SHALL be a wkid property value of an object in http://schemas.opengis.net/gsr/1.0/sr.json.

simplify/ParametersSR |

## Geometry Array resources

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

simplify/ResponseValid |

## Example

In this example, a polygon with one ring is simplified into a polygon with two rings:

http://example.com/rest/services/Geometry/GeometryServer/simplify?sr=4326&geometries={"geometryType":"GeometryPolygon","geometries":[{"rings":[[[-117,34],[-115,36],[-115,33],[-117,36],[-117,34]]]}]}&f=json

**Request**

GET /rest/services/Geometry/GeometryServer/simplify?sr=4326&geometries={"geometryType":"GeometryPolygon","geometries":[{"rings":[[[-117,34],[-115,36],[-115,33],[-117,36],[-117,34]]]}]}&f=json HTTP/1.1

Host: example.com

**Response**

HTTP/1.1 200 OK

Content-Type: application/json

Content-Length: nnn

{

 "geometryType" : "GeometryPolygon",

 "geometries" :

 [

 {

 "rings" :

 [

 [

 [-116.2, 34.8],

 [-117, 34],

 [-117, 36],

 [-116.2, 34.8]

 ],

 [

 [-116.2, 34.8],

 [-115, 36],

 [-115, 33],

 [-116.2, 34.8]

 ]

 ]

 }

 ]

}

# Buffer

## Overview

The Buffer operation is performed on a Geometry Service resource. The result of this operation is buffer polygons at the specified distances for the input geometry array. Users can provide arguments to the Buffer operation as query parameters. An option is available to union buffers at each distance.

Table 11 – Buffer overview

|  |  |  |
| --- | --- | --- |
| **Resource** | **Parameters** | **Resource representation** |
| Buffer | f=jsongeometriesinSRoutSRbufferSRdistancesunit | JSON representation validAll JSON schema elements supported |

## Buffer URI

In the following URI templates, these variables are used:

* geometryServiceRootURI: the URL of the geometry service without any parameter

Table 12 – Buffer reference

|  |  |
| --- | --- |
| **URI template** | {+geometryServiceRootURI}/Buffer{?f,geometries,inSR,outSR,bufferSR,distances,unit} |
| **HTTP methods** | GETPOST (application/x-www-form-urlencoded) |
| **Parent Resource** | Geometry Service Root |

Table 13 – Buffer parameters

|  |  |
| --- | --- |
| **Parameter** | **Details** |
| f | The response format.  |
| Required | Yes |
| Syntax | "json"  |
| Example | f=json |
| geometries | The array of geometries to be buffered. Three representations are supported: JSON, a simplified syntax for point geometries, and a remote reference.**JSON:**An array of input geometries. All geometries in the array are of the type defined by the geometryType property.See GeoServices REST API – Core, subclause 9.8 for the schema of the geometry array.**Remote reference (URL-based):**For a large set of geometries, a URL may be provided to the input geometries stored in a JSON representation (see above) in a file on a public server. The URL is the value of a property "url" in a JSON object.**Simple syntax for point geometries:**When using points, the geometries can alternatively be specified with a simpler comma-separated syntax. |
| Required | Yes |
| Syntax | JSON / X "," Y \*("," X "," Y)  |
| Examples | **JSON:**geometries={"geometryType":"GeometryPoint","geometries":[{"x":-104.53,"y":34.74},{"x":-63.53,"y":10.23}]}**Remote reference (URL-based):**geometries={"url":"http://example.com/mygeometries/afile.json"}**Simple syntax for point geometries:**geometries=-104.53,34.74,-63.53,10.23 |
| inSR | The spatial reference for the input geometries.The spatial reference is specified as either a well-known ID (WKID) or a spatial reference JSON object. See the Geoservices REST API Core standard for the requirements related to spatial references. |
| Required | Yes |
| Syntax | POSINT / JSON |
| Example | inSR=4326 |
| outSR | The spatial reference for the output geometries.The spatial reference is specified as either a well-known ID (WKID) or a spatial reference JSON object. See the Geoservices REST API Core standard for the requirements related to spatial references.  |
| Required | No. Default: the spatial reference specified by the bufferSR parameter; if no value for bufferSR is specified, they are in the spatial reference specified by the inSR parameter. |
| Syntax | POSINT / JSON |
| Example | outSR=4326 |
| bufferSR | The spatial reference in which the geometries are buffered.The spatial reference is specified as either a well-known ID (WKID) or a spatial reference JSON object. See the Geoservices REST API Core standard for the requirements related to spatial references. |
| Required | No. Default: the spatial reference specified by the outSR parameter. If no value for outSR is specified, they are buffered in the spatial reference specified by the inSR parameter. |
| Syntax | POSINT / JSON |
| Example | bufferSR=4326 |
| distance | The distances the input geometries are buffered. The distance units are specified by the unit parameter. |
| Required | Yes |
| Syntax | NUMBER \*( "," NUMBER) |
| Example | distances=100,123.45 |
| unit | The distance units to be applied to the calculation. The value is a well-known id (WKID) of a unit. |
| Required | Yes |
| Syntax | POSINT |
| Example | unit=9001 |
| unionResults | If true, all geometries buffered at a given distance are combined into a single (possibly multipart) polygon, and the unioned geometry is placed in the output array. |
| Required | No. Default: false |
| Syntax | Boolean  |
| Example | unionResults=true |

**Request Requirements**

|  |
| --- |
| * + 1. A request on a Buffer resource SHALL conform to the URI template in Table 12 and be accessed using a HTTP method identified in the same table.

buffer/Request |

|  |
| --- |
| * + 1. A request on a Buffer resource SHALL support all parameters and values specified in Table 13.

buffer/ParametersBasic |

|  |
| --- |
| * + 1. If the geometries parameter uses a JSON syntax, it SHALL either validate against the JSON Schema **http://schemas.opengis.net/gsr-gs/1.0/url.json** or conform to the requirements for geometry arrays (see requirements class **http://www.opengis.net/spec/gsr-gs/1.0/req/geometry**) and contain a "geometryType" property but no "spatialReference" property.

buffer/ParametersGeometries |

|  |
| --- |
| * + 1. The values of the inSR parameter SHALL either be a well-known ID (WKID) or a spatial reference JSON object. The values of the outSR and bufferSR parameters, if provided, SHALL either be a well-known ID (WKID) or a spatial reference JSON object. WKID values SHALL be a wkid property value of an object in http://schemas.opengis.net/gsr/1.0/sr.json.

buffer/ParametersSR |

|  |
| --- |
| * + 1. The values of the distance parameter SHALL be positive.

buffer/ParametersDistance |

|  |
| --- |
| * + 1. The values of the unit parameter SHALL be wkid properties of one of the entries in the document http://schemas.opengis.net/gsr/1.0/srLengthUnits.json.

buffer/ParametersUnit |

## Geometry Array resources

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

buffer/ResponseValid |

|  |
| --- |
| * + 1. If unionResults is false, the geometries SHALL be ordered in the sequence of the input geometries and then for each input geometry in the sequence of the distance values. If unionResults is true, there reslt SHALL contain one geometry for each distance value in the sequence of the distance values.

buffer/ResponseOrder |

|  |
| --- |
| * + 1. If bufferSR is a geodetic coordinate reference system and the unit is linear (e.g., feet or meter) then geodesic buffering SHALL be performed for points and multipoints; polylines and polygons SHALL be ignored (the unit has to be angular for these geometry types).

buffer/GeodesicBuffering |

## Example

In this example, the points [-117, 34] and [117,34] are buffered in geodetic WGS84 coordinates (WKID 4326) at distances of 1,000 and 1,500 meters. The geometry is buffered using the Web Mercator projection (WKID 3857), and the output polygon is returned in geodetic WGS84 coordinates (WKID 4326):

http://example.com/rest/services/Geometry/GeometryServer/buffer?f=json&

geometries=-117,34,117,34&inSR=4326&outSR=4326&bufferSR=3857&unit=9001& distances=1000,1500

**Request**

GET /rest/services/Geometry/GeometryServer/buffer?f=json&geometries=-117,34,117,34&inSR=4326&outSR=4326&bufferSR=3857&unit=9001&distances=1000,1500 HTTP/1.1

Host: example.com

**Response**

HTTP/1.1 200 OK

Content-Type: application/json

Content-Length: nnn

{

 "geometries" :

 [

 {

 "rings" :

 [

 [

 [-117, 34.0074470447458],

 [-116.999435943166, 34.0074323503471],

 …,

 [-117.000564056834, 34.0074323503471],

 [-117, 34.0074470447458]

 ]

 ]

 },

 {

 "rings" :

 [

 [

 [117, 34.0074470447458],

 [117.000564056834, 34.0074323503471],

 …,

 [116.998874112407, 34.0073883251278],

 [116.999435943166, 34.0074323503471],

 [117, 34.0074470447458]

 ]

 ]

 },

 {

 "rings" :

 [

 [

 [-117, 34.0111703222461],

 [-116.999306337717, 34.0111555123756],

 …,

 [-117.000693662283, 34.0111555123756],

 [-117, 34.0111703222461]

 ]

 ]

 },

 {

 "rings" :

 [

 [

 [117, 34.0111703222461],

 [117.000693662283, 34.0111555123756],

 …,

 [116.999306337717, 34.0111555123756],

 [117, 34.0111703222461]

 ]

 ]

 }

 ]

}

# Areas And Lengths

## Overview

The Areas and Lengths operation is performed on a Geometry Service resource. This operation calculates areas and perimeter lengths for each polygon specified in the input array. Users can provide arguments to the Areas and Lengths operation as query parameters.

Table 14 – Areas And Lengths overview

|  |  |  |
| --- | --- | --- |
| **Resource** | **Parameters** | **Resource representation** |
| areasAndLengths | f=jsonpolygonssrlengthUnitareaUnit | JSON representation validAll JSON schema elements supported |

## Areas and Lengths URI

In the following URI templates, these variables are used:

* geometryServiceRootURI: the URL of the geometry service without any parameter

Table 15 – Areas And Lengths reference

|  |  |
| --- | --- |
| **URI template** | {+geometryServiceRootURI}/AreasAndLengths{?f,polygons,sr,lengthUnit,areaUnit} |
| **HTTP methods** | GETPOST (application/x-www-form-urlencoded) |
| **Parent Resource** | Geometry Service Root |

Table 16 – Areas And Lengths parameters

|  |  |
| --- | --- |
| **Parameter** | **Details** |
| f | The response format.  |
| Required | Yes |
| Syntax | "json"  |
| Example | f=json |
| polygons | The array of polygons whose areas and lengths are to be computed. The spatial reference of the polygons is specified by the sr parameter. Two representations are supported: JSON and a remote reference.**JSON:**An array of input polygons.See GeoServices REST API – Core, subclause 9.6 for the schema of the polygon.**Remote reference (URL-based):**For a large set of geometries, a URL may be provided to the input geometries stored in a JSON representation (see above) in a file on a public server. The URL is the value of a property "url" in a JSON object. |
| Required | Yes |
| Syntax | JSON  |
| Examples | **JSON:**polygons=[{"rings":[[[-117,34],[-116,34],[-117,33],[-117,34]],[[-115,44],[-114,43],[-115,43],[-115,44]]]},{"rings":[[[32.49,17.83],[31.96,17.59],[30.87,17.01],[30.11,16.86],[32.49,17.83]]]}]**Remote reference (URL-based):**polygons={"url":"http://example.com/mygeometries/afile.json"} |
| sr | The spatial reference for the input polygons.The spatial reference is specified as either a well-known ID (WKID) or a spatial reference JSON object. See the Geoservices REST API Core standard for the requirements related to spatial references. |
| Required | Yes |
| Syntax | POSINT / JSON |
| Example | sr=4326 |
| lengthUnit | The length unit in which perimeters of polygons will be calculated.The value is a well-known id (WKID) of a unit. |
| Required | No. Default: If the unit is not specified, it is derived from the spatial reference. |
| Syntax | POSINT |
| Example | lengthUnit=9035 |
| areaUnit | The area unit in which areas of polygons will be calculated. The value is specified as a JSON object with a property "areaUnit" and a well-known id for an area unit. |
| Required | No. Default: If the unit is not specified, it is derived from the spatial reference. |
| Syntax | JSON |
| Example | areaUnit={"areaUnit":"Meter"} |

**Request Requirements**

|  |
| --- |
| * + 1. A request on an Areas And Lengths resource SHALL conform to the URI template in Table 15 and be accessed using a HTTP method identified in the same table.

areasAndLengths/Request |

|  |
| --- |
| * + 1. A request on an Areas And Lengths resource SHALL support all parameters and values specified in Table 16.

areasAndLengths/ParametersBasic |

|  |
| --- |
| * + 1. The polygons parameter SHALL either validate against the JSON Schema **http://schemas.opengis.net/gsr-gs/1.0/url.json** or conform to the requirements for polygons (see requirements class **http://www.opengis.net/spec/gsr-gs/1.0/req/geometry**) and contain no "spatialReference" property.

areasAndLengths/ParametersGeometries |

|  |
| --- |
| * + 1. The value of the sr parameter SHALL either be a well-known ID (WKID) or a spatial reference JSON object. A WKID value SHALL be a wkid property value of an object in http://schemas.opengis.net/gsr/1.0/sr.json.

areasAndLengths/ParametersSR |

|  |
| --- |
| * + 1. The value of the lengthUnit parameter SHALL be a wkid property of one of the entries in the document http://schemas.opengis.net/gsr/1.0/srLengthUnits.json.

areasAndLengths/ParametersLengthUnit |

|  |
| --- |
| * + 1. The value of the lengthUnit parameter SHALL be a unit property of one of the entries in the document http://schemas.opengis.net/gsr/1.0/areaUnits.json.

areasAndLengths/ParametersAreaUnit |

## Areas And Lengths resources

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

areasAndLengths/ResponseValid |

|  |
| --- |
| * + 1. The number of items in each array in the response SHALL be the same as the number of polygons in the request.

areasAndLengths/ResponseCount |

## Example

In this example, the area and length of a polygon are calculated. The length is returned in miles and the area in acres:

http://example.com/rest/services/Geometry/GeometryServer/areasAndLengths?

sr=102009&polygons=[{"rings":[[[-628833.344099998,206205.236200001],

[-630269.659900002,192298.906100001],[-631848.233800001,173991.394400001],

[-616471.690300003,341822.557500001],[-620213.661300004,301450.162799999],

[-625923.431999996,237538.0579],[-628833.344099998,206205.236200001]]]}]&

lengthUnit=9035&areaUnit={"areaUnit":"Acres"}&f=json

**Request**

GET /rest/services/Geometry/GeometryServer/areasAndLengths?

sr=102009&polygons=[{"rings":[[[-628833.344099998,206205.236200001],

[-630269.659900002,192298.906100001],[-631848.233800001,173991.394400001],

[-616471.690300003,341822.557500001],[-620213.661300004,301450.162799999],

[-625923.431999996,237538.0579],[-628833.344099998,206205.236200001]]]}]&

lengthUnit=9035&areaUnit={"areaUnit":"Acres"}&f=json HTTP/1.1

Host: example.com

**Response**

HTTP/1.1 200 OK

Content-Type: application/json

Content-Length: nnn

{

 "areas" : [ 615.362788718949 ],

 "lengths" : [ 209.444905018474 ]

}

# Lengths

## Lengths Overview

The Lengths operation is performed on a Geometry Service resource. This operation calculates the lengths of each polyline specified in the input array. Users can provide arguments to the Lengths operation as query parameters.

