Open Geospatial Consortium Inc.

Date:   2012-06-22

Reference number of this OGC® project document: OGC 12-059r1

Version: 0.0.3

Category: OGC® Implementation Specification

Editor:   Clemens Portele, Satish Sankaran

GeoServices REST API — Part 6: Image 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 3

4 Terms and Definitions 4

5 Conventions 4

6 Image Service overview 4

7 Image Service Core 5

7.1 Overview 5

7.2 Image Service Root 6

7.2.1 Image Service Root URI 6

7.2.2 Image Service Root resources 8

7.2.3 Example 9

7.3 Export Image 9

7.3.1 Overview 9

7.3.2 Export Image URI 10

7.3.3 Image resources 16

7.3.4 Examples 17

8 Export Image with support for coordinate transformation 18

8.1 Overview 18

8.2 Export Image 18

8.2.1 Dependency 18

8.2.2 Export Image request 18

8.2.3 Image resources 19

8.2.4 Examples 19

9 Export Image with support for time 20

9.1 Overview 20

9.2 Image Service Root resources 20

9.3 Export Image 21

9.3.1 Dependency 21

9.3.2 Export Image URI 21

9.3.3 Image resources 22

10 Identify 22

10.1 Overview 22

10.2 Identify URI 22

10.3 Identified Items resources 25

10.4 Examples 25

11 Query 29

11.1 Overview 29

11.2 Query URI 30

11.3 Feature Set and Feature ID Set resources 35

11.4 Examples 35

12 Temporal Query 37

12.1 Overview 37

12.2 Query 38

12.2.1 Dependency 38

12.2.2 Query request 38

12.2.3 Feature Set and Feature ID Set resources 38

13 Raster Catalog 39

13.1 Overview 39

13.2 Raster Catalog Item 40

13.2.1 Overview 40

13.2.2 Raster Catalog Item URI 40

13.2.3 Raster Catalog Item resources 41

13.2.4 Example 41

13.3 Raster Image 42

13.3.1 Overview 42

13.3.2 Raster Catalog Item URI 42

13.3.3 Raster Image resources 45

13.3.4 Example 46

13.4 Raster Thumbnail 46

13.4.1 Overview 46

13.4.2 Raster Thumbnail URI 47

13.4.3 Example 47

13.5 Raster Info 47

13.5.1 Overview 47

13.5.2 Raster Info URI 47

13.5.3 Raster Info resources 48

13.5.4 Example 49

13.6 Download Rasters 49

13.6.1 Overview 49

13.6.2 Download Rasters URI 49

13.6.3 Raster Files resources 52

13.6.4 Example 52

13.7 Raster File 53

13.7.1 Overview 53

13.7.2 Raster File URI 53

13.7.3 Example 54

A.1 Conformance class: imgservice 56

A.1.1 Test: imgservice/root 56

A.1.2 Test: imgservice/exportImage 56

A.2 Conformance class: convert 57

A.2.1 Test: convert/exportImage 57

A.3 Conformance class: time 57

A.3.1 Test: time/timeInfo 57

A.3.2 Test: time/exportImage 57

A.4 Conformance class: identify 58

A.4.1 Test: identify/identify 58

A.5 Conformance class: query 58

A.5.1 Test: query/query 58

A.6 Conformance class: querytime 58

A.6.1 Test: querytime/query 58

A.7 Conformance class: catalog 59

A.7.1 Test: catalog/catItem 59

A.7.2 Test: catalog/rasterImg 59

A.7.3 Test: catalog/rasterThumb 59

A.7.4 Test: catalog/rasterInfo 60

A.7.5 Test: catalog/download 60

A.7.6 Test: catalog/rasterFile 60

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. 28 May 2012
 | 1. 0.0.1
 | 1. CP, SS
 | 1. All
 | 1. First draft
 |
| 1. 31 May 2012
 | 1. 0.0.2
 | 1. CP
 |  | 1. RFC package for OAB review
 |
| 1. 22 Jun 2012
 | 1. 0.0.3
 | 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 6 of the GeoServices REST API series:

Part 1: Core

Part 2: Catalog

Part 3: Map Service

Part 4: Feature Service

Part 5: Geometry Service

Part 6: Image Service

Part 7: Geoprocessing Service

Part 8: Geocoding Service

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

# Scope

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

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

# Conformance

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

This Standard establishes 7 requirements classes and corresponding conformance classes, extending the core conformance class of the GeoServices REST API series.

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

Any implementation claiming conformance with a conformance class shall pass all the tests in the associated abstract test suite. Table 1 summarizes the requirements and conformance tests associated per conformance class.

Table 1 – Conformance class summary

|  |  |  |
| --- | --- | --- |
| **imgservice** | **Title** | Image Service Core |
| **Standardization target type** | Web service |
| **Dependencies** | **http://www.opengis.net/spec/gsr/1.0/conf/core****http://www.opengis.net/spec/gsr/1.0/conf/geometry** |
| **Requirements** | All requirements in Clause 7 |
| **Conformance tests** | Annex A.1 |
| **convert** | **Title** | Export Image with support for coordinate transformation |
| **Standardization target type** | Web service |
| **Dependencies** | **conