Open Geospatial Consortium Inc.

Date:   2012-06-22

Reference number of this OGC® project document:    OGC 12-057r1

Version: 0.0.4

Category: OGC® Implementation Specification

Editor:   Clemens Portele, Satish Sankaran

GeoServices REST API — Part 4: Feature 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 4

4 Terms and Definitions 4

5 Conventions 4

6 Feature Service overview 4

7 Feature Service Core 6

7.1 Overview 6

7.2 Feature Service Root 6

7.2.1 Overview 6

7.2.2 Feature Service Root URI 6

7.2.3 Feature Service Root resources 7

7.2.4 Example 8

7.3 Layer/Table 8

7.3.1 Overview 8

7.3.2 Layer/Table URI 9

7.3.3 Layer/Table resources 10

7.3.4 Example 10

7.4 Feature 15

7.4.1 Overview 15

7.4.2 Feature URI 15

7.4.3 Feature resources 16

7.4.4 Example 17

7.5 Image 17

7.5.1 Overview 17

7.5.2 Image URI 17

7.5.3 Image resources 18

7.5.4 Example 18

8 Query 18

8.1 Overview 18

8.2 Feature Service Root resources 19

8.3 Query 20

8.3.1 Query URI 20

8.3.2 Feature Set and Feature ID Set resources 25

8.3.3 Examples 26

9 Temporal Query 29

9.1 Overview 29

9.2 Layer/Table resources 29

9.3 Query 29

9.3.1 Dependency 29

9.3.2 Query request 30

9.3.3 Feature Set and Feature ID Set resources 30

10 Query Related Records 30

10.1 Overview 30

10.2 Query Related Records 31

10.2.1 Query Related Records URI 31

10.2.2 Related Records resources 34

10.2.3 Example 34

11 Feature Editing 35

11.1 Overview 35

11.2 Feature Service Root resources 36

11.3 Add Features 36

11.3.1 Overview 36

11.3.2 Add Features URI 37

11.3.3 Add Features Result resources 38

11.3.4 Example 39

11.4 Update Features 40

11.4.1 Overview 40

11.4.2 Update Features URI 40

11.4.3 Update Features Result resources 42

11.4.4 Example 42

11.5 Delete Features 44

11.5.1 Overview 44

11.5.2 Delete Features URI 44

11.5.3 Delete Features Result resources 48

11.5.4 Example 48

11.6 Apply Edits 49

11.6.1 Overview 49

11.6.2 Apply Edits Features URI 49

11.6.3 Apply Edits Result resources 51

11.6.4 Example 52

12 Attachments and HTML Popups 54

12.1 Overview 54

12.2 Attachment Infos 54

12.2.1 Overview 54

12.2.2 Attachment Infos URI 55

12.2.3 Attachment Infos resources 56

12.2.4 Example 56

12.3 Attachment 56

12.3.1 Overview 56

12.3.2 Attachment URI 57

12.3.3 Attachment resources 57

12.3.4 Example 57

12.4 HTML Popup 58

12.4.1 Overview 58

12.4.2 HTML Popup URI 58

12.4.3 HTML Popup resources 59

12.4.4 Example 59

12.5 Add Attachment 60

12.5.1 Overview 60

12.5.2 Add Attachement URI 60

12.5.3 Add Attachement Result resources 61

12.5.4 Example 61

12.6 Update Attachment 63

12.6.1 Overview 63

12.6.2 Update Attachement URI 63

12.6.3 Update Attachement Result resources 64

12.6.4 Example 65

12.7 Delete Attachments 66

12.7.1 Overview 66

12.7.2 Delete Attachements URI 66

12.7.3 Delete Attachements Result resources 67

12.7.4 Example 68

13 Templates 68

13.1 Overview 68

13.2 Layer/Table resources 69

A.1 Conformance class: featureservice 71

A.1.1 Test: featureservice/root 71

A.1.2 Test: featureservice/layerOrTable 71

A.1.3 Test: featureservice/feature 72

A.1.4 Test: featureservice/image 72

A.2 Conformance class: query 72

A.2.1 Test: query/query 72

A.3 Conformance class: query-time 73

A.3.1 Test: query-time/query 73

A.4 Conformance class: queryrel 73

A.4.1 Test: queryrel/query 73

A.5 Conformance class: editing 74

A.5.1 Test: editing/basic 74

A.5.2 Test: editing/add 74

A.5.3 Test: editing/update 74

A.5.4 Test: editing/delete 75

A.5.5 Test: editing/applyEdits 75

A.5.6 Test: attachments/infos 76

A.5.7 Test: attachments/attachment 76

A.5.8 Test: attachments/popup 77

A.5.9 Test: attachments/add 77

A.5.10 Test: attachments/update 77

A.5.11 Test: attachments/delete 78

A.6 Conformance class: templates 78

A.6.1 Test: templates/basic 78

Preface

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

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

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

Submitting organizations

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

Esri Inc.

interactive instruments GmbH

Oracle USA

52°North

Submission contact points

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

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

Revision history

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Date | Release | Author | Paragraph modified | Description |
| 1. 23 Jan 2012
 | 1. 0.0.1
 | 1. CP, SS
 | 1. All
 | 1. First draft
 |
| 1. 21 May 2012
 | 1. 0.0.2
 | 1. CP
 | 1. All
 | 1. Include changes identified in review process
 |
| 1. 31 May 2012
 | 1. 0.0.3
 | 1. CP
 |  | 1. RFC package for OAB review
 |
| 1. 22 Jun 2012
 | 1. 0.0.4
 | 1. CP
 |  | 1. References updated
2. Editorial changes
 |

Changes to the OGC® Abstract Specification

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

Versioning Rules

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

Foreword

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

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

This document is part 4 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.

# 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 feature 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 7 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-fs/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

|  |  |  |
| --- | --- | --- |
| **featureservice** | **Title** | Feature Service Core |
| **Standardization target type** | Web service |
| **Dependencies** | **http://www.opengis.net/spec/gsr/1.0/conf/core****http://www.opengis.net/spec/gsr/1.0/conf/feature****http://www.opengis.net/spec/gsr/1.0/conf/symbol** |
| **Requirements** | All requirements in Clause 7 |
| **Conformance tests** | Annex A.1 |
| **query** | **Title** | Query |
| **Standardization target type** | Web service |
| **Dependencies** | **conf/featureservice** |
| **Requirements** | All requirements in Clause 8 |
| **Conformance tests** | Annex A.2 |
| **queryTemporal** | **Title** | Temporal Query |
| **Standardization target type** | Web service |
| **Dependencies** | **conf/query** |
| **Requirements** | All requirements in Clause 9 |
| **Conformance tests** | Annex A.3 |
| **queryRelated** | **Title** | Query Related Records |
| **Standardization target type** | Web service |
| **Dependencies** | **conf/query** |
| **Requirements** | All requirements in Clause 10 |
| **Conformance tests** | Annex A.4 |
| **editing** | **Title** | Feature Editing |
| **Standardization target type** | Web service |
| **Dependencies** | **conf/featureservice** |
| **Requirements** | All requirements in Clause 11 |
| **Conformance tests** | Annex A.5 |
| **attachements** | **Title** | Attachements and HTML popups |
| **Standardization target type** | Web service |
| **Dependencies** | **conf/editing****http://tools.ietf.org/html/rfc1867** |
| **Requirements** | All requirements in Clause 12 |
| **Conformance tests** | Annex A.6 |
| **templates** | **Title** | Editing templates |
| **Standardization target type** | Web service |
| **Dependencies** | **conf/editing** |
| **Requirements** | All requirements in Clause 13 |
| **Conformance tests** | Annex A.7 |

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

Form-based File Upload in HTML, IETF RFC 1867 (1995), http://tools.ietf.org/html/rfc1867

# 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.

# Feature Service overview

A feature service allows clients to access, query and edit features in feature layers and tables.

Features include geometry, attributes, and symbology and are organized into layers and subtypes within a layer. Features are explained in more detail in the GeoServices REST API – Core document, Clause 10 ("Features").

The Feature Service supports several operations:

Query: Returns a subset of features in a layer or records in a table based on query criteria.

Query Related Records: Returns relationships between features of two layers/tables based on query criteria.

Add Features, Update Features, Delete Features, Apply Edits: Editing features of a layer/table.

Add Attachment, Update Attachment, Delete Attachments: Editing attachements associated with features.

* The following figure 2 provides an overview of the resources in a Feature Service. Resources in green color are controller resources, also called "operations", that a) either edit information in the server or b) process the operation parameters and the information in the server and create resources that are not persistently stored on the server and which are not made available with their own URI, but returned in the HTTP response to the operation, for example, as a result to a query. Controller resources of type b) could also be viewed simply as accessing existing resources on the server, shown in white colour in the figure, while in general these will be dynamically created by the controller resource. This is the case with queries. Editing features or attachments ("Add Features", "Apply Edits", "Add Attachment", etc) is a controller resource of type a) and creates new resources or updates/deletes existing ones.

Figure 2 – Resource overview

# Feature Service Core

## Overview

The GeoServices REST API Feature Service Core conformance class provides read access to the layers and tables of the service including all features in the layers and tables.

Table 2 – Feature Service Core overview

|  |  |  |
| --- | --- | --- |
| **Resource** | **Parameters** | **Resource representation** |
| Feature Service Root | f=json | JSON representation validAll JSON schema elements supported |
| Layer/Table | f=json | JSON representation validJSON schema elements supported: all except: timeInfo, templates and hasAttachments |
| Feature | f=json | JSON representation validAll JSON schema elements supported |
| Image | - | Image accessible |

## Feature Service Root

### Overview

The GeoServices REST API Feature Service Root resource provides basic information about the feature service: the feature layers and tables that it contains, the service description, and so on.

### Feature Service Root URI

In the following URI templates, these variables are used:

* featureServiceRootURI: the URL of the service

If the Feature Service is referenced from a Catalog Service, featureServiceRootURI is the same as

{+catServiceRootURI}/{featureServiceName}/FeatureServer

where featureServiceName is the name of the feature service referenced in the catalogue.

Table 3 – Feature Service Root reference

|  |  |
| --- | --- |
| **URI template** | {+featureServiceRootURI}{?f} |
| **HTTP methods** | GET |
| **Parent Resource Type** | n/a |
| **Child Resource Types**  | Layer/Table |

Table 4 – Feature Service Root parameters

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

**Request Requirements**

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

featureservice/request |

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

featureservice/parameters |

### Feature Service Root resources

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

featureservice/valid |

|  |
| --- |
| * + 1. Each layer id in the JSON representation of a Feature Service Root resource SHALL be unique within the feature service.

featureservice/uniqueLayerId |

|  |
| --- |
| * + 1. Each table id in the JSON representation of a Feature Service Root resource SHALL be unique within the feature service.

featureservice/uniqueTableId |

### Example

URL to the 311Incidents feature service running on example.com:

http://example.com/rest/services/311Incidents/FeatureServer?f=json

**Request**

GET /rest/services/311Incidents/FeatureServer?f=json HTTP/1.1

Host: example.com

**Response**

HTTP/1.1 200 OK

Content-Type: application/json

Content-Length: nnn

{

 "serviceDescription": "Edit parcels, buildings and owner information.",

 "layers": [

 { "id": 0, "name": "Parcels" }

 { "id": 1, "name": "Buildings" }

 ],

 "tables": [

 { "id": 3, "name": "Owners" }

 ]

}

## Layer/Table

### Overview

The Layer/Table resource represents a single layer or table in a feature service.

For tables, it provides basic information about the table such as its ID, name, fields, and relationships with other tables or feature layers.

For layers, it provides additional information such as its geometry type, min and max scales, and spatial reference.

Both tables and layers publish one or more subtypes. This resource includes information about these types as well. Each type includes information about the type, such as the type ID, name, and definition expression. Feature layer subtypes also include a default symbol.