Table 17 – Lengths overview

|  |  |  |
| --- | --- | --- |
| **Resource** | **Parameters** | **Resource representation** |
| Lengths | f=jsonpolylinessrlengthUnitgeodesic | JSON representation validAll JSON schema elements supported |

## Lengths URI

In the following URI templates, these variables are used:

* geometryServiceRootURI: the URL of the geometry service without any parameter

Table 18 – Lengths reference

|  |  |
| --- | --- |
| **URI template** | {+geometryServiceRootURI}/lengths{?f,polylines,sr,lengthUnit,geodesic} |
| **HTTP methods** | GETPOST (application/x-www-form-urlencoded) |
| **Parent Resource** | Geometry Service Root |

Table 19 – Lengths parameters

|  |  |
| --- | --- |
| **Parameter** | **Details** |
| f | The response format. |
| Required | Yes |
| Syntax | "json" |
| Example | f=json |
| Polylines | The array of polylines whose lengths are to be computed. The spatial reference of the polylines is specified by the sr parameter. Two representations are supported: JSON and a remote reference.**JSON:**An array of input polylines.See GeoServices REST API – Core, subclause 9.5 for the schema of the polygon.**Remote reference (URL-based):**For a large set of geometries, a URL may be provided to the input geometries stored in a JSON representation (see above) in a file on a public server. The URL is the value of a property "url" in a JSON object. |
| Required | Yes |
| Syntax | JSON  |
| Example | **JSON:**polylines=[[{"paths":[[[-117,34],[-116,34],[-117,33]],[[-115,44],[-114,43],[-115,43]]]},{"paths": :[[[32.49,17.83],[31.96,17.59],[30.87,17.01],[30.11,16.86]]]}]**Remote reference (URL-based):**polygons={"url":"http://example.com/mygeometries/afile.json"} |
| sr | The spatial reference for the input polylines.The spatial reference is specified as either a well-known ID (WKID) or a spatial reference JSON object. See the Geoservices REST API Core standard for the requirements related to spatial references.  |
| Required | Yes |
| Syntax | POSINT / JSON |
| Example | sr=4326 |
| lengthUnit | The length unit in which lengths of polylines will be calculated.The value is a well-known id (WKID) of a unit. |
| Required | No. Default: Meter (9001) |
| Syntax | POSINT |
| Example | lengthUnit=9035 |
| geodesic | If polylines are in a geodetic coordinate reference system, geodesic should be set to true to calculate the ellipsoidal shortest path distance between each pair of vertices in the polylines. |
| Required | No. Default: false |
| Syntax | Boolean |
| Example | geodesic=true|false |

**Request Requirements**

|  |
| --- |
| * + 1. A request on a Lengths resource SHALL conform to the URI template in Table 18 and be accessed using a HTTP method identified in the same table.

lengths/Request |

|  |
| --- |
| * + 1. A request on a Lengths resource SHALL support all parameters and values specified in Table 19.

lengths/ParametersBasic |

|  |
| --- |
| * + 1. The polygons parameter SHALL either validate against the JSON Schema **http://schemas.opengis.net/gsr-gs/1.0/url.json** or conform to the requirements for polylines (see requirements class **http://www.opengis.net/spec/gsr-gs/1.0/req/geometry**) and contain no "spatialReference" property.

lengths/ParametersGeometries |

|  |
| --- |
| * + 1. The value of the sr parameter SHALL either be a well-known ID (WKID) or a spatial reference JSON object. A WKID value SHALL be a wkid property value of an object in http://schemas.opengis.net/gsr/1.0/sr.json.

lengths/ParametersSR |

|  |
| --- |
| * + 1. The value of the lengthUnit parameter SHALL be a wkid property of one of the entries in the document http://schemas.opengis.net/gsr/1.0/srLengthUnits.json.

lengths/ParametersLengthUnit |

## Lengths resources

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

lengths/ResponseValid |

|  |
| --- |
| * + 1. The number of items in the array in the response SHALL be the same as the number of polylines in the request.

lengths/ResponseCount |

## Example

In this example, the lengths of two input polylines are calculated:

http://example.com/rest/services/Geometry/GeometryServer/lengths?sr=4269&

polylines=[{"paths":[[[-117,34],[-116,34],[-117,33]],[[-115,44],[-114,43],

[-115,43]]]},{"paths":[[[32.49,17.83],[31.96,17.59],[30.87,17.01],

[30.11,16.86]]]}]&lengthUnit=9036&geodesic=true&f=json

**Request**

GET /rest/services/Geometry/GeometryServer/lengths?sr=4269&polylines=

[{"paths":[[[-117,34],[-116,34],[-117,33]],[[-115,44],[-114,43],

[-115,43]]]},{"paths":[[[32.49,17.83],[31.96,17.59],[30.87,17.01],

[30.11,16.86]]]}]&lengthUnit=9036&geodesic=true&f=json HTTP/1.1

Host: example.com

**Response**

HTTP/1.1 200 OK

Content-Type: application/json

Content-Length: nnn

{

"lengths" : [ 456.036465954783, 277.294288451794 ]

}

# Relation

## Relation Overview

The Relation operation is performed on a Geometry Service resource. The operation determines the pairs of geometries from the input geometry arrays that participate in the specified spatial relation.

This operation computes the set of pairs of geometries from parameters geometries1 and geometries2 that belong to the specified relation. Both arrays are assumed to be in the spatial reference specified by sr, which is a required parameter. The relations are evaluated in 2D. Z coordinates are not used. Geometry types may not be mixed within an array.

Users can provide arguments to the Relation operation as query parameters.

Table 20 – Relation overview

|  |  |  |
| --- | --- | --- |
| **Resource** | **Parameters** | **Resource representation** |
| Lengths | f=jsongeometries1geometries2srrelationrelationParam | JSON representation validAll JSON schema elements supported |

## Relation URI

In the following URI templates, these variables are used:

* geometryServiceRootURI: the URL of the geometry service without any parameter

Table 21 – Relation reference

|  |  |
| --- | --- |
| **URI template** | {+geometryServiceRootURI}/relation{?f,geometries1,geometries2,sr,relation,relationParam} |
| **HTTP methods** | GETPOST (application/x-www-form-urlencoded) |
| **Parent Resource** | Geometry Service Root |

Table 22 – Relation parameters

|  |  |
| --- | --- |
| **Parameter** | **Details** |
| f | The response format. |
| Required | Yes |
| Syntax | "json" |
| Example | f=json |
| geometries1, geometries2 | The first/second array of geometries for computing the relations.Two representations are supported: JSON and a remote reference.**JSON:**An array of input geometries.See GeoServices REST API – Core, subclause 9.8 for the schema of the geometry array. **Remote reference (URL-based):**For a large set of geometries, a URL may be provided to the input geometries stored in a JSON representation (see above) in a file on a public server. The URL is the value of a property "url" in a JSON object. |
| Required | Yes |
| Syntax | JSON  |
| Example | **JSON:**geometries1={"geometryType":"GeometryPoint", "geometries":[{"x":-104.53,"y":34.74},{"x":-63.53,"y":10.23}]}**Remote reference (URL-based):**geometries2={"url":"http://example.com/mygeometries/afile.json"} |
| sr | The spatial reference for the input geometries.The spatial reference is specified as either a well-known ID (WKID) or a spatial reference JSON object. See the Geoservices REST API Core standard for the requirements related to spatial references.  |
| Required | Yes |
| Syntax | POSINT / JSON |
| Example | sr=4326 |
| relation | The spatial relationship to be tested between the two input geometry arrays.Pre-defined values:* GeometryRelationCross: Two polylines cross if they share only points in common, at least one of which is not an endpoint. A polyline and an polygon cross if they share a polyline in common on the interior of the polygon which is not equivalent to the entire polyline. Cross is a Clementini operator. If either one of the geometries is empty, the geometries do not cross.
* GeometryRelationDisjoint: Two geometries are disjoint if their intersection is empty. Two geometries intersect if disjoint is "false". Disjoint is a Clementini operator.
* GeometryRelationIn: The base geometry is within the comparison geometry if the base geometry is the intersection of the geometries and the intersection of their interiors is not empty. Within is a Clementini operator. Within is the opposite operator of contains. An empty geometry is within another geometry, unless the other geometry is empty.
* GeometryRelationInteriorIntersection: Geometries intersect excluding boundary touch.
* GeometryRelationIntersection: Geometry interiors intersect or boundaries touch, same as 'not disjoint'.
* GeometryRelationLineCoincidence: The boundaries of the geometries must share an intersection, but the relationship between the interiors of the shapes is not considered (they could overlap, one could be contained in the other, or their interiors could be disjoint). This relation applies to polylines and polygons.
* GeometryRelationLineTouch: Two geometries are said to touch when the intersection of the geometries is non-empty, but the intersection of their interiors is empty. This evaluates if the touch occurs along a boundary (not a point). Valid for polygons.
* GeometryRelationOverlap: Two polylines share a common sub-line, or two polygons share a common sub-area. Overlaps is a Clementini operator. Two geometries do not overlap if either one is empty.
* GeometryRelationPointTouch: Two geometries are said to touch when the intersection of the geometries is non-empty, but the intersection of their interiors is empty. This evaluates if the touch occurs at a point (not a boundary).
* GeometryRelationRelation: Allows specification of any relationship defined using the 'Shape Comparison Language'.
* GeometryRelationTouch: The union of GeometryRelationPointTouch and GeometryRelationLineTouch. Two geometries are said to touch when the intersection of the geometries is non-empty, but the intersection of their interiors is empty. Touches is a Clementini operator. For example, a point touches a polyline only if the point is coincident with one of the polyline end points. If either one of the two geometries is empty, the geometries are not touched.
* GeometryRelationWithin: Same as GeometryRelationIn but also allows polylines that are strictly on the boundaries of polygons to be considered "in" the polygon.
* GeometryRelationRelation: The relationParam parameter describes the spatial relationship and has to be specified.
 |
| Required | Yes.  |
| Syntax | "GeometryRelationCross" / "GeometryRelationDisjoint" / "GeometryRelationIn" / "GeometryRelationInteriorIntersection" / "GeometryRelationIntersection" / "GeometryRelationLineCoincidence" / "GeometryRelationLineTouch" / "GeometryRelationOverlap" / "GeometryRelationPointTouch" / "GeometryRelationTouch" / "GeometryRelationWithin" / "GeometryRelationRelation" |
| Example | realtion=GeometryRelationDisjoint |
| relationParam | This string describes the spatial relationship to be tested when the relation parameter is specified as GeometryRelationRelation.For additional information on how to construct this string, see the relationParam description in table "Query parameters" of the GeoServices REST API – Map Service (table 12). |
| Required | Yes, if relation = "GeometryRelationRelation". Otherwise the parameter is ignored.  |
| Syntax | "'" 9\*("F" / "T" / "\*") "'" |
| Example | relationParam='FFFTTT\*\*\*' |

**Request Requirements**

|  |
| --- |
| * + 1. A request on a Relation resource SHALL conform to the URI template in Table 21 and be accessed using a HTTP method identified in the same table.

relation/Request |

|  |
| --- |
| * + 1. A request on a Relation resource SHALL support all parameters and values specified in Table 22.

relation/ParametersBasic |

|  |
| --- |
| * + 1. The geometries1 and geometries2 parameters SHALL either validate against the JSON Schema **http://schemas.opengis.net/gsr-gs/1.0/url.json** or conform to the requirements for geometry arrays (see requirements class **http://www.opengis.net/spec/gsr-gs/1.0/req/geometry**) and contain no "spatialReference" property.

relation/ParametersGeometries |

|  |
| --- |
| * + 1. The value of the sr parameter SHALL either be a well-known ID (WKID) or a spatial reference JSON object. A WKID value SHALL be a wkid property value of an object in http://schemas.opengis.net/gsr/1.0/sr.json.

relation/ParametersSR |

|  |
| --- |
| * + 1. The geometry types in each array in the parameters geometries1 and geometries2 SHALL NOT be mixed.

relation/ParametersGeometryTypes |

|  |
| --- |
| * + 1. If the relation parameter has the value "GeometryRelationRelation", the relationParam parameter SHALL be provided.

relation/ParametersRelationParam |

## Relation resources

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

relation/ResponseValid |

|  |
| --- |
| * + 1. The relationships SHALL be evaluated in 2D, i.e. Z-coordinates are not used.

relation/2D |

## Example

This example determines which of the 3 input points lie within the input polygon. The response shows that both the first (geometry1Index=0) and the third point (geometry1Index=2) lie within the polygon (geometry2Index=0).

http://example.com/rest/services/Geometry/GeometryServer/relation?f=json& sr=4326&relation=esriGeometryRelationWithin&

geometries1={"geometryType":"GeometryPoint","geometries":

[{"x":-104.53,"y":34.74},{"x":-63.53,"y":10.23},{"x":-104.67,"y":34.54}]}&

geometries2={"geometryType":"GeometryPolygon","geometries":

[{"rings":[[[-105,34],[-104,34],[-104,35],[-105,35],[-105,34]]]}]}

**Request**

GET /rest/services/Geometry/GeometryServer/relation?f=json& sr=4326&relation=esriGeometryRelationWithin&

geometries1={"geometryType":"GeometryPoint","geometries":

[{"x":-104.53,"y":34.74},{"x":-63.53,"y":10.23},{"x":-104.67,"y":34.54}]}&

geometries2={"geometryType":"GeometryPolygon","geometries":

[{"rings":[[[-105,34],[-104,34],[-104,35],[-105,35],[-105,34]]]}]} HTTP/1.1

Host: example.com

**Response**

HTTP/1.1 200 OK

Content-Type: application/json

Content-Length: nnn

{

 "relations" : [

 { "geometry1Index" : 0, "geometry2Index" : 0 },

 { "geometry1Index" : 2, "geometry2Index" : 0 }

 ]

}

# Label Points

## Label Points Overview

The Label Points operation is performed on a Geometry Service resource. This operation calculates an interior point for each polygon specified in the input array. These interior points can be used by clients for labeling the polygons. Users can provide arguments to the Label Points operation as query parameters.

