Open Geospatial Consortium Inc.

Date:   2012-06-22

Reference number of this OGC® project document:    OGC 12-056r1

Version: 0.0.8

Category: OGC® Implementation Specification

Editor:   Clemens Portele, Satish Sankaran

GeoServices REST API — Part 3: Map Service

**Copyright notice**

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

**Warning**

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

Document type:    OGC® Publicly Available Standard

Document subtype:    n/a

Document stage:    Draft

Document language:  English

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

Draft copyright:

Copyright: Esri Inc.

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

Contents

1. Scope 1

2. Conformance 1

3. References 6

4. Terms and Definitions 7

5. Conventions 7

6. Map Service overview 7

7. Map Service Core 9

7.1 Overview 9

7.2 Map Service Root 11

7.2.1 Map Service Root URI 11

7.2.2 Map Service Root resources 12

7.2.3 Example 13

7.3 Export Map 15

7.3.1 Overview 15

7.3.2 Export Map URI 15

7.3.3 Map resources 17

7.3.4 Examples 18

7.4 Layer/Table 19

7.4.1 Overview 19

7.4.2 Layer/Table URI 19

7.4.3 Layer/Table resources 20

7.4.4 Example 20

7.5 Image 22

7.5.1 Overview 22

7.5.2 Image URI 22

7.5.3 Image resources 23

7.5.4 Example 23

7.6 All Layers and Tables 23

7.6.1 Overview 23

7.6.2 All Layers and Tables URI 23

7.6.3 All Layers and Tables resources 24

7.6.4 Example 25

8. Export Map with support for coordinate transformation 27

8.1 Overview 27

8.2 Export Map 28

8.2.1 Dependency 28

8.2.2 Export Map request 28

8.2.3 Map resources 29

8.2.4 Examples 29

9. Export Map with support for time 30

9.1 Overview 30

9.2 Layer/Table resources 30

9.3 Export Map 31

9.3.1 Dependency 31

9.3.2 Export Map URI 31

9.3.3 Map resources 32

10. Export Map with support for feature filtering 32

10.1 Overview 32

10.2 Export Map 33

10.2.1 Dependency 33

10.2.2 Export Map URI 33

10.2.3 Map resources 33

11. Map Tile 34

11.1 Overview 34

11.2 Map Service Root resources 34

11.3 Map Tile URI 35

11.3.1 Map Tile resources 36

12. Identify 36

12.1 Overview 36

12.2 Map Service Root resources 37

12.2.1 Identify URI 37

12.2.2 Identified Features resources 40

12.2.3 Examples 40

13. Identify with support for coordinate transformation 42

13.1 Overview 42

13.2 Identify 42

13.2.1 Dependency 42

13.2.2 Identify URI 42

13.2.3 Identifyied Features resources 43

14. Identify with support for time 44

14.1 Overview 44

14.2 Layer/Table resources 44

14.3 Identify 44

14.3.1 Dependency 44

14.3.2 Identify URI 44

14.3.3 Identified Features resources 45

15. Identify with support for feature filtering 45

15.1 Overview 45

15.2 Identify 45

15.2.1 Dependency 45

15.2.2 Identify URI 45

15.2.3 Identified Features resources 45

16. Data/Feature 45

16.1 Overview 45

16.2 Map Service Root resources 46

16.3 Feature 46

16.3.1 Overview 46

16.3.2 Feature URI 46

16.3.3 Feature resources 47

16.3.4 Example 47

17. Find 48

17.1 Overview 48

17.2 Find 49

17.2.1 Find URI 49

17.2.2 Identified Features resources 50

17.2.3 Example 51

18. Find with support for coordinate transformation 52

18.1 Overview 52

18.2 Find 52

18.2.1 Dependency 52

18.2.2 Find URI 52

18.2.3 Identified Features resources 53

19. Find with support for feature filtering 53

19.1 Overview 53

19.2 Find 53

19.2.1 Dependency 53

19.2.2 Find URI 54

19.2.3 Identified Features resources 54

20. Query 55

20.1 Overview 55

20.2 Query 56

20.2.1 Query URI 56

20.2.2 Feature Set and Feature ID Set resources 62

20.2.3 Examples 62

21. Temporal Query 64

21.1 Overview 64

21.2 Query 64

21.2.1 Dependency 64

21.2.2 Query request 64

21.2.3 Feature Set and Feature ID Set resources 65

22. Query Related Records 65

22.1 Overview 65

22.2 Query Related Records 66

22.2.1 Query Related Records URI 66

22.2.2 Related Records resources 69

22.2.3 Example 69

23. Attachments and HTML Popups 71

23.1 Overview 71

23.2 Attachment Infos 71

23.2.1 Overview 71

23.2.2 Attachment Infos URI 71

23.2.3 Attachment Infos resources 72

23.2.4 Example 72

23.3 Attachment 73

23.3.1 Overview 73

23.3.2 Attachment Infos URI 73

23.3.3 Attachment resources 74

23.4 HTML Popup 74

23.4.1 Overview 74

23.4.2 HTML Popup URI 74

23.4.3 HTML Popup resources 75

23.4.4 Example 75

A.1 Conformance class: mapservice 76

A.1.1 Test: mapservice/root 76

A.1.2 Test: mapservice/layerIds 76

A.1.3 Test: mapservice/exportMap 77

A.1.4 Test: mapservice/layerOrTable 77

A.1.5 Test: mapservice/image 78

A.1.6 Test: mapservice/allLayersAndTables 78

A.2 Conformance class: convert 79

A.2.1 Test: convert/exportMap 79

A.3 Conformance class: time 79

A.3.1 Test: time/layerOrTable 79

A.3.2 Test: time/exportMap 80

A.4 Conformance class: filter 80

A.4.1 Test: filter/exportMap 80

A.5 Conformance class: tile 80

A.5.1 Test: tile/requests 80

A.6 Conformance class: identify 81

A.6.1 Test: identify/requests 81

A.7 Conformance class: identify-convert 81

A.7.1 Test: identify-convert/requests 81

A.8 Conformance class: identify-time 82

A.8.1 Test: identify-time/requests 82

A.9 Conformance class: identify-filter 82

A.9.1 Test: identify-filter/requests 82

A.10 Conformance class: data 83

A.10.1 Test: data/feature 83

A.11 Conformance class: find 83

A.11.1 Test: find/requests 83

A.12 Conformance class: find-convert 83

A.12.1 Test: find-convert/requests 83

A.13 Conformance class: find-filter 84

A.13.1 Test: find-filter/requests 84

A.14 Conformance class: query 84

A.14.1 Test: query/requests 84

A.15 Conformance class: query-time 85

A.15.1 Test: query-time/requests 85

A.16 Conformance class: queryrel 85

A.16.1 Test: queryrel/requests 85

A.17 Conformance class: attachements 86

A.17.1 Test: attachements/attachmentInfo 86

A.17.2 Test: attachements/popup 86

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. 15 Aug 2011
 | 1. DRAFT 1
 | 1. CP, SS
 | 1. All
 | 1. First internal draft
 |
| 1. 29 Aug 2011
 | 1. DRAFT 2
 | 1. CP, SS
 | 1. Several
 | 1. Figure 1 replaced, HTTP verbs added, references and dependencies updated
 |
| 1. 16 Oct 2011
 | 1. DRAFT 3
 | 1. CP, SS
 | 1. Clauses 5-11, Clauses 12+ not yet updated
 | 1. Revised based on discussion in Boulder and aligned with the core document, request/URI requirements formalised
 |
| 1. 25 Nov 2011
 | 1. DRAFT 4
 | 1. CP, SS
 | 1. All, in particular 12+
 |  |
| 1. 23 Jan 2012
 | 1. DRAFT 5
 | 1. CP, SS
 | 1. All
 | 1. Harmonisation with other parts
 |
| 1. 01 May 2012
 | 1. 0.0.6
 | 1. CP
 | 1. All
 | 1. Include changes identified in review process
 |
| 1. 31 May 2012
 | 1. 0.0.7
 | 1. CP
 |  | 1. RFC package for OAB review
 |
| 1. 22 Jun 2012
 | 1. 0.0.8
 | 1. CP
 |  | 1. References updated
 |

Changes to the OGC® Abstract Specification

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

Versioning Rules

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

Foreword

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

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

This document is part 3 of the GeoServices REST API series:

Part 1: Core

Part 2: Catalog

Part 3: Map Service

Part 4: Feature Service

Part 5: Geometry Service

Part 6: Image Service

Part 7: Geoprocessing Service

Part 8: Geocoding Service

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

GeoServices REST API — Part 3: Map Service

# Scope

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

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

# Conformance

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

This Standard establishes 17 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-ms/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.

Some conformance classes are parameterized (parameters are listed in curly brackets). This means that the class’s tests depend on some parameter that must be defined before the tests can be executed. For example, the mapservice conformance class needs to specify the image formats like "png" or "jpeg" for which an implementation claims conformance and which have to tested.

Any implementation claiming conformance with a conformance class shall pass all the tests in the associated abstract test suite. For parameterized conformance classes, any certificate of conformance shall specify all parameter values used to pass the tests.

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

Table 1 – Conformance class summary

|  |  |  |
| --- | --- | --- |
| **mapservice** {imageFormat} | **Title** | Map 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 |
| **convert** | **Title** | Export Map with support for coordinate transformation |
| **Standardization target type** | Web service |
| **Dependencies** | **conf/mapservice** |
| **Requirements** | All requirements in Clause 8 |
| **Conformance tests** | Annex A.2 |
| **time** | **Title** | Export Map with support for time |
| **Standardization target type** | Web service |
| **Dependencies** | **conf/mapservice** |
| **Requirements** | All requirements in Clause 9 |
| **Conformance tests** | Annex A.3 |
| **filter** | **Title** | Export Map with support for feature filtering |
| **Standardization target type** | Web service |
| **Dependencies** | **conf/mapservice** |
| **Requirements** | All requirements in Clause 10 |
| **Conformance tests** | Annex A.4 |
| **tile** | **Title** | Map Tile |
| **Standardization target type** | Web service |
| **Dependencies** | **conf/mapservice** |
| **Requirements** | All requirements in Clause 11 |
| **Conformance tests** | Annex A.5 |
| **identify** | **Title** | Query/Identify |
| **Standardization target type** | Web service |
| **Dependencies** | **conf/mapservice** |
| **Requirements** | All requirements in Clause 12 |
| **Conformance tests** | Annex A.6 |
| **identify-convert** | **Title** | Identify with support for coordinate transformation |
| **Standardization target type** | Web service |
| **Dependencies** | **conf/identify** |
| **Requirements** | All requirements in Clause 13 |
| **Conformance tests** | Annex A.7 |
| **identify-time** | **Title** | Identify with support for time |
| **Standardization target type** | Web service |
| **Dependencies** | **conf/identify** |
| **Requirements** | All requirements in Clause 14 |
| **Conformance tests** | Annex A.8 |
| **identify-filter** | **Title** | Identify with support for feature filtering |
| **Standardization target type** | Web service |
| **Dependencies** | **conf/identify** |
| **Requirements** | All requirements in Clause 15 |
| **Conformance tests** | Annex A.9 |
| **data** | **Title** | Data/Feature |
| **Standardization target type** | Web service |
| **Dependencies** | **conf/mapservice** |
| **Requirements** | All requirements in Clause 16 |
| **Conformance tests** | Annex A.10 |
| **find** | **Title** | Find |
| **Standardization target type** | Web service |
| **Dependencies** | **conf/data** |
| **Requirements** | All requirements in Clause 17 |
| **Conformance tests** | Annex A.11 |
| **find-convert** | **Title** | Find with support for coordinate transformation |
| **Standardization target type** | Web service |
| **Dependencies** | **conf/find** |
| **Requirements** | All requirements in Clause 18 |
| **Conformance tests** | Annex A.12 |
| **find-filter** | **Title** | Find with support for feature filtering |
| **Standardization target type** | Web service |
| **Dependencies** | **conf/find** |
| **Requirements** | All requirements in Clause 19 |
| **Conformance tests** | Annex A.13 |
| **query** | **Title** | Query |
| **Standardization target type** | Web service |
| **Dependencies** | **conf/data** |
| **Requirements** | All requirements in Clause 20 |
| **Conformance tests** | Annex A.14 |
| **querytime** | **Title** | Temporal Query  |
| **Standardization target type** | Web service |
| **Dependencies** | **conf/query** |
| **Requirements** | All requirements in Clause 21 |
| **Conformance tests** | Annex A.15 |
| **queryrel** | **Title** | Query related records |
| **Standardization target type** | Web service |
| **Dependencies** | **conf/data** |
| **Requirements** | All requirements in Clause 22 |
| **Conformance tests** | Annex A.16 |
| **attachements** | **Title** | Attachments and Popups |
| **Standardization target type** | Web service |
| **Dependencies** | **conf/data** |
| **Requirements** | All requirements in Clause 23 |
| **Conformance tests** | Annex A.17 |