### Layer/Table URI

In the following URI templates, these variables are used:

* featureServiceURI: URL of a Feature Service Root resource without any parameter
* layerOrTableId: id of the layer or table

Table 5 – Layer/Table reference

|  |  |
| --- | --- |
| **URI template**  | {+featureServiceURI}/{layerOrTableId}{?f} |
| **HTTP methods** | GET |
| **Parent Resource Type** | Feature Service Root |
| **Child Resource Types** | FeatureImage |
| **Child Operations** | Add Features (only if conformance class "Feature Editing" is supported)Update Features (only if conformance class "Feature Editing" is supported)Delete Features (only if conformance class "Feature Editing" is supported)Apply Edits (only if conformance class "Feature Editing" is supported)Query (only if conformance class "Query" is supported)QueryRelatedRecords (only if conformance class "Query Related Records" is supported) |

Table 6 – Layer/Table parameters

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

**Request Requirements**

|  |
| --- |
| * + 1. The request of a Layer/Table resource SHALL conform to the URI template in Table 5 and be accessed using a HTTP method identified in the same table.

featureservice/layerOrTableRequest |

|  |
| --- |
| * + 1. The request of a Layer/Table resource SHALL support all parameters and values specified in Table 6.

featureservice/layerOrTableParameters |

### Layer/Table resources

|  |
| --- |
| * + 1. The JSON representation of a map service resource SHALL validate against the JSON Schema **http://schemas.opengis.net/gsr/1.0/layerOrTable.json** or in case of an exception against JSON Schema http://schemas.opengis.net/gsr/1.0/exception.json.

featureservice/layerOrTableValid |

### Example

URL to the first layer/table in the 311Incidents feature service running on example.com:

http://example.com/rest/services/311Incidents/FeatureServer/0?f=json

**Request**

GET /rest/services/311Incidents/FeatureServer/0?f=json HTTP/1.1

Host: example.com

**Response**

HTTP/1.1 200 OK

Content-Type: application/json

Content-Length: nnn

{

 "id" : 0,

 "name" : "Incidents",

 "type" : "Feature Layer",

 "displayField" : "req\_id",

 "description" : "",

 "copyrightText" : "",

 "relationships" : [

 {

 "id" : 1,

 "name" : "ServiceRequest\_IncidentPriority",

 "relatedTableId" : 1

 }

 ],

 "geometryType" : "GeometryPoint",

 "minScale" : 0,

 "maxScale" : 0,

 "extent" : {

 "xmin" : -122.514435102,

 "ymin" : 5.6843418860808E-14,

 "xmax" : 138.625776397,

 "ymax" : 67.1577965990001,

 "spatialReference" : {

 "wkid" : 4326

 }

 },

 "drawingInfo" : {"renderer" :

 {

 "type" : "uniqueValue",

 "field1" : "req\_type",

 "field2" : null,

 "field3" : null,

 "fieldDelimiter" : ", ",

 "defaultSymbol" : null,

 "defaultLabel" : "\u003call other values\u003e",

 "uniqueValueInfos" : [

 {

 "value" : "Blocked Street or Sidewalk",

 "label" : "Blocked Street or Sidewalk",

 "description" : "",

 "symbol" :

 {

 "type" : "PMS",

 "url" : "1DD4FC53",

 "imageData" : "iVBORw0KGgoAAAANSUhEUgAAABoAAAAaCAYAAACpSkzOAAAAAXNSR0IB2cksfwAAAAlwSFlzAAAOxAAADsQBlSsOGwAAAsVJREFUSInt001IVFEUwPH/a2bewwyCCZEZMI2gBoRKSdtEROBKooyiDwqt8QNpYQmjYuIHDZGOVDM5RJAwUvRBizYFLmrXoo+F0iKklROkSOQmyrnqvNNinLHRmdcY5Kqzuvfde8/vnnN5dtYp7P+hlbFnp0uq9isCw7PaP4Oqq1zy/PY0KGDRKYGR7NhfQ2mIgoGGWSq3O+VEd2bsT5AYGChU2uHqKpcUbJpG80CkB2qrwFQGx/b+4EmnU05eW41ZQTIz85XCwoKMlWgeSK6fO2AkKpuHmt3w+LJLTt2cTsMsK6qpOYJhGCilAPA1OmWgdaldgGmaicESElf5yDwcLYVHTS45fXcZywqJCObDLchCPqZyiqNxKrW2EHKjBiBUv5Xz3nqYv59CRBlIDDpG9bR8lhVtOPON+B03xED1uzHaE5hs3oX5+QO+F4rGJgfPXsPh3cuIJ6ATjUZzap0kB7bmKeLX3cg8zHWVkOefRK8dZa6rBACHXef4oCJ2z5kVsaxI9boxehMV6B1TxDrdSAzUAx/G2QB5/knqvlxAz0u0yArJCpU6DUwFP1tK2BicBMD8rsMCLLx6inE2AMC2bdsJBm9Qd96LJ/AyK5IVevbmI83XrjISiWiENCkuLiYajS5viKTnKyoqskSyQWKacUYikdQHb0MT3V2dABogExOf8Hh2pOYOXYfEu66tolAo+PtUc9hsAtDX55eeni7iS/9PX59fABw2m1Ux2aFweIhweCh5SwA6OtoA6O6+kkRS+212x9ohEcm0Ly3Gx8fT5m2+Vm4Fw1xquZg7lEuUlZXB8nsIgK4blmdWQjI2Nk55eVoiAIaHR8TrreXgwUMrc2iA6Lo9ieb2Hy3GM7fO663VABkIDFJZUb4Kq/fWZTyXEervDyAST95yVbx9995yPWeovd1nmWBfZcWagYzQv4x1g34BI70ZygdCoCgAAAAASUVORK5CYII=",

 "contentType" : "image/png",

 "color" : null,

 "width" : 19,

 "height" : 19,

 "angle" : 0,

 "xoffset" : 0,

 "yoffset" : 0

 }

 },

 {

 "value" : "Damaged Property",

 "label" : "Damaged Property",

 "description" : "",

 "symbol" :

 {

 "type" : "PMS",

 "url" : "DF3100A6",

 "imageData" : "iVBORw0KGgoAAAANSUhEUgAAABQAAAAMCAYAAABiDJ37AAAAAXNSR0IB2cksfwAAAAlwSFlzAAAOxAAADsQBlSsOGwAAAY1JREFUKJGl0j+Iz2EYAPDPw+lJZ+CY/B2kbCKkDEZMN8hoMBoMFmVSSqKUWIyXSThlQAZ/LgMZZFNncUhRvq6L8nTyGu6H3P3uDJ56h+d9n/fzvs/bO+A/ozGICr7DwKzFDTI34YfBwc+6biLoZtUslrlT1WpsbqyLzOuq7s8BZW5UdR6vVB3AWGNT41Zkjqra0tgWVUvxSOZYVI1H1cdfxF9gVD1oLMfalnlK1apgT3BE1X68DUZwI5hSNecJBubMkI1xDATvMY0lLfNaVB3v5Q8x1e9Nf4NtaGiHrnuJp8Gh3umfcFLmYVXLeqXf8LixO5joCzYWta9fDwa7Gq+DN1hppv1hVcPB3cZIsAJrsN28YObpqNraGA9GW+Y7VctlTqsaDJ7IHI6qC1jSuBLcnrflqDqLfYu4Cq3qOZ6pWo8vOKNqFBeDU/2gv0Em9bBe/qENDQ3rupst81ZUnWjckbm0VR0NLi0I9ovouheNc1F1EpPBsah63zL39vsu/wR7N71sZvyZq7q30J6fOkKkEEs/eqsAAAAASUVORK5CYII=",

 "contentType" : "image/png",

 "color" : null,

 "width" : 15,

 "height" : 9,

 "angle" : 0,

 "xoffset" : 0,

 "yoffset" : 0

 }

 },

 {

 "value" : "Graffiti Complaint - Public Property",

 "label" : "Graffiti Complaint",

 "description" : "",

 "symbol" :

 {

 "type" : "PMS",

 "url" : "B2E6E7A0",

 "imageData" : "iVBORw0KGgoAAAANSUhEUgAAABoAAAAaCAYAAACpSkzOAAAAAXNSR0IB2cksfwAAAAlwSFlzAAAOxAAADsQBlSsOGwAAAiVJREFUSInt1EFIVFEUxvH/4DgnmUR8MTpjUJQPImgTIQbWoqCFixSEgoTaJFJZibawjNJCkiQIF5GImxaGGGIiZFiEBSa1ikLIGpLpWUNGb8J8OWfMsUVGFGZag5D4re4958KPe+BeN4sU99KFdovkpUBFu2rlTkjLEGmYUO/LLuzjCYW6VfuBF4Dcg2eo5teu9Wd3hexEOD+Nbn2jafLUtms+Oo5vDO4OhkK3ikX6Jg2jKVM14w1s6bTtWmD4r6EKwzj6yLLWxWHqs/qvFpjJqwYsq6hVdceZcNiogw9NhpGTZRjbr9j2MOACphcMXbbtSoA74OkjlHksiFUiUlMgcrEXbqL6eMBxzo9DFLhdLnJgBLY9VD0ZhufzhopFuiYhvkv1EGABvPJ6R9c4Tvp+r7esRdW6pppfDys6INqoeqnZNDsiwaAdXsiNrqsW/tos9Pkin0T871XH9gQCeTfC4bZTEC0RuRCBzaXB4MFSkRPpsHUEarMgr1O1FXj7W2i2lA8Ntc1Wb1GtroGUDphoVj1bBoHVIvtGoe6wSNoAPHii2jtvaK6cg4mZZWqOaXZmJ0lu9Nv+9N4pihqCg1U90P3P0PdUS6DdnyS5MSBp6gsuIA4bNxlGfY9tJw4Sg/QMdzIA0+5kXDN1Q1IFfjz2hP51rjl6S/j3XoaWof8H8nk89/vtdxtSPZ6VLnBPQzwWi407qq8TCh0JhaqAqj+dW7TRfQVgAs+jjDbsggAAAABJRU5ErkJggg==",

 "contentType" : "image/png",

 "color" : null,

 "width" : 19,

 "height" : 19,

 "angle" : 0,

 "xoffset" : 0,

 "yoffset" : 0

 }

 }

 ]

 },

 "transparency" : 0,

 "labelingInfo" : null},

 "hasAttachments" : true,

 "htmlPopupType" : "ServerHTMLPopupTypeAsHTMLText",

 "objectIdField" : "objectid",

 "globalIdField" : "",

 "typeIdField" : "req\_type",

 "fields" : [

 {

 "name" : "objectid",

 "type" : "FieldTypeOID",

 "alias" : "Object ID",

 "editable" : false,

 "domain" : null

 },

 {

 "name" : "req\_id",

 "type" : "FieldTypeString",

 "alias" : "Request ID",

 "editable" : true,

 "length" : 20,

 "domain" : null

 },

 {

 "name" : "req\_type",

 "type" : "FieldTypeString",

 "alias" : "Request Type",

 "editable" : true,

 "length" : 40,

 "domain" : null

 },

 {

 "name" : "req\_date",

 "type" : "FieldTypeString",

 "alias" : "Request Date",

 "editable" : true,

 "length" : 30,

 "domain" : null

 },

 {

 "name" : "req\_time",

 "type" : "FieldTypeString",

 "alias" : "Request Time",

 "editable" : true,

 "length" : 20,

 "domain" : null

 },

 {

 "name" : "address",

 "type" : "FieldTypeString",

 "alias" : "Address",

 "editable" : true,

 "length" : 60,

 "domain" : null

 }

,

 {

 "name" : "status",

 "type" : "FieldTypeSmallInteger",

 "alias" : "Status",

 "editable" : true,

 "domain" :

 {

 "type" : "codedValue",

 "name" : "StatusCodes",

 "codedValues" : [

 {

 "name" : "New",

 "code" : 1

 },

 {

 "name" : "Open",

 "code" : 2

 },

 {

 "name" : "Closed",

 "code" : 3

 }

 ]

 }

 }

 ],

 "types" : [

 {

 "id" : "Blocked Street or Sidewalk",

 "name" : "Blocked Street or Sidewalk",

 "domains" :

 {

 },

 "templates" : [

 {

 "name" : "Blocked Street or Sidewalk",

 "description" : "",

 "drawingTool" : "FeatureEditToolPoint",

 "prototype" : {

 "attributes" : {

 "status" : 1,

 "req\_id" : null,

 "req\_type" : "Blocked Street or Sidewalk",

 "req\_date" : null,

 "req\_time" : null,

 "address" : null,

 "x\_coord" : null,

 "y\_coord" : null,

 "district" : null

 }

 }

 }

 ]

 },

 {

 "id" : "Damaged Property",

 "name" : "Damaged Property",

 "domains" :

 {

 },

 "templates" : [

 {

 "name" : "Damaged Property",

 "description" : "",

 "drawingTool" : "FeatureEditToolPoint",

 "prototype" : {

 "attributes" : {

 "status" : 1,

 "req\_id" : null,

 "req\_type" : "Damaged Property",

 "req\_date" : null,

 "req\_time" : null,

 "address" : null,

 "x\_coord" : null,

 "y\_coord" : null,

 "district" : null

 }

 }

 }

 ]

 },

 {

 "id" : "Graffiti Complaint - Public Property",

 "name" : "Graffiti Complaint",

 "domains" :

 {

 },

 "templates" : [

 {

 "name" : "Graffiti Complaint",

 "description" : "",

 "drawingTool" : "FeatureEditToolPoint",

 "prototype" : {

 "attributes" : {

 "status" : 1,

 "req\_id" : null,

 "req\_type" : "Graffiti Complaint - Public Property",

 "req\_date" : null,

 "req\_time" : null,

 "address" : null,

 "x\_coord" : null,

 "y\_coord" : null,

 "district" : null

 }

 }

 }

 ]

 }

 ],

 "templates" : [

 ],

 "capabilities" : "Query,Editing"

}

## Feature

### Overview

The Feature resource represents a single feature in a layer in a feature service.