Table 23 – Label Points overview

|  |  |  |
| --- | --- | --- |
| **Resource** | **Parameters** | **Resource representation** |
| Lengths | f=jsonpolygonssr | JSON representation validAll JSON schema elements supported |

## Label Points URI

In the following URI templates, these variables are used:

* geometryServiceRootURI: the URL of the geometry service without any parameter

Table 24 – Label Points reference

|  |  |
| --- | --- |
| **URI template** | {+geometryServiceRootURI}/labelPoints{?f,polygons,sr} |
| **HTTP methods** | GETPOST (application/x-www-form-urlencoded) |
| **Parent Resource** | Geometry Service Root |

Table 25 – Label Points parameters

|  |  |
| --- | --- |
| **Parameter** | **Details** |
| f | The response format |
| Required | Yes |
| Syntax | "json" |
| Example | f=json |
| polygons | The array of polygons whose label points are to be computed. The spatial reference of the polygons is specified by the sr parameter. Two representations are supported: JSON and a remote reference.**JSON:**An array of input polygons.See GeoServices REST API – Core, subclause 9.6 for the schema of the polygon.**Remote reference (URL-based):**For a large set of geometries, a URL may be provided to the input geometries stored in a JSON representation (see above) in a file on a public server. The URL is the value of a property "url" in a JSON object. |
| Required | Yes |
| Syntax | JSON  |
| Example | **JSON:**polygons= [{"rings":[[[-117,34],[-116,34],[-117,33],[-117,34]],[[-115,44],[-114,43],[-115,43],[-115,44]]]}, {"rings":[[[32.49,17.83],[31.96,17.59],[30.87,17.01],[30.11,16.86],[32.49,17.83]]]}]**Remote reference (URL-based):**polygons={"url":"http://example.com/mygeometries/afile.json"} |
| sr | The spatial reference of the input polygons.The spatial reference is specified as either a well-known ID (WKID) or a spatial reference JSON object. See the Geoservices REST API Core standard for the requirements related to spatial references. |
| Required | Yes |
| Syntax | POSINT |
| Example | sr=4326 |

**Request Requirements**

|  |
| --- |
| * + 1. A request on a LabelPoints resource SHALL conform to the URI template in Table 24 and be accessed using a HTTP method identified in the same table.

labelPoints/Request |

|  |
| --- |
| * + 1. A request on a LabelPoints resource SHALL support all parameters and values specified in Table 25.

labelPoints/ParametersBasic |

|  |
| --- |
| * + 1. If the polygons parameter uses a JSON syntax, it SHALL either validate against the JSON Schema http://schemas.opengis.net/gsr-gs/1.0/url.json or conform to the requirements for polygons (see requirements class **http://www.opengis.net/spec/gsr-gs/1.0/req/geometry**) and contain no "spatialReference" property.

labelPoints/ParametersGeometries |

|  |
| --- |
| * + 1. The value of the sr parameter SHALL either be a well-known ID (WKID) or a spatial reference JSON object. A WKID value SHALL be a wkid property value of an object in http://schemas.opengis.net/gsr/1.0/sr.json.

labelPoints/ParametersSR |

## Label Points resources

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

labelPoints/ResponseValid |

## Example

Compute a label point within an input polygon:

http://example.com/rest/services/Geometry/GeometryServer/labelPoints?f=json&sr=4326&polygons=[{"rings":[[[-105,34],[-104,34],[-104,35],[-105,35],

[-105,34]]]}]

**Request**

GET /rest/services/Geometry/GeometryServer/labelPoints?f=json&sr=4326&

polygons=[{"rings":[[[-105,34],[-104,34],[-104,35],[-105,35],[-105,34]]]}] HTTP/1.1

Host: example.com

**Response**

HTTP/1.1 200 OK

Content-Type: application/json

Content-Length: nnn

{

"labelPoints": [

 { "x": -104.5, "y": 34.5 }

]

}

# Distance

## Distance Overview

The Distance operation is performed on a Geometry Service resource. It reports the planar (projected space)/geodesic shortest distance between A and B. Users can provide arguments to the Distance operation as query parameters.

Table 26 – Distance overview

|  |  |  |
| --- | --- | --- |
| **Resource** | **Parameters** | **Resource representation** |
| Distance | f=jsongeometry1geometry2SRdistanceUnitgeodesic | JSON representation validAll JSON schema elements supported |

## Distance URI

In the following URI templates, these variables are used:

* geometryServiceRootURI: the URL of the geometry service without any parameter

Table 27 – Distance reference

|  |  |
| --- | --- |
| **URI template** | {+geometryServiceRootURI}/distance{?f,geometry1,geometry2,s,distanceUnit,geodesic} |
| **HTTP methods** | GETPOST (application/x-www-form-urlencoded) |
| **Parent Resource** | Geometry Service Root |

Table 28 – Distance parameters

|  |  |
| --- | --- |
| **Parameter** | **Details** |
| f | The response format |
| Required | Yes |
| Syntax | "json" |
| Example | f=json |
| geometry1, geometry2 | The geometry from (geometry1) and to (geometry2) which the distance is to be measured. See GeoServices REST API – Core, subclauses 9.3-9.7 for the schema of geometry objects.NOTE The use of simple syntax is not supported. |
| Required | Yes |
| Syntax | JSON  |
| Example | geometry1={"geometryType":"GeometryPoint","geometry":{"x":-104.53,"y":34.74}} |
| sr | The spatial reference of the input geometries. The spatial reference is specified as either a well-known ID (WKID) or a spatial reference JSON object. See the Geoservices REST API Core standard for the requirements related to spatial references.If the planar distance is being measured, the spatial reference MAY be a projected coordinate reference system. If the geodesic distance is being measured, the spatial reference MAY be either a projected or a geodetic coordinate reference system.  |
| Required | Yes |
| Syntax | JSON / POSINT |
| Example | sr=4326 |
| distanceUnit | Specifies the units for measuring distance between the geometry1 and geometry2 geometries. This is specified as a numerical constant from the wkid property of an object in http://schemas.opengis.net/gsr/1.0/srLengthUnits.json. For example, the value for meters is 9001.  |
| Required | No. Default is the unit used by the spatial reference. |
| Syntax | POSINT |
| Example | distanceUnit=9001 |
| geodesic | If true, the geodetic distance is measured between the geometries geometry1 and geometry2. |
| Required | No. Default: false |
| Syntax | Boolean |
| Example | geodesic=true |

**Request Requirements**

|  |
| --- |
| * + 1. A request on a Distance resource SHALL conform to the URI template in Table 27 and be accessed using a HTTP method identified in the same table.

distance/Request |

|  |
| --- |
| * + 1. A request on a Distance resource SHALL support all parameters and values specified in Table 28.

distance/ParametersBasic |

|  |
| --- |
| * + 1. The geometry1 and geometry2 parameters SHALL conform to the requirements for geometry objects (see requirements class **http://www.opengis.net/spec/gsr-gs/1.0/req/geometry**) and contain a "geometryType" property.

distance/ParametersGeometry |

|  |
| --- |
| * + 1. The value of the sr parameter SHALL either be a well-known ID (WKID) or a spatial reference JSON object. A WKID value SHALL be a wkid property value of an object in http://schemas.opengis.net/gsr/1.0/sr.json.

distance/ParametersSR |

|  |
| --- |
| * + 1. If the distanceUnit parameter is present, it SHALL be a value of a wkid property of an object in http://schemas.opengis.net/gsr/1.0/srLengthUnits.json.

distance/ParametersDistanceUnit |

## Distance resources

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

distance/ResponseValid |

## Example

Compute the geodesic distance in miles between two points:

http://example.com/rest/services/Geometry/GeometryServer/distance?sr=4326&f=json&geodesic=true&distanceUnit=9035&

geometry1={"geometryType":"GeometryPoint",

"geometry":{"x":-117.47697998046874,"y":34.121858211839566}}&

geometry2={"geometryType":"GeometryPoint",

"geometry":{"x":-117.41586853027343,"y":34.108125301683316}}

**Request**

GET /rest/services/Geometry/GeometryServer/distance?sr=4326&f=json&

geodesic=true&distanceUnit=9035&

geometry1={"geometryType":"GeometryPoint",

"geometry":{"x":-117.47697998046874,"y":34.121858211839566}}&

geometry2={"geometryType":"GeometryPoint",

"geometry":{"x":-117.41586853027343,"y":34.108125301683316}} HTTP/1.1

Host: example.com

**Response**

HTTP/1.1 200 OK

Content-Type: application/json

Content-Length: nnn

{

 "distance" : 10

}

# Densify

## Densify Overview

The Densify operation is performed on a Geometry Service resource. This operation densifies geometries by plotting points between existing vertices. Users can provide arguments to the Densify operation as query parameters.

Table 29 –Densify overview

|  |  |  |
| --- | --- | --- |
| **Resource** | **Parameters** | **Resource representation** |
| Lengths | f=jsongeometriessrmaxSegmentLengthlengthUnit (optional) | JSON representation validAll JSON schema elements supported |

## Densify URI

In the following URI templates, these variables are used:

* geometryServiceRootURI: the URL of the geometry service without any parameter

Table 30 – Densify reference

|  |  |
| --- | --- |
| **URI template** | {+geometryServiceRootURI}/densify{?f,geometries,sr,maxSegmentLength,lengthUnit} |
| **HTTP methods** | GETPOST (application/x-www-form-urlencoded) |
| **Parent Resource** | Geometry Service Root |

Table 31 – Densify parameters

|  |  |
| --- | --- |
| **Parameter** | **Details** |
| f | The response format |
|  | Required | Yes |
| Syntax | "json" |
| Example | f=json |
| geometries | The array of geometries to be densified. See GeoServices REST API – Core, subclause 9.8 for the schema of the geometry array. |
|  | Required | Yes |
| Syntax | JSON  |
| Example | geometries= {"geometryType":"GeometryPolyline","geometries":[{"paths":[[[-117,34],[-116,34],[-117,33]],[[-115,44],[-114,43],[-115,43]]]},{"paths":[[[32.49,17.83],[31.96,17.59],[30.87,17.01],[30.11,16.86]]]}]} |
| sr | The spatial reference of the input geometries. The spatial reference is specified as either a well-known ID (WKID) or a spatial reference JSON object. See the Geoservices REST API Core standard for the requirements related to spatial references. |
|  | Required | Yes |
| Syntax | JSON / POSINT |
| Example | sr=4326 |
| maxSegmentLength | All segments longer than the value specified for maxSegmentLength are replaced with sequences of lines no longer than the maximum segment length. |
|  | Required | Yes |
| Syntax | NUMBER |
| Example | maxSegmentLength=10000 |
| geodesic | A flag that can be set to true if a geodetic coordinate reference system is used. |
|  | Required | Yes |
| Syntax | Boolean |
| Example | geodesic=false |
| lengthUnit | The length unit of the maxSegmentLength value. This is specified as a numerical constant from the wkid property of an object in http://schemas.opengis.net/gsr/1.0/srLengthUnits.json. For example, the value for meters is 9001.  |
|  | Required | No. Default: if the sr is a projected coordinate reference system, the unit is determined by the unit of the spatial reference; otherwise, it is meter. |
| Syntax | POSINT |
| Example | lengthUnit=9001 |

**Request Requirements**

|  |
| --- |
| * + 1. A request on a Densify resource SHALL conform to the URI template in Table 30 and be accessed using a HTTP method identified in the same table.

densify/Request |

|  |
| --- |
| * + 1. A request on a Densify resource SHALL support all parameters and values specified in Table 31.

densify/ParametersBasic |

|  |
| --- |
| * + 1. The geometries parameter SHALL conform to the requirements for geometry arrays (see requirements class **http://www.opengis.net/spec/gsr-gs/1.0/req/geometry**) and contain a "geometryType" property.

densify/ParametersGeometries |

|  |
| --- |
| * + 1. The value of the sr parameter SHALL either be a well-known ID (WKID) or a spatial reference JSON object. A WKID value SHALL be a wkid property value of an object in http://schemas.opengis.net/gsr/1.0/sr.json.

densify/ParametersSR |

|  |
| --- |
| * + 1. If the lengthUnit parameter is present, it SHALL be a value of a wkid property of an object in http://schemas.opengis.net/gsr/1.0/srLengthUnits.json.

densify/ParametersLengthUnit |

## Geometry Array resources

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

densify/ResponseValid |

## Example

Densify a polyline:

http://example.com/rest/services/Geometry/GeometryServer/densify?sr=3395&

geometries={"geometryType":"GeometryPolyline","geometries":

[{"paths":[[[-17313284.793,2209625.866],[-17312808.186926104, 2210504.3164105085],[-17308518.732261017,2218410.3701050845],

[-17260185.82890302,2310809.9320710143],[-17307752.671522036,

2223194.8742101695],[-14501308.957,7392483.288],[-13773503.446,

6003036.405 ]]]}]}&maxSegmentLength=10000&geodesic=false&f=json

**Request**

GET /rest/services/Geometry/GeometryServer/densify?sr=3395&

geometries={"geometryType":"GeometryPolyline","geometries":

[{"paths":[[[-17313284.793,2209625.866],[-17312808.186926104, 2210504.3164105085],[-17308518.732261017,2218410.3701050845],

[-17260185.82890302,2310809.9320710143],[-17307752.671522036,

2223194.8742101695],[-14501308.957,7392483.288],[-13773503.446,

6003036.405 ]]]}]}&maxSegmentLength=10000&geodesic=false&f=json HTTP/1.1

Host: example.com

**Response**

HTTP/1.1 200 OK

Content-Type: application/json

Content-Length: nnn

{

 "geometryType" : "GeometryPolyline",

 "geometries" :

 [

 {

 "paths" :

 [

 [

 [-17313284.793, 2209625.866],

 [-17312808.1869261, 2210504.31641051],

 [-17308518.732261, 2218410.37010508],

 …

 [-13792046.2615669, 6038436.32558599],

 [-13787410.5576752, 6029586.34543949],

 [-13782774.8537834, 6020736.36529299],

 [-13778139.1498917, 6011886.3851465],

 [-13773503.446, 6003036.405]

 ]

 ]

 }

 ]

}

# Generalize

## Generalize Overview

The Generalize operation is performed on a Geometry Service resource. This operation densifies geometries by plotting points between existing vertices. Users can provide arguments to the Generize operation as query parameters.