Figure 1 – Hierarchy of conformance classes

# References

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

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

# Terms and Definitions

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

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

map

portrayal of geographic information as a digital image file suitable for display on a computer screen

[OGC WMS 1.3, same as ISO 19128:2005]

# Conventions

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

# Map Service overview

Map services offer access to map and layer content. A map service will either fulfill requests with typically precreated tiles from a cache or by dynamically rendering the map each time a request comes in. Map services using a tile cache can significantly improve performance when returning maps, while dynamic map services offer more flexibility.

Map services can be used to expose tabular data, too, whether this is associated with geographic features or not. The Map Service Root resource includes a tables property that contains some basic information about tables. The child layer resource is a Layer/Table resource in that it represents either a layer or a table depending on the ID that was specified.

If the map service supports querying and exporting maps based on time, the Map Service Root resource includes a timeInfo property, which includes information about the map's time extent and the map's native time reference.

The Map Service Root resource provides basic information about the map, including the layers that it contains; whether the map has a tile cache; and the map's spatial reference, initial and full extents, map units, and copyright text. It also provides some metadata associated with the service such as its service description, its author, and keywords. If the map is cached, additional information about its tiling scheme, such as the origin of the cached tiles, the levels of detail, and tile size, is included.

The Map Service Root resource supports several operations:

Export Map: Used to export a dynamically drawn map image.

Identify: Returns information about features in one or more layers at a given location. This location commonly originates from a click of the mouse.

Find: Returns information about features in one or more fields in one or more layers based on a keyword.

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

The following figure provides an overview of the resources in a Map Service. Resources in green color are controller resources (also called "operations") that query the underlying layers/tables and create resources that are not persistently stored on the server and made available with their own URI, but returned in the response from the controller resource. These resources are shown in white color.

Figure 2 – Resource overview

Map services do not expose feature editing capabilities. They provide read-only access to feature and attribute content. Feature services expose editing capabilities.

# Map Service Core

## Overview

An implementation of the GeoServices REST API Map Service Core provides capabilities that are needed by most applications using map services. Additional capabilities, which are often more complex to implement, are specified in additional conformance classes that depend on the core.

Every GeoServices REST API Map Service will provide access to maps as bitmap graphics. A map will show one or more layers. The core conformance class requires only support for basic parameters like the image size, the extent of the map, the image format and transparency.

In addition, the Map Service provides basic information about the service and the layers so that users can access the map resources.

Table 2 – Map Service Core overview

|  |  |  |
| --- | --- | --- |
| **Resource** | **Parameters** | **Resource representation** |
| Map Service Root | f=json | JSON representation validJSON schema elements supported:- serviceDescription- mapName- description- copyright- layers- spatial reference- initialExtent- fullExtent- documentInfo- capabilities (value contains "Map") |
| Export Map | f=jsonf=imagebboxsizedpiformat={imageFormat}layerstransparent | JSON representation validAll JSON schema elements supportedImage has correct format and size |
| Layer/Table | f=json | JSON representation validJSON schema elements supported:- id- name- type (value "Feature Layer")- description- definitionExpression- geometryType- copyrightText- parentLayer- subLayers- minScale- maxScale- defaultVisibility- extent- field information |
| Image | - | - |
| All Layers/Tables | f=json | JSON representation validAll JSON schema elements supported (Layer/Table resources as described in the row above) |

## Map Service Root

### Map Service Root URI

In the following URI templates, these variables are used:

* mapServiceRootURI: the URL of the service

If the Map Service is referenced from a Catalog Service, mapServiceRootURI is the same as

{+catServiceRootURI}/{mapServiceName}/MapServer

where mapServiceName is the name of a map service referenced in the catalogue.

Table 3 – Map Service Root reference

|  |  |
| --- | --- |
| **URI template** | {+mapServiceRootURI}{?f} |
| **HTTP methods** | GET |
| **Parent Resource Type** | - |
| **Child Resource Types**  | Layer/TableAll Layers and TablesMap Tile (only if conformance class "tile" is supported)Export MapIdentify (only if conformance class "identify" is supported)Find (only if conformance class "find" is supported) |

Table 4 – Map Service Root parameters

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

**Request Requirements**

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

mapservice/request |

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

mapservice/parameters |

### Map Service Root resources

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

mapservice/valid |

|  |
| --- |
| * + 1. Each layer or table id in the JSON representation of a Map Service Root resource SHALL be unique within the map service.

mapservice/uniqueLayerOrTableId |

|  |
| --- |
| * + 1. Each parent and sublayer id in the JSON representation of a Map Service Root resource SHALL be the id of a layer in the Map Service Root resource.

mapservice/layersExist |

|  |
| --- |
| * + 1. The units property in the JSON representation of a Map Service Root resource SHALL be one of the entries in the documents http://schemas.opengis.net/gsr/1.0/lengthUnits.json.

mapservice/units |

|  |
| --- |
| * + 1. The supportedImageFormatTypes property in the JSON representation of a Map Service Root resource SHALL be a comma separated list. The following format names are well-known (comparisons SHALL be case insensitive): "PNG": as specified by media type "image/png"; "PNG8": PNG with 8-bit color depth; "PNG24": PNG with 24-bit color depth; "PNG32": PNG with 32-bit color depth; "JPG": as specified by media type "image/jpeg";"PDF": as specified by media type "image/png"; "BMP": file according to the Device Independent Bitmap (DIB) file format as specified by Microsoft; "GIF": as specified by media type "image/gif"; "SVG": as specified by media type "image/svg+xml". Additional values MAY be provided in which case the details of the image format are not specified by this standard.

mapservice/imageFormats |

|  |
| --- |
| * + 1. The supportedImageFormatTypes property in the JSON representation of a Map Service Root resource SHALL include the value "PNG".

mapservice/png |

|  |
| --- |
| * + 1. The documentInfo property in the JSON representation of a Map Service Root resource SHALL be an object that contains only string-valued properties.

mapservice/documentInfo |

|  |
| --- |
| * + 1. The capabilities property in the JSON representation of a Map Service Root resource SHALL be a comma separated list and include the value "Map".

mapservice/capMap |

### Example

URL for the StateCityHighway service on example.com:

http://example.com/rest/services/StateCityHighway/MapServer?f=json

**Request**

GET /rest/services/StateCityHighway/MapServer?f=json HTTP/1.1

Host: example.com

**Response**

HTTP/1.1 200 OK

Content-Type: application/json

Content-Length: nnn

{

"serviceDescription" : "Test Map Service Description",

"mapName" : "Street Map Pro Data",

"description": "Street Map USA",

"copyrightText" : "Esri",

"layers": [

 {"id" : 0, "name" : "Cities", "defaultVisibility" : true, "parentLayerId" : -1, "subLayerIds" : null},

 {"id" : 1, "name" : "States", "defaultVisibility" : true, "parentLayerId" : -1, "subLayerIds" : null},

 {"id" : 2, "name" : "Counties", "defaultVisibility" : false, "parentLayerId" : -1, "subLayerIds" : [3, 4]},

 {"id" : 3, "name" : "Large Counties", "defaultVisibility" : false, "parentLayerId" : 2, "subLayerIds" : null},

 {"id" : 4, "name" : "Small Counties", "defaultVisibility" : false, "parentLayerId" : 2, "subLayerIds" : null}

],

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

"singleFusedMapCache" : true,

"tileInfo": {

 "rows" : 512, "cols" : 512, "dpi" : 96, "format" : "JPEG", "compressionQuality" : 75,

 "origin" : {"x" : -130.0, "y" : 50.0},

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

 "lods": [

 {"level" : 0, "resolution" : 0.3515625, "scale" : 147748799.285417},

 {"level" : 1, "resolution" : 0.17578125, "scale" : 73874399.6427087},

 {"level" : 2, "resolution" : 0.087890625, "scale" : 36937199.8213544},

 {"level" : 3, "resolution" : 0.0439453125, "scale" : 18468599.9106772},

 {"level" : 4, "resolution" : 0.02197265625, "scale" : 9234299.95533859},

 {"level" : 5, "resolution" : 0.010986328125, "scale" : 4617149.97766929},

 {"level" : 6, "resolution" : 0.0054931640625, "scale" : 2308574.98883465},

 {"level" : 7, "resolution" : 0.00274658203125, "scale" : 1154287.49441732},

 {"level" : 8, "resolution" : 0.001373291015625, "scale" : 577143.747208662},

 {"level" : 9, "resolution" : 0.0006866455078125, "scale" : 288571.873604331},

 {"level" : 10, "resolution" : 0.00034332275390625, "scale" : 144285.936802165},

 {"level" : 11, "resolution" : 0.000171661376953125, "scale" : 72142.9684010827},

 {"level" : 12, "resolution" : 8.58306884765629E-05, "scale" : 36071.4842005414},

 {"level" : 13, "resolution" : 4.29153442382814E-05, "scale" : 18035.7421002707},

 {"level" : 14, "resolution" : 2.14576721191407E-05, "scale" : 9017.87105013534},

 {"level" : 15, "resolution" : 1.07288360595703E-05, "scale" : 4508.93552506767}

 ]

},

"initialExtent" : {

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

 "spatialReference" : {"wkid" : 4326}

},

"fullExtent" : {

 "xmin" : -130.0, "ymin" : 24.0, "xmax" : -65.0, "ymax" : 50.0,

 "spatialReference" : {"wkid" : 4326}

},

"units" : "DecimalDegrees",

"supportedImageFormatTypes": "PNG32,PNG24,PNG,JPG,DIB,TIFF,EMF,PS,PDF,GIF,SVG,SVGZ",

"documentInfo": {

 "Title" : "StreetMap USA.mxd",

 "Author" : "Esri Data Team",

 "Comments" : "Esri Data and Maps 2004",

 "Subject" : "Street level data for the US",

 "Category" : "vector",

 "Keywords" : "StreetMap USA"

} ,

"capabilities" : "Map,Query,Data"

}

## Export Map

### Overview

The Export Map operation is performed on a controller resource of the map service. The result of this operation provides a Map resource with information about the exported map image such as its URL, width and height, extent, and scale. The Map resource is not stored on the server, it is returned directly in the response to the request.