### Feature URI

In the following URI templates, these variables are used:

* featureServiceURI: URL of a Feature Service Root resource without any parameter
* layerOrTableId: id of the layer or table
* featureId: id of the feature in the layer or table

Table 7 – Feature reference

|  |  |
| --- | --- |
| **URI template**  | {+featureServiceURI}/{layerOrTableId}/{featureId}{?f} |
| **HTTP methods** | GET |
| **Parent Resource Type** | Layer/Table |
| **Child Resource Types** | Attachment Infos (only if conformance class "Attachments" is supported)HTML Popup (only if conformance class "Attachments" is supported) |
| **Child Operations** | Add Attachment (only if conformance class "Attachments" is supported)Update Attachment (only if conformance class "Attachments" is supported)Delete Attachment (only if conformance class "Attachments" is supported) |

Table 8 – Feature parameters

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

**Request Requirements**

|  |
| --- |
| * + 1. The request of a Feature resource SHALL conform to the URI template in Table 5 and be accessed using a HTTP method identified in the same table.

featureservice/featureRequest |

|  |
| --- |
| * + 1. The request of a Feature resource SHALL support all parameters and values specified in Table 6.

featureservice/featureParameters |

### Feature resources

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

featureservice/featureValid |

### Example

URL for feature 1 in layer 1 of the Watershed service on example.com:

http://example.com/rest/services/Watershed/FeatureServer/1/1?f=json

**Request**

GET /rest/services/Watershed/FeatureServer/1/1?f=json HTTP/1.1

Host: example.com

**Response**

HTTP/1.1 200 OK

Content-Type: application/json

Content-Length: nnn

{

 "feature" : {

 "geometry" : {

 "rings" : [

 [ [-97.06138,32.837], [-97.06133,32.836], [-97.06124,32.834], [-97.06127,32.832], [-97.06138,32.837] ]

 ]

 },

 "attributes" : {

 "OBJECTID" : 37,

 "OWNER" : "Joe Smith",

 "VALUE" : 94820.37,

 "APPROVED" : true,

 "LASTUPDATE" : 1227663551096

 }

 }

}

## Image

### Overview

The Image resource represents an individual image associated with a picture symbol. This resource is available only if the layer includes picture marker symbols or picture fill symbols. **The url property of these symbols SHOULD be used as the imageId value in the image URL.**

### Image URI

In the following URI templates, these variables are used:

* featureServiceURI: URL of a Feature Service Root resource without any parameter
* layerId: id of the layer
* imageId: id of the image

Table 9 – Image reference

|  |  |
| --- | --- |
| **URI template**  | {+featureServiceURI}/{layerId}/images/{imageId} |
| **HTTP methods** | GET |
| **Parent Resource Type** | Layer/Table |

**Request Requirements**

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

featureservice/imageRequest |

### Image resources

|  |
| --- |
| * + 1. The image bytes SHALL be returned to the client. If the image is not found, an HTTP status code of 404 SHALL be returned.

featureservice/image |

### Example

Return an image associated with a symbol for layer 0 in the 311Incidents feature service:

http://example.com/rest/services/311Incidents/FeatureServer/0/images/1DD4FC53

**Request**

GET /rest/services/311Incidents/FeatureServer/0/images/1DD4FC53 HTTP/1.1

Host: example.com

**Response**

HTTP/1.1 200 OK

Content-Type: image/png

Content-Length: nnn

iVBORw0KGgoAAAANSUhEUgAAADIA…

# Query

## Overview

The Query operation is performed on a controller resource of the feature service Layer/Table resource. The result of this operation is either a feature set or an array of feature IDs (if returnIdsOnly is set to true). The result is not stored on the server and is returned in the response to the request.

In the feature set response, the layer features include their geometries. The records for tables do not.

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

Table 10 – Query overview

|  |  |  |
| --- | --- | --- |
| **Resource** | **Parameters** | **Resource representation** |
| Feature Service Root | - | JSON schema elements supported:- capabilities (value contains "Query") |
| Query | f=jsongeometrygeometryTypewherereturnGeometryinSRoutSRspatialRelrelationParamobjectIdsoutFieldsreturnIdsOnly | JSON representation validAll JSON schema elements |

## Feature Service Root resources

|  |
| --- |
| * + 1. The Feature Service Root resource SHALL contain the value "Query" in the property "capabilities".

featureservice/capQuery |

**Example**

...

 "capabilities" : "Query,Editing"

...

## Query

### Query URI

In the following URI templates, these variables are used:

* featureServiceURI: URL of a Feature Service Root resource without any parameter
* layerOrTableId: id of the layer or table to query

Table 11 – Query reference

|  |  |
| --- | --- |
| **URI template** | {+featureServiceURI}/{layerOrTableId}/query{?f,geometry,geometryType, where,returnGeometry,inSR,outSR,maxAllowableOffset,spatialRel,relationParam, objectIds,outFields,returnIdsOnly,time} |
| **HTTP methods** | GETPOST (application/x-www-form-urlencoded) |
| **Parent Resource** | Layer/Table |

Table 12 – Query parameters

|  |  |
| --- | --- |
| **Parameter** | **Details** |
| f | The response format.  |
| Required | Yes |
| Syntax | "json" / "image" |
| Example | f=json |
| geometry | The geometry to apply as a spatial filter. The type of the geometry is specified by the geometryType parameter. In addition to the JSON structures, for points and envelopes, users may specify the geometries with a simpler comma-separated syntax.**JSON:**An input geometry. The geometry is of the type defined by the geometryType parameter.See GeoServices REST API – Core, Clause 9 for the schema of the geometries.**Simple syntax for point geometries:**When using points, the geometries may alternatively be specified with a simpler comma-separated syntax.**Simple syntax for envelope geometries:**When using envelopes, the geometries may alternatively be specified with a simpler comma-separated syntax (first the lower left corner, then the upper right corner).NOTE Coordinates always use a period as the decimal separator even in countries where a comma is traditionally used. |
| Required | No. Default is that no spatial filter is applied. |
| Syntax | JSON / X "," Y / XMIN "," YMIN "," XMAX "," YMAX  |
| Examples | **JSON:**geometryType=GeometryPoint&geometry={x: -104, y: 35.6}**Simple syntax for point geometries:**geometryType=GeometryPoint&geometry=-104,35.6**Simple syntax for envelope geometries:**geometryType=GeometryEnvelope&geometry=-104,35.6,-94.32,41 |
| geometryType | The type of geometry specified by the geometry parameter. The well-known geometry types include point ("GeometryPoint"), multi point ("GeometryMultiPoint"), polyline ("GeometryPolyline"), polygon ("GeometryPolygon"), and envelope ("GeometryEnvelope"). See GeoServices REST API – Core, Clause 9 for additional information about these geometry types. |
| Required | No. Default: "GeometryPoint" |
| Syntax | "GeometryPoint" / "GeometryMultiPoint" / "GeometryPolyline" / "GeometryPolygon" / "GeometryEnvelope" |
| Example | geometryType=GeometryPolygon |
| where | A WHERE clause for the query filter. Any legal SQL WHERE clause operating on the fields in the layer is allowed.The parameter is ignored, if a value for objectIds is specified. |
| Required | No. Default: no filter |
| Syntax | STRING |
| Example | where=POP2000 > 350000 |
| returnGeometry | If true, the result set includes the geometry associated with each result.If the outFields parameter is set to the wildcard, it implies returnGeometry=true, and setting returnGeometry to false has no effect. |
| Required | No. Default: "true". |
| Syntax | BOOLEAN |
| Example | returnGeometry=false |
| inSR | The spatial reference of the input geometry. The spatial reference is specified as either a well-known ID (WKID) or a spatial reference JSON object. See Geoservices REST API Core standard for more requirements related to spatial references. |
| Required | No. Default: The geometry is assumed to be in the spatial reference of the map. |
| Syntax | POSINT / JSON |
| Example | sr=4326 |
| outSR | 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 Geoservices REST API Core standard for more requirements related to spatial references. |
| Required | No. Default: The output geometry is exported in the spatial reference of the map. |
| Syntax | POSINT / JSON |
| Example | sr=4326 |
| spatialRel | The spatial relationship to be applied on the input geometry while performing the query. The supported spatial relationships include intersects, contains, envelope intersects, and within.Pre-defined values: SpatialRelIntersects: Returns a feature if any spatial relationship is found. Applies to all shape type combinations.SpatialRelContains: Returns a feature if its shape is wholly contained within the search geometry. Valid for all shape type combinations.SpatialRelCrosses: Returns a feature if the intersection of the interiors of the two shapes is not empty and has a lower dimension than the maximum dimension of the two shapes. Two lines that share an endpoint in common do not cross. Valid for line/line, line/area, multipoint/area, and multipoint/line shape type combinations.SpatialRelEnvelopeIntersects: Returns a feature if the envelope of the two shapes intersects.SpatialRelIndexIntersects: Returns a feature if the envelope of the query geometry intersects the index entry for the target geometry.SpatialRelOverlaps: Returns a feature if the intersection of the two shapes results in an object of the same dimension but different from both of the shapes. Applies to area/area, line/line, and multipoint/multipoint shape type combinations.SpatialRelTouches: Returns a feature if the two shapes share a common boundary. However, the intersection of the interiors of the two shapes must be empty. In the point/line case, the point may touch an endpoint only of the line. Applies to all combinations except point/point.SpatialRelWithin: Returns a feature if its shape wholly contains the search geometry. Valid for all shape type combinations.SpatialRelRelation: Defines a custom spatial relationship as specified by the relationParam parameter. |
| Required | No. Default: "SpatialRelIntersects" |
| Syntax | "SpatialRelIntersects" / "SpatialRelContains" / "SpatialRelCrosses" / "SpatialRelEnvelopeIntersects" / "SpatialRelIndexIntersects" / "SpatialRelOverlaps" / "SpatialRelTouches" / "SpatialRelWithin" / "SpatialRelRelation" |
| Example | spatialRel=SpatialRelContains |
| relationParam | The spatial relate function that can be applied while performing the query operation. An example for this spatial relate function is 'FFFTTT\*\*\*'.The relate function is supported as a unary function for testing against the entire 9IM array. The 9IM array has nine elements of comparison—three elements of each shape versus three elements of the other shape. These elements are boundary, interior, and exterior.Each element for the first geometry (G1) can be tested against each element of the second geometry (G2) for truth or falsehood. Array elements can be selectively ignored.Relate is a unary function and is not compared to true or false.In the example, shapes G1 and G2 are compared. The string 'FFFTTT\*\*\*', delimited in single quotes, is used to specify whether the intersection of each of the elements in the 9IM array is true (T), false (F), or not tested (\*). There are exactly nine elements in the string, which correspond, from left to right, to the following nine relationships:

|  |  |  |
| --- | --- | --- |
| 1 | G1.interior | G2.interior |
| 2 | G1.interior | G2.boundary |
| 3 | G1.interior | G2.exterior |
| 4 | G1.boundary | G2.interior |
| 5 | G1.boundary | G2.boundary |
| 6 | G1.boundary | G2.exterior |
| 7 | G1.exterior | G2.interior |
| 8 | G1.exterior | G2.boundary |
| 9 | G1.exterior | G2.exterior |

In the previous example, relationships 1–3 must be false, relationships 4–6 must be true, and relationships 7–9 are not tested. The truth criteria for any given geometry relationship is that the dimension of intersection between the geometry is not null. This function does not evaluate the dimension of intersection between the geometries, whether or not the intersection exists.Any two geometries' exteriors always intersect, and the dimension of intersection is 2 (area). |
| Required | Yes, if spatialRel = "SpatialRelRelation". Otherwise the parameter is ignored.  |
| Syntax | "'" 9\*("F" / "T" / "\*") "'" |
| Example | relationParam='FFFTTT\*\*\*' |
| objectIds | The object identifiers of this layer/table to be queried |
| Required | No. Default: No filtering based on object identifiers. |
| Syntax | POSINT \*("," POSINT) |
| Example | objectIds=37,462 |
| outFields | The list of fields to be included in the returned result set. This list is a comma-delimited list of field names. If the shape field is specified in the list of return fields, it is ignored. To request geometry, returnGeometry can be set to true.A wildcard (\*) can also be specified as the value of this parameter. In this case, the query results include all the field values. |
| Required | No. Default:  |
| Syntax | NAME \*("," NAME) / "\*" |
| Example | outFields=AREANAME,ST,POP2000outFields=\* (wildcard usage) |
| returnIdsOnly | If true, the response only includes an array of object identifiers. Otherwise, the response is a feature set. |
| Required | No. Default: "false" |
| Syntax | BOOLEAN |
| Example | returnIdsOnly=true |

**Request Requirements**

|  |
| --- |
| * + 1. The Query resource SHALL accept requests that conform to the URI template in Table 11 and use any HTTP method identified in the same table.

query/request |

|  |
| --- |
| * + 1. The Query resource SHALL support all parameters and values specified in Table 12.

query/parameters |

### Feature Set and Feature ID Set resources

|  |
| --- |
| * + 1. The JSON representation of a response to a request on a Query resource SHALL validate against the JSON Schema **http://schemas.opengis.net/gsr/1.0/featureSet.json** (if returnIdsOnly=false), against the JSON Schema **http://schemas.opengis.net/gsr/1.0/featureIdSet.json** (if returnIdsOnly=true), or in case of an exception against JSON Schema http://schemas.opengis.net/gsr/1.0/exception.json.

query/valid |

|  |
| --- |
| * + 1. All parameters related to geometry SHALL be ignored when querying tables.

query/tables |

### Examples

Make a query using a WHERE clause:

http://example.com/rest/services/Earthquakes/FeatureServer/0/query?where=magnitude+%3E+4.5&outFields=\*&returnGeometry=true&returnIdsOnly=false&f=json

**Request**

GET /rest/services/Earthquakes/FeatureServer/0/query?where=magnitude+%3E+4.5&outFields=\*&returnGeometry=true&returnIdsOnly=false&f=json HTTP/1.1

Host: example.com

**Response**

HTTP/1.1 200 OK

Content-Type: application/json

Content-Length: nnn

{

 "objectIdFieldName" : "objectid",

 "globalIdFieldName" : "",

 "geometryType" : "GeometryPoint",

 "spatialReference" : {

 "wkid" : 4326

 },

 "fields" : [

 {

 "name" : "objectid",

 "type" : "FieldTypeOID",

 "alias" : "Object ID"

 },

 {

 "name" : "datetime",

 "type" : "FieldTypeDate",

 "alias" : "Earthquake Date",

 "length" : 36

 },

 {

 "name" : "depth",

 "type" : "FieldTypeDouble",

 "alias" : "Depth"

 },

 {

 "name" : "eqid",

 "type" : "FieldTypeString",

 "alias" : "Earthquake ID",

 "length" : 50

 },

 {

 "name" : "latitude",

 "type" : "FieldTypeDouble",

 "alias" : "Latitude"