Table 32 –Generalize overview

|  |  |  |
| --- | --- | --- |
| **Resource** | **Parameters** | **Resource representation** |
| Generalize | f=jsongeometriessrmaxDeviationdeviationUnit | JSON representation validAll JSON schema elements supported |

## Generalize URI

In the following URI templates, these variables are used:

* geometryServiceRootURI: the URL of the geometry service without any parameter

Table 33 – Generalize reference

|  |  |
| --- | --- |
| **URI template** | {+geometryServiceRootURI}/generalize{?f,geometries,sr,maxDeviation,deviationUnit} |
| **HTTP methods** | GETPOST (application/x-www-form-urlencoded) |
| **Parent Resource** | Geometry Service Root |

Table 34 – Generalize parameters

|  |  |
| --- | --- |
| **Parameter** | **Details** |
| f | The response format |
|  | Required | Yes |
| Syntax | "json" |
| Example | f=json |
| geometries | The array of geometries to be generalized. See GeoServices REST API – Core, subclause 9.8 for the schema of the geometry array. |
|  | Required | Yes |
| Syntax | JSON  |
| Example | geometries= {"geometryType":"GeometryPolyline","geometries":[{"paths":[[[-117,34],[-116,34],[-117,33]],[[-115,44],[-114,43],[-115,43]]]},{"paths":[[[32.49,17.83],[31.96,17.59],[30.87,17.01],[30.11,16.86]]]}]} |
| sr | The spatial reference of the input geometries. The spatial reference is specified as either a well-known ID (WKID) or a spatial reference JSON object. See the Geoservices REST API Core standard for the requirements related to spatial references. |
|  | Required | Yes |
| Syntax | JSON / POSINT |
| Example | sr=4326 |
| maxDeviation | Specifies the maximum deviation for constructing a generalized geometry based on the input geometries. |
|  | Required | Yes |
| Syntax | NUMBER |
| Example | maxDeviation=0.01 |
| deviationUnit | A unit for maximum deviation. This is specified as a numerical constant from the wkid property of an object in http://schemas.opengis.net/gsr/1.0/srLengthUnits.json. For example, the value for meters is 9001. |
|  | Required | No. Default: The unit is the unit used in the spatial reference. |
| Syntax | POSINT |
| Example | deviationUnit=9001 |

**Request Requirements**

|  |
| --- |
| * + 1. A request on a Generalize resource SHALL conform to the URI template in Table 33 and be accessed using a HTTP method identified in the same table.

generalize/Request |

|  |
| --- |
| * + 1. A request on a Generalize resource SHALL support all parameters and values specified in Table 34.

generalize/ParametersBasic |

|  |
| --- |
| * + 1. The geometries parameter SHALL conform to the requirements for geometry arrays (see requirements class **http://www.opengis.net/spec/gsr-gs/1.0/req/geometry**) and contain a "geometryType" property.

generalize/ParametersGeometries |

|  |
| --- |
| * + 1. The value of the sr parameter SHALL either be a well-known ID (WKID) or a spatial reference JSON object. A WKID value SHALL be a wkid property value of an object in http://schemas.opengis.net/gsr/1.0/sr.json.

generalize/ParametersSR |

|  |
| --- |
| * + 1. If the derivationUnit parameter is present, it SHALL be a value of a wkid property of an object in http://schemas.opengis.net/gsr/1.0/srLengthUnits.json.

generalize/ParametersDerivationUnit |

## Geometry Array resources

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

generalize/ResponseValid |

## Example

Generalize a polygon:

http://example.com/rest/services/Geometry/GeometryServer/generalize?sr=4326&geometries={"geometryType":"GeometryPolygon","geometries":[{"rings":[[

[-87.099342820011174,40.228084543758385],

[-87.100223146960218,40.381421968321966],

[-87.101720814594017,40.480793699969276],

[-87.100295810761097,40.57694817663144],

[-86.777024960686575,40.576769365423601],

[-86.700217232694484,40.574640906530412],

[-86.700551416568786,40.443071019286172],

[-86.704228938064603,40.225843915639118],

[-87.099342820011174,40.228084543758385]]]}]}&maxDeviation=0.01&f=json

**Request**

GET /rest/services/Geometry/GeometryServer/generalize?sr=4326&f=json&

geometries={"geometryType":"GeometryPolygon","geometries":[{"rings":[[

[-87.099342820011174,40.228084543758385],

[-87.100223146960218,40.381421968321966],

[-87.101720814594017,40.480793699969276],

[-87.100295810761097,40.57694817663144],

[-86.777024960686575,40.576769365423601],

[-86.700217232694484,40.574640906530412],

[-86.700551416568786,40.443071019286172],

[-86.704228938064603,40.225843915639118],

[-87.099342820011174,40.228084543758385]]]}]}&maxDeviation=0.01 HTTP/1.1

Host: example.com

**Response**

HTTP/1.1 200 OK

Content-Type: application/json

Content-Length: nnn

{

 "geometryType" : "GeometryPolygon",

 "geometries" :

 [

 {

 "rings" :

 [

 [

 [-87.0993428200112, 40.2280845437584],

 [-87.1002958107611, 40.5769481766314],

 [-86.7002172326945, 40.5746409065304],

 [-87.0993428200112, 40.2280845437584]

 ]

 ]

 }

 ]

}

# Convex Hull

## Convex Hull Overview

The Convex Hull operation is performed on a Geometry Service resource. It returns the convex hull of the input geometry. The input geometry can be a point, multipoint, polyline, or polygon. The hull is typically a polygon but can also be a polyline or point in degenerate cases.

Users can provide arguments to the Convex Hull operation as query parameters.

Table 35 –Convex Hull overview

|  |  |  |
| --- | --- | --- |
| **Resource** | **Parameters** | **Resource representation** |
| ConvexHull | f=jsongeometriessr | JSON representation validAll JSON schema elements supported |

## Convex Hull URI

In the following URI templates, these variables are used:

* geometryServiceRootURI: the URL of the geometry service without any parameter

Table 36 – Convex Hull reference

|  |  |
| --- | --- |
| **URI template** | {+geometryServiceRootURI}/convexHull{?f,geometries,sr} |
| **HTTP methods** | GETPOST (application/x-www-form-urlencoded) |
| **Parent Resource** | Geometry Service Root |

Table 37 – Convex Hull parameters

|  |  |
| --- | --- |
| **Parameter** | **Details** |
| f | The response format |
|  | Required | Yes |
| Syntax | "json" |
| Example | f=json |
| geometries | The geometries whose convex hull is to be created.See GeoServices REST API – Core, subclause 9.8 for the schema of the geometry array. |
|  | Required | Yes |
| Syntax | JSON  |
| Example | geometries= {"geometryType":"GeometryPolyline","geometries":[{"paths":[[[-117,34],[-116,34],[-117,33]],[[-115,44],[-114,43],[-115,43]]]},{"paths":[[[32.49,17.83],[31.96,17.59],[30.87,17.01],[30.11,16.86]]]}]} |
| sr | The spatial reference of the output geometry. The spatial reference is specified as either a well-known ID (WKID) or a spatial reference JSON object. See the Geoservices REST API Core standard for the requirements related to spatial references. |
|  | Required | Yes |
| Syntax | JSON / POSINT |
| Example | sr=4326 |

**Request Requirements**

|  |
| --- |
| * + 1. A request on a ConvexHull resource SHALL conform to the URI template in Table 36 and be accessed using a HTTP method identified in the same table.

convexHull/Request |

|  |
| --- |
| * + 1. A request on a ConvexHull resource SHALL support all parameters and values specified in Table 37.

convexHull/ParametersBasic |

|  |
| --- |
| * + 1. The geometries parameter SHALL conform to the requirements for geometry arrays (see requirements class **http://www.opengis.net/spec/gsr-gs/1.0/req/geometry**) and contain a "geometryType" property.

convexHull/ParametersGeometries |

|  |
| --- |
| * + 1. The value of the sr parameter SHALL either be a well-known ID (WKID) or a spatial reference JSON object. A WKID value SHALL be a wkid property value of an object in http://schemas.opengis.net/gsr/1.0/sr.json.

convexHull/ParametersSR |

## Polygon resources

|  |
| --- |
| * + 1. The JSON representation of a response to a Convex Hull resource request SHALL validate against the JSON Schema **http://schemas.opengis.net/gsr-gs/1.0/singleGeometry.json** and the value of the geometry property SHALL validate against the JSON Schema **http://schemas.opengis.net/gsr-gs/1.0/polygon.json** or in case of an exception against JSON Schema http://schemas.opengis.net/gsr/1.0/exception.json.

convexHull/ResponseValid |

## Example

Compute the convex hull for three points:

http://example.com/rest/services/Geometry/GeometryServer/convexHull?sr=4326&geometries={"geometryType":"GeometryPoint","geometries":[

{"x":-117.2332208251953,"y":34.086152645433316,

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

{"x":-117.21536804199218,"y":34.0854659999255,

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

{"x":-117.22498107910155,"y":34.06623992570675,

"spatialReference":{"wkid":4326}}]}&f=json

**Request**

GET /rest/services/Geometry/GeometryServer/convexHull? sr=4326&geometries={"geometryType":"GeometryPoint","geometries":[

{"x":-117.2332208251953,"y":34.086152645433316,

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

{"x":-117.21536804199218,"y":34.0854659999255,

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

{"x":-117.22498107910155,"y":34.06623992570675,

"spatialReference":{"wkid":4326}}]}&f=json HTTP/1.1

Host: example.com

**Response**

HTTP/1.1 200 OK

Content-Type: application/json

Content-Length: nnn

{

 "geometryType" : "GeometryPolygon",

 "geometry" :

 {

 "rings" :

 [

 [

 [-117.224981079102, 34.0662399257068],

 [-117.233220825195, 34.0861526454333],

 [-117.215368041992, 34.0854659999255],

 [-117.224981079102, 34.0662399257068]

 ]

 ]

 }

}

# Offset

## Offset Overview

The Offset operation is performed on a Geometry Service resource. Offset constructs the offset of the given input geometries. If the offset parameter is positive, the constructed offset will be on the right side of the geometry. Left side offsets are constructed with negative parameters.

Tracing the geometry from its first vertex to the last will give a direction along the geometry. It is to the right and left perspective of this direction that the positive and negative parameters dictate where the offset is constructed. In these terms, it can be inferred where the offset of even horizontal geometries will be constructed.

Users can provide arguments to the Offset operation as query parameters.

Table 38 –Offset overview

|  |  |  |
| --- | --- | --- |
| **Resource** | **Parameters** | **Resource representation** |
| Offset | f=jsongeometriessroffsetDistanceoffsetUnitoffsetHowbevelratio | JSON representation validAll JSON schema elements supported |

## Offset URI

In the following URI templates, these variables are used:

* geometryServiceRootURI: the URL of the geometry service without any parameter

Table 39 – Offset reference

|  |  |
| --- | --- |
| **URI template** | {+geometryServiceRootURI}/offset{?f,geometries,sr, offsetDistance,offsetUnit, offsetHow,bevelratio} |
| **HTTP methods** | GETPOST (application/x-www-form-urlencoded) |
| **Parent Resource** | Geometry Service Root |

Table 40 – Offset parameters

|  |  |
| --- | --- |
| **Parameter** | **Details** |
| f | The response format |
|  | Required | Yes |
| Syntax | "json" |
| Example | f=json |
| geometries | The array of geometries to be generalized. See GeoServices REST API – Core, subclause 9.8 for the schema of the geometry array. |
|  | Required | Yes |
| Syntax | JSON  |
| Example | geometries= {"geometryType":"GeometryPolyline","geometries":[{"paths":[[[-117,34],[-116,34],[-117,33]],[[-115,44],[-114,43],[-115,43]]]},{"paths":[[[32.49,17.83],[31.96,17.59],[30.87,17.01],[30.11,16.86]]]}]} |
| sr | The spatial reference of the input geometries. The spatial reference is specified as either a well-known ID (WKID) or a spatial reference JSON object. See the Geoservices REST API Core standard for the requirements related to spatial references. |
|  | Required | Yes |
| Syntax | JSON / POSINT |
| Example | sr=4326 |
| offsetDistance | Specifies the distance for constructing an offset based on the input geometries. If the offsetDistance parameter is positive, the constructed offset is on the right side of the curve. Left side offsets are constructed with negative values. |
|  | Required | Yes |
| Syntax | NUMBER |
| Example | offsetDistance=10 |
| offsetUnit | A unit for offset distance. This is specified as a numerical constant from the wkid property of an object in http://schemas.opengis.net/gsr/1.0/srLengthUnits.json. For example, the value for meters is 9001. |
|  | Required | No. Default: The unit is the unit used in the spatial reference. |
| Syntax | POSINT |
| Example | offsetUnit=9001 |
| offsetHow | This parameter determines how outer corners between segments are handled. * GeometryOffsetMitered: attempts to allow extended offsets to naturally intersect, but if that intersection occurs too far from the corner, the corner is eventually beveled at a fixed distance
* GeometryOffsetBevelled: squares off the corner after a given ratio distance (parameter bevelRatio)
* GeometryOffsetRounded: rounds the corner between extended offsets
 |
|  | Required | Yes |
| Syntax | "GeometryOffsetMitered" / "GeometryOffsetBevelled" / "GeometryOffsetRounded" |
| Example | offsetHow=GeometryOffsetMitered |
| bevelRatio | The bevelRatio value is multiplied by the offset distance, and the result determines how far a mitered offset intersection can be located before it is beveled. |
|  | Required | When GeometryOffsetMitered is specified, the input bevel ratio is ignored and 10 is used internally. When GeometryOffsetBevelled is specified, 1.1 is used if bevelRatio is not specified. The bevelRatio parameter is ignored for rounded offsets. |
| Syntax | NUMBER |
| Example | bevelRatio=2 |

**Request Requirements**

|  |
| --- |
| * + 1. A request on a Generalize resource SHALL conform to the URI template in Table 39 and be accessed using a HTTP method identified in the same table.

offset/Request |

|  |
| --- |
| * + 1. A request on a Generalize resource SHALL support all parameters and values specified in Table 40.

offset/ParametersBasic |

|  |
| --- |
| * + 1. The geometries parameter SHALL conform to the requirements for geometry arrays (see requirements class **http://www.opengis.net/spec/gsr-gs/1.0/req/geometry**) and contain a "geometryType" property.

offset/ParametersGeometries |

|  |
| --- |
| * + 1. The value of the sr parameter SHALL either be a well-known ID (WKID) or a spatial reference JSON object. A WKID value SHALL be a wkid property value of an object in http://schemas.opengis.net/gsr/1.0/sr.json.

offset/ParametersSR |

|  |
| --- |
| * + 1. If the offsetUnit parameter is present, it SHALL be a value of a wkid property of an object in http://schemas.opengis.net/gsr/1.0/srLengthUnits.json.

offset/ParametersOffsetUnit |

## Geometry Array resources

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

offset/ResponseValid |

## Example

Offset multiple polylines by one meter:

http://example.com/rest/services/Geometry/GeometryServer/offset?sr=2229&geometries={"geometryType":"GeometryPolyline","spatialReference":{"wkid":2229},"geometries":[{"paths":[[[6805566.1574656013,1846311.2154481949],[6805527.5463925907,1847577.0219133438],[6805567.9057296626,1846206.4173309559],[6805566.1574656013,1846311.2154481949],[6805489.2513970956,1845286.8614968264],[6805567.9057296626,1846206.4173309559]]]},{"paths":[[[6805493.9062097641,1845121.674091635],[6805489.2513970956,1845286.8614968264],[6805496.3884592885,1844963.0200417505],[6805493.9062097641,1845121.674091635],[6805512.6586404499,1843725.7847297059],[6805496.3884592885,1844963.0200417505],[6805514.211678369,1843607.5195617655],[6805512.6586404499,1843725.7847297059],[6805523.8066700343,1842901.206113206],[6805514.211678369,1843607.5195617655],[6805532.3821443468,1842246.5626597235],[6805523.8066700343,1842901.206113206]]]}]}&offsetDistance=1&offsetUnit=9001&offsetHow=GeometryOffsetMitered&bevelRatio=2f=json