Apart from the response format of JSON, users can also request a format of image while performing this operation. When users export with the format of image, the server responds by directly streaming the image bytes to the client. With this approach, no information is associated with the exported map other than the actual image.

The extent displayed in the exported map image will not exactly match the extent sent in the bounding box (bbox) parameter when the aspect ratio of the image size does not match the aspect ratio of the bbox. The aspect ratio is the height divided by the width. In these cases, the extent will resized to prevent map images from appearing stretched. The exported map's extent is sent along with the JSON response and can be used in client-side calculations, so it is important that the client-side code update its extent based on the response.

Users provide arguments to the export operation as query parameters. These parameters include the request extent, size information, layer information, and transparency.

### Export Map URI

In the following URI templates, these variables are used:

* mapServiceURI: URL of a Map Service Root resource without any parameter

Table 5 – Export Map reference

|  |  |
| --- | --- |
| **URI template** | {+mapServiceURI}/export{?f,bbox,size,dpi,format,layers,transparent} |
| **HTTP methods** | GETPOST (application/x-www-form-urlencoded) |
| **Parent Resource** | Map Service |

Table 6 – Export Map parameters

|  |  |
| --- | --- |
| **Parameter** | **Details** |
| f | The response format.  |
| Required | Yes |
| Syntax | "json" / "image" |
| Example | f=json |
| bbox | The extent (bounding box) of the exported image. The bbox is assumed to be in the spatial reference of the map service unless. This behaviour MAY be changed in an extension (see Clause 8).NOTE The bbox coordinates always use a period as the decimal separator even in countries where traditionally a comma is used. |
| Required | Yes |
| Syntax | XMIN "," YMIN "," XMAX "," YMAX  |
| Example | bbox=-104,35.6,-94.32,41 |
| size | The size (width \* height) of the exported image in pixels. |
| Required | No. Default: "400,400" |
| Syntax | WIDTH "," HEIGHT |
| Example | size=600,550 |
| dpi | The device resolution of the exported image (dots per inch). |
| Required | No. Default: "96" |
| Syntax | POSINT |
| Example | dpi=200 |
| format | The format of the exported image. |
| Required | No. Default: "PNG" |
| Syntax | FORMATNAMEThe supported values are listed in the property supportedImageFormatTypes of the parent Map Service Root resource. |
| Example | format=JPEG |
| layers | Determines which layers appear on the exported map. There are four ways to specify which layers are shown:show: Only the layers specified in this list will be exported.hide: All layers except those specified in this list will be exported.include: In addition to the layers exported by default, the layers specified in this list will be exported.exclude: The layers exported by default, excluding those specified in this list, will be exported.The layers are identified by their layer IDs returned by the Map Service Root resource. |
| Required | No. Default: All layers with default visibility set to "true". |
| Syntax | ("show"/"hide"/"include"/"exclude") ":" LAYERID \*("," LAYERID) |
| Example | layers=show:2,4,7 |
| transparent | If true, the image is exported with the background color of the map set as its transparent color. |
| Required | No. Default: "false". |
| Syntax | BOOLEAN |
| Example | transparent=true |

**Request Requirements**

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

mapservice/mapRequest |

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

mapservice/mapParameters |

### Map resources

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

mapservice/mapValid |

|  |
| --- |
| * + 1. The image representation of a Map resource and the image file referenced from the the href property of the JSON representation of a Map resource SHALL have the width, height and dpi as specified by the URI parameters.

mapservice/imageSize |

|  |
| --- |
| * + 1. If aspect ratio is inconsistent between the parameters bbox and size, the extent of the returned map image SHALL be adjusted to have the same aspect ratio as the image. If the Map resource is returned, the bbox SHALL be the adjusted envelope.

mapservice/aspectRatio |

|  |
| --- |
| * + 1. The image SHALL be in the requested format, if the format is listed in the Map Service Root resource as a supported image format type. The following format names are well-known (comparisons SHALL be case insensitive): "PNG": as specified by media type "image/png"; "png8": "png" with 8-bit color depth;"png24": "png" with 24-bit color depth; "png32": "png" with 32-bit color depth;"jpg": as specified by media type "image/jpeg";"pdf": as specified by media type "image/png";"bmp": file according to the Device Independent Bitmap (DIB) file format as specified by Microsoft ;"gif": as specified by media type "image/gif";"svg": as specified by media type "image/svg+xml".

mapservice/imageFormat |

### Examples

Export a map in the JSON representation. Include only the bounding box parameter:

http://example.com/rest/services/StateCityHighway/MapServer/export?f=json&bbox=-127.8,15.4,-63.5,60.5

**Request**

GET /rest/services/StateCityHighway/MapServer/export?f=json&bbox=-127.8,15.4,-63.5,60.5 HTTP/1.1

Host: example.com

**Response**

HTTP/1.1 200 OK

Content-Type: application/json

Content-Length: nnn

{

"href" : "http://example.com/output/map42ef5eae899942a9b564138e184a55c9.png",

"width" : 400,

"height" : 400,

"extent" : {

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

 "spatialReference" : {"wkid" : 4326}

},

"scale" : 2.53E7

}

Export a map and directly request the image file. Include only the bounding box parameter:

http://example.com/rest/services/StateCityHighway/MapServer/export?f=image&bbox=-127.8,15.4,-63.5,60.5

**Request**

GET /rest/services/StateCityHighway/MapServer/export?f=image&bbox=-127.8,15.4,-63.5,60.5 HTTP/1.1

Host: example.com

**Response**

HTTP/1.1 200 OK

Content-Type: image/png

Content-Length: nnn

iVBORw0KGgoAAAANSUhEUgAAADIA…

## Layer/Table

### Overview

The Layer/Table resource represents a single layer or table in a map service. It provides basic information about the layer or table such as its name, type, and fields. For layers, it provides additional information such as the layer's parent and sublayers, minimum and maximum scales, extent, and copyright text. It also provides information regarding the relationship of this layer/table with other layers/tables in the map service.

### Layer/Table URI

In the following URI templates, these variables are used:

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

Table 7 – Layer/Table reference

|  |  |
| --- | --- |
| **URI template**  | {+mapServiceURI}/{layerOrTableId}{?f} |
| **HTTP methods** | GET |
| **Parent Resource Type** | Map Service |
| **Child Resource Types** | Feature (only if conformance class "Data/Feature" is supported) |
| **Child Operations** | Query (only if conformance class "Query" is supported)QueryRelatedRecords (only if conformance class "Query/Relationships" is supported) |

Table 8 – Layer/Table parameters

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

**Request Requirements**

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

mapservice/layerOrTableRequest |

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

mapservice/layerOrTableParameters |

### Layer/Table resources

|  |
| --- |
| * + 1. The JSON representation of a Layer/Table 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.

mapservice/layerOrTableValid |

### Example

URL for layer 2 in the the StateCityHighway service:

http://example.com/rest/services/StateCityHighway/MapServer/2?f=json

**Request**

GET /rest/services/StateCityHighway/MapServer/2?f=json HTTP/1.1

Host: example.com

**Response**

HTTP/1.1 200 OK

Content-Type: application/json

Content-Length: nnn

{

 "id" : 0,

 "name" : "Wells",

 "type" : "Feature Layer",

 "description" : "",

 "definitionExpression" : "",

 "geometryType" : "GeometryPoint",

 "copyrightText" : "",

 "parentLayer" : null,

 "subLayers" : [],

 "minScale" : 0,

 "maxScale" : 0,

 "defaultVisibility" : false,

 "extent" : {

 "xmin" : -102.048629,

 "ymin" : 5.6843418860808E-14,

 "xmax" : 5.6843418860808E-14,

 "ymax" : 40.0020000000001,

 "spatialReference" : {

 "wkid" : 4267

 }

 },

 "hasAttachments" : false,

 "timeInfo" : {

 "startTimeField" : "COMPLETION",

 "endTimeField" : "PLUG\_DATE",

 "trackIdField" : null,

 "timeExtent" : [

 -2556057600000,

 1246060800000

 ], "timeReference" : null,

 "timeInterval" : 3,

 "timeIntervalUnits" : "TimeUnitsYears",

 "exportOptions" : {

 "useTime" : true,

 "timeDataCumulative" : false,

 "timeOffset" : null,

 "timeOffsetUnits" : null

 }

 },

 "drawingInfo" : {"renderer" :

 {

 "type" : "simple",

 "symbol" :

 {

 "type" : "SMS",

 "style" : "SMSCircle",

 "color" : [

 166,

 36,

 0,

 255

 ],

 "size" : 4,

 "angle" : 0,

 "xoffset" : 0,

 "yoffset" : 0,

 "outline" :

 {

 "color" : [

 0,

 0,

 0,

 255

 ],

 "width" : 1

 }

 },

 "label" : "",

 "description" : ""

 },

 "scaleSymbols" : true,

 "transparency" : 0,

 "brightness" : 0,

 "contrast" : 0,

 "labelingInfo" : null},

 "displayField" : "FIELD\_NAME",

 "fields" : [

 {

 "name" : "OBJECTID",

 "type" : "FieldTypeOID",

 "alias" : "OBJECTID"},

 {

 "name" : "Shape",

 "type" : "FieldTypeGeometry",

 "alias" : "Shape"},

 {

 "name" : "KID",

 "type" : "FieldTypeDouble",

 "alias" : "KID"},

 {

 "name" : "STATE\_CODE",

 "type" : "FieldTypeSmallInteger",

 "alias" : "STATE\_CODE"}

 ],

 "relationships" : [

 {

 "id" : 3,

 "name" : "Well 2 Tops",

 "relatedTableId" : 2},

 {

 "id" : 2,

 "name" : "Wells 2 Field",

 "relatedTableId" : 1}

 ],

 "capabilities" : "Map,Query,Data"

}

## 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:

* mapServiceURI: URL of a Map Service Root resource without any parameter
* layerId: id of the layer

Table 9 – Image reference

|  |  |
| --- | --- |
| **URI template**  | {+mapServiceURI}/{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.

mapservice/imageRequest |

### Image resources

**Response Requirements**

|  |
| --- |
| * + 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.

mapservice/image |

### Example

URL of an image:

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

**Request**

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

Host: example.com

**Response**

HTTP/1.1 200 OK

Content-Type: image/png

Content-Length: nnn

iVBORw0KGgoAAAANSUhEUgAAADIA…

## All Layers and Tables

### Overview

The All Layers and Tables resource represents all the layers and stand-alone tables under a map service. It provides basic information about the layers and tables such as their names, types, parent and sublayers, fields, minimum and maximum scales, extents, and copyright text.