 },

 {

 "name" : "longitude",

 "type" : "FieldTypeDouble",

 "alias" : "Longitude"

 },

 {

 "name" : "magnitude",

 "type" : "FieldTypeDouble",

 "alias" : "Magnitude"

 },

 {

 "name" : "numstations",

 "type" : "FieldTypeInteger",

 "alias" : "Number of Stations"

 },

 {

 "name" : "region",

 "type" : "FieldTypeString",

 "alias" : "Region",

 "length" : 200

 },

 {

 "name" : "source",

 "type" : "FieldTypeString",

 "alias" : "Source",

 "length" : 50

 },

 {

 "name" : "version",

 "type" : "FieldTypeString",

 "alias" : "Version",

 "length" : 50

 }

 ],

 "features" : [

 {

 "geometry" : {

 "x" : -178.24479999999991,

 "y" : 50.012500000000045

 },

 "attributes" : {

 "objectid" : 3745682,

 "datetime" : 1272210710000,

 "depth" : 31.100000000000001,

 "eqid" : "2010vma5",

 "latitude" : 50.012500000000003,

 "longitude" : -178.2448,

 "magnitude" : 4.7999999999999998,

 "numstations" : 112,

 "region" : "Andreanof Islands, Aleutian Islands, Alaska",

 "source" : "us",

 "version" : "Q"

 }

 },

 {

 "geometry" : {

 "x" : -72.865099999999927,

 "y" : -37.486599999999953

 },

 "attributes" : {

 "objectid" : 3745685,

 "datetime" : 1272210142999,

 "depth" : 40.600000000000001,

 "eqid" : "2010vma4",

 "latitude" : -37.486600000000003,

 "longitude" : -72.865099999999998,

 "magnitude" : 4.9000000000000004,

 "numstations" : 58,

 "region" : "Bio-Bio, Chile",

 "source" : "us",

 "version" : "7"

 }

 }

 ]

}

Query a table using a WHERE clause and return ObjectIDs only:

http://example.com/rest/services/Earthquakes/FeatureServer/0/query?where=magnitude+%3E+4.5&returnIdsOnly=true&f=json

**Request**

GET /rest/services/Earthquakes/FeatureServer/0/query?where=magnitude+%3E+4.5&returnIdsOnly=true&f=json HTTP/1.1

Host: example.com

**Response**

HTTP/1.1 200 OK

Content-Type: application/json

Content-Length: nnn

{

"objectIdFieldName":"objectid",

"objectIds":[1,2,3,4,5,7]

}

NOTE 1 In this case, the resource does - in general - not exist on the server but is compiled based on the query parameters. This is inline with HTTP 1.1, section 9.3: "The GET method means retrieve whatever information (in the form of an entity) is identified by the Request-URI. If the Request-URI refers to a data-producing process, it is the produced data which shall be returned as the entity in the response and not the source text of the process, unless that text happens to be the output of the process."

The same query may be submitted using POST, too.

**Request**

POST /rest/services/Earthquakes/FeatureServer/0/query HTTP/1.1

Host: example.com

Content-Length: nnn

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

where=magnitude+%3E+4.5&returnIdsOnly=true&f=json

The response is the same as in the GET example above.

NOTE 2 The use of POST has the disadvantage that the result is not cachable and not representable by just a URL, but HTTP POST is needed whenever the size of the URL may be long. This isn’t usually a factor for most API designs, but in the context of geographic information it happens quite frequently (serialized geometries can easily be bigger than 2000 characters).

# Temporal Query

## Overview

This conformance class adds support for temporal aspects by adding additional parameters for time-aware layers. The time parameter can be used to specify the time instant or the time extent to query.

Table 13 – Temporal Query overview

|  |  |  |
| --- | --- | --- |
| **Resource** | **Parameters** | **Resource representation** |
| Query | time | - |

## Layer/Table resources

|  |
| --- |
| * + 1. If a layer/table supports querying based on time, the associated Layer/Table resource SHALL include a timeInfo property.

query-time/timeInfo |

The timeInfo property provides information such as the start time field (or the time instance field), the end time field, the track ID field, the layer's time extent, and the suggested draw time interval.

**Example**

...

 "timeInfo" : {

 "startTimeField" : "COMPLETION",

 "endTimeField" : "PLUG\_DATE",

 "trackIdField" : null,

 "timeExtent" : [

 -2556057600000,

 1246060800000

 ],

 "timeReference" : null,

 "timeInterval" : 3,

 "timeIntervalUnits" : "TimeUnitsYears"

 },

...

## Query

### Dependency

This conformance class extends the requirements for the Query operation as specified in Clause 3.

### Query request

Table 14 – Additonal Query parameters

|  |  |
| --- | --- |
| time | The time instant or the time extent to query. All values are in milliseconds since 1 Jan. 1970 00:00:00 UTC.A single value identifies a time instance, two values separated by a comma describe a time extent (start and end time).A null value specified for start time or end time represents infinity for start or end time, respectively. |
| Required | No. Default: Time is not considered in the query. |
| Syntax | (POSINT / "NULL") ["," (POSINT / "NULL")] |
| Example | time=1199145600000 (1 Jan. 2008 00:00:00 UTC)time=1199145600000, 1230768000000 (1 Jan. 2008 00:00:00 UTC to 1 Jan. 2009 00:00:00 UTC) |

**Request Requirements**

|  |
| --- |
| * + 1. The Query resource SHALL support all parameters and values specified in Table 14.

query-time/parameters |

### Feature Set and Feature ID Set resources

This conformance class does not add any requirements on the JSON representation of the responses to requests in the Query resource.

# Query Related Records

## Overview

The Query Related Records operation is performed on a controller resource of a Layer/Table. The result of this operation is feature sets grouped by source layer/table ObjectIDs. Each feature set contains GeoServices REST API feature objects including the values for the fields requested by the user. The result is not stored on the server and is returned in the response to the request.

For related layers, if a user requests geometry information, the geometry of each feature is also returned in the feature set. For related tables, the feature set does not include geometries.

Users can provide arguments to the Query Related Records operation as query parameters.

Each feature set contains an array of field information objects for fields requested in the outFields parameter. See the Layer/Table section of this specification for details on fields. Note that the domains member is not included in field information objects returned with the response.

Table 15 – Query Related Records overview

|  |  |  |
| --- | --- | --- |
| **Resource** | **Parameters** | **Resource representation** |
| Query Related Records | f=jsondefinitionExpressionreturnGeometryoutSRobjectIdsoutFieldsrelationshipId | JSON representation validAll JSON schema elements |

## Query Related Records

### Query Related Records URI

In the following URI templates, these variables are used:

* featureServiceURI: URL of a Feature Service Root resource without any parameter
* layerOrTableId: id of the layer or table to query

Table 16 – Query Related Records reference

|  |  |
| --- | --- |
| **URI template** | {+featureServiceURI}/{layerOrTableId}/queryRelatedRecords{?f, definitionExpression,returnGeometry,outSR, objectIds,outFields,relationshipId} |
| **HTTP methods** | GETPOST (application/x-www-form-urlencoded) |
| **Parent Resource** | Layer/Table |

Table 17 – Query Related Records parameters

|  |  |
| --- | --- |
| **Parameter** | **Details** |
| f | The response format.  |
| Required | Yes |
| Syntax | "json" |
| Example | f=json |
| definitionExpression | The definition expression to be applied to the related layer/table. From the list of records that are related to the specified object identifiers, only those records that conform to this expression are returned. |
| Required | No. Default: no filter |
| Syntax | STRING |
| Example | definitionExpression=POP2000 > 100000 |
| returnGeometry | If true, the result set includes the geometry associated with each feature.If the outFields parameter is set to the wildcard, it implies returnGeometry=true, and setting returnGeometry to false has no effect.This parameter only applies to related layers. It will be ignored for related tables. |
| Required | No. Default: "true". |
| Syntax | BOOLEAN |
| Example | returnGeometry=false |
| outSR | 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 more requirements related to spatial references.Note that this parameter only applies to related layers. It is ignored for related tables. |
| Required | No. Default: The output geometry is exported in the spatial reference of the map. |
| Syntax | POSINT / JSON |
| Example | sr=4326 |
| objectIds | The object identifiers of the layer/table to be queried. Records related to these object identifiers will be queried. |
| Required | No. Default: No filtering based on object identifiers. |
| Syntax | POSINT \*("," POSINT) |
| Example | objectIds=37,462 |
| outFields | The list of fields from the related layer/table to be included in the returned feature set. This is a comma-delimited list of field names. If you specify the shape field in the list of return fields, it is ignored. To request geometry, set returnGeometry to true. Users can also specify the wildcard (\*) as the value of this parameter. In this case, the results will include all the field values. |
| Required | No. Default:  |
| Syntax | NAME \*("," NAME) / "\*" |
| Example | outFields=AREANAME,ST,POP2000outFields=\* (wildcard usage) |
| relationshipId | The ID of the relationship to be queried. The relationships that this layer/table participates in are included in the Layer/Table resource response. Records in layers/tables corresponding to the related layer/table of the relationship are queried. |
| Required | Yes |
| Syntax | POSINT |
| Example | relationshipId=4 |

**Request Requirements**

|  |
| --- |
| * + 1. The Query Related Records resource SHALL accept requests that conform to the URI template in Table 16 and use any HTTP method identified in the same table.

queryrel/request |

|  |
| --- |
| * + 1. The Query Related Records resource SHALL support all parameters and values specified in Table 17.

queryrel/parameters |

### Related Records resources

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

queryrel/valid |

### Example

Query related records defined by relationship ID 2; are related to ObjectIDs 3, 4, and 5; and are in layer 0:

http://example.com/rest/services/Petroleum/FeatureServer/0/queryRelatedRecords?objectIds=3,4,5&relationshipId=2&returnGeometry=true&outFields=\*&f=json

**Request**

GET /rest/services/Petroleum/FeatureServer/0/queryRelatedRecords?objectIds=3,4,5&relationshipId=2&returnGeometry=true&outFields=\*&f=json HTTP/1.1

Host: example.com

**Response**

HTTP/1.1 200 OK

Content-Type: application/json

Content-Length: nnn

{

 "geometryType" : "GeometryPolygon",

 "spatialReference" : {

 "wkid" : 4267

 },

 "fields" : [

 {

 "name" : "OBJECTID",

 "type" : "FieldTypeOID",

 "alias" : "OBJECTID"},

 {

 "name" : "FIELD\_KID",

 "type" : "FieldTypeString",

 "alias" : "FIELD\_KID",

 "length" : 25},

 {

 "name" : "APPROXACRE",

 "type" : "FieldTypeDouble",

 "alias" : "APPROXACRE"},

 {

 "name" : "FIELD\_NAME",

 "type" : "FieldTypeString",

 "alias" : "FIELD\_NAME",

 "length" : 150}

],

 "relatedRecordGroups" : [

 {

 "objectId" : 3,

 "relatedRecords" : [

 {

 "attributes" : {

 "OBJECTID" : 5540,

 "FIELD\_KID" : "1000147595",

 "APPROXACRE" : 95929,

 "FIELD\_NAME" : "LOST SPRINGS",

 },

 "geometry" : {

 "rings" : [

 [

 [-96.929599633999942,38.52426809800005],

 [-96.929602437999961,38.522448437000037],

 [-96.92959118999994,38.529723252000053],

 [-96.929594022999936,38.527905578000059],

 [-96.929596839999988,38.526087119000067],

 [-96.929599633999942,38.52426809800005]

 ]

 ]

 }

 }

 ]

 }

 ]

}

|  |
| --- |
| * + 1. All parameters related to geometry SHALL be ignored when querying tables.

queryrel/tables |

# Feature Editing

## Overview

This conformance class adds capabilities to add, update or delete features in feature layers and records in tables.

Table 18 – Feature Service Core overview

|  |  |  |
| --- | --- | --- |
| **Resource** | **Parameters** | **Resource representation** |
| Feature Service Root | - | JSON schema elements supported:- capabilities (value contains "Editing") |
| Add Features | f=jsonfeatures | JSON representation validAll JSON schema elements supported |
| Update Features | f=jsonfeatures | JSON representation validAll JSON schema elements supported |
| Delete Features | f=jsonobjectIdswheregeometrygeometryTypeinSRspatialRel | JSON representation validAll JSON schema elements supported |
| Apply Edits | f=jsonaddsupdatesdeletes | JSON representation validAll JSON schema elements supported |

## Feature Service Root resources

|  |
| --- |
| * + 1. The Feature Service Root resource SHALL contain the value "Editing" in the property "capabilities".

editing/capEditing |

**Example**

...

 "capabilities" : "Query,Editing"

...

## Add Features

### Overview

This operation adds features to the associated feature layer or table. The Add Features operation is performed on a controller resource under the feature service Layer/Table resource. The result of this operation is an array of edit results. Each edit result identifies a single feature and indicates if the edits were successful or not. If not, it also includes an error code and an error description.

Users provide arguments to the Add Features operation as query parameters.