**Request**

GET /rest/services/Geometry/GeometryServer/offset?sr=2229&geometries={"geometryType":"GeometryPolyline","spatialReference":{"wkid":2229},"geometries":[{"paths":[[[6805566.1574656013,1846311.2154481949],[6805527.5463925907,1847577.0219133438],[6805567.9057296626,1846206.4173309559],[6805566.1574656013,1846311.2154481949],[6805489.2513970956,1845286.8614968264],[6805567.9057296626,1846206.4173309559]]]},{"paths":[[[6805493.9062097641,1845121.674091635],[6805489.2513970956,1845286.8614968264],[6805496.3884592885,1844963.0200417505],[6805493.9062097641,1845121.674091635],[6805512.6586404499,1843725.7847297059],[6805496.3884592885,1844963.0200417505],[6805514.211678369,1843607.5195617655],[6805512.6586404499,1843725.7847297059],[6805523.8066700343,1842901.206113206],[6805514.211678369,1843607.5195617655],[6805532.3821443468,1842246.5626597235],[6805523.8066700343,1842901.206113206]]]}]}&offsetDistance=1&offsetUnit=9001&offsetHow=GeometryOffsetMitered&bevelRatio=2&f=json HTTP/1.1

Host: example.com

**Response**

HTTP/1.1 200 OK

Content-Type: application/json

Content-Length: nnn

{

 "geometryType" : "GeometryPolyline",

 "geometries" :

 [

 {

 "paths" :

 [

 [

 [6805569.43677369, 1846311.31547739],

 [6805530.62569506, 1847583.67882845],

 [6805524.07389809, 1847583.48243998],

 [6805564.81883934, 1846199.78272534],

 [6805571.29520759, 1846199.93209211],

 [6805569.3307883, 1846317.68744017],

 [6805563.36634781, 1846317.86122269],

 [6805562.87287955, 1846311.28844923]

 ]

 ]

 },

 {

 "paths" :

 [

 [

 [6805497.18574129, 1845121.76650549],

 [6805492.34638359, 1845293.50294242],

 [6805485.82700549, 1845293.33924976],

 [6805493.25276803, 1844956.39812766],

 [6805499.7713768, 1844956.52095142],

 [6805497.08404916, 1845128.28265713],

 [6805490.53757939, 1845128.18747252],

 [6805509.46623861, 1843719.18004924],

 [6805516.02546532, 1843719.26723603],

 [6805499.58272771, 1844969.62427911],

 [6805493.02163891, 1844969.53800327],

 [6805511.0173989, 1843600.91535478],

 [6805511.01872677, 1843600.91537222],

 [6805529.18919647, 1842239.95819138],

 [6805535.74863772, 1842240.04494229],

 [6805527.08722191, 1842901.24908666]

 ]

 ]

 }

 ]

}

# Trim/Extend

## Trim/Extend Overview

The Trim/Extend operation is performed on a Geometry Service resource. This operation trims or extends each polyline specified in the input array, applying the user-specified guide polylines. When trimming features, the part to the left of the oriented cutting line is preserved in the output and the other part is discarded. An empty polyline is added to the output array if the corresponding input polyline is neither cut nor extended.

Users can provide arguments to the Trim/Extend operation as query parameters.

Table 41 –Trim/Extend overview

|  |  |  |
| --- | --- | --- |
| **Resource** | **Parameters** | **Resource representation** |
| Trim/Extend | f=jsonpolylinestrimExtendTosrextendHow | JSON representation validAll JSON schema elements supported |

## Trim/Extend URI

In the following URI templates, these variables are used:

* geometryServiceRootURI: the URL of the geometry service without any parameter

Table 42 – Trim/Extend reference

|  |  |
| --- | --- |
| **URI template** | {+geometryServiceRootURI}/trimExtend{?f,polylines,trimExtndTo,sr,extendHow} |
| **HTTP methods** | GETPOST (application/x-www-form-urlencoded) |
| **Parent Resource** | Geometry Service Root |

Table 43 – Trim/Extend parameters

|  |  |
| --- | --- |
| **Parameter** | **Details** |
| f | The response format |
|  | Required | Yes |
| Syntax | "json" |
| Example | f=json |
| polylines | The array of polylines to be trimmed or extended. See GeoServices REST API – Core, subclause 9.4 for the schema of a polyline. |
|  | Required | Yes |
| Syntax | JSON  |
| Example | polylines=[{"paths":[[[-117,34],[-116,34],[-117,33]],[[-115,44],[-114,43],[-115,43]]]}, {"paths":[[[32.49,17.83],[31.96,17.59],[30.87,17.01],[30.11,16.86]]]}] |
| trimExtendTo | A polyline which is used as a guide for trimming / extending input polylines. The spatial reference of the polylines is specified by sr. See GeoServices REST API – Core, subclause 9.4 for the schema of a polyline. |
|  | Required | Yes |
| Syntax | JSON |
| Example | trimExtendTo={"paths":[[[-117,34],[-116,34],[-117,33]],[[-115,44],[-114,43],[-115,43]]]} |
| sr | The spatial reference of the input geometries (polylines and trimExtendTo). The spatial reference is specified as either a well-known ID (WKID) or a spatial reference JSON object. See the Geoservices REST API Core standard for the requirements related to spatial references. |
|  | Required | Yes |
| Syntax | JSON / POSINT |
| Example | sr=4326 |
| extendHow | Describes how to perform the Trim/Extend operation. The values:* 0: By default, extension considers both ends of paths. The old ends remain and new points are added to the extended ends. The new points have attributes that are extrapolated from adjacent existing segments.
* 1: If an extension is performed at an end, relocate the end point to the new position instead of leaving the old point and adding a new point at the new position.
* 2: If an extension is performed at an end, do not extrapolate the end segment's attributes for the new point. Instead, make its attributes the same as the current end.
* 4: If an extension is performed at an end, do not extrapolate the end segment's attributes for the new point. Instead, make its attributes empty.
* 8: Do not extend the from end of any path.
* 16: Do not extend the to end of any path.
 |
|  | Required | No. Default: 0. |
| Syntax | INTEGER |
| Example | extendHow=2 |

**Request Requirements**

|  |
| --- |
| * + 1. A request on a Trim/Extend resource SHALL conform to the URI template in Table 42 and be accessed using a HTTP method identified in the same table.

trimExtend/Request |

|  |
| --- |
| * + 1. A request on a Trim/Extend resource SHALL support all parameters and values specified in Table 43.

trimExtend/ParametersBasic |

|  |
| --- |
| * + 1. The polylines and the trimExtendTo parameters SHALL conform to the requirements for polyline objects (see requirements class **http://www.opengis.net/spec/gsr-gs/1.0/req/geometry**).

trimExtend/ParametersGeometries |

|  |
| --- |
| * + 1. The value of the sr parameter SHALL either be a well-known ID (WKID) or a spatial reference JSON object. A WKID value SHALL be a wkid property value of an object in http://schemas.opengis.net/gsr/1.0/sr.json.

trimExtend/ParametersSR |

## Geometry Array resources

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

trimExtend/ResponseValid |

## Example

Trims/Extend two polyline segments:

http://example.com/rest/services/Geometry/GeometryServer/trimExtend?sr=2229&polylines=[{"paths":[[[6805512.658537939,1843725.7846097648],[6805496.38855736,1844963.0199961811]]]},{"paths":[[[6805532.382251769,1842246.5625026077],[6805523.806809604,1842901.206206441]]]}]&trimExtendTo={"paths":[[[6804206.368171528,1843554.492957607],[6805395.769992188,1843570.1779655963],[6805514.211684436,1843607.5194263458],[6805740.688921779,1843619.888168022]]]}&extendHow=2&f=json

**Request**

GET /rest/services/Geometry/GeometryServer/trimExtend?sr=2229&polylines=[{"paths":[[[6805512.658537939,1843725.7846097648],[6805496.38855736,1844963.0199961811]]]},{"paths":[[[6805532.382251769,1842246.5625026077],[6805523.806809604,1842901.206206441]]]}]&trimExtendTo={"paths":[[[6804206.368171528,1843554.492957607],[6805395.769992188,1843570.1779655963],[6805514.211684436,1843607.5194263458],[6805740.688921779,1843619.888168022]]]}&extendHow=2&f=json HTTP/1.1

Host: example.com

**Response**

HTTP/1.1 200 OK

Content-Type: application/json

Content-Length: nnn

{

 "geometries" :

 [

 {

 "paths" :

 [

 [

 [6805514.21365289, 1843607.51942632],

 [6805512.65853788, 1843725.78460972],

 [6805496.3885573, 1844963.01999615]

 ]

 ]

 },

 {

 "paths" :

 [

 [

 [6805532.38225172, 1842246.56250256],

 [6805523.80680956, 1842901.2062064],

 [6805514.55420339, 1843607.53812706]

 ]

 ]

 }

 ]

}

# Auto Complete

## Auto Complete Overview

The Auto Complete operation is performed on a Geometry Service resource. The Auto Complete operation simplifies the process of constructing new polygons that are adjacent to other polygons. It constructs polygons that fill in the gaps between existing polygons and a set of polylines.

Users can provide arguments to the Auto Complete operation as query parameters.

Table 44 –Auto Complete overview

|  |  |  |
| --- | --- | --- |
| **Resource** | **Parameters** | **Resource representation** |
| Auto Complete | f=jsonpolygonspolylinessr | JSON representation validAll JSON schema elements supported |

## AutoComplete URI

In the following URI templates, these variables are used:

* geometryServiceRootURI: the URL of the geometry service without any parameter

Table 45 – Auto Complete reference

|  |  |
| --- | --- |
| **URI template** | {+geometryServiceRootURI}/autoComplete{?f,polygons,polylines,sr} |
| **HTTP methods** | GETPOST (application/x-www-form-urlencoded) |
| **Parent Resource** | Geometry Service Root |

Table 46 – Auto Complete parameters

|  |  |
| --- | --- |
| **Parameter** | **Details** |
| f | The response format |
|  | Required | Yes |
| Syntax | "json" |
| Example | f=json |
| polygons | The array of polygons that will provide some boundaries for new polygons. The spatial reference of the polygons is specified by sr. See GeoServices REST API – Core, subclause 9.5 for the schema of a polygon. |
|  | Required | Yes |
| Syntax | JSON  |
| Example | polygons=[{"rings":[[[-117,34],[-116,34],[-117,33],[-117,34]],[[-115,44],[-114,43],[-115,43],[-115,44]]]},{"rings":[[[32.49,17.83],[31.96,17.59],[30.87,17.01],[30.11,16.86],[32.49,17.83]]]}] |
| polylines | An array of polylines that will provide the remaining boundaries for new polygons. The spatial reference of the polylines is specified by sr. See GeoServices REST API – Core, subclause 9.4 for the schema of a polyline. |
|  | Required | Yes |
| Syntax | JSON  |
| Example | polylines=[{"paths":[[[-117,34],[-116,34],[-117,33]],[[-115,44],[-114,43],[-115,43]]]}, {"paths":[[[32.49,17.83],[31.96,17.59],[30.87,17.01],[30.11,16.86]]]}] |
| sr | The spatial reference of the input geometries (polylines and polygons). The spatial reference is specified as either a well-known ID (WKID) or a spatial reference JSON object. See the Geoservices REST API Core standard for the requirements related to spatial references. |
|  | Required | Yes |
| Syntax | JSON / POSINT |
| Example | sr=4326 |

**Request Requirements**

|  |
| --- |
| * + 1. A request on a Auto Complete resource SHALL conform to the URI template in Table 45 and be accessed using a HTTP method identified in the same table.

autoComplete/Request |

|  |
| --- |
| * + 1. A request on a Auto Complete resource SHALL support all parameters and values specified in Table 46.

autoComplete/ParametersBasic |

|  |
| --- |
| * + 1. The polylines and the polygons parameters SHALL conform to the requirements for polyline and polygon objects respectively (see requirements class **http://www.opengis.net/spec/gsr-gs/1.0/req/geometry**).

autoComplete/ParametersGeometries |

|  |
| --- |
| * + 1. The value of the sr parameter SHALL either be a well-known ID (WKID) or a spatial reference JSON object. A WKID value SHALL be a wkid property value of an object in http://schemas.opengis.net/gsr/1.0/sr.json.

autoComplete/ParametersSR |

## Geometry Array resources

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

autoComplete/ResponseValid |

## Example

Creates a polygon from a base polygon and a polyline:

http://example.com/rest/services/Geometry/GeometryServer/autoComplete?sr=4269&polygons=[{"rings":[[[0,0],[110,0],[110,-60],[0,-60],[0,0]],[[120,0],[180,0],[180,-60],[120,-60],[120,0]]]}]&polylines=[{"paths":[[[109,0],[121,0]],[[109,-60],[121,-60]]]}]&f=json

**Request**

GET /rest/services/Geometry/GeometryServer/autoComplete?sr=4269& polygons=[{"rings":[[[0,0],[110,0],[110,-60],[0,-60],[0,0]],[[120,0],[180,0],[180,-60],[120,-60],[120,0]]]}]&polylines=[{"paths":[[[109,0],[121,0]],[[109,-60],[121,-60]]]}]&f=json

 HTTP/1.1

Host: example.com

**Response**

HTTP/1.1 200 OK

Content-Type: application/json

Content-Length: nnn

{

 "geometries" :

 [

 {

 "spatialReference" : {

 "wkid" : 4269

 },

 "rings" :

 [

 [

 [120, -59.9999999999999],

 [110, -59.9999999999999],

 [110, 5.6843418860808E-14],

 [120, 5.6843418860808E-14],

 [120, -59.9999999999999]

 ]

 ]

 }

 ]

}

# Cut

## Cut Overview

The Cut operation is performed on a Geometry Service resource. This operation splits the input polyline or polygon where it crosses a cutting polyline.

Users can provide arguments to the Cut operation as query parameters.