### All Layers and Tables URI

In the following URI templates, these variables are used:

* mapServiceURI: URL of a Map Service Root resource without any parameter

Table 10 – All Layers and Tables reference

|  |  |
| --- | --- |
| **URI template** | {+mapServiceURI}/layers{?f} |
| **HTTP methods** | GET |
| **Parent Resource Type** | Map Service |

Table 11 – All Layers and Tables parameters

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

**Request Requirements**

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

mapservice/allLayersAndTablesRequest |

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

mapservice/allLayersAndTablesParameters |

### All Layers and Tables resources

|  |
| --- |
| * + 1. The JSON representation of an All Layers and Tables resource SHALL validate against the JSON Schema **http://schemas.opengis.net/gsr-ms/1.0/allLayersAndTables.json** or in case of an exception against JSON Schema http://schemas.opengis.net/gsr/1.0/exception.json.

mapservice/allLayersAndTablesValid |

|  |
| --- |
| * + 1. Each layer or table in the JSON representation of an All Layers and Tables resource SHALL be available as a Layer/Table resource.

mapservice/allLayersAndTablesExist |

|  |
| --- |
| * + 1. The Layer/Table representations in the "layers" or "tables" property SHALL be layers or tables respectively.

mapservice/layersTables |

### Example

URL for the layers on the StateCityHighway service:

http://example.com/rest/services/StateCityHighway/MapServer/layers?f=json

**Request**

GET /rest/services/StateCityHighway/MapServer/layers?f=json HTTP/1.1

Host: example.com

**Response**

HTTP/1.1 200 OK

Content-Type: application/json

Content-Length: nnn

{

 "layers" : [

 {

 "id" : 0,

 "name" : "Daily fish passage",

 "type" : "Feature Layer",

 "description" : "",

 "definitionExpression" : "",

 "geometryType" : "GeometryPoint",

 "copyrightText" : "",

 "parentLayer" : null,

 "subLayers" : [],

 "minScale" : 0,

 "maxScale" : 0,

 "defaultVisibility" : true,

 "extent" : {

 "xmin" : -13649384.2021,

 "ymin" : 5677265.4122,

 "xmax" : -13072321.1393,

 "ymax" : 6098060.9347,

 "spatialReference" : {

 "wkid" : 3857

 }

 },

 "hasAttachments" : false,

 "htmlPopupType" : "ServerHTMLPopupTypeAsHTMLText",

 "drawingInfo" : {"renderer" :

 {

 "type" : "simple",

 "symbol" :

 {

 "type" : "SMS",

 "style" : "SMSCircle",

 "color" : [

 0,

 153,

 56,

 255

 ],

 "size" : 4,

 "angle" : 0,

 "xoffset" : 0,

 "yoffset" : 0,

 "outline" :

 {

 "color" : [

 0,

 0,

 0,

 255

 ],

 "width" : 1

 }

 },

 "label" : "",

 "description" : ""

 },

 "transparency" : 0,

 "labelingInfo" : null},

 "displayField" : "DAM",

 "fields" : [

 {

 "name" : "OBJECTID",

 "type" : "FieldTypeOID",

 "alias" : "OBJECTID"},

 {

 "name" : "SHAPE",

 "type" : "FieldTypeGeometry",

 "alias" : "SHAPE"},

 {

 "name" : "DATE",

 "type" : "FieldTypeDate",

 "alias" : "DAY",

 "length" : 8},

 {

 "name" : "STEELHEAD",

 "type" : "FieldTypeInteger",

 "alias" : "STEELHEAD"},

 {

 "name" : "SOCKEYE",

 "type" : "FieldTypeInteger",

 "alias" : "SOCKEYE"}

 ],

 "typeIdField" : null,

 "types" : null,

 "relationships" : [],

 "capabilities" : "Map,Query,Data"

 }

 ],

 "tables" : [

 {

 "id" : 1,

 "name" : "dam\_locations",

 "type" : "Table",

 "description" : null,

 "definitionExpression" : "",

 "hasAttachments" : false,

 "htmlPopupType" : "ServerHTMLPopupTypeNone",

 "displayField" : "NAME",

 "fields" : [

 {

 "name" : "OBJECTID",

 "type" : "FieldTypeInteger",

 "alias" : "OBJECTID"},

 {

 "name" : "NAME",

 "type" : "FieldTypeString",

 "alias" : "NAME",

 "length" : 50},

 {

 "name" : "X",

 "type" : "FieldTypeDouble",

 "alias" : "X"},

 {

 "name" : "Y",

 "type" : "FieldTypeDouble",

 "alias" : "Y"}

 ],

 "typeIdField" : null,

 "types" : null,

 "relationships" : [],

 "capabilities" : "Map,Query,Data"

 }

 ]

}

# Export Map with support for coordinate transformation

## Overview

This conformance class adds support for transformations between coordinate reference systems by adding parameters to support the use of a different coordinate reference system for both the bounding box and the returned image.

Table 12 – Export Map with support for coordinate transformation overview

|  |  |  |
| --- | --- | --- |
| **Resource** | **Parameters** | **Resource representation** |
| Map | imageSRbboxSR | - |

## Export Map

### Dependency

This conformance class extends the requirements for the Export Map operation as specified in 7.3.

### Export Map request

Table 13 – Additonal Export Map parameters

|  |  |
| --- | --- |
| **Parameter** | **Details** |
| imageSR | The spatial reference of the exported image. The spatial reference is specified as either a well-known ID (WKID) or a spatial reference JSON object. See the GeoServices REST API Core standard for the requirements related to spatial references. |
| Required | No. Default: The image is exported in the spatial reference of the map. |
| Syntax | POSINT / JSON |
| Example | imageSR=4326 |
| bboxSR | The spatial reference of the coordinates in the bbox parameter.The spatial reference is specified as either a well-known ID (WKID) or a spatial reference JSON object. See the GeoServices REST API Core standard for the requirements related to spatial references. |
| Required | No. Default: The bbox is in the spatial reference of the map. |
| Syntax | POSINT / JSON |
| Example | bboxSR=4326 |

**Request Requirements**

|  |
| --- |
| * + 1. The Export Map resource SHALL support all parameters and values specified in Table 13.

convert/parameters |

|  |
| --- |
| * + 1. The value of a imageSR or bboxSR paramater, if provided, SHALL either be a well-known ID (WKID) or a spatial reference JSON object.

convert/validSR |

### Map resources

|  |
| --- |
| * + 1. The extent property in the JSON representation of the Map resource SHALL use the coordinate reference system specified in the imageSR parameter.

convert/imageSR |

### Examples

Export a map and change the imageSR value to WKID 102004 (USA\_Contiguous\_Lambert\_Conformal\_Conic projection):

http://example.com/rest/services/StateCityHighway/MapServer/export?bbox=-127.8,15.4,-63.5,60.5&imageSR=102004&f=json

**Request**

GET /rest/services/StateCityHighway/MapServer/export?bbox=-127.8,15.4,-63.5,60.5&imageSR=102004&f=json HTTP/1.1

Host: example.com

**Response**

HTTP/1.1 200 OK

Content-Type: application/json

Content-Length: nnn

{

 "href" : "http://example.com/output/map42ef5eae899942a9b564138e184a55c9.png",

 "width" : 400,

 "height" : 400,

 "extent" : {

 "xmin" : -3600302.08028698,

 "ymin" : -3583476.08362469,

 "xmax" : 3676184.81539067,

 "ymax" : 3693010.81205295,

 "spatialReference" : {

 "wkid" : 102004

 }

 },

 "scale" : 68754069.3798789

}

Export a map and change the imageSR value to WKID 102004 (USA\_Contiguous\_Lambert\_Conformal\_Conic projection), image size to a width and height of 800 x 600, format to GIF, and transparent to false:

http://example.com/rest/services/StateCityHighway/MapServer/export?bbox=-115.8,30.4,-85.5,50.5&size=800,600&imageSR=102004& transparent=false&dpi=&f=image&format=gif

**Request**

GET /rest/services/StateCityHighway/MapServer/export?bbox=-115.8,30.4,-85.5,50.5&size=800,600&imageSR=102004& transparent=false&dpi=&f=image&format=gif HTTP/1.1

Host: example.com

**Response**

HTTP/1.1 200 OK

Content-Type: image/gif

Content-Length: nnn

R0lGODdhMAAwAPAAAAAAAP///ywAAAAAM…

# Export Map with support for time

## Overview

This conformance class adds support for temporal aspects by adding additional parameters.

For time-aware map services, the time parameter can be used to specify the time instant or the time extent for which to export the map. Users can also control time-based behavior on a per-layer basis by using the layerTimeOptions parameter.

Table 14 – Export Map with support for time

|  |  |  |
| --- | --- | --- |
| **Resource** | **Parameters** | **Resource representation** |
| Map | timetimeLayerOptions | - |

## Layer/Table resources

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

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",

 "exportOptions" : {

 "useTime" : true,

 "timeDataCumulative" : false, "timeOffset" : null, "timeOffsetUnits" : null

 }

 },

...

## Export Map

### Dependency

This conformance class extends the requirements for the Export Map operation as specified in 7.3.

### Export Map URI

Table 15 – Additonal Export Map parameters

|  |  |
| --- | --- |
| **Parameter** | **Details** |
| time | The time instant or the time extent of the exported map image. 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 generation of the map. |
| 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) |
| layerTimeOptions | The time options per layer. Users can control whether the layer will use the time extent specified by the time parameter or not, whether to draw the layer features cumulatively or not, and the time offsets for the layer. |
| Required | No. Default: Time is not considered in the generation of the map. |
| Syntax | JSONThe parameter is a JSON object that validates against the JSON Schema **http://schemas.opengis.net/gsr-ms/1.0/layerTimeOptions.json**. |
| Example | layerTimeOptions={ "0" : { "useTime" : true, "timeDataCumulative" : false, "timeOffset" : 1, "timeOffsetUnits" : "TimeUnitsYears" }, "3" : { "useTime" : false }} |

**Request Requirements**

|  |
| --- |
| * + 1. The Export Map resource SHALL support all parameters and values specified in Table 15.

time/parameters |

### Map resources

This conformance class does not add any requirements on the JSON representation of the Map resources and bitmap images.

# Export Map with support for feature filtering

## Overview

This conformance class adds support for filtering the displayed features in the exported map by specifying definition expressions per layer by adding an additional parameter.

Table 16 – Export Map with support for feature filtering

|  |  |  |
| --- | --- | --- |
| **Resource** | **Parameters** | **Resource representation** |
| Map | layerDefs | - |

## Export Map

### Dependency

This conformance class extends the requirements for the Export Map operation as specified in 7.3.