### Add Features URI

In the following URI templates, these variables are used:

* featureServiceURI: URL of a Feature Service Root resource without any parameter
* layerOrTableId: id of the layer or table

Table 19 – Add Features reference

|  |  |
| --- | --- |
| **URI template**  | {+featureServiceURI}/{layerOrTableId}/addFeatures{?f,features} |
| **HTTP methods** | POST (application/x-www-form-urlencoded) |
| **Parent Resource Type** | Layer/Table |

Table 20 – Add Features parameters

|  |  |
| --- | --- |
| **Parameter** | **Details** |
| f | The response format.  |
| Required | Yes |
| Syntax | "json" |
| Example | f=json |
| features | The array of features to be added.  |
| Required | Yes |
| Syntax | "[" JSON \*("," JSON) "]" |
| Example | features=[ { "geometry" : {"x" : -118.15, "y" : 33.80},  "attributes" : { "OWNER" : "Joe Smith", "VALUE" : 94820.37, "APPROVED" : true, "LASTUPDATE" : 1227663551096 } }, { "geometry" : { "x" : -118.37, "y" : 34.086 },  "attributes" : { "OWNER" : "John Doe", "VALUE" : 17325.90, "APPROVED" : false, "LASTUPDATE" : 1227628579430 } }] |

**Request Requirements**

|  |
| --- |
| * + 1. The request of a Feature resource SHALL conform to the URI template in Table 5 and be accessed using a HTTP method identified in the same table.

editing/addRequest |

|  |
| --- |
| * + 1. The request of a Feature resource SHALL support all parameters and values specified in Table 6.

editing/addParameters |

|  |
| --- |
| * + 1. Features to be added to a feature layer SHALL include geometry, records to be added to a table SHALL not include geometry.

editing/addGeometry |

|  |
| --- |
| * + 1. The parameter "features" SHALL be an array of JSON objects which validate against the JSON Schema **http://schemas.opengis.net/gsr/1.0/feature.json**.

editing/addParameterValid |

### Add Features Result resources

|  |
| --- |
| * + 1. The JSON representation of a feature resource SHALL validate against the JSON Schema **http://schemas.opengis.net/gsr-fs/1.0/addResults.json** or in case of an exception against JSON Schema http://schemas.opengis.net/gsr/1.0/exception.json.

editing/addValid |

|  |
| --- |
| * + 1. The property "error" SHALL be provided, if and only if the property "success" is false.

editing/addErrors |

|  |
| --- |
| * + 1. The response SHALL contain one entry per feature/record submitted.

editing/addResponseComplete |

### Example

Add an array of features using the Add Features operation on a feature service Layer resource:

http://example.com/rest/services/311Incidents/FeatureServer/0/addFeatures

An input array of features might look like the following:

[

 {

 "attributes" : {

 "req\_id" : "508389",

 "req\_type" : "Graffiti Complaint - Public Property",

 "req\_date" : "09\/19\/2009",

 "req\_time" : "18:44",

 "address" : "11TH ST and HARRISON ST",

 "x\_coord" : "6008925.0",

 "y\_coord" : "2108713.8",

 "district" : "6",

 "status" : 1

 },

 "geometry" : {

 "x" : -122.41247978999991,

 "y" : 37.770630098000083

 }

 }

]

**Request**

POST /rest/services/311Incidents/FeatureServer/0/addFeatures HTTP/1.1

Host: example.com

Content-Length: nnn

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

f=json&features=[{"attributes":{"req\_id":"508389","req\_type":"Graffiti+Complaint+-+Public+Property","req\_date":"09\/19\/2009","req\_time":"18:44", "address":"11TH+ST+and+HARRISON+ST","x\_coord":"6008925.0","y\_coord":"2108713.8","district":"6","status":1},"geometry":{"x":-122.41247978999991, "y":37.770630098000083}}]

**Response**

HTTP/1.1 200 OK

Content-Type: application/json

Content-Length: nnn

{

"addResults" : [

 {

 "objectId" : 37,

 "globalId" : null,

 "success" : true

 }

 ]

}

The newly created feature resources are identified by the "objectId" properties in the response for items with "success"=true. They can be accessed at http://example.com/rest/services/311Incidents/FeatureServer/0/{objectId}.

NOTE 1 Since more than one resource (feature) may be created, there is no single resource that is created on the server as a result. HTTP 1.1, 9.5, states for such cases: "The action performed by the POST method might not result in a resource that can be identified by a URI. In this case, either 200 (OK) or 204 (No Content) is the appropriate response status, depending on whether or not the response includes an entity that describes the result."

## Update Features

### Overview

This operation updates features in a feature layer or table. The Update Features operation is performed on a controller resource under the feature service Layer/Table resource. The result of this operation is an array of edit results. Each edit result identifies a single feature and indicates if the edits were successful or not. If not, it also includes an error code and an error description.

Users provide arguments to the operation as query parameters.

### Update Features URI

In the following URI templates, these variables are used:

* featureServiceURI: URL of a Feature Service Root resource without any parameter
* layerOrTableId: id of the layer or table

Table 21 – Update Features reference

|  |  |
| --- | --- |
| **URI template**  | {+featureServiceURI}/{layerOrTableId}/updateFeatures{?f,features} |
| **HTTP methods** | POST (application/x-www-form-urlencoded) |
| **Parent Resource Type** | Layer/Table |

Table 22 – Update Features parameters

|  |  |
| --- | --- |
| **Parameter** | **Details** |
| f | The response format.  |
| Required | Yes |
| Syntax | "json" |
| Example | f=json |
| features | The array of features to be updated.  |
| Required | Yes |
| Syntax | "[" JSON \*("," JSON) "]" |
| Example | features=[ { "geometry" : {"x" : -118.15, "y" : 33.80},  "attributes" : { "OBJECTID" : 37, "OWNER" : "Joe Smith", "VALUE" : 94820.37, "APPROVED" : true, "LASTUPDATE" : 1227663551096 } }, { "geometry" : { "x" : -118.37, "y" : 34.086 },  "attributes" : { "OBJECTID" : 462, "OWNER" : "John Doe", "VALUE" : 17325.90, "APPROVED" : false, "LASTUPDATE" : 1227628579430 } }] |

**Request Requirements**

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

editing/updateRequest |

|  |
| --- |
| * + 1. The request of a Feature resource SHALL support all parameters and values specified in Table 22.

editing/updateParameters |

|  |
| --- |
| * + 1. Features to be updated in a feature layer SHALL include geometry, records to be updated in a table SHALL not include geometry.

editing/updateGeometry |

|  |
| --- |
| * + 1. Features and records to be updated SHALL include the object identifier property.

editing/updateObjectId |

|  |
| --- |
| * + 1. The parameter "features" SHALL be an array of JSON objects which validate against the JSON Schema **http://schemas.opengis.net/gsr/1.0/feature.json**.

editing/updateParameterValid |

### Update Features Result resources

|  |
| --- |
| * + 1. The JSON representation of a feature resource SHALL validate against the JSON Schema **http://schemas.opengis.net/gsr-fs/1.0/updateResults.json** or in case of an exception against JSON Schema http://schemas.opengis.net/gsr/1.0/exception.json.

editing /updateValid |

|  |
| --- |
| * + 1. The property "error" SHALL be provided, if and only if the property "success" is false.

editing/updateErrors |

|  |
| --- |
| * + 1. If a feature or record is submitted with an object identifier that does not exist in the layer/table, an error SHALL be reported.

editing/objectIdErrors |

|  |
| --- |
| * + 1. The response SHALL contain one entry per feature/record submitted.

editing/updateResponseComplete |

### Example

Update an array of features using the Update Features operation on a feature service Layer resource:

http://example.com/rest/services/311Incidents/FeatureServer/0/updateFeatures

An input array of features might look like the following:

[

 {

 "attributes" : {

 "objectid": 37

 "req\_id" : "508389",

 "req\_type" : "Graffiti Complaint - Private Property",

 "req\_date" : "09\/19\/2009",

 "req\_time" : "18:44",

 "address" : "11TH ST and HARRISON ST",

 "x\_coord" : "6008925.0",

 "y\_coord" : "2108713.8",

 "district" : "6",

 "status" : 2

 },

 "geometry" : {

 "x" : -122.41247978999991,

 "y" : 37.770630098000083

 }

 }

]

**Request**

POST /rest/services/311Incidents/FeatureServer/0/updateFeatures HTTP/1.1

Host: example.com

Content-Length: nnn

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

f=json&features=[{"attributes":{"objectid":37,"req\_id":"508389","req\_type":"Graffiti+Complaint+-+Public+Property","req\_date":"09\/19\/2009", "req\_time":"18:44","address":"11TH+ST+and+HARRISON+ST","x\_coord":"6008925.0","y\_coord":"2108713.8","district":"6","status":2},"geometry":{"x":

-122.41247978999991,"y":37.770630098000083}}]

**Response**

HTTP/1.1 200 OK

Content-Type: application/json

Content-Length: nnn

{

 "updateResults" : [

 {

 "objectId" : 37,

 "globalId" : null,

 "success" : true

 }

 ]

}

The updated Feature resources are identified by the "objectId" properties in the response for items with "success"=true. They may be accessed at http://example.com/rest/services/311Incidents/FeatureServer/0/{objectId}.

NOTE Since more than one resource (feature) may be created, there is no single resource that is created on the server as a result. HTTP 1.1, section 9.5, states for such cases: "The action performed by the POST method might not result in a resource that can be identified by a URI. In this case, either 200 (OK) or 204 (No Content) is the appropriate response status, depending on whether or not the response includes an entity that describes the result."

## Delete Features

### Overview

This operation deletes features in a feature layer or table. The Update Features operation is performed on a controller resource under the feature service Layer/Table resource. The result of this operation is an array of edit results. Each edit result identifies a single feature and indicates if the edits were successful or not. If not, it also includes an error code and an error description.

Users provide arguments to the operation as query parameters.

### Delete Features URI

In the following URI templates, these variables are used:

* featureServiceURI: URL of a Feature Service Root resource without any parameter
* layerOrTableId: id of the layer or table

Table 23 – Delete Features reference

|  |  |
| --- | --- |
| **URI template**  | {+featureServiceURI}/{layerOrTableId}/deleteFeatures{?f,objectIds,where,geometry, geometryType,inSR,spatialRel} |
| **HTTP methods** | POST (application/x-www-form-urlencoded) |
| **Parent Resource Type** | Layer/Table |

Table 24 – Delete Features parameters

|  |  |
| --- | --- |
| **Parameter** | **Details** |
| f | The response format.  |
| Required | Yes |
| Syntax | "json" / "image" |
| Example | f=json |
| geometry | The geometry to apply as a spatial filter. The type of the geometry is specified by the geometryType parameter. In addition to the JSON structures, for points and envelopes, users may specify the geometries with a simpler comma-separated syntax.The parameter is ignored, if a value for objectIds is specified.**JSON:**An input geometry. The geometry is of the type defined by the geometryType parameter.See GeoServices REST API – Core, Clause 9 for the schema of the geometries.**Simple syntax for point geometries:**When using points, the geometries may alternatively be specified with a simpler comma-separated syntax.**Simple syntax for envelope geometries:**When using envelopes, the geometries may alternatively be specified with a simpler comma-separated syntax (first the lower left corner, then the upper right corner).NOTE Coordinates always use a period as the decimal separator even in countries where a comma is traditionally used. |
| Required | No. Default is that no spatial filter is applied. |
| Syntax | JSON / X "," Y / XMIN "," YMIN "," XMAX "," YMAX  |
| Examples | **JSON:**geometryType=GeometryPoint&geometry={x: -104, y: 35.6}**Simple syntax for point geometries:**geometryType=GeometryPoint&geometry=-104,35.6**Simple syntax for envelope geometries:**geometryType=GeometryEnvelope&geometry=-104,35.6,-94.32,41 |
| geometryType | The type of geometry specified by the geometry parameter. The well-known geometry types include point ("GeometryPoint"), multi point ("GeometryMultiPoint"), polyline ("GeometryPolyline"), polygon ("GeometryPolygon"), and envelope ("GeometryEnvelope"). The parameter is ignored, if a value for objectIds is specified.See GeoServices REST API – Core, Clause 9 for additional information about these geometry types. |
| Required | No. Default: "GeometryEnvelope" |
| Syntax | "GeometryPoint" / "GeometryMultiPoint" / "GeometryPolyline" / "GeometryPolygon" / "GeometryEnvelope" |
| Example | geometryType=GeometryPolygon |
| where | A WHERE clause for the query filter. Any legal SQL WHERE clause operating on the fields in the layer is allowed. The parameter is ignored, if a value for objectIds is specified. |
| Required | No. Default: no filter |
| Syntax | STRING |
| Example | where=POP2000 > 350000 |
| inSR | The spatial reference of the input geometry. The spatial reference is specified as either a well-known ID (WKID) or a spatial reference JSON object. See Geoservices REST API Core standard for more requirements related to spatial references.The parameter is ignored, if a value for objectIds is specified. |
| Required | No. Default: The geometry is assumed to be in the spatial reference of the map. |
| Syntax | POSINT / JSON |
| Example | sr=4326 |
| spatialRel | The spatial relationship to be applied on the input geometry while performing the query. The supported spatial relationships include intersects, contains, envelope intersects, and within.The parameter is ignored, if a value for objectIds is specified.Pre-defined values: SpatialRelIntersects: Returns a feature if any spatial relationship is found. Applies to all shape type combinations.SpatialRelContains: Returns a feature if its shape is wholly contained within the search geometry. Valid for all shape type combinations.SpatialRelCrosses: Returns a feature if the intersection of the interiors of the two shapes is not empty and has a lower dimension than the maximum dimension of the two shapes. Two lines that share an endpoint in common do not cross. Valid for line/line, line/area, multipoint/area, and multipoint/line shape type combinations.SpatialRelEnvelopeIntersects: Returns a feature if the envelope of the two shapes intersects.SpatialRelIndexIntersects: Returns a feature if the envelope of the query geometry intersects the index entry for the target geometry.SpatialRelOverlaps: Returns a feature if the intersection of the two shapes results in an object of the same dimension but different from both of the shapes. Applies to area/area, line/line, and multipoint/multipoint shape type combinations.SpatialRelTouches: Returns a feature if the two shapes share a common boundary. However, the intersection of the interiors of the two shapes must be empty. In the point/line case, the point may touch an endpoint only of the line. Applies to all combinations except point/point.SpatialRelWithin: Returns a feature if its shape wholly contains the search geometry. Valid for all shape type combinations.SpatialRelRelation: Defines a custom spatial relationship as specified by the relationParam parameter. |
| Required | No. Default: "SpatialRelIntersects" |
| Syntax | "SpatialRelIntersects" / "SpatialRelContains" / "SpatialRelCrosses" / "SpatialRelEnvelopeIntersects" / "SpatialRelIndexIntersects" / "SpatialRelOverlaps" / "SpatialRelTouches" / "SpatialRelWithin" / "SpatialRelRelation" |
| Example | spatialRel=SpatialRelContains |
| objectIds | The object identifiers of this layer/table to be deleted.If this parameter is specified, all other parameter are ignored. |
| Required | No. Default: No filtering based on object identifiers. |
| Syntax | POSINT \*("," POSINT) |
| Example | objectIds=37,462 |