Table 47 –Cut overview

|  |  |  |
| --- | --- | --- |
| **Resource** | **Parameters** | **Resource representation** |
| Auto Complete | f=jsoncuttertargetsr | JSON representation validAll JSON schema elements supported |

## Cut URI

In the following URI templates, these variables are used:

* geometryServiceRootURI: the URL of the geometry service without any parameter

Table 48 – Cut reference

|  |  |
| --- | --- |
| **URI template** | {+geometryServiceRootURI}/cut{?f,cutter,target,sr} |
| **HTTP methods** | GETPOST (application/x-www-form-urlencoded) |
| **Parent Resource** | Geometry Service Root |

Table 49 – Cut parameters

|  |  |
| --- | --- |
| **Parameter** | **Details** |
| f | The response format |
|  | Required | Yes |
| Syntax | "json" |
| Example | f=json |
| cutter | The polyline that will be used to divide the target into pieces where it crosses the target. The spatial reference of the polylines is specified by the sr parameter. See GeoServices REST API – Core, subclause 9.4 for the schema of a polyline. |
|  | Required | Yes |
| Syntax | JSON  |
| Example | cutter= {"paths":[[[-117,34],[-116,34],[-117,33]],[[-115,44],[-114,43],[-115,43]]]} |
| target | The array of polylines/polygons to be cut. The spatial reference of the polylines is specified by sr. See GeoServices REST API – Core, subclause 9.8 for the schema of a geometry array. |
|  | Required | Yes |
| Syntax | JSON  |
| Example | target={"geometryType":"GeometryPolyline","geometries":[{"paths":[[[-117,34],[-116,34],[-117,33]],[[-115,44],[-114,43],[-115,43]]]},{"paths":[[[32.49,17.83],[31.96,17.59],[30.87,17.01],[30.11,16.86]]]}]} |
| sr | The spatial reference of the input geometries (cutter and target). The spatial reference is specified as either a well-known ID (WKID) or a spatial reference JSON object. See the Geoservices REST API Core standard for the requirements related to spatial references. |
|  | Required | Yes |
| Syntax | JSON / POSINT |
| Example | sr=4326 |

**Request Requirements**

|  |
| --- |
| * + 1. A request on a Cut resource SHALL conform to the URI template in Table 48 and be accessed using a HTTP method identified in the same table.

cut/Request |

|  |
| --- |
| * + 1. A request on a Cut resource SHALL support all parameters and values specified in Table 49.

cut/ParametersBasic |

|  |
| --- |
| * + 1. The cutter and the target parameters SHALL conform to the requirements for polyline and geometry array objects respectively (see requirements class **http://www.opengis.net/spec/gsr-gs/1.0/req/geometry**).

cut/ParametersGeometries |

|  |
| --- |
| * + 1. The value of the sr parameter SHALL either be a well-known ID (WKID) or a spatial reference JSON object. A WKID value SHALL be a wkid property value of an object in http://schemas.opengis.net/gsr/1.0/sr.json.

cut/ParametersSR |

## Geometry Array resources

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

cut/ResponseValid |

## Example

Cut a polyline geometry:

http://example.com/rest/services/Geometry/GeometryServer/cut?sr=2229&target={"geometryType":"GeometryPolyline","spatialReference":{"wkid":2229},"geometries":[{"paths":[[[6805743.810634688,1843230.507057026],[6805740.688921779,1843619.888168022],[6802621.935316771,1843581.5805018544],[6805496.38855736,1844963.0199961811]]]}]}&cutter={"paths":[[[6805210,1843869],[6805842,1843529],[6805259,1843173]]]}&f=json

**Request**

GET /rest/services/Geometry/GeometryServer/cut?sr=2229& target={"geometryType":"GeometryPolyline","spatialReference":{"wkid":2229},"geometries":[{"paths":[[[6805743.810634688,1843230.507057026],[6805740.688921779,1843619.888168022],[6802621.935316771,1843581.5805018544],[6805496.38855736,1844963.0199961811]]]}]}&cutter={"paths":[[[6805210,1843869],[6805842,1843529],[6805259,1843173]]]}&f=json

 HTTP/1.1

Host: example.com

**Response**

HTTP/1.1 200 OK

Content-Type: application/json

Content-Length: nnn

{

 "geometryType" : "GeometryPolyline",

 "geometries" :

 [

 {

 "paths" :

 [

 [

 [6805743.81063464, 1843230.50705698],

 [6805741.90775131, 1843467.87994181]

 ],

 [

 [6805740.98190014, 1843583.34526207],

 [6805740.68892172, 1843619.88816798],

 [6805674.56471014, 1843619.07583365]

 ]

 ]

 },

 {

 "paths" :

 [

 [

 [6805741.90775131, 1843467.87994181],

 [6805740.98190014, 1843583.34526207]

 ],

 [

 [6805674.56471014, 1843619.07583365],

 [6802621.93531673, 1843581.58050181],

 [6805496.3885573, 1844963.01999615]

 ]

 ]

 }

 ],

 "cutIndexes" : [

 0,

 0

 ]

}

# Difference

## Difference Overview

The Difference operation is performed on a Geometry Service resource. This operation constructs the set-theoretic difference between an array of geometries and another geometry

Users can provide arguments to the Difference operation as query parameters.

Table 50 –Difference overview

|  |  |  |
| --- | --- | --- |
| **Resource** | **Parameters** | **Resource representation** |
| Difference | f=jsongeometriesgeometrysr | JSON representation validAll JSON schema elements supported |

## Difference URI

In the following URI templates, these variables are used:

* geometryServiceRootURI: the URL of the geometry service without any parameter

Table 51 – Difference reference

|  |  |
| --- | --- |
| **URI template** | {+geometryServiceRootURI}/difference{?f,geometries,geometry,sr} |
| **HTTP methods** | GETPOST (application/x-www-form-urlencoded) |
| **Parent Resource** | Geometry Service Root |

Table 52 – Difference parameters

|  |  |
| --- | --- |
| **Parameter** | **Details** |
| f | The response format |
|  | Required | Yes |
| Syntax | "json" |
| Example | f=json |
| geometries | An array of points, multipoints, polylines or polygons. See GeoServices REST API – Core, subclause 9.8 for the schema of a geometry array. |
|  | Required | Yes |
| Syntax | JSON  |
| Example | geometries={"geometryType":"GeometryPolyline","geometries":[{"paths":[[[-117,34],[-116,34],[-117,33]],[[-115,44],[-114,43],[-115,43]]]},{"paths":[[[32.49,17.83],[31.96,17.59],[30.87,17.01],[30.11,16.86]]]}]} |
| geometry | A single geometry of any type, of dimension equal to or greater than the elements of "geometries". See GeoServices REST API – Core, subclauses 9.3 to 9.7 for the schema of geometry objects. |
|  | Required | Yes |
| Syntax | JSON  |
| Example | geometry={"geometryType":"GeometryPolygon","geometry":{"rings":[[[-117,34],[-116,34],[-117,33],[-117,34]],[[-115,44],[-114,43],[-115,43],[-115,44]]]}} |
| sr | The spatial reference of the input geometries (geometries and geometry). The spatial reference is specified as either a well-known ID (WKID) or a spatial reference JSON object. See the Geoservices REST API Core standard for the requirements related to spatial references. |
|  | Required | Yes |
| Syntax | JSON / POSINT |
| Example | sr=4326 |

**Request Requirements**

|  |
| --- |
| * + 1. A request on a Difference resource SHALL conform to the URI template in Table 51 and be accessed using a HTTP method identified in the same table.

difference/Request |

|  |
| --- |
| * + 1. A request on a Difference resource SHALL support all parameters and values specified in Table 52.

difference/ParametersBasic |

|  |
| --- |
| * + 1. The geometries and the geometry parameters SHALL conform to the requirements for polyline and geometry array objects respectively (see requirements class **http://www.opengis.net/spec/gsr-gs/1.0/req/geometry**). The geometry in the geometry parameter SHALL be of a dimension equal to or greater than the geometries in the geometries parameter.

difference/ParametersGeometries |

|  |
| --- |
| * + 1. The value of the sr parameter SHALL either be a well-known ID (WKID) or a spatial reference JSON object. A WKID value SHALL be a wkid property value of an object in http://schemas.opengis.net/gsr/1.0/sr.json.

difference/ParametersSR |

## Geometry Array resources

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

difference/ResponseValid |

## Example

Calculate the difference between two polygons:

http://example.com/rest/services/Geometry/GeometryServer/difference?sr=4269&geometries={"geometryType":"GeometryPolygon","spatialReference":{"wkid":4269},"geometries":[{"rings":[[[-75.489280661,39.714858219],[-75.475974268,39.720084384],[-75.474768457,39.741832093],[-75.460394119,39.763362027],[-74.738824727,40.177725644],[-74.916654342,39.170638542],[-75.014407077,39.198363837],[-75.119958112,39.184691781],[-75.415672275,39.374971842],[-75.55276304,39.490514307],[-75.516688884,39.566568416],[-75.570234187,39.617734963],[-75.489280661,39.714858219]]]}]}&geometry={"geometryType":"GeometryPolygon","spatialReference":{"wkid":4269},"geometry":{"rings":[[[-75.460394119,39.763362027],[-74.738824727,40.177725644],[-75.460394119,39.763362027]]]}}&f=json

**Request**

GET /rest/services/Geometry/GeometryServer/difference?sr=4269&geometries={"geometryType":"GeometryPolygon","spatialReference":{"wkid":4269},"geometries":[{"rings":[[[-75.489280661,39.714858219],[-75.475974268,39.720084384],[-75.474768457,39.741832093],[-75.460394119,39.763362027],[-74.738824727,40.177725644],[-74.916654342,39.170638542],[-75.014407077,39.198363837],[-75.119958112,39.184691781],[-75.415672275,39.374971842],[-75.55276304,39.490514307],[-75.516688884,39.566568416],[-75.570234187,39.617734963],[-75.489280661,39.714858219]]]}]}&geometry={"geometryType":"GeometryPolygon","spatialReference":{"wkid":4269},"geometry":{"rings":[[[-75.460394119,39.763362027],[-74.738824727,40.177725644],[-75.460394119,39.763362027]]]}}&f=json HTTP/1.1

Host: example.com

**Response**

HTTP/1.1 200 OK

Content-Type: application/json

Content-Length: nnn

{

 "geometryType" : "GeometryPolygon",

 "geometries" :

 [

 {

 "rings" :

 [

 [

 [-74.738824727, 40.1777256440001],

 [-74.916654342, 39.1706385420001],

 [-75.014407077, 39.198363837],

 [-75.1199581119999, 39.184691781],

 [-75.415672275, 39.374971842],

 [-75.5527630399999, 39.4905143070001],

 [-75.516688884, 39.5665684160001],

 [-75.570234187, 39.617734963],

 [-75.489280661, 39.7148582190001],

 [-75.475974268, 39.7200843840001],

 [-75.474768457, 39.741832093],

 [-75.4603941189999, 39.763362027],

 [-74.738824727, 40.1777256440001]

 ]

 ]

 }

 ]

}

# Intersect

## Intersect Overview

The Intersect operation is performed on a Geometry Service resource. This operation constructs the set-theoretic intersection between an array of geometries and another geometry.

Users can provide arguments to the Intersect operation as query parameters.

Table 53 –Intersect overview

|  |  |  |
| --- | --- | --- |
| **Resource** | **Parameters** | **Resource representation** |
| Intersects | f=jsongeometriesgeometrysr | JSON representation validAll JSON schema elements supported |

## Intersect URI

In the following URI templates, these variables are used:

* geometryServiceRootURI: the URL of the geometry service without any parameter

Table 54 – Intersect reference

|  |  |
| --- | --- |
| **URI template** | {+geometryServiceRootURI}/inetersect{?f,geometries,geometry,sr} |
| **HTTP methods** | GETPOST (application/x-www-form-urlencoded) |
| **Parent Resource** | Geometry Service Root |

Table 55 – Difference parameters

|  |  |
| --- | --- |
| **Parameter** | **Details** |
| f | The response format |
|  | Required | Yes |
| Syntax | "json" |
| Example | f=json |
| geometries | An array of points, multipoints, polylines or polygons. See GeoServices REST API – Core, subclause 9.8 for the schema of a geometry array. |
|  | Required | Yes |
| Syntax | JSON  |
| Example | geometries={"geometryType":"GeometryPolyline","geometries":[{"paths":[[[-117,34],[-116,34],[-117,33]],[[-115,44],[-114,43],[-115,43]]]},{"paths":[[[32.49,17.83],[31.96,17.59],[30.87,17.01],[30.11,16.86]]]}]} |
| geometry | A single geometry of any type, of dimension equal to or greater than the elements of "geometries". See GeoServices REST API – Core, subclauses 9.3 to 9.7 for the schema of geometry objects. |
|  | Required | Yes |
| Syntax | JSON  |
| Example | geometry={"geometryType":"GeometryPolygon","geometry":{"rings":[[[-117,34],[-116,34],[-117,33],[-117,34]],[[-115,44],[-114,43],[-115,43],[-115,44]]]}} |
| sr | The spatial reference of the input geometries (geometries and geometry). The spatial reference is specified as either a well-known ID (WKID) or a spatial reference JSON object. See the Geoservices REST API Core standard for the requirements related to spatial references. |
|  | Required | Yes |
| Syntax | JSON / POSINT |
| Example | sr=4326 |

**Request Requirements**

|  |
| --- |
| * + 1. A request on a Intersect resource SHALL conform to the URI template in Table 51 and be accessed using a HTTP method identified in the same table.

intersect/Request |

|  |
| --- |
| * + 1. A request on a Inetrsect resource SHALL support all parameters and values specified in Table 52.

intersect/ParametersBasic |

|  |
| --- |
| * + 1. The geometries and the geometry parameters SHALL conform to the requirements for polyline and geometry array objects respectively (see requirements class **http://www.opengis.net/spec/gsr-gs/1.0/req/geometry**). The geometry in the geometry parameter SHALL be of a dimension equal to or greater than the geometries in the geometries parameter.

intersect/ParametersGeometries |

|  |
| --- |
| * + 1. The value of the sr parameter SHALL either be a well-known ID (WKID) or a spatial reference JSON object. A WKID value SHALL be a wkid property value of an object in http://schemas.opengis.net/gsr/1.0/sr.json.

intersect/ParametersSR |

## Geometry Array resources

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

intersect/ResponseValid |

## Example

Calculate the inetrsection between two polygons:

http://example.com/rest/services/Geometry/GeometryServer/intersect?sr=4269&f=json&geometries={"geometryType":"GeometryPolygon","spatialReference":{"wkid":4269},"geometries":[{"rings":[[

[-75.48928066099995,39.714858219000064],

[-75.4759742679999,39.720084384000074],

[-75.47476845699993,39.741832093000085],

[-75.46039411899994,39.763362027000085],

[-74.73882472699995,40.17772564400008],

[-74.9166543419999,39.17063854200006],

[-75.01440707699993,39.198363837000045],

[-75.11995811199995,39.18469178100008],

[-75.4156722749999,39.374971842000036],

[-75.55276303999995,39.49051430700007],

[-75.5166888839999,39.56656841600005],

[-75.57023418699993,39.61773496300009],

[-75.48928066099995,39.714858219000064]]]}]}& geometry={"geometryType":"GeometryPolygon","spatialReference":{"wkid":4269},"geometry":{"rings":[[

[-75.48928066099995,39.714858219000064],

[-75.4759742679999,39.720084384000074],

[-75.47476845699993,39.741832093000085],

[-75.46039411899994,39.763362027000085],

[-74.73882472699995,40.17772564400008],

[-75.48928066099995,39.714858219000064]]]}}

**Request**

GET /rest/services/Geometry/GeometryServer/intersect?sr=4269&f=json&geometries={"geometryType":"GeometryPolygon","spatialReference":{"wkid":4269},"geometries":[{"rings":[[

[-75.48928066099995,39.714858219000064],

[-75.4759742679999,39.720084384000074],

[-75.47476845699993,39.741832093000085],

[-75.46039411899994,39.763362027000085],

[-74.73882472699995,40.17772564400008],

[-74.9166543419999,39.17063854200006],

[-75.01440707699993,39.198363837000045],

[-75.11995811199995,39.18469178100008],

[-75.4156722749999,39.374971842000036],

[-75.55276303999995,39.49051430700007],

[-75.5166888839999,39.56656841600005],

[-75.57023418699993,39.61773496300009],

[-75.48928066099995,39.714858219000064]]]}]}& geometry={"geometryType":"GeometryPolygon","spatialReference":{"wkid":4269},"geometry":{"rings":[[

[-75.48928066099995,39.714858219000064],

[-75.4759742679999,39.720084384000074],

[-75.47476845699993,39.741832093000085],

[-75.46039411899994,39.763362027000085],

[-74.73882472699995,40.17772564400008],

[-75.48928066099995,39.714858219000064]]]}} HTTP/1.1

Host: example.com

**Response**

HTTP/1.1 200 OK

Content-Type: application/json

Content-Length: nnn

{

 "geometryType" : "GeometryPolygon",

 "geometries" :

 [

 {

 "rings" :

 [

 [

 [-75.475803134, 39.7231709100001],

 [-75.474768457, 39.741832093],

 [-75.4603941189999, 39.763362027],

 [-74.738824727, 40.1777256440001],

 [-75.475803134, 39.7231709100001]

 ]

 ]

 }

 ]

}

# Reshape

## Reshape Overview

The Reshape operation is performed on a Geometry Service resource. It reshapes a polyline or a part of a polygon using a reshaping line.