### Export Map URI

Table 17 – Additonal Export Map parameters

|  |  |
| --- | --- |
| **Parameter** | **Details** |
| layerDefs | Allows you to filter the features of individual layers in the exported map by specifying definition expressions for those layers.The filter expressions for each layer are applied to select the relevant features of the layer shown in the map. |
| Required | No. Default: No filter is considered in the generation of the map. |
| Syntax | (NUMID ":" EXPR \*(";" NUMID ":" EXPR)) / JSONIf the parameter matches the JSON rule, it is a JSON object that validates against the JSON Schema **http://schemas.opengis.net/gsr-ms/1.0/layerTimeOptions.json**. |
| Example | 0:POP2000 > 1000000;5:AREA > 100000 (text representation){"0":"POP2000 > 1000000","5":"AREA > 100000"} (JSON representation) |

**Request Requirements**

|  |
| --- |
| * + 1. The Export Map resource SHALL support all parameters and values specified in Table 17.

filter/parameters |

### Map resources

This conformance class does not add any requirements on the JSON representation of the Map resources.

# Map Tile

## Overview

For maps containing a server-side cache of tiles, this resource represents a single tile. The image bytes for the tile at the specified level, row, and column are directly streamed to the client. If the tile is not found, an HTTP status code of 404 (not found) is returned.

Table 18 – Map Tile overview

|  |  |  |
| --- | --- | --- |
| **Resource** | **URI parameters** | **Resource representation** |
| Map Tile | - | Image of correct size and format |
| Map Service Root | - | JSON schema elements supported:- singleFuesMapCache- tileInfo |

## Map Service Root resources

|  |
| --- |
| * + 1. If the parent Map Service Root resource includes the singleFusedMapCache property with a value of "true" and the tileInfo property, then the map service SHALL provide Map Tile resources in accordance with the tileInfo information.

tile/availability |

The singleFusedMapCache property specifies whether the map is cached. A single fused cache contains image tiles that are created by grouping all the layers together at each scale, or level of detail.

The resolution at each zoom level is the units in the given spatial reference per pixel. The scale at each zoom level is derived from the resolution and the dpi value.

**Example**

...

"singleFusedMapCache" : true,

"tileInfo": {

 "rows" : 512, "cols" : 512, "dpi" : 96, "format" : "JPEG", "compressionQuality" : 75,

 "origin" : {"x" : -130.0, "y" : 50.0},

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

 "lods": [

 {"level" : 0, "resolution" : 0.3515625, "scale" : 147748799.285417},

 {"level" : 1, "resolution" : 0.17578125, "scale" : 73874399.6427087},

 {"level" : 2, "resolution" : 0.087890625, "scale" : 36937199.8213544},

 {"level" : 3, "resolution" : 0.0439453125, "scale" : 18468599.9106772},

 {"level" : 4, "resolution" : 0.02197265625, "scale" : 9234299.95533859},

 {"level" : 5, "resolution" : 0.010986328125, "scale" : 4617149.97766929},

 {"level" : 6, "resolution" : 0.0054931640625, "scale" : 2308574.98883465},

 {"level" : 7, "resolution" : 0.00274658203125, "scale" : 1154287.49441732},

 {"level" : 8, "resolution" : 0.001373291015625, "scale" : 577143.747208662},

 {"level" : 9, "resolution" : 0.0006866455078125, "scale" : 288571.873604331},

 {"level" : 10, "resolution" : 0.00034332275390625, "scale" : 144285.936802165},

 {"level" : 11, "resolution" : 0.000171661376953125, "scale" : 72142.9684010827},

 {"level" : 12, "resolution" : 8.58306884765629E-05, "scale" : 36071.4842005414},

 {"level" : 13, "resolution" : 4.29153442382814E-05, "scale" : 18035.7421002707},

 {"level" : 14, "resolution" : 2.14576721191407E-05, "scale" : 9017.87105013534},

 {"level" : 15, "resolution" : 1.07288360595703E-05, "scale" : 4508.93552506767}

 ]

},

...

## Map Tile URI

In the following URI templates, these variables are used:

* mapServiceURI: URL of a Map Service Root resource without any parameter
* level: level of detail
* row: row of the tile from tile level {level}
* column: column of the tile from tile level {level}

Table 19 – Map Tile reference

|  |  |
| --- | --- |
| **URI template** | {+mapServiceURI}/tile/{level}/{row}/{column} |
| **HTTP methods** | GET |
| **Parent Resource Type** | Map Service Root |

**Example**

Request a map tile for the Chicago service on example.com:

http://example.com/rest/services/Chicago/MapServer/tile/5/7/10

**Request requirements**

|  |
| --- |
| * + 1. The Map Tile resource SHALL accept requests that conform to the URI template in Table 19 and use any HTTP method identified in the same table.

tile/request |

### Map Tile resources

|  |
| --- |
| * + 1. The image representation of a Map Tile resource SHALL have the width, height and dpi as specified by the tileInfo/cols, tileInfo/rows and tileInfo/Dpi properties of the Map Service Root resource.

tile/imageSize |

|  |
| --- |
| * + 1. The image SHALL be in the format specified by the tileInfo/format property of the Map Service Root resource. The following format names are well-known (comparisons SHALL be case insensitive): "PNG": as specified by media type "image/png"; "PNG8": PNG with 8-bit color depth; "PNG24": PNG with 24-bit color depth; "PNG32": PNG with 32-bit color depth; "JPG": as specified by media type "image/jpeg";"PDF": as specified by media type "image/png"; "BMP": file according to the Device Independent Bitmap (DIB) file format as specified by Microsoft; "GIF": as specified by media type "image/gif"; "SVG": as specified by media type "image/svg+xml".

tile/imageFormat |

# Identify

## Overview

The Identify operation is performed on a controller resource of the map service to discover features at a geographic location. Each identified result includes its name, layer ID, layer name, geometry, geometry type, and other attributes of that result as name-value pairs.

The identified features are not stored on the server, but returned directly in the response to the request.

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

Table 20 – Identify overview

|  |  |  |
| --- | --- | --- |
| **Resource** | **URI parameters** | **Resource representation** |
| Identify | f=jsongeometrygeometryTypelayersimageDisplayreturnGeometrymaxAllowableOffset | JSON representation validAll JSON schema elements supported |
| Map Service Root | - | JSON schema elements supported:- capabilities (value contains "Query") |

## Map Service Root resources

|  |
| --- |
| * + 1. If the parent Map Service Root resource has a capabilities property that includes the value "Query", then the map service SHALL provide the Identify resource.

identify/availability |

**Example**

...

 "capabilities" : "Map,Query"

...

### Identify URI

In the following URI templates, these variables are used:

* mapServiceURI: URL of a Map Service Root resource without any parameter

Table 21 – Identify reference

|  |  |
| --- | --- |
| **URI template** | {+mapServiceURI}/identify(?f, geometry,geometryType,layers,imageDisplay, returnGeometry,maxAllowableOffset} |
| **HTTP methods** | GETPOST (application/x-www-form-urlencoded) |
| **Parent Resource Type** | Map Service Root |

Table 22 – Identify parameters

|  |  |
| --- | --- |
| **Parameter** | **Details** |
| f | The response format.  |
| Required | Yes |
| Syntax | "json" / "image" |
| Example | f=json |
| geometry | The geometry to identify on. 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 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 | Yes |
| 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 |
| layers | The layers to perform the Identify operation on. There are three ways to specify which layers to identify on:top: Only the topmost layer at the specified locationvisible: All visible layers at the specified locationall: All layers at the specified locationUsers may specify the layer options as mentioned above either by themselves or in conjunction with a list of layer IDs. When both the layer option and the layer IDs are specified, the server treats it as a Boolean AND operator. For example, if the parameter is specified as layers=visible:2,5, only layers with IDs 2 and 5, if visible, are identified.The layers are identified by their layer IDs returned by the Map Service Root resource. |
| Required | No. Default: "top", i.e. the topmost layer  |
| Syntax | ("top"/"visible"/"all") [":" LAYERID \*("," LAYERID)] |
| Example | layers=visible:2,5 |
| imageDisplay | The screen image display parameters (width, height, and dpi) of the map currently being viewed.The mapExtent and the imageDisplay parameters are used by the server to determine the layers visible in the current extent. They are also used to calculate the distance on the map to search based on the tolerance in screen pixels. |
| Required | Yes |
| Syntax | POSINT "," POSINT "," POSINTThe values are: width, height, and dpi. |
| Example | imageDisplay=600,550,96 |
| returnGeometry | If true, the result set includes the geometries associated with each result.  |
| Required | No. Default: "true". |
| Syntax | BOOLEAN |
| Example | returnGeometry=false |

**Request Requirements**

|  |
| --- |
| * + 1. The Identify resource SHALL accept requests that conform to the URI template in Table 21and use any HTTP method identified in the same table.

identify/request |

|  |
| --- |
| * + 1. The Identify resource SHALL support all parameters and values specified in Table 22.

identify/parameters |

### Identified Features resources

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

identify/valid |

### Examples

Identify and include geometry using simple point syntax, and image display. Default values for return format, geometry type, spatial reference, layers, and return geometry are used:

http://example.com/rest/services/StateCityHighway/MapServer/identify?geometryType=GeometryPoint&geometry=-120,40&imageDisplay=400,300,96&f=json

**Request**

GET /rest/services/StateCityHighway/MapServer/identify?geometryType=GeometryPoint&geometry=-120,40&imageDisplay=400,300,96&f=json HTTP/1.1

Host: example.com

**Response**

HTTP/1.1 200 OK

Content-Type: application/json

Content-Length: nnn

{

"results" : [

 {

 "layerId" : 3,

 "layerName" : "Cities",

 "value" : "Joe City",

 "displayFieldName" : "City Name",

 "attributes" : {

 "City Name" : "Joe City",

 "CLASS" : "city",

 "ST" : "CA"

 },

 "geometryType" : "GeometryPoint",

 "geometry" : { "x" : -118.375, "y" : 34.086, "spatialReference" : {"wkid" : 4326} }

 },

 {

 "layerId" : 59,

 "layerName" : "Parcel",

 "value" : "Parcel 649",

 "displayFieldName" : "NAME",

 "attributes" : {

 "NAME" : "Parcel 649",

 "SUB\_REGION" : "Pacific",

 "STATE\_ABBR" : "CA"

 },

 "geometryType" : "GeometryPolygon",

 "geometry" : { "spatialReference" : {"wkid" : 4326}, "rings" : [[[-118.35,32.81],[-118.42.806],[-118.511,32.892],[-118.35,32.81]]]}

 }

]

}

Identify and include geometry using a JSON structure. The response is in JSON format:

http://example.com/rest/services/StateCityHighway/MapServer/identify?geometryType=GeometryPoint&geometry={“x”:-120,”y”:40}& imageDisplay=400,300,96&f=json

**Request**

GET /rest/services/StateCityHighway/MapServer/identify?geometryType=GeometryPoint&geometry={“x”:-120,”y”:40}&imageDisplay=400,300,96&f=json HTTP/1.1

Host: example.com

**Response**

 See the first example above

Identify, specifying a particular layer. In this example, only layer 2 is desired. Since this is not the top layer, the syntax layer=all:2 is used:

http://example.com/rest/services/StateCityHighway/MapServer/identify?geometryType=GeometryPoint&geometry={“x”: -120,”y”:40}&layers=all:2&

imageDisplay=400,300,96&returnGeometry=true&f=json

**Request**

GET /rest/services/StateCityHighway/MapServer/identify?geometryType=GeometryPoint&geometry={“x”:-120,”y”:40}&layers=all:2&imageDisplay=400,300,96&

returnGeometry=true&f=json HTTP/1.1

Host: example.com

**Response**

 See the first example above

# Identify with support for coordinate transformation

## Overview

This conformance class adds support for transformations between coordinate reference systems by adding a parameters to support the use of a different coordinate reference system for both the input and output geometries.