**Request Requirements**

|  |
| --- |
| * + 1. The request of a Feature resource SHALL conform to the URI template in Table 23 and be accessed using a HTTP method identified in the same table.

editing/deleteRequest |

|  |
| --- |
| * + 1. The request of a Feature resource SHALL support all parameters and values specified in Table 24.

editing/deleteParameters |

### Delete Features Result resources

|  |
| --- |
| * + 1. The JSON representation of a Delete Features resource SHALL validate against the JSON Schema **http://schemas.opengis.net/gsr-fs/1.0/deleteResults.json**, if the objectIds parameter was provided, be a JSON response of '{"success":true}', if no objectIds paramater was provided or in case of an exception it SHALL validate against JSON Schema http://schemas.opengis.net/gsr/1.0/exception.json.

editing/deleteValid |

|  |
| --- |
| * + 1. The property "error" SHALL be provided, if and only if the property "success" is false.

editing/deleteErrors |

|  |
| --- |
| * + 1. If the request contained the parameter objectIds, the response SHALL contain one entry per object identifier.

editing/deleteResponseComplete |

### Example

Delete an array of features using the Delete Features operation on a feature service Layer resource:

http://example.com/rest/services/311Incidents/FeatureServer/0/deleteFeatures

The input to the Delete Features operation can be a list of unique IDs, a WHERE clause to apply as an attribute filter, or a geometry to apply as a spatial filter. In the request below we filter by object identifiers.

**Request**

POST /rest/services/311Incidents/FeatureServer/0/deleteFeatures HTTP/1.1

Host: example.com

Content-Length: nnn

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

f=json&objectIds=37,462

**Response**

HTTP/1.1 200 OK

Content-Type: application/json

Content-Length: nnn

{

 "deleteResults" : [

 {

 "objectId" : 37,

 "globalId" : null,

 "success" : true

 },

 {

 "objectId" : 462,

 "globalId" : null,

 "success" : false,

 "error" : {

 "code" : 60,

 "description" : "Features whose last update was less than 2 days ago cannot be deleted."

 }

 }

 ]

}

## Apply Edits

### Overview

This operation adds, updates and deletes features to the associated feature layer or table. The is performed on a controller resource under the feature service Layer/Table resource. The result of this operation are three arrays of edit results (for additions, updates, and deletions, respectively). Each edit result identifies a single feature and indicates if the edits were successful or not. If not, it also includes an error code and an error description.

Users provide arguments to the as query parameters.

### Apply Edits Features URI

In the following URI templates, these variables are used:

* featureServiceURI: URL of a Feature Service Root resource without any parameter
* layerOrTableId: id of the layer or table

Table 25 – Apply Edits reference

|  |  |
| --- | --- |
| **URI template**  | {+featureServiceURI}/{layerOrTableId}/applyEdits{?f,adds,updates,deletes} |
| **HTTP methods** | POST (application/x-www-form-urlencoded) |
| **Parent Resource Type** | Layer/Table |

Table 26 – Apply Edits parameters

|  |  |
| --- | --- |
| **Parameter** | **Details** |
| f | The response format.  |
| Required | Yes |
| Syntax | "json" |
| Example | f=json |
| adds | The array of features to be added.  |
| Required | No. Default is that no features will be added. |
| Syntax | "[" JSON \*("," JSON) "]" |
| Example | adds=[ { "geometry" : {"x" : -118.15, "y" : 33.80},  "attributes" : { "OWNER" : "Joe Smith", "VALUE" : 94820.37, "APPROVED" : true, "LASTUPDATE" : 1227663551096 } }, { "geometry" : { "x" : -118.37, "y" : 34.086 },  "attributes" : { "OWNER" : "John Doe", "VALUE" : 17325.90, "APPROVED" : false, "LASTUPDATE" : 1227628579430 } }] |
| updates | The array of features to be updated.  |
| Required | No. Default is that no features will be updated. |
| Syntax | "[" JSON \*("," JSON) "]" |
| Example | updates=[ { "geometry" : {"x" : -118.15, "y" : 33.80},  "attributes" : { "OBJECTID" : 37, "OWNER" : "Joe Smith", "VALUE" : 94820.37, "APPROVED" : true, "LASTUPDATE" : 1227663551096 } }, { "geometry" : { "x" : -118.37, "y" : 34.086 },  "attributes" : { "OBJECTID" : 462, "OWNER" : "John Doe", "VALUE" : 17325.90, "APPROVED" : false, "LASTUPDATE" : 1227628579430 } }] |
| deletes | The object identifiers of this layer/table to be deleted.If this parameter is specified, all other parameter are ignored. |
| Required | No. Default is that no features will be deleted. |
| Syntax | POSINT \*("," POSINT) |
| Example | deletes=37,462 |

**Request Requirements**

|  |
| --- |
| * + 1. The request of a Feature resource SHALL conform to the URI template in Table 25 and be accessed using a HTTP method identified in the same table.

editing/applyRequest |

|  |
| --- |
| * + 1. The request of a Feature resource SHALL support all parameters and values specified in Table 26.

editing/applyParameters |

|  |
| --- |
| * + 1. Features to be added to or updated in a feature layer SHALL include geometry, records to be added to or updated in a table SHALL not include geometry.

editing/applyGeometry |

|  |
| --- |
| * + 1. The parameters "adds" and "updates" SHALL be arrays of JSON objects which validate against the JSON Schema **http://schemas.opengis.net/gsr/1.0/feature.json**.

editing/applyParameterValid |

### Apply Edits Result resources

|  |
| --- |
| * + 1. The JSON representation of a feature resource SHALL validate against the JSON Schema **http://schemas.opengis.net/gsr-fs/1.0/editResults.json** or in case of an exception against JSON Schema http://schemas.opengis.net/gsr/1.0/exception.json.

editing/applyValid |

|  |
| --- |
| * + 1. The property "error" SHALL be provided, if and only if the property "success" is false.

editing/applyErrors |

|  |
| --- |
| * + 1. The response SHALL contain one entry per feature/record/object identifer submitted.

editing/applyResponseComplete |

### Example

Add, update and delete features using the Apply Edits operation on a feature service Layer resource:

http://example.com/rest/services/311Incidents/FeatureServer/0/applyEdits

where the payload might look like the following:

f=json&

adds=[

 {

 "attributes" : {

 "req\_id" : "508389",

 "req\_type" : "Graffiti Complaint - Public Property",

 "req\_date" : "09\/19\/2009",

 "req\_time" : "18:44",

 "address" : "11TH ST and HARRISON ST",

 "x\_coord" : "6008925.0",

 "y\_coord" : "2108713.8",

 "district" : "6",

 "status" : 1

 },

 "geometry" : {

 "x" : -122.41247978999991,

 "y" : 37.770630098000083

 }

 }

]&

updates=[

 {

 "attributes" : {

 "objectid": 1234567

 "req\_id" : "508385",

 "req\_type" : "Graffiti Complaint - Private Property",

 "req\_date" : "09\/17\/2009",

 "req\_time" : "17:26",

 "address" : "12TH ST and MAIN ST",

 "x\_coord" : "6008846.0",

 "y\_coord" : "2109264.8",

 "district" : "6",

 "status" : 23

 },

 "geometry" : {

 "x" : -122.41243757345734,

 "y" : 37.77063745743753

 }

 }

]

&

deletes=9,625&

**Request**

POST /rest/services/311Incidents/FeatureServer/0/applyEdits HTTP/1.1

Host: example.com

Content-Length: nnn

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

f=json&…

**Response**

HTTP/1.1 200 OK

Content-Type: application/json

Content-Length: nnn

{

 "addResults" : [

 {

 "objectId" : 37,

 "globalId" : null,

 "success" : true

 }

 ],

 "updateResults" : [

 {

 "objectId" : 1234567,

 "globalId" : null,

 "success" : false,

 "error" : {

 "code" : 30,

 "description" : "STATUS 23 not valid."

 }

 }

 ],

 "deleteResults" : [

 {

 "objectId" : 9,

 "globalId" : null,

 "success" : true

 },

 {

 "objectId" : 625,

 "globalId" : null,

 "success" : false,

 "error" : {

 "code" : 60,

 "description" : "Features whose last update was less than 2 days ago cannot be deleted."

 }

 }

 ]

}

# Attachments and HTML Popups

## Overview

The resources specified in this conformance class provide access to attachments and HTML popups (information intended for display in an HTML pop-up balloon) associated with a feature.

Table 27 – Attachments and HTML Popups overview

|  |  |  |
| --- | --- | --- |
| **Resource** | **Parameters** | **Resource representation** |
| Attachement Info | f=json | JSON representation validAll JSON schema elements |
| Attachment | - | - |
| HTML Popup | f=json | JSON representation validAll JSON schema elements |
| Add Attachement | f=jsonattachement | JSON representation validAll JSON schema elements |
| Update Attachement | f=jsonattachementattachmentId | JSON representation validAll JSON schema elements |
| Delete Attachements | f=jsonattachementIds | JSON representation validAll JSON schema elements |

## Attachment Infos

### Overview

The Attachment Infos resource returns information about attachments associated with a feature. This resource is available only if the layer has advertised that it has attachments. A layer has attachments if its hasAttachments property is set to true.

Each attachment info includes, for example, the ID, content type, size, and name of the attachment. The Attachment Infos resource has one child resource, the attachments.

### Attachment Infos URI

In the following URI templates, these variables are used:

* featureServiceURI: URL of a Feature Service Root resource without any parameter
* layerOrTableId: id of the layer or table
* featureId: id of the feature in the layer or table

Table 28 – Attachment Infos reference

|  |  |
| --- | --- |
| **URI template**  | {+featureServiceURI}/{layerOrTableId}/{featureId}/attachments{?f} |
| **HTTP methods** | GET |
| **Parent Resource Type** | Feature |
| **Child Resource Types** | Attachment |

Table 29 – Attachment Infos parameters

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

**Request Requirements**

|  |
| --- |
| * + 1. The Attachment Infos resource SHALL accept requests that conform to the URI template in Table 28 and use any HTTP method identified in the same table.

attachments/infoRequest |

|  |
| --- |
| * + 1. The Attachment Infos resource SHALL support all parameters and values specified in Table 29.

attachements/infoParameters |

### Attachment Infos resources

|  |
| --- |
| * + 1. The JSON representation of an Attachment Infos resource SHALL validate against the JSON Schema **http://schemas.opengis.net/gsr/1.0/attachmentInfos.json** or in case of an exception against JSON Schema http://schemas.opengis.net/gsr/1.0/exception.json.

attachments/infoValid |

### Example

The URL to get information on the attachments of feature 818654 in layer/table 0:

http://example.com/rest/services/311Incidents/FeatureServer/0/818654/attachments

**Request**

GET /rest/services/311Incidents/FeatureServer/0/818654/attachments HTTP/1.1

Host: example.com

**Response**

HTTP/1.1 200 OK

Content-Type: application/json

Content-Length: nnn

{

 "attachmentInfos": [

 {

 "id": 3,

 "contentType": "video/quicktime",

 "size": 397540,

 "name": "360 degree view"

 },

 {

 "id": 2,

 "contentType": "application/pdf",

 "size": 270133,

 "name": "Sales Deed"

 },

 {

 "id": 1,

 "contentType": "image/jpg",

 "size": 45325,

 "name": "Picture of the house"

 }

 ]

}

## Attachment

### Overview

The Attachment resource represents an individual attachment associated with a feature. This resource is available only if the layer has advertised that it has attachments. A layer has attachments if its hasAttachments property is set to true.

### Attachment URI

In the following URI templates, these variables are used:

* featureServiceURI: URL of a Feature Service Root resource without any parameter
* layerOrTableId: id of the layer or table
* featureId: id of the feature in the layer or table
* attachmentId: id of the attachement in the Attachment Info

Table 30 – Attachment reference

|  |  |
| --- | --- |
| **URI template**  | {+featureServiceURI}/{layerOrTableId}/{featureId}/attachments/{attachmentId}{?f} |
| **HTTP methods** | GET |
| **Parent Resource Type** | Attachment Infos |

**Request Requirements**

|  |
| --- |
| * + 1. The Attachment resource SHALL accept requests that conform to the URI template in Table 30 and use any HTTP method identified in the same table.

attachements/request |

### Attachment resources

|  |
| --- |
| * + 1. The attachment SHALL be returned to the client. If the attachment is not found, an HTTP status code of 404 SHALL be returned.

attachments/valid |

### Example

The URL to attachment 1 of feature 818654 in layer/table 0:

http://example.com/rest/services/311Incidents/FeatureServer/0/818654/attachments/1

**Request**

GET /rest/services/311Incidents/FeatureServer/0/818654/attachments/1 HTTP/1.1

Host: example.com

**Response**

HTTP/1.1 200 OK

Content-Type: application/pdf

Content-Length: nnn

…

## HTML Popup

### Overview

The HTML Popup resource provides details about any HTML pop-ups that are to appear in association with each feature in a pop-up balloon.