Users can provide arguments to the Reshape operation as query parameters.

Table 56 –Reshape overview

|  |  |  |
| --- | --- | --- |
| **Resource** | **Parameters** | **Resource representation** |
| Reshape | f=jsontargetreshapersr | JSON representation validAll JSON schema elements supported |

## Reshape URI

In the following URI templates, these variables are used:

* geometryServiceRootURI: the URL of the geometry service without any parameter

Table 57 – Reshape reference

|  |  |
| --- | --- |
| **URI template** | {+geometryServiceRootURI}/reshape{?f,target,reshaper,sr} |
| **HTTP methods** | GETPOST (application/x-www-form-urlencoded) |
| **Parent Resource** | Geometry Service Root |

Table 58 – Reshape parameters

|  |  |
| --- | --- |
| **Parameter** | **Details** |
| f | The response format |
|  | Required | Yes |
| Syntax | "json" |
| Example | f=json |
| target | The polyline or polygon to be reshaped.See GeoServices REST API – Core, subclauses 9.4 and 9.5 for the schema of polyline and polygon respectively. |
|  | Required | Yes |
| Syntax | JSON  |
| Example | target={"geometryType":"GeometryPolygon","geometry":{"rings":[[[-117,34],[-116,34],[-117,33],[-117,34]],[[-115,44],[-114,43],[-115,43],[-115,44]]]}} |
| reshaper | The single-part polyline that does the reshaping.See GeoServices REST API – Core, subclauses 9.4 for the schema of polyline. |
|  | Required | Yes |
| Syntax | JSON  |
| Example | reshaper={"paths":[[[-117,34],[-116,34],[-117,33]],[[-115,44],[-114,43],[-115,43]]]} |
| sr | The spatial reference of the input geometries. The spatial reference is specified as either a well-known ID (WKID) or a spatial reference JSON object. See the Geoservices REST API Core standard for the requirements related to spatial references. |
|  | Required | Yes |
| Syntax | JSON / POSINT |
| Example | sr=4326 |

**Request Requirements**

|  |
| --- |
| * + 1. A request on a Reshape resource SHALL conform to the URI template in Table 57 and be accessed using a HTTP method identified in the same table.

reshape/Request |

|  |
| --- |
| * + 1. A request on a Reshape resource SHALL support all parameters and values specified in Table 58.

reshape/ParametersBasic |

|  |
| --- |
| * + 1. The target and reshaper parameters SHALL conform to the requirements for polyline and polygon objects (see requirements class **http://www.opengis.net/spec/gsr-gs/1.0/req/geometry**). The target property SHALL contain a "geometryType" property. The reshaper property SHALL consist of a single path.

reshape/ParametersGeometries |

|  |
| --- |
| * + 1. The value of the sr parameter SHALL either be a well-known ID (WKID) or a spatial reference JSON object. A WKID value SHALL be a wkid property value of an object in http://schemas.opengis.net/gsr/1.0/sr.json.

reshape/ParametersSR |

## Single Geometry resources

|  |
| --- |
| * + 1. The JSON representation of a response to a Reshape resource request SHALL validate against the JSON Schema **http://schemas.opengis.net/gsr-gs/1.0/singleGeometry.json** and the value of the geometry property SHALL validate against either the JSON Schema **http://schemas.opengis.net/gsr-gs/1.0/polyline.json** or **http://schemas.opengis.net/gsr-gs/1.0/polygon.json** - or in case of an exception against JSON Schema http://schemas.opengis.net/gsr/1.0/exception.json.

reshape/ResponseValid |

If the reshaper geometry does not intersect the target polyline, or does not define a closed loop for the target polygon, then an empty geometry of the same type as the target geometry is returned. The orientation of Reshaper is not considered.

The following rules for polygon and polyline reshaping are used:

Polygons: Only one ring is explicitly reshaped – the first one in index order that intersects the reshaping path. The longest portion of the original ring boundary is preserved. The reshaped polygon is re-simplified after the reshape operation has been applied, so the rules for simplification will determine what happens if multiple rings interact with the reshaping path. Figure 3 illustratess how preserving the longest portion of the boundary can invert the ring in some cases. If more complicated reshaping behavior is desired, a combination of auto-complete, cutting and union can be used.

|  |  |
| --- | --- |
| http://atlas.resources.ca.gov/ArcGIS/SDK/SOAP/images/ReshapePolygon1.png | http://atlas.resources.ca.gov/ArcGIS/SDK/SOAP/images/ReshapePolygon2.png |
| http://atlas.resources.ca.gov/ArcGIS/SDK/SOAP/images/ReshapePolygon3.png | http://atlas.resources.ca.gov/ArcGIS/SDK/SOAP/images/ReshapePolygon4.png |

 Figure 3 - Reshaper operation on a polygon

Polylines: Multiple paths can be affected by the reshaping path. The extreme points of intersection of the reshaping path with the polyline determine the portion of the reshaping path to be included in the output. Only those points are used to modify the polyline. No other intersection points are inserted into the result.

Two examples are shown below. The circled intersection points are the only ones that will be added to the output.

|  |  |
| --- | --- |
| http://atlas.resources.ca.gov/ArcGIS/SDK/SOAP/images/ReshapePolylineA.png |  http://atlas.resources.ca.gov/ArcGIS/SDK/SOAP/images/ReshapePolylineB.png |
| http://atlas.resources.ca.gov/ArcGIS/SDK/SOAP/images/ReshapePolylineC.png | http://atlas.resources.ca.gov/ArcGIS/SDK/SOAP/images/ReshapePolylineD.png |

  Figure 4 - Reshaper operation on a single-part polyline

Figure 4 illustrates reshaping a polyline with a path that intersects the polyline in several locations. Only the extreme intersection points relative to the reshaping path are used in the result.

When the extreme points of intersection on the reshaping path are on two different parts, the orientation of the polyline determines which segments of those parts are included in the output. Other parts, intersecting or not, remain unaffected and are copied to the result. The next figure shows several cases. The orientation of the polyline is shown using red arrows.

|  |  |
| --- | --- |
| http://atlas.resources.ca.gov/ArcGIS/SDK/SOAP/images/ReshapePolyline1.png | http://atlas.resources.ca.gov/ArcGIS/SDK/SOAP/images/ReshapePolyline2.png |
| http://atlas.resources.ca.gov/ArcGIS/SDK/SOAP/images/ReshapePolyline3.png | http://atlas.resources.ca.gov/ArcGIS/SDK/SOAP/images/ReshapePolyline4.png |
| http://atlas.resources.ca.gov/ArcGIS/SDK/SOAP/images/ReshapePolyline5A.png | http://atlas.resources.ca.gov/ArcGIS/SDK/SOAP/images/ReshapePolyline6A.png |

   Figure 5 - Reshaper operation on a multipart polyline

Figure 5 illustrates several cases of reshaping a multipart polyline. The orientation of the polyline is shown with red arrows in the upper left graphic. In the first row, the unaffected part is copied to the output. In the second row, the orientation of the parts, and the ordering of intersection points on the reshaping path, determine which portion of the parts is preserved in the output. In the third case, the upper part is copied to output unchanged because the extreme points of intersection relative to the reshaping path are not on it.

## Example

Reshape a polygon:

http://example.com/rest/services/Geometry/GeometryServer/reshape?sr=2229&f=json&target={"geometryType":"GeometryPolygon","spatialReference":{"wkid":2229},"geometry":{"rings":[[[6807691.607592106,1841423.2521413565],[6807403.895241022,1841226.589476943],[6807330.383577019,1841176.5013225228],[6807144.497465864,1841049.4458023459],[6806716.6797519475,1840757.4020951092],[6806479.870514274,1840595.731486693],[6806183.101422772,1840393.1846476793],[6806075.24238652,1840319.1683915257],[6805779.309579447,1840125.002769351],[6805524.879970193,1839930.444431439],[6804797.212462276,1841029.5203172714],[6804973.141244277,1841152.8425771892],[6805745.21122244,1841695.4566494375],[6806023.85141319,1841890.9388700128],[6806319.65692395,1842098.571656853],[6806763.038583115,1842410.1881596],[6806913.661969528,1842516.0485121906],[6807150.782230198,1842187.7000875175],[6807308.85212402,1841967.8783486933],[6807691.258839533,1841423.802665189]]]}}&reshaper={"paths":[[[6804973.141244277,1841152.8425771892],[6804797.212462276,1841029.5203172714],[6804463.906370357,1841533.088863939],[6804224.755601943,1841930.77901344],[6804233.406175196,1842251.835129857],[6804206.368171528,1843554.492957607],[6805395.769992188,1843570.1779655963],[6805514.211684436,1843607.5194263458],[6805740.688921779,1843619.888168022],[6806080.253859445,1843657.1859936863],[6806290.270171687,1843380.829262942],[6806717.911376774,1842787.4210009426],[6806913.661969528,1842516.0485121906],[6806763.038583115,1842410.1881596],[6806319.65692395,1842098.571656853],[6806027.585329607,1841887.9188629389],[6805745.21122244,1841695.4566494375],[6804969.741316691,1841150.3071491867]]]}

**Request**

GET /rest/services/Geometry/GeometryServer/reshape?sr=2229&f=json& target={"geometryType":"GeometryPolygon","spatialReference":{"wkid":2229},"geometry":{"rings":[[[6807691.607592106,1841423.2521413565],[6807403.895241022,1841226.589476943],[6807330.383577019,1841176.5013225228],[6807144.497465864,1841049.4458023459],[6806716.6797519475,1840757.4020951092],[6806479.870514274,1840595.731486693],[6806183.101422772,1840393.1846476793],[6806075.24238652,1840319.1683915257],[6805779.309579447,1840125.002769351],[6805524.879970193,1839930.444431439],[6804797.212462276,1841029.5203172714],[6804973.141244277,1841152.8425771892],[6805745.21122244,1841695.4566494375],[6806023.85141319,1841890.9388700128],[6806319.65692395,1842098.571656853],[6806763.038583115,1842410.1881596],[6806913.661969528,1842516.0485121906],[6807150.782230198,1842187.7000875175],[6807308.85212402,1841967.8783486933],[6807691.258839533,1841423.802665189]]]}}&reshaper={"paths":[[[6804973.141244277,1841152.8425771892],[6804797.212462276,1841029.5203172714],[6804463.906370357,1841533.088863939],[6804224.755601943,1841930.77901344],[6804233.406175196,1842251.835129857],[6804206.368171528,1843554.492957607],[6805395.769992188,1843570.1779655963],[6805514.211684436,1843607.5194263458],[6805740.688921779,1843619.888168022],[6806080.253859445,1843657.1859936863],[6806290.270171687,1843380.829262942],[6806717.911376774,1842787.4210009426],[6806913.661969528,1842516.0485121906],[6806763.038583115,1842410.1881596],[6806319.65692395,1842098.571656853],[6806027.585329607,1841887.9188629389],[6805745.21122244,1841695.4566494375],[6804969.741316691,1841150.3071491867]]]} HTTP/1.1

Host: example.com

**Response**

HTTP/1.1 200 OK

Content-Type: application/json

Content-Length: nnn

{

 "geometryType" : "GeometryPolygon",

 "geometry" :

 {

 "rings" :

 [

 [

 [6804973.14124422, 1841152.84257714],

 [6804797.21246222, 1841029.52031723],

 [6804463.9063703, 1841533.08886389],

 [6804224.75560188, 1841930.7790134],

 [6804233.40617514, 1842251.83512981],

 [6804206.36817147, 1843554.49295756],

 [6805395.76999214, 1843570.17796557],

 [6805514.21168439, 1843607.51942632],

 [6805740.68892172, 1843619.88816798],

 [6806080.25385939, 1843657.18599364],

 [6806290.27017164, 1843380.8292629],

 [6806717.91137671, 1842787.4210009],

 [6806913.66196947, 1842516.04851215],

 [6806763.03858306, 1842410.18815956],

 [6806319.65692389, 1842098.57165681],

 [6806027.58532955, 1841887.91886289],

 [6805745.21122238, 1841695.45664939],

 [6804973.14124422, 1841152.84257714]

 ]

 ]

 }

}

# Union

## Union Overview

The Union operation is performed on a Geometry Service resource. This operation constructs the set-theoretic union of the geometries in the input array. All inputs must be of the same type.

Users can provide arguments to the Union operation as query parameters.