Table 23 – Identify with support for coordinate transformation overview

|  |  |  |
| --- | --- | --- |
| **Resource** | **URI parameters** | **Resource representation** |
| Identified Features | srmaxAllowableOffset | - |

## Identify

### Dependency

This conformance class extends the requirements for the Identify operation as specified in Clause 12.

### Identify URI

Table 24 – Additonal Identify parameters

|  |  |
| --- | --- |
| **Parameter** | **Details** |
| sr | The spatial reference of the output geometries. 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 geometries are assumed to be in the spatial reference of the map, and the output geometries are also in the spatial reference of the map. |
| Syntax | POSINT / JSON |
| Example | sr=4326 |
| maxAllowableOffset | Specifies the maximum allowable offset to be used for generalizing geometries returned by the Find operation.The maxAllowableOffset value is in the units of the spatial reference. If a value for sr is not specified, maxAllowableOffset is assumed to be in the units of the spatial reference of the map. |
| Required | No. Default: no generalization |
| Syntax | NUMBER |
| Example | maxAllowableOffset=2 |

**Request Requirements**

|  |
| --- |
| * + 1. An Identify resource SHALL support all parameters and values specified in Table 24.

identify-convert/parameters |

|  |
| --- |
| * + 1. The value of a imageSR or bboxSR paramater, if provided, SHALL either be a well-known ID (WKID) or a spatial reference JSON object.

identify-convert/validSR |

### Identifyied Features resources

|  |
| --- |
| * + 1. The geometries returned in the JSON representation of the Identified Features resource SHALL use the coordinate reference system specified in the sr parameter.

identify-convert/sr |

# Identify with support for time

## Overview

This conformance class adds support for temporal aspects by adding additional parameters.

For time-aware map services, the time parameter can be used to specify the time instant or the time extent for which to identify features. Users can also control time-based behavior on a per-layer basis by using the layerTimeOptions parameter.

Table 25 – Identify with support for time

|  |  |  |
| --- | --- | --- |
| **Resource** | **URI parameters** | **Resource representation** |
| Identified Features | timetimeLayerOptions | - |

## Layer/Table resources

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

identify-time/timeInfo |

NOTE 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.

## Identify

### Dependency

This conformance class extends the requirements for the Identify operation as specified in Clause 12.

### Identify URI

The additional parameters in the Identify operation are the same as specified in Table 15.

**Request Requirements**

|  |
| --- |
| * + 1. The Identify controller resource SHALL support all parameters and values specified in Table 25.

identfy-time/parameters |

### Identified Features resources

This conformance class does not add any requirements on the JSON representation of the Identified Features resources.

# Identify with support for feature filtering

## Overview

This conformance class adds support for filtering the identified features by specifying definition expressions per layer by adding an additional parameter.

Table 26 – Identify with support for feature filtering

|  |  |  |
| --- | --- | --- |
| **Resource** | **URI parameters** | **Resource representation** |
| Identify | layerDefs | - |

## Identify

### Dependency

This conformance class extends the requirements for the Identify operation as specified in Clause 12.

### Identify URI

The additional parameters in the Identify operation are the same as specified in Table 17.

**Request Requirements**

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

identify-filter/parameters |

### Identified Features resources

This conformance class does not add any requirements on the JSON representation of the Map resources and bitmap images.

# Data/Feature

## Overview

This conformance class provides read access to individual features and records.

Table 27 – Data/Feature overview

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

## Map Service Root resources

|  |
| --- |
| * + 1. If the parent Map Service Root resource has a capabilities property that includes the value "Data", then the map service SHALL provide the Feature resources.

data/availability |

**Example**

...

 "capabilities" : "Map,Query,Data"

...

## Feature

### Overview

The Feature resource represents a single feature in a layer.

### Feature URI

In the following URI templates, these variables are used:

* mapServiceURI: URL of a Map 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 – Feature reference

|  |  |
| --- | --- |
| **URI template**  | {+mapServiceURI}/{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) |

Table 29 – Feature parameters

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

**Request Requirements**

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

data/request |

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

data/parameters |

### 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.

data/featureValid |

### Example

URL for feature 1 in layer 1 of the Watershed service:

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

**Request**

GET /rest/services/Watershed/MapServer/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

 }

 }

}

# Find

## Overview

The Find operation is performed on a controller resource to search the attributes of features. Each result includes its value, feature ID, field name, layer ID, layer name, geometry, geometry type, and attributes in the form of name-value pairs.

The Find operation supports several query parameters as defined in the parameters table below.

This conformance class requires only support for basic parameters like search text. Additional capabilities, which are often more complex to implement, are specified in additional conformance classes.

Table 30 – Find overview

|  |  |  |
| --- | --- | --- |
| **Resource** | **Parameters** | **Resource representation** |
| Find | f=jsonsearchTextcontainssearchFieldslayersreturnGeometry | JSON representation validAll JSON schema elements supported |

## Find

### Find URI

In the following URI templates, these variables are used:

* mapServiceURI: URL of a Map Service Root resource without any parameter

Table 31 – Find reference

|  |  |
| --- | --- |
| **URI template** | {+mapServiceURI}/find{?f, searchText,contains,searchFields, layers,returnGeometry} |
| **HTTP methods** | GETPOST (application/x-www-form-urlencoded) |
| **Parent Resource** | Map Service |

Table 32 – Find parameters

|  |  |
| --- | --- |
| **Parameter** | **Details** |
| f | The response format.  |
| Required | Yes |
| Syntax | "json" / "image" |
| Example | f=json |
| searchText | The search string. This is the text that is searched across the layers and the fields that the user specifies. |
| Required | Yes |
| Syntax | STRING  |
| Example | searchText=Los |
| contains | If false, the operation searches for an exact match of the searchText string. An exact match is case sensitive. Otherwise, it searches for a value that contains the searchText string provided; this search is not case sensitive. |
| Required | No. Default: "true" |
| Syntax | BOOLEAN |
| Example | contains=false |
| searchFields | The names of the fields to search. The fields are specified as a comma-separated list of field names returned by the Layer resource. |
| Required | No. Default: all fields are searched. |
| Syntax | FIELD \*("," FIELD) |
| Example | searchFields=AREANAME,SUB\_REGION |
| layers | The layers to perform the Find operation on. The layers are specified as a comma-separated list of layer IDs returned by the Map Service Root resource. |
| Required | Yes |
| Syntax | LAYERID \*("," LAYERID) |
| Example | layers=2,4,7 |
| returnGeometry | If true, the result set includes the geometries associated with each result.  |
| Required | No. Default: "true". |
| Syntax | BOOLEAN |
| Example | returnGeometry=false |

**Request Requirements**

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

find/request |

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

find/parameters |

### Identified Features resources

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

find/valid |

### Example

Find operation that includes search text and a layer:

http://example.com/rest/services/StatesCitiesRivers/MapServer/find?searchText=island&contains=true&layers=0,2&returnGeometry=true

**Request**

GET /rest/services/StatesCitiesRivers/MapServer/find?searchText=island&contains=true&layers=0,2&returnGeometry=true HTTP/1.1

Host: example.com

**Response**

HTTP/1.1 200 OK

Content-Type: application/json

Content-Length: nnn

{

"results" : [

 { "layerId" : 3,

 "layerName" : "Cities",

 "displayFieldName" : "City Name"

 "foundFieldName" : "City Name",

 "value" : "Joe City",

 "attributes" : {

 "City Name" : "Joe City",

 "CLASS" : "city",

 "ST" : "CA"

 },

 "geometryType" : "GeometryPoint",

 "geometry" : { "x" : -118.375, "y" : 34.086, "spatialReference" : {"wkid" : 4326} }

 },

 {

 "layerId" : 59,

 "layerName" : "Parcel",

 "displayFieldName" : "NAME"

 "foundFieldName" : "NAME",

 "value" : "Joe's Parcel",

 "attributes" : {

 "NAME" : "Parcel 649",

 "SUB\_REGION" : "Pacific",

 "STATE\_ABBR" : "CA"

 },

 "geometryType" : "GeometryPolygon",

 "geometry" : { "spatialReference" : {"wkid" : 4326}, "rings" : [[[-118.35,32.81],[-118.42.806],[-118.511,32.892],[-118.35,32.81]]]}

 }

]

}

# Find with support for coordinate transformation

## Overview

This conformance class adds support for transformations between coordinate reference systems by adding parameters to support the use of a different coordinate reference system for both the bounding box and the returned image.

Table 33 – Export Map with support for coordinate transformation overview

|  |  |  |
| --- | --- | --- |
| **Resource** | **Parameters** | **Resource representation** |
| Find | srmaxAllowedAoffset | - |

## Find

### Dependency

This conformance class extends the requirements for the Find operation as specified in Clause 17.

### Find URI

Table 34 – Additonal Find parameters

|  |  |
| --- | --- |
| **Parameter** | **Details** |
| sr | The spatial reference of the output geometries. 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 geometries is exported in the spatial reference of the map. |
| Syntax | POSINT / JSON |
| Example | sr=4326 |
| maxAllowableOffset | Specifies the maximum allowable offset to be used for generalizing geometries returned by the Find operation.The maxAllowableOffset value is in the units of the spatial reference. If a value for sr is not specified, maxAllowableOffset is assumed to be in the units of the spatial reference of the map. |
| Required | No. Default: no generalization |
| Syntax | NUMBER |
| Example | maxAllowableOffset=2 |

**Request Requirements**

|  |
| --- |
| * + 1. The Find controller resource SHALL support all parameters and values specified in Table 34.

find-convert/parameters |

### Identified Features resources

|  |
| --- |
| * + 1. The extent property in the JSON representation of a response to a request on an Identify resource SHALL use the coordinate reference system specified in the sr parameter.

find-convert/sr |

# Find with support for feature filtering

## Overview

This conformance class adds support for filtering the displayed features in the exported map by specifying definition expressions per layer by adding an additional parameter.

Table 35 – Export Map with support for feature filtering

|  |  |  |
| --- | --- | --- |
| **Resource** | **Parameters** | **Resource representation** |
| Find | layerDefs | - |

## Find

### Dependency

This conformance class extends the requirements for the Find operation as specified in Clause 17.

### Find URI

Table 36 – Additonal Find parameters

|  |  |
| --- | --- |
| **Parameter** | **Details** |
| layerDefs | Allows you to filter the features of individual layers by specifying definition expressions for those layers.The filter expressions for each layer are applied to select the relevant features of the layer. |
| Required | No. Default: No filter is considered in the generation of the map. |
| Syntax | (NUMID ":" EXPR \*(";" NUMID ":" EXPR)) / JSONIf the parameter matches the JSON rule, it is a JSON object that validates against the following JSON Schema:{ "id" : "http://www.opengis.net/gsr-ms/1.0/layerDefs#", "title":"layer filter definitions", "type":"object",  "patternProperties":{ "\*":{ "title":"filter expression for the layer", "type":"string"} }} |
| Example | 0:POP2000 > 1000000;5:AREA > 100000 (text representation){"0":"POP2000 > 1000000","5":"AREA > 100000"} (JSON representation) |

**Request Requirements**

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

find-filter/parameters |

### Identified Features resources

This conformance class does not add any requirements on the JSON representation of the Identified Features resources.