This resource is available when a Layer/Table resource's htmlPopupType parameter is not ServerHTMLPopupTypeNone.

### HTML Popup URI

In the following URI templates, these variables are used:

* featureServiceURI: URL of a Feature Service Root resource without any parameter
* layerOrTableId: id of the layer or table
* featureId: id of the feature in the layer or table

Table 31 – HTML Popup reference

|  |  |
| --- | --- |
| **URI template**  | {+featureServiceURI}/{layerOrTableId}/{featureId}/htmlPopup{?f} |
| **HTTP methods** | GET |
| **Parent Resource Type** | Feature |

Table 32 – HTML Popup parameters

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

**Request Requirements**

|  |
| --- |
| * + 1. The HTML Popup resource SHALL accept requests that conform to the URI template in Table 31 and use any HTTP method identified in the same table.

attachments/popupRequest |

|  |
| --- |
| * + 1. The HTML Popup resource SHALL support all parameters and values specified in Table 32.

attachments/popupParameters |

### HTML Popup resources

|  |
| --- |
| * + 1. The JSON representation of a HTML Popup resource SHALL validate against the JSON Schema **http://schemas.opengis.net/gsr/1.0/htmlPopup.json** or in case of an exception against JSON Schema http://schemas.opengis.net/gsr/1.0/exception.json.

attachements/popupValid |

### Example

The URL to the HTML Popup of feature 818654 in layer/table 0:

http://example.com/rest/services/311Incidents/FeatureServer/0/818654/htmlPopup?f=json

**Request**

GET /rest/services/311Incidents/FeatureServer/0/818654/htmlPopup?f=json HTTP/1.1

Host: example.com

**Response**

HTTP/1.1 200 OK

Content-Type: application/json

Content-Length: nnn

{

 "htmlPopupType" : "ServerHTMLPopupTypeAsHTMLText",

 "content": "A <b>Sample HTML</b> pop up."

}

## Add Attachment

### Overview

This operation adds an attachement to the associated feature. The operation is performed on a controller resource under the feature service Feature resource.

The result of this operation is an edit result. The edit result indicates if the edit was successful or not. If not, it also includes an error code and an error description. If successful, the objectId of the result is the ID of the new attachment.

Since this request uploads a file, a multipart request as per Internet Engineering Task Force (IETF) RFC1867 is used.

This operation is available only if the layer has advertised that it has attachments. A layer has attachments if its hasAttachments property is set to true.

### Add Attachement URI

In the following URI templates, these variables are used:

* featureServiceURI: URL of a Feature Service Root resource without any parameter
* layerOrTableId: id of the layer or table
* featureId: id of the feature in the layer or table

Table 33 – Add Attachement reference

|  |  |
| --- | --- |
| **URI template**  | {+featureServiceURI}/{layerOrTableId}/{featureId}/addAttachment |
| **HTTP methods** | POST (multipart/formdata) |
| **Parent Resource Type** | Feature |

Table 34 – Add Attachement parameters

|  |  |
| --- | --- |
| **Parameter** | **Details** |
| f | The response format.  |
| Required | Yes |
| Syntax | "json" |
| Example | See example below |
| attachment | The file to be uploaded as a new feature attachment. The content type, size, and name of the attachment will be derived from the uploaded file. |
| Required | Yes |
| Syntax | \*CHAR |
| Example | See example below |

**Request Requirements**

|  |
| --- |
| * + 1. The request to an Add Attachement resource SHALL conform to the URI template in Table 33 and be accessed using a HTTP method identified in the same table.

attachements/addRequest |

|  |
| --- |
| * + 1. The request to an Add Attachement resource SHALL support all parameters and values specified in Table 34.

attachements/addParameters |

### Add Attachement Result resources

|  |
| --- |
| * + 1. The JSON representation of a feature resource SHALL validate against the JSON Schema **http://schemas.opengis.net/gsr-fs/1.0/addAttResult.json** or in case of an exception against JSON Schema http://schemas.opengis.net/gsr/1.0/exception.json.

attachements/addValid |

|  |
| --- |
| * + 1. The property "error" SHALL be provided, if and only if the property "success" is false.

attachements/addErrors |

### Example

Add an attachment using the Add Attachment operation on a Feature resource. In this sample URL, the Add Attachment operation is performed on feature ID 818654 belonging to layer 0 of the 311Incidents feature service.

http://example.com/rest/services/311Incidents/FeatureServer/0/818654/addAttachment

**Request**

POST /rest/services/311Incidents/FeatureServer/0/818654/addAttachment HTTP/1.1

Host: example.com

Content-Length: nnn

Content-type: multipart/form-data, boundary=AaB03x

--AaB03x

content-disposition: form-data; name="f"

json

--AaB03x

content-disposition: form-data; name="attachement"; filename="file.pdf"

Content-Type: application/pdf

... contents of file.pdf ...

--AaB03x--

**Response**

HTTP/1.1 200 OK

Content-Type: application/json

Content-Length: nnn

{

 "addAttachmentResult": {

 "objectId" : 58,

 "globalId" : null,

 "success" : true

 }

}

The newly created Attachement resources is identified by the "objectId" property in the response. It may be accessed at http://example.com/rest/services/311Incidents/FeatureServer/0/818654/attachments/{objectId}.

NOTE HTTP 1.1 recommends to return 201 (Created) and a Location header with the URI of the new resource. Also, the new resource should be a subordinate to the POST URI. These recommendations are not followed by the GeoServices REST API for the following reasons (note that these are recommendations, not requirements):

* When this specification was originally created, HTTP support was limited in the different client environments, so the response code 200 is used. Inline with the recommendations of HTTP 1.1, an enitiy that describes the status of the request and refers to the new resource is returned.
* One of the inconveniences of using JSONP which is needed for cross-domain scripting is that the server always needs to respond with an HTTP status of 200. Otherwise a browser’s network stack rejects the response from the server, and the client callbacks are never even called. To overcome this, the GeoServices REST API - like other APIs - responds with errors wrapped inside the response.

## Update Attachment

### Overview

This operation updates an attachement associated with a feature. The operation is performed on a controller resource under the feature service Feature resource.

The result of this operation is an edit result. The edit result indicates if the edit was successful or not. If not, the result includes an error code and an error description. If successful, the ObjectID of the result is the ID of the updated attachment.

Since this request uploads a file, a multipart request as per Internet Engineering Task Force (IETF) RFC1867 is used.

This operation is available only if the layer has advertised that it has attachments. A layer has attachments if its hasAttachments property is set to true.

### Update Attachement URI

In the following URI templates, these variables are used:

* featureServiceURI: URL of a Feature Service Root resource without any parameter
* layerOrTableId: id of the layer or table
* featureId: id of the feature in the layer or table

Table 35 – Update Attachement reference

|  |  |
| --- | --- |
| **URI template**  | {+featureServiceURI}/{layerOrTableId}/{featureId}/updateAttachment |
| **HTTP methods** | POST (multipart/formdata) |
| **Parent Resource Type** | Feature |

Table 36 – Update Attachement parameters

|  |  |
| --- | --- |
| **Parameter** | **Details** |
| f | The response format.  |
| Required | Yes |
| Syntax | "json" |
| Example | See example below |
| attachment | The file to be uploaded as the updated feature attachment. The content type, size, and name of the attachment will be derived from the uploaded file. |
| Required | Yes |
| Syntax | \*CHAR |
| Example | See example below |
| attachementId | The ID of the attachment to be updated. |
| Required | Yes |
| Syntax | POSINT |
| Example | attachmentId=58 |

**Request Requirements**

|  |
| --- |
| * + 1. The request to an Update Attachement resource SHALL conform to the URI template in Table 35 and be accessed using a HTTP method identified in the same table.

attachements/updateRequest |

|  |
| --- |
| * + 1. The request to an Update Attachement resource SHALL support all parameters and values specified in Table 36.

attachements/updateParameters |

### Update Attachement Result resources

|  |
| --- |
| * + 1. The JSON representation of a feature resource SHALL validate against the JSON Schema **http://schemas.opengis.net/gsr-fs/1.0/updAttResults.json** or in case of an exception against JSON Schema http://schemas.opengis.net/gsr/1.0/exception.json.

attachements/updateValid |

|  |
| --- |
| * + 1. The property "error" SHALL be provided, if and only if the property "success" is false.

attachements/updateErrors |

### Example

Update an attachment using the Update Attachment operation on a Feature resource. In this example, the Update Attachment operation is performed on attachment 58 in feature 818654 belonging to layer 0 of the 311Incidents feature service:

http://example.com/rest/services/311Incidents/FeatureServer/0/818654/updateAttachment

**Request**

POST /rest/services/311Incidents/FeatureServer/0/818654/updateAttachment HTTP/1.1

Host: example.com

Content-Length: nnn

Content-type: multipart/form-data, boundary=AaB03x

--AaB03x

content-disposition: form-data; name="f"

json

--AaB03x

content-disposition: form-data; name="attachmentId"

58

--AaB03x

content-disposition: form-data; name="attachement"; filename="file.html"

Content-Type: text/html

... contents of file.html ...

--AaB03x--

**Response**

HTTP/1.1 200 OK

Content-Type: application/json

Content-Length: nnn

{

 "updateAttachmentResult": {

 "objectId" : 58,

 "globalId" : null,

 "success" : true

 }

}

NOTE The natural HTTP method would have been PUT. However, the GeoServices REST API does not use PUT or DELETE. Instead POST and a rather simple controller resource is used instead. The reasons for not using PUT and DELETE are discussed in the part 1 of the GeoServices REST API series. In summary:

* Lacking web browser support for HTTP PUT and DELETE
* Some rich internet application clients do not fully support PUT and DELETE
* Sometimes firewalls and proxies strip out HTTP PUTs and DELETEs. This can be mitigated by forcing SSL for all requests, but this is not practical. Some RESTful APIs recommend HTTP method overloading to get around this, which would be a hack. In this case one would use POST, but in the header (or in a query parameter) one specifies that one really wants to do a PUT or DELETE.

## Delete Attachments

### Overview

This operation deletes attachements associated with a feature. The operation is performed on a controller resource under the feature service Feature resource.

The result of this operation is an array of results indicating whether the individual edits were successful or not. If not, the result includes an error code and an error description. If successful, the ObjectID in the results is the ID of the deleted attachment.

This operation is available only if the layer has advertised that it has attachments. A layer has attachments if its hasAttachments property is set to true.

Users provide arguments to the operation as query parameters.

### Delete Attachements URI

In the following URI templates, these variables are used:

* featureServiceURI: URL of a Feature Service Root resource without any parameter
* layerOrTableId: id of the layer or table
* featureId: id of the feature in the layer or table

Table 37 – Delete Attachements reference

|  |  |
| --- | --- |
| **URI template**  | {+featureServiceURI}/{layerOrTableId}/{featureId}/deleteAttachments |
| **HTTP methods** | POST (application/x-www-form-urlencoded) |
| **Parent Resource Type** | Feature |

Table 38 – Delete Attachements parameters

|  |  |
| --- | --- |
| **Parameter** | **Details** |
| f | The response format.  |
| Required | Yes |
| Syntax | "json" |
| Example | f=json |
| attachementIds | The IDs of the attachments to be deleted. |
| Required | Yes |
| Syntax | POSINT \*("," POSINT) |
| Example | attachmentIds=58,4 |

**Request Requirements**

|  |
| --- |
| * + 1. The request to an Delete Attachement resource SHALL conform to the URI template in Table 37 and be accessed using a HTTP method identified in the same table.

attachements/deleteRequest |

|  |
| --- |
| * + 1. The request to an Delete Attachement resource SHALL support all parameters and values specified in Table 38.

attachements/deleteParameters |

### Delete Attachements Result resources

|  |
| --- |
| * + 1. The JSON representation of a feature resource SHALL validate against the JSON Schema **http://schemas.opengis.net/gsr-fs/1.0/delAttResults.json** or in case of an exception against JSON Schema http://schemas.opengis.net/gsr/1.0/exception.json.

attachements/deleteValid |

|  |
| --- |
| * + 1. In each result, the property "error" SHALL be provided, if and only if the property "success" is false.

attachements/deleteErrors |

### Example

Delete two attachments of feature ID 818654 belonging to layer 0 of the 311Incidents feature service:

**Request**

POST /rest/services/311Incidents/FeatureServer/0/818654/deleteAttachments HTTP/1.1

Host: example.com

Content-Length: nnn

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

f=json&attachementIds=58,4

**Response**

HTTP/1.1 200 OK

Content-Type: application/json

Content-Length: nnn

{

 "deleteAttachmentResults": [

 {

 "objectId": 58,

 "globalId": null,

 "success": true

 },

 {

 "objectId": 4,

 "globalId": null,

 "success": false,

 "error": {

 "code": 50,

 "description": "Attachment not found"

 }

 }

 ]

}

# Templates

## Overview

Creating features may be accomplished through the use of feature templates. Feature templates are associated with layers and define all the information required to create a feature:

* the attributes a feature is created with ("feature prototype")
* the user inteface tool that may be used to create that feature and its geometry.