Table 59 –Union overview

|  |  |  |
| --- | --- | --- |
| **Resource** | **Parameters** | **Resource representation** |
| ConvexHull | f=jsongeometriessr | JSON representation validAll JSON schema elements supported |

## Union URI

In the following URI templates, these variables are used:

* geometryServiceRootURI: the URL of the geometry service without any parameter

Table 60 – Union reference

|  |  |
| --- | --- |
| **URI template** | {+geometryServiceRootURI}/union{?f,geometries,sr} |
| **HTTP methods** | GETPOST (application/x-www-form-urlencoded) |
| **Parent Resource** | Geometry Service Root |

Table 61 – Union parameters

|  |  |
| --- | --- |
| **Parameter** | **Details** |
| f | The response format |
|  | Required | Yes |
| Syntax | "json" |
| Example | f=json |
| geometries | An array of points, multipoints, polylines or polygons that are subjected to a "union.See GeoServices REST API – Core, subclause 9.8 for the schema of a geometry array. |
|  | Required | Yes |
| Syntax | JSON  |
| Example | geometries={"geometryType":"GeometryPolyline","geometries":[{"paths":[[[-117,34],[-116,34],[-117,33]],[[-115,44],[-114,43],[-115,43]]]},{"paths":[[[32.49,17.83],[31.96,17.59],[30.87,17.01],[30.11,16.86]]]}]} |
| sr | The spatial reference of the input geometries. The spatial reference is specified as either a well-known ID (WKID) or a spatial reference JSON object. See the Geoservices REST API Core standard for the requirements related to spatial references. |
|  | Required | Yes |
| Syntax | JSON / POSINT |
| Example | sr=4326 |

**Request Requirements**

|  |
| --- |
| * + 1. A request on a Union resource SHALL conform to the URI template in Table 60 and be accessed using a HTTP method identified in the same table.

union/Request |

|  |
| --- |
| * + 1. A request on a Reshape resource SHALL support all parameters and values specified in Table 61.

union/ParametersBasic |

|  |
| --- |
| * + 1. The geometries parameter SHALL conform to the requirements for geometry arrays (see requirements class **http://www.opengis.net/spec/gsr-gs/1.0/req/geometry**) and it SHALL contain a "geometryType" property.

union/ParametersGeometries |

|  |
| --- |
| * + 1. The value of the sr parameter SHALL either be a well-known ID (WKID) or a spatial reference JSON object. A WKID value SHALL be a wkid property value of an object in http://schemas.opengis.net/gsr/1.0/sr.json.

union/ParametersSR |

## Single Geometry resources

|  |
| --- |
| * + 1. The JSON representation of a response to a Union resource request SHALL validate against the JSON Schema **http://schemas.opengis.net/gsr-gs/1.0/singleGeometry.json** and the value of the geometry property SHALL validate against the JSON Schema for the geometry type used in the geometries parameter or in case of an exception against JSON Schema http://schemas.opengis.net/gsr/1.0/exception.json.

union/ResponseValid |

## Example

Make a union of two multipoint geometries:

http://example.com/rest/services/Geometry/GeometryServer/union?sr=102113&f=json&geometries={"geometryType":"GeometryMultipoint","spatialReference":{"wkid":102113},"geometries":[{"points":[

[-8418433.3989,5262954.0867],

[-8353363.717800001,5381714.5528],

[-8250336.7228,5337002.1307],

[-8250336.7228,5509587.154700004]]},

{"points":[

[-1.37211748048E7,4991906.4582],

[-1.35126474057E7,4800770.159900002],

[-1.35126474057E7,4991906.4582],

[-1.3470941925900001E7,4468209.699000001],

[-1.32936936367E7,4494424.728500001],

[-1.31998563072E7,4133116.2506000027],

[-1.3126871717500001E7,4260779.642999999],

[-1.29391970583E7,4044613.1774000004],

[-1.29183443184E7,4222325.328299999],

[-1.28870652086E7,3906909.467299998]]}]}

**Request**

GET /rest/services/Geometry/GeometryServer/union?sr=102113&f=json& geometries={"geometryType":"GeometryMultipoint","spatialReference":{"wkid":102113},"geometries":[{"points":[

[-8418433.3989,5262954.0867],

[-8353363.717800001,5381714.5528],

[-8250336.7228,5337002.1307],

[-8250336.7228,5509587.154700004]]},

{"points":[

[-1.37211748048E7,4991906.4582],

[-1.35126474057E7,4800770.159900002],

[-1.35126474057E7,4991906.4582],

[-1.3470941925900001E7,4468209.699000001],

[-1.32936936367E7,4494424.728500001],

[-1.31998563072E7,4133116.2506000027],

[-1.3126871717500001E7,4260779.642999999],

[-1.29391970583E7,4044613.1774000004],

[-1.29183443184E7,4222325.328299999],

[-1.28870652086E7,3906909.467299998]]}]} HTTP/1.1

Host: example.com

**Response**

HTTP/1.1 200 OK

Content-Type: application/json

Content-Length: nnn

{

 "geometryType" : "GeometryMultipoint",

 "geometry" :

 {

 "points" :

 [

 [-13721174.8048, 4991906.4582],

 [-13512647.4057, 4800770.1599],

 [-13512647.4057, 4991906.4582],

 [-13470941.9259, 4468209.699],

 [-13293693.6367, 4494424.7285],

 [-13199856.3072, 4133116.2506],

 [-13126871.7175, 4260779.643],

 [-12939197.0583, 4044613.1774],

 [-12918344.3184, 4222325.3283],

 [-12887065.2086, 3906909.4673],

 [-8418433.3989, 5262954.0867],

 [-8353363.7178, 5381714.5528],

 [-8250336.7228, 5337002.1307],

 [-8250336.7228, 5509587.1547]

 ]

 }

}

Annex A
(normative)

Abstract Test Suite

Conformance class: geometryservice

* 1. Test: geometryservice/root

|  |  |
| --- | --- |
| Requirements | **geometryservice/request, geometryservice/parameters, geometryservice/valid, geometryservice/supportedOperations** |
| Test purpose | Verify that the service supports the request and response requirements. |
| Test method | Set up a test service. Access the root resource, inspect the response and validate it against the relevant JSON Schemas.  |
| Test type | Basic |

Conformance class: project

* 1. Test: project/test

|  |  |
| --- | --- |
| Requirements | **project/Request, project/ParametersBasic, project/ParametersGeometries, project/ParametersSR, project/ResponseValid** |
| Test purpose | Verify that the Project resource supports the request and response requirements. |
| Test method | In the test geometry service, construct valid Project requests and vary each parameter. Validate the resource against the relevant JSON Schemas. |
| Test type | Capability |

Conformance class: simplify

* 1. Test: simplify/test

|  |  |
| --- | --- |
| Requirements | **simplify/Request, simplify/ParametersBasic, simplify/ParametersGeometries, simplify/ParametersSR, simplify/ResponseValid** |
| Test purpose | Verify that the Simplify resource supports the request and response requirements. |
| Test method | In the test geometry service, construct valid requests and vary each parameter. Validate the resource against the relevant JSON Schemas. |
| Test type | Capability |

Conformance class: buffer

* 1. Test: buffer/test

|  |  |
| --- | --- |
| Requirements | **buffer/Request, buffer/ParametersBasic, buffer/ParametersGeometries, buffer/ParametersSR, buffer/ParametersDistance, buffer/ParametersUnit, buffer/ResponseValid, buffer/ResponseOrder, buffer/GeodesicBuffering** |
| Test purpose | Verify that the Buffer resource supports the request and response requirements. |
| Test method | In the test geometry service, construct valid requests and vary each parameter. Validate the resource against the relevant JSON Schemas. |
| Test type | Capability |

Conformance class: areasAndLengths

* 1. Test: areasAndLengths/test

|  |  |
| --- | --- |
| Requirements | **areasAndLengths/Request, areasAndLengths/ParametersBasic, areasAndLengths/ParametersGeometries, areasAndLengths/ParametersSR, areasAndLengths/ParametersLengthUnit, areasAndLengths/ParametersAreaUnit, areasAndLengths/ResponseValid, areasAndLengths/ResponseCount** |
| Test purpose | Verify that the Areas And Lengths resource supports the request and response requirements. |
| Test method | In the test geometry service, construct valid requests and vary each parameter. Validate the resource against the relevant JSON Schemas. |
| Test type | Capability |

Conformance class: lengths

* 1. Test: lengths/test

|  |  |
| --- | --- |
| Requirements | **lengths/Request, lengths/ParametersBasic, lengths/ParametersGeometries, lengths/ParametersSR, lengths/ParametersLengthUnit, lengths/ResponseValid, lengths/ResponseCount** |
| Test purpose | Verify that the Lengths resource supports the request and response requirements. |
| Test method | In the test geometry service, construct valid requests and vary each parameter. Validate the resource against the relevant JSON Schemas. |
| Test type | Capability |

Conformance class: relation

* 1. Test: relation/test

|  |  |
| --- | --- |
| Requirements | **relation/Request, relation/ParametersBasic, relation/ParametersGeometries, relation/ParametersSR, relation/ParametersGeometryTypes, relation/ParametersRelationParam, relation/ResponseValid, relation/2D** |
| Test purpose | Verify that the Relation resource supports the request and response requirements. |
| Test method | In the test geometry service, construct valid requests and vary each parameter. Validate the resource against the relevant JSON Schemas. |
| Test type | Capability |

Conformance class: labelPoints

* 1. Test: labelPoints/test

|  |  |
| --- | --- |
| Requirements | **labelPoints/Request, labelPoints/ParametersBasic, labelPoints/ParametersGeometries, labelPoints/ParametersSR, labelPoints/ResponseValid** |
| Test purpose | Verify that the Label Points resource supports the request and response requirements. |
| Test method | In the test geometry service, construct valid requests and vary each parameter. Validate the resource against the relevant JSON Schemas. |
| Test type | Capability |

Conformance class: distance

* 1. Test: distance/test

|  |  |
| --- | --- |
| Requirements | **distance/Request, distance/ParametersBasic, distance/ParametersGeometry, distance/ParametersSR, distance/ParametersDistanceUnit, distance/ResponseValid** |
| Test purpose | Verify that the Distance resource supports the request and response requirements. |
| Test method | In the test geometry service, construct valid requests and vary each parameter. Validate the resource against the relevant JSON Schemas. |
| Test type | Capability |

Conformance class: densify

* 1. Test: densify/test

|  |  |
| --- | --- |
| Requirements | **densify/Request, densify/ParametersBasic, densify/ParametersGeometries, densify/ParametersSR, densify/ParametersLengthUnit, densify/ResponseValid** |
| Test purpose | Verify that the Densify resource supports the request and response requirements. |
| Test method | In the test geometry service, construct valid requests and vary each parameter. Validate the resource against the relevant JSON Schemas. |
| Test type | Capability |

Conformance class: generalize

* 1. Test: generalize/test

|  |  |
| --- | --- |
| Requirements | **generalize/Request, generalize/ParametersBasic, generalize/ParametersGeometries, generalize/ParametersSR, generalize/ParametersDerivationUnit, generalize/ResponseValid** |
| Test purpose | Verify that the Generalize resource supports the request and response requirements. |
| Test method | In the test geometry service, construct valid requests and vary each parameter. Validate the resource against the relevant JSON Schemas. |
| Test type | Capability |

Conformance class: convexHull

* 1. Test: convexHull/test

|  |  |
| --- | --- |
| Requirements | **convexHull/Request, convexHull/ParametersBasic, convexHull/ParametersGeometries, convexHull/ParametersSR, convexHull/ResponseValid** |
| Test purpose | Verify that the Convex Hull resource supports the request and response requirements. |
| Test method | In the test geometry service, construct valid requests and vary each parameter. Validate the resource against the relevant JSON Schemas. |
| Test type | Capability |

Conformance class: offset

* 1. Test: offset/test

|  |  |
| --- | --- |
| Requirements | **offset/Request, offset/ParametersBasic, offset/ParametersGeometries, offset/ParametersSR, offset/ParametersOffsetUnit, offset/ResponseValid** |
| Test purpose | Verify that the Offset resource supports the request and response requirements. |
| Test method | In the test geometry service, construct valid requests and vary each parameter. Validate the resource against the relevant JSON Schemas. |
| Test type | Capability |

Conformance class: trimExtend

* 1. Test: trimExtend/test

|  |  |
| --- | --- |
| Requirements | **trimExtend/Request, trimExtend/ParametersBasic, trimExtend/ParametersGeometries, trimExtend/ParametersSR, trimExtend/ResponseValid** |
| Test purpose | Verify that the Trim/Extend resource supports the request and response requirements. |
| Test method | In the test geometry service, construct valid requests and vary each parameter. Validate the resource against the relevant JSON Schemas. |
| Test type | Capability |

Conformance class: autoComplete

* 1. Test: autoComplete/test

|  |  |
| --- | --- |
| Requirements | **autoComplete/Request, autoComplete/ParametersBasic, autoComplete/ParametersGeometries, autoComplete/PolylinesParametersSR, autoComplete/ResponseValid** |
| Test purpose | Verify that the Auto Complete resource supports the request and response requirements. |
| Test method | In the test geometry service, construct valid requests and vary each parameter. Validate the resource against the relevant JSON Schemas. |
| Test type | Capability |

Conformance class: cut

* 1. Test: cut/test

|  |  |
| --- | --- |
| Requirements | **cut/Request, cut/ParametersBasic, cut/ParametersGeometries, cut/ParametersSR, cut/ResponseValid** |
| Test purpose | Verify that the Cut resource supports the request and response requirements. |
| Test method | In the test geometry service, construct valid requests and vary each parameter. Validate the resource against the relevant JSON Schemas. |
| Test type | Capability |

Conformance class: difference

* 1. Test: difference/test

|  |  |
| --- | --- |
| Requirements | **difference/Request, difference/ParametersBasic, difference/ParametersGeometries, difference/ParametersSR, difference/ResponseValid** |
| Test purpose | Verify that the Difference resource supports the request and response requirements. |
| Test method | In the test geometry service, construct valid requests and vary each parameter. Validate the resource against the relevant JSON Schemas. |
| Test type | Capability |

Conformance class: intersect

* 1. Test: intersect/test

|  |  |
| --- | --- |
| Requirements | **intersect/Request, intersect/ParametersBasic, intersect/ParametersGeometries, intersect/ParametersSR, intersect/ResponseValid** |
| Test purpose | Verify that the Intersect resource supports the request and response requirements. |
| Test method | In the test geometry service, construct valid requests and vary each parameter. Validate the resource against the relevant JSON Schemas. |
| Test type | Capability |

Conformance class: reshape

* 1. Test: reshape/test

|  |  |
| --- | --- |
| Requirements | **reshape/Request, reshape/ParametersBasic, reshape/ParametersGeometries, reshape/ParametersSR, reshape/ResponseValid** |
| Test purpose | Verify that the Reshape resource supports the request and response requirements. |
| Test method | In the test geometry service, construct valid requests and vary each parameter. Validate the resource against the relevant JSON Schemas. |
| Test type | Capability |

Conformance class: union

* 1. Test: union/test

|  |  |
| --- | --- |
| Requirements | **union/Request, union/ParametersBasic, union/ParametersGeometries, union/ParametersSR, union/ResponseValid** |
| Test purpose | Verify that the Union resource supports the request and response requirements. |
| Test method | In the test geometry service, construct valid requests and vary each parameter. Validate the resource against the relevant JSON Schemas. |
| Test type | Capability |

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