# Query

## Overview

The Query operation is performed on a controller resource of the map service Layer/Table resource. The result of this operation is is either a feature set, an informational structure for storing multiple feature objects, 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.

The result feature set contains feature objects, including the values for the fields requested by the user.

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 37 – Query overview

|  |  |  |
| --- | --- | --- |
| **Resource** | **Parameters** | **Resource representation** |
| Query | f=jsongeometrygeometryTypetextwherereturnGeometryinSRoutSRmaxAllowableOffsetspatialRelrelationParamobjectIdsoutFieldsreturnIdsOnly | JSON representation validAll JSON schema elements |

## Query

### Query URI

In the following URI templates, these variables are used:

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

Table 38 – Query reference

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

Table 39 – Query parameters

|  |  |
| --- | --- |
| **Parameter** | **Details** |
| f | The response format.  |
| Required | Yes |
| Syntax | "json" / "image" |
| Example | f=json |
| geometry | The geometry to identify on. 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 | Yes |
| 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 |
| text | Description: A literal search text. If the layer has a display field associated with it, the server searches for this text in this field. This parameter is shorthand for a WHERE clause as follows:where <displayField> like '%<text>%' The text is case sensitive. |
| Required | No. Default: no literal text search. The parameter is ignored if the where parameter is specified. |
| Syntax | STRING |
| Example | text = Los |
| where | A WHERE clause for the query filter. Any legal SQL WHERE clause operating on the fields in the layer is allowed. |
| 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 |
| maxAllowableOffset | Specifies the maximum allowable offset to be used for generalizing geometries returned by the Query operation.The maxAllowableOffset value is in the units of the spatial reference. If a value for sr is not specified, maxAllowableOffset is assumed to be in the units of the spatial reference of the map. |
| Required | No. Default: no generalization |
| Syntax | NUMBER |
| Example | maxAllowableOffset=2 |
| 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 38 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 39.

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

Query using the text parameter on the states layer of the StateCityHighway service on example.com:

http://example.com/rest/services/StateCityHighway/MapServer/1/query?text=CA&f=json

**Request**

GET /rest/services/StateCityHighway/MapServer/query?text=CA&f=json HTTP/1.1

Host: example.com

**Response**

HTTP/1.1 200 OK

Content-Type: application/json

Content-Length: nnn

{

"displayFieldName" : "AREANAME",

"fields" : [

{

 "name" : "ST",

 "alias" : "ST",

 "type" : "FieldTypeString",

 "length" : 2

},

{

 "name" : "POP2000",

 "alias" : "Population - 2000",

 "type" : "FieldTypeInteger"

},

{

 "name" : "AREANAME",

 "alias" : "City Name",

 "type" : "FieldTypeString",

 "length" : 255

}

],

"geometryType" : "GeometryPoint",

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

"features" : [

 {

 "attributes" : {

 "ST" : "CA",

 "POP2000" : 3694820,

 "AREANAME" : "Los Angeles"

 },

 "geometry" : { "x" : -118.37, "y" : 34.086 }

 },

 {

 "attributes" : {

 "ST" : "CA",

 "POP2000" : 461522,

 "AREANAME" : "Long Beach"

 },

 "geometry" : { "x" : -118.15, "y" : 33.80 }

 }

]

}

Other queries may vary the parameters, but return JSON representations using the same schema:

Query using a where statement on the same layer:

http://example.com/rest/services/StateCityHighway/MapServer/1/query?where=STATE\_NAME='Florida'&f=json

Query strings are case sensitive. In this example, uppercase is used to make the query case insensitive:

http://example.com/rest/services/StateCityHighway/MapServer/1/query?where=UPPER(STATE\_NAME)=UPPER('colorado')&f=json

Query the same states layer using geometry (envelope):

http://example.com/rest/services/StateCityHighway/MapServer/1/query?geometry=-125.4,35.2,-118.7,43.8&geometryType=GeometryEnvelope&f=json

Query the states layer by both geometry (envelope) and a where statement:

http://example.com/rest/services/StateCityHighway/MapServer/1/query?geometry=-125.4,35.2,-118.7,43.8&geometryType=GeometryEnvelope&

where=POP1999>5000000&f=json

Query the states layer by where statement, specifying a list of fields to return and requesting no geometry in the results:

http://example.com/rest/services/StateCityHighway/MapServer/1/query?where=POP1999>15000000&returnGeometry=false&outFields=STATE\_NAME,MALES,FEMALES,POP1999&f=json

Query the states layer by text parameter, requesting the geometry with the well-known ID of 102113 (Web Mercator):

http://example.com/rest/services/StateCityHighway/MapServer/1/query?text=New+York&outSR=102113&f=json

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

http://example.com/rest/services/311Incidents/MapServer/1/ query?objectIds=&where=agree\_with\_incident+%3D+1&geometryType=GeometryEnvelope&spatialRel=SpatialRelIntersects&returnGeometry=true&returnIdsOnly=true&f=json

In this last query, the response is different due to the returnIdsOnly parameter:

**Request**

GET /rest/services/311Incidents/MapServer/1/query?objectIds=&where=agree\_with\_incident+%3D+1&geometryType=GeometryEnvelope&spatialRel=SpatialRelIntersects&returnGeometry=true&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]

}

# 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 40 – Temporal Query overview

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

## Query

### Dependency

This conformance class extends the requirements for the Query operation as specified in Clause 20.

### Query request

Table 41 – 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 41.

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 42 – Query Related Records overview

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

## Query Related Records

### Query Related Records URI

In the following URI templates, these variables are used:

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

Table 43 – Query Related Records reference

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

Table 44 – 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 |
| maxAllowableOffset | Specifies the maximum allowable offset to be used for generalizing geometries returned by the Query Related Records operation.The maxAllowableOffset value is in the units of the spatial reference. If a value for outSR is not specified, maxAllowableOffset is assumed to be in the units of the spatial reference of the map. |
| Required | No. Default: no generalization |
| Syntax | NUMBER |
| Example | maxAllowableOffset=2 |
| 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 43 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 44.

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 |

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

queryrel/tables |

### Example

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

**Request**

GET /rest/services/KSPetro/MapServer/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

 ]

 ]

 ]

 }

 }

 ]

 }

 ]

}

# 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 45 – 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 |

## 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:

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

Table 46 – Feature reference

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

Table 47 – 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 46 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 47.

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

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

**Request**

GET /rest/services/311Incidents/MapServer/0/818654/attachments?f=json 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 Infos URI

In the following URI templates, these variables are used:

* mapServiceURI: URL of a Map 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 48 – Attachment reference

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

**Example**

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

**Request Requirements**

|  |
| --- |
| * + 1. The Attachment resource SHALL accept requests that conform to the URI template in Table 48 and use any HTTP method identified in the same table.

attachements/request |

### Attachment resources

**Response Requirements**

|  |
| --- |
| * + 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 |

## 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 resource's htmlPopupType parameter is not ServerHTMLPopupTypeNone.

### HTML Popup URI

In the following URI templates, these variables are used:

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

Table 49 – HTML Popup reference

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

Table 50 – 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 49 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 50.

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

http://example.com/rest/services/StantonCounty/MapServer/0/1/htmlPopup?f=json

**Request**

GET /rest/services/ StantonCounty/MapServer/0/1/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."

}

Annex A
(normative)

Abstract Test Suite

Conformance class: mapservice

* 1. Test: mapservice/root

|  |  |
| --- | --- |
| Requirements | **mapservice/request, mapservice/parameters, mapservice/valid, mapservice/units, mapservice/imageFormats, mapservice/png, mapservice/documentInfo, mapservice/capMap** |
| Test purpose | Verify that the Map Service Root resource supports the request and response requirements. |
| Test method | Set up a test service. Construct valid requests for the Map Service Root resource.Inspect the responses and validate them against the JSON Schema http://schemas.opengis.net/gsr-ms/1.0/root.json or for exceptions against http://schemas.opengis.net/gsr/1.0/exception.json. Inspect the Map Service Root resource and verify that* all spatial reference are correct
* the units property is an allowed value.
* the supportedImageFormatTypes property is a comma separated list and contain the value "PNG"
* the documentInfo property contains only string-valued properties
* the capabilities property is a comma separated list and include the value "Map"
 |
| Test type | Capability |

* 1. Test: mapservice/layerIds

|  |  |
| --- | --- |
| Requirements | **mapservice/uniqueLayerTableId, mapservice/layersExist** |
| Test purpose | Verify uniqueness and consistency of identifiers |
| Test method | Set up a test service and attempt to define two layers and/or tables with identical id. Verify that the implementation rejects layer/table ids that are not unique.Set up hierarchical layers and verify that all layer references point to layers described in the Map Service Root resource.  |
| Test type | Capability |

* 1. Test: mapservice/exportMap

|  |  |
| --- | --- |
| Requirements | **mapservice/mapRequest, mapservice/mapParameters, mapservice/mapValid, mapservice/imageSize, mapservice/aspectRatio, mapservice/imageFormat** |
| Test purpose | Verify that the Export Map resource supports the request and response requirements. |
| Test method | Set up a test service. Construct valid requests for the Export Map resource varying the values of all parameters.Inspect the responses and validate those with (f=json) against the JSON Schema http://schemas.opengis.net/gsr-ms/1.0/map.json or for exceptions against http://schemas.opengis.net/gsr/1.0/exception.json. Inspect the image representation of a Map resource and the image file referenced from the the href property of the JSON representation of a Map resource and verify that width, height and dpi are as specified by the query parameters. Submit requests where the aspect ratio is inconsistent between the parameters bbox and size and inspect the response to verify that the extent of the returned map image has been adjusted to have the same aspect ratio as the image. For f=json requests verify inspect the Map resource and verify that the bbox is the adjusted envelope.Inspect the bitmap image and verify that it is in the requested image format. |
| Test type | Capability |

* 1. Test: mapservice/layerOrTable

|  |  |
| --- | --- |
| Requirements | **mapservice/layerOrTableRequest, mapservice/layerOrTableParameters, mapservice/layerOrTableValid** |
| Test purpose | Verify that the Layer/Table resources support the request and response requirements. |
| Test method | Set up a test service. Construct valid requests for all Layer/Table resources listed in the Map Service Root resource.Inspect the responses and validate them 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: mapservice/image

|  |  |
| --- | --- |
| Requirements | **mapservice/imageRequest, mapservice/image** |
| Test purpose | Verify that the Image resources support the request and response requirements. |
| Test method | Set up a test service. Construct valid requests for selected images in one or more layers listed in the Map Service Root resource.Inspect the responses and verify that the response is a single image file or a 404 response.  |
| Test type | Capability |