Templates have a name and description.

A layer can have multiple templates associated with it, where each template has different default settings.

EXAMPLE For a roads layer with classifications of freeway, major highway, and local road, one could have three different templates with each one having a different default attribute for the type of road. This makes it easier for clients to create the new type of road the users wants from the start.

The feature templates of a layer/table that may be applied to all instances of that layer/table are provided in the templates property of a Layer/Table resource object. Templates applicable for each subtype of the layer/table are listed under each subtype (path in the JSON object: #/types/templates).

## Layer/Table resources

|  |
| --- |
| * + 1. If a layer/table supports feature templates, the associated Layer/Table resource SHALL include template objects.

templates/layerOrTable |

**Example**

...

 "templates" : [

 {

 "name" : "Graffiti Complaint",

 "description" : "",

 "drawingTool" : "FeatureEditToolPoint",

 "prototype" : {

 "attributes" : {

 "status" : 1,

 "req\_id" : null,

 "req\_type" : "Graffiti Complaint - Public Property",

 "req\_date" : null,

 "req\_time" : null,

 "address" : null,

 "x\_coord" : null,

 "y\_coord" : null,

 "district" : null

 }

 }

 }

 ]

...

The drawing tool property SHOULD be one of the following values:

|  |  |
| --- | --- |
|  |  |
| FeatureEditToolNone | No tool |
| FeatureEditToolPoint | A client should allow capturing a point geometry. |
| FeatureEditToolLine | A client should allow capturing a polyline geometry. |
| FeatureEditToolPolygon | A client should allow capturing a polygon geometry. |
| FeatureEditToolAutoCompletePolygon | A client should allow capturing a polygon geometry where polygon rings are automatically completed. |
| FeatureEditToolRectangle | A client should allow capturing a polygon geometry with rectangular shape. |
| FeatureEditToolFreehand | A client should allow capturing a freehand geometry. |

Annex A
(normative)

Abstract Test Suite

Conformance class: featureservice

* 1. Test: featureservice/root

|  |  |
| --- | --- |
| Requirements | **featureservice/request, featureservice/parameters, featureservice/valid, featureservice/uniqueLayerId, featureservice/uniqueTableId** |
| Test purpose | Verify that the service root resource supports the request and response requirements. |
| Test method | Set up a test service. Construct valid requests for the Feature Service Root resource.Inspect the responses and validate it against the JSON Schema http://schemas.opengis.net/gsr-fs/1.0/root.json or for exceptions against http://schemas.opengis.net/gsr/1.0/exception.json. Verify that * each layer id is unique within the feature service.
* each table id is unique within the feature service.
 |
| Test type | Capability |

* 1. Test: featureservice/layerOrTable

|  |  |
| --- | --- |
| Requirements | **featureservice/layerOrTableRequest, featureservice/layerOrTableParameters, featureservice/layerOrTableValid** |
| Test purpose | Verify that the layer/table resource supports the request and response requirements. |
| Test method | Construct valid requests for all layer/table resources in the test service.Inspect the responses and validate it against the JSON Schema http://schemas.opengis.net/gsr/1.0/layerOrTable.json or for exceptions against http://schemas.opengis.net/gsr/1.0/exception.json.  |
| Test type | Capability |

* 1. Test: featureservice/feature

|  |  |
| --- | --- |
| Requirements | **featureservice/featureRequest, featureservice/featureParameters, featureservice/featureValid** |
| Test purpose | Verify that the feature resources support the request and response requirements. |
| Test method | Construct valid requests for a representative number of feature resources in the test service.Inspect the responses and validate it against the JSON Schema http://schemas.opengis.net/gsr/1.0/singleObject.json or for exceptions against http://schemas.opengis.net/gsr/1.0/exception.json.  |
| Test type | Capability |

* 1. Test: featureservice/image

|  |  |
| --- | --- |
| Requirements | **featureservice/imageRequest, featureservice/image** |
| Test purpose | Verify that the image resources support the request and response requirements. |
| Test method | Construct valid requests for a representative number of image resources in the test service.Inspect the responses and verify that the image or a HTTP status code of 404 is returned.  |
| Test type | Capability |

Conformance class: query

* 1. Test: query/query

|  |  |
| --- | --- |
| Requirements | **query/request, query/parameters, query/valid, query/tables** |
| Test purpose | Verify that the Query resource supports the request and response requirements. |
| Test method | In the test feature service, construct valid Query requests and vary each parameter. Validate the responses against the relevant JSON Schemas (http://schemas.opengis.net/gsr/1.0/featureSet.json if returnIdsOnly=false, http://schemas.opengis.net/gsr/1.0/featureIdSet.json if returnIdsOnly=true, or http://schemas.opengis.net/gsr/1.0/exception.json in case of an exception).Verify that all parameters related to geometry are ignored when querying tables. |
| Test type | Capability |

Conformance class: query-time

* 1. Test: query-time/query

|  |  |
| --- | --- |
| Requirements | **query-time/timeInfo, query-time/parameters, query-time/valid** |
| Test purpose | Verify that the Query resource supports the time-related request and response requirements. |
| Test method | Inspect the layer/table resources and verify that the timeInfo property is provided. Analyse the timeInfo values and construct valid Query requests and vary each parameter. Validate the responses against the relevant JSON Schemas (http://schemas.opengis.net/gsr/1.0/featureSet.json if returnIdsOnly=false, http://schemas.opengis.net/gsr/1.0/featureIdSet.json if returnIdsOnly=true, or http://schemas.opengis.net/gsr/1.0/exception.json in case of an exception). |
| Test type | Capability |

Conformance class: queryrel

* 1. Test: queryrel/query

|  |  |
| --- | --- |
| Requirements | **queryrel/request, queryrel/parameters, queryrel/valid, queryrel/tables** |
| Test purpose | Verify that the Query Related Records resource supports the request and response requirements. |
| Test method | Inspect the layer/table resource and verify that the relationships property is provided. In the test feature service, construct valid Query Related Record requests and vary each parameter. Validate the responses against the relevant JSON Schemas (http://schemas.opengis.net/gsr/1.0/relatedRecords.json or http://schemas.opengis.net/gsr/1.0/exception.json in case of an exception).Verify that all parameters related to geometry are ignored when querying tables. |
| Test type | Capability |

Conformance class: editing

* 1. Test: editing/basic

|  |  |
| --- | --- |
| Requirements | **editing/capEditing** |
| Test purpose | Verify that the feature service supports feature editing requirements. |
| Test method | Inspect the root resource of the test service and verify that the capabilities property includes the value "Editing".  |
| Test type | Basic |

* 1. Test: editing/add

|  |  |
| --- | --- |
| Requirements | **editing/addRequest, editing/addParameters, editing/addGeometry, editing/addParameterValid, editing/addValid, editing/addErrors, editing/addResponseComplete** |
| Test purpose | Verify that the Add Features resource supports the request and response requirements. |
| Test method | Construct valid requests for a several new features of several layers and tables - taking the requirements on the parameters into account.Inspect the responses and validate it against the JSON Schema http://schemas.opengis.net/gsr-fs/1.0/addResults.json or for exceptions against http://schemas.opengis.net/gsr/1.0/exception.json. Verify for each requested feature insertion that one entry exists in the result and that the property "error" is provided, if and only if the property "success" is false. |
| Test type | Capability |

* 1. Test: editing/update

|  |  |
| --- | --- |
| Requirements | **editing/updateRequest, editing/updateParameters, editing/updateGeometry, editing/updateParameterValid, editing/updateValid, editing/updateErrors, editing/updateResponseComplete, editing/updateObjectId, editing/objectIdErrors** |
| Test purpose | Verify that the Update Features resource supports the request and response requirements. |
| Test method | Construct valid requests for a several existing features of several layers and tables - taking the requirements on the parameters into account.Inspect the responses and validate it against the JSON Schema http://schemas.opengis.net/gsr-fs/1.0/updateResults.json or for exceptions against http://schemas.opengis.net/gsr/1.0/exception.json. Verify for each requested feature update that one entry exists in the result and that the property "error" is provided, if and only if the property "success" is false.Verify that if a feature or record is submitted with an object identifier that does not exist in the layer/table, an error is reported. |
| Test type | Capability |

* 1. Test: editing/delete

|  |  |
| --- | --- |
| Requirements | **editing/deleteRequest, editing/deleteParameters, editing/deleteValid, editing/deleteErrors, editing/deleteResponseComplete** |
| Test purpose | Verify that the Delete Features resource supports the request and response requirements. |
| Test method | Construct valid requests for deleting several existing features of several layers and tables - taking the requirements on the parameters into account.Inspect the responses and validate it against the JSON Schema http://schemas.opengis.net/gsr-fs/1.0/deleteResults.json or for exceptions against http://schemas.opengis.net/gsr/1.0/exception.json. Verify for each requested feature deletion that one entry exists in the result and that the property "error" is provided, if and only if the property "success" is false.Verify that if a feature or record is submitted with an object identifier that does not exist in the layer/table, an error is reported. |
| Test type | Capability |

* 1. Test: editing/applyEdits

|  |  |
| --- | --- |
| Requirements | **editing/applyRequest, editing/applyParameters, editing/applyGeometry, editing/applyParameterValid, editing/applyValid, editing/applyErrors, editing/applyResponseComplete** |
| Test purpose | Verify that the Apply Edits resource supports the request and response requirements. |
| Test method | Construct valid requests for inserting, updating and deleting several features of several layers and tables - taking the requirements on the parameters into account.Inspect the responses and validate it against the JSON Schema http://schemas.opengis.net/gsr-fs/1.0/applyEdits.json or for exceptions against http://schemas.opengis.net/gsr/1.0/exception.json. Verify for each requested feature edit that one entry exists in the result and that the property "error" is provided, if and only if the property "success" is false.Verify that if a feature or record is submitted with an object identifier that does not exist in the layer/table, an error is reported. |
| Test type | Capability |

* 1. Test: attachments/infos

|  |  |
| --- | --- |
| Requirements | **attachments/infoRequest, attachments/infoParameters, attachments/infoValid** |
| Test purpose | Verify that the attachment infos resources support the request and response requirements. |
| Test method | Construct valid requests for a representative number of attachment infos resources in the test service.Inspect the responses and validate it against the JSON Schema http://schemas.opengis.net/gsr/1.0/attachmentInfos.json or for exceptions against http://schemas.opengis.net/gsr/1.0/exception.json.  |
| Test type | Capability |

* 1. Test: attachments/attachment

|  |  |
| --- | --- |
| Requirements | **attachements/request, attachements/valid** |
| Test purpose | Verify that the attachement resources support the request and response requirements. |
| Test method | Construct valid requests for a representative number of attachment resources in the test service, referenced from attachment info resources.Inspect the responses and verify that the image or a HTTP status code of 404 is returned.  |
| Test type | Capability |

* 1. Test: attachments/popup

|  |  |
| --- | --- |
| Requirements | **attachments/popupRequest, attachments/popupParameters, attachments/popupValid** |
| Test purpose | Verify that the HTML popup resources support the request and response requirements. |
| Test method | Construct valid requests for a representative number of HTML popup resources in the test service.Inspect the responses and validate it against the JSON Schema http://schemas.opengis.net/gsr/1.0/htmlPopup.json or for exceptions against http://schemas.opengis.net/gsr/1.0/exception.json.  |
| Test type | Capability |

* 1. Test: attachments/add

|  |  |
| --- | --- |
| Requirements | **attachements/addRequest, attachements/addParameters, attachements/addValid, attachements/addErrors** |
| Test purpose | Verify that the Add Attachment resource supports the request and response requirements. |
| Test method | Construct valid requests for a several new attachements of several features using different content types - taking the requirements on the parameters into account.Inspect the responses and validate it against the JSON Schema http://schemas.opengis.net/gsr-fs/1.0/addAttResults.json or for exceptions against http://schemas.opengis.net/gsr/1.0/exception.json. Verify that the property "error" is provided, if and only if the property "success" is false. |
| Test type | Capability |

* 1. Test: attachments/update

|  |  |
| --- | --- |
| Requirements | **attachements/updateRequest, attachements/updateParameters, attachements/updateValid, attachements/updateErrors** |
| Test purpose | Verify that the Update Attachment resource supports the request and response requirements. |
| Test method | Construct valid requests for updating several attachements of several features using different content types - taking the requirements on the parameters into account.Inspect the responses and validate it against the JSON Schema http://schemas.opengis.net/gsr-fs/1.0/updAttResults.json or for exceptions against http://schemas.opengis.net/gsr/1.0/exception.json. Verify that the property "error" is provided, if and only if the property "success" is false. |
| Test type | Capability |

* 1. Test: attachments/delete

|  |  |
| --- | --- |
| Requirements | **attachementsdeleteeRequest, attachements/deleteParameters, attachements/deleteValid, attachements/deleteErrors** |
| Test purpose | Verify that the Delete Attachment resource supports the request and response requirements. |
| Test method | Construct valid requests for deleting several attachements of several features - taking the requirements on the parameters into account.Inspect the responses and validate it against the JSON Schema http://schemas.opengis.net/gsr-fs/1.0/delAttResults.json or for exceptions against http://schemas.opengis.net/gsr/1.0/exception.json. Verify that the property "error" is provided, if and only if the property "success" is false. |
| Test type | Capability |

Conformance class: templates

* 1. Test: templates/basic

|  |  |
| --- | --- |
| Requirements | **templates/layerOrTable** |
| Test purpose | Verify that the feature service supports feature template requirements. |
| Test method | Inspect the layer/table resource of the test service and verify that the layer/table includes template objects.  |
| Test type | Basic |

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