* 1. Test: mapservice/allLayersAndTables

|  |  |
| --- | --- |
| Requirements | **mapservice/allLayersAndTablesRequest, mapservice/allLayersAndTablesParameters, mapservice/allLayersAndTablesValid, mapservice/allLayersAndTablesExist, mapservice/layersTables** |
| Test purpose | Verify that the All Layers And Tables resource supports the request and response requirements. |
| Test method | Set up a test service. Construct valid requests for the All Layers and Tables resource.Inspect the responses and validate them against the JSON Schema http://schemas.opengis.net/gsr-ms/1.0/allLayersAndTables.json or for exceptions against http://schemas.opengis.net/gsr/1.0/exception.json. Inspect that each layer listed in the layers property is a "Feature layer" and that each table listed in the tables property is a "Table".For each layer and table, access the Layer/Table resource and verify that it exists. |
| Test type | Capability |

Conformance class: convert

* 1. Test: convert/exportMap

|  |  |
| --- | --- |
| Requirements | **convert/parameters, convert/validSR, convert/imageSR** |
| Test purpose | Verify that the Export Map resource supports the request and response requirements. |
| Test method | Set up a test service. Construct valid requests for the Export Map resource that use the parameters added by the requirements class.Inspect the responses and validate those with (f=json) against the JSON Schema http://schemas.opengis.net/gsr-ms/1.0/map.json or for exceptions against http://schemas.opengis.net/gsr/1.0/exception.json. In the request use spatial references that use both correct and incorrect values for wkid as well as wkt. Verify that incorrect values raise an exception.For requests with f=json verify that the extent property in the JSON representation of the Map resource uses the coordinate reference system specified in the imageSR parameter. |
| Test type | Capability |

Conformance class: time

* 1. Test: time/layerOrTable

|  |  |
| --- | --- |
| Requirements | **time/timeInfo** |
| Test purpose | Verify that the Layer/Table resource supports the request and response requirements. |
| Test method | Set up a test service that supports temporal aspects. Inspect the Layer/Table resource 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 and
* verify that it contains the timeInfo property.
 |
| Test type | Capability |

* 1. Test: time/exportMap

|  |  |
| --- | --- |
| Requirements | **time/parameters** |
| Test purpose | Verify that the Export Map resource supports the request and response requirements. |
| Test method | Set up a test service. Construct valid requests for the Export Map resource that use the parameters added by the requirements class.Inspect the responses and validate those with (f=json) against the JSON Schema http://schemas.opengis.net/gsr-ms/1.0/map.json or for exceptions against http://schemas.opengis.net/gsr/1.0/exception.json.  |
| Test type | Capability |

Conformance class: filter

* 1. Test: filter/exportMap

|  |  |
| --- | --- |
| Requirements | **filter/parameters** |
| Test purpose | Verify that the Export Map resource supports the request and response requirements. |
| Test method | Set up a test service. Construct valid requests for the Export Map resource that use the parameters added by the requirements class.Inspect the responses and validate those with (f=json) against the JSON Schema http://schemas.opengis.net/gsr-ms/1.0/map.json or for exceptions against http://schemas.opengis.net/gsr/1.0/exception.json.  |
| Test type | Capability |

Conformance class: tile

* 1. Test: tile/requests

|  |  |
| --- | --- |
| Requirements | **tile/availability, tile/request, tile/imageSize, tile/imageFormat** |
| Test purpose | Verify that the Map Tile resources are available and support the request and response requirements. |
| Test method | Set up a test service so that a tile cache is available. Verify that the Map Service Root resource includes the singleFusedMapCache property with a value of "true" and the tileInfo property.Access a representative set of Map Tile resources in accordance with the tileInfo information and verify that all requested tiles are available.For each accessed tile, inspect that it has the width, height and dpi as specified by the tileInfo/cols, tileInfo/rows and tileInfo/Dpi properties of the Map Service Root resource.Inspect the bitmap image and verify that it is in the image format specified by the tileInfo/format property of the Map Service Root resource. |
| Test type | Capability |

Conformance class: identify

* 1. Test: identify/requests

|  |  |
| --- | --- |
| Requirements | **identify/availability, identify/request, identify/parameters, identify/valid** |
| Test purpose | Verify that the Identify resource supports the request and response requirements. |
| Test method | Set up a test service. Verify that the Map Service Root resource includes the value "Query".Construct valid requests for the Identify resource varying the values of all parameters. Inspect the responses and validate them against the JSON Schema http://schemas.opengis.net/gsr-ms/1.0/identifiedFeatures.json or for exceptions against http://schemas.opengis.net/gsr/1.0/exception.json.  |
| Test type | Capability |

Conformance class: identify-convert

* 1. Test: identify-convert/requests

|  |  |
| --- | --- |
| Requirements | **identify-convert/parameters, identify-convert/validSR, identify-convert/sr** |
| Test purpose | Verify that the identify resource supports the request and response requirements. |
| Test method | Set up a test service. Construct valid requests for the Identify resource that use the parameters added by the requirements class.Inspect the responses and validate them against the JSON Schema http://schemas.opengis.net/gsr-ms/1.0/identifiedFeatures.json or for exceptions against http://schemas.opengis.net/gsr/1.0/exception.json. In the request use spatial references that use both correct and incorrect values for wkid as well as wkt. Verify that incorrect values raise an exception.Verify that all geometries returned in the JSON representation of the Identified Features resource use the coordinate reference system specified in the sr parameter. |
| Test type | Capability |

Conformance class: identify-time

* 1. Test: identify-time/requests

|  |  |
| --- | --- |
| Requirements | **identify-time/timeInfo, identfy-time/parameters** |
| Test purpose | Verify that the Identify resource supports the request and response requirements. |
| Test method | Set up a test service that supports temporal aspects. Inspect the Layer/Table resources and verify that timeInfo properties are provided correctly.Construct valid requests for the Identify resource that use the parameters added by the requirements class.Inspect the responses and validate them against the JSON Schema http://schemas.opengis.net/gsr-ms/1.0/identifiedFeatures.json or for exceptions against http://schemas.opengis.net/gsr/1.0/exception.json.  |
| Test type | Capability |

Conformance class: identify-filter

* 1. Test: identify-filter/requests

|  |  |
| --- | --- |
| Requirements | **identify-filter/parameters** |
| Test purpose | Verify that the Identify resource supports the request and response requirements. |
| Test method | Set up a test service. Construct valid requests for the Identify resource that use the parameters added by the requirements class.Inspect the responses and validate them against the JSON Schema http://schemas.opengis.net/gsr-ms/1.0/identifiedFeatures.json or for exceptions against http://schemas.opengis.net/gsr/1.0/exception.json.  |
| Test type | Capability |

Conformance class: data

* 1. Test: data/feature

|  |  |
| --- | --- |
| Requirements | **data/availability, data/request, data/parameters, data/featureValid** |
| Test purpose | Verify that the Feature resources support the request and response requirements. |
| Test method | Set up a test service. Verify that the Map Service Root resource includes the value "Data".Construct valid requests for each layer or table to Feature resources. Inspect the responses and validate them against the JSON Schema http://schemas.opengis.net/gsr/1.0/singleFeature.json or for exceptions against http://schemas.opengis.net/gsr/1.0/exception.json.  |
| Test type | Capability |

Conformance class: find

* 1. Test: find/requests

|  |  |
| --- | --- |
| Requirements | **find/request, find/parameters, find/valid** |
| Test purpose | Verify that the Find resource supports the request and response requirements. |
| Test method | Set up a test service. Construct valid requests for the Identify resource varying the values of all parameters. Inspect the responses and validate them against the JSON Schema http://schemas.opengis.net/gsr-ms/1.0/identifiedFeatures.json or for exceptions against http://schemas.opengis.net/gsr/1.0/exception.json. |
| Test type | Capability |

Conformance class: find-convert

* 1. Test: find-convert/requests

|  |  |
| --- | --- |
| Requirements | **find-convert/parameters, find-convert/sr** |
| Test purpose | Verify that the Find resource supports the request and response requirements. |
| Test method | Set up a test service. Construct valid requests for the Find resource that use the parameters added by the requirements class.Inspect the responses and validate them against the JSON Schema http://schemas.opengis.net/gsr-ms/1.0/identifiedFeatures.json or for exceptions against http://schemas.opengis.net/gsr/1.0/exception.json. In the request use spatial references that use both correct and incorrect values for wkid as well as wkt. Verify that incorrect values raise an exception.Verify that all geometries returned in the JSON representation of the Identified Features resource use the coordinate reference system specified in the sr parameter. |
| Test type | Capability |

Conformance class: find-filter

* 1. Test: find-filter/requests

|  |  |
| --- | --- |
| Requirements | **find-filter/parameters** |
| Test purpose | Verify that the Find resource supports the request and response requirements. |
| Test method | Set up a test service. Construct valid requests for the Find resource that use the parameters added by the requirements class.Inspect the responses and validate them against the JSON Schema http://schemas.opengis.net/gsr-ms/1.0/identifiedFeatures.json or for exceptions against http://schemas.opengis.net/gsr/1.0/exception.json.  |
| Test type | Capability |

Conformance class: query

* 1. Test: query/requests

|  |  |
| --- | --- |
| Requirements | **query/request, query/parameters, query/valid, query/tables** |
| Test purpose | Verify that the Query resource supports the request and response requirements. |
| Test method | Set up a test service. Construct valid requests for the Query resource varying the values of all parameters. Inspect the responses and validate them 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.Verify that all parameters related to geometries are ignored when querying tables. |
| Test type | Capability |

Conformance class: query-time

* 1. Test: query-time/requests

|  |  |
| --- | --- |
| Requirements | **query-time/parameters** |
| Test purpose | Verify that the Query resource supports the request and response requirements. |
| Test method | Set up a test service that supports temporal aspects. Inspect the Layer/Table resources and verify that timeInfo properties are provided correctly.Construct valid requests for the Query resource that use the parameters added by the requirements class.Inspect the responses and validate them 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. |
| Test type | Capability |

Conformance class: queryrel

* 1. Test: queryrel/requests

|  |  |
| --- | --- |
| 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 | Set up a test service with relationships between layers and tables. Construct valid requests for the Query Related Recordsresource varying the values of all parameters. Inspect the responses and validate them 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.Verify that all parameters related to geometries are ignored when querying tables. |
| Test type | Capability |

Conformance class: attachements

* 1. Test: attachements/attachmentInfo

|  |  |
| --- | --- |
| Requirements | **attachments/infoRequest, attachements/infoParameters, attachments/infoValid, attachements/request, attachments/valid** |
| Test purpose | Verify that the Attachment Infos and Attachment resources supports the request and response requirements. |
| Test method | Set up a test service with attachments associated to features using different mime types. Construct valid requests to the Attachment Infos resource for the features with attachments. Inspect the responses and validate them 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.Access each attachment referenced and inspect the responses and verify that the response is of the stated mime type.Request attachements with ids that do not exist and verify that the results is a HTTP 404 response. |
| Test type | Capability |

* 1. Test: attachements/popup

|  |  |
| --- | --- |
| Requirements | **attachments/popupRequest, attachments/popupParameters, attachements/popupValid** |
| Test purpose | Verify that the HTML Popup resources supports the request and response requirements. |
| Test method | Set up a test service with HTML popups associated to features. Construct valid requests to the HTML Popup resource for the features with such popups. Inspect the responses and validate them 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. |
| Test type | Capability |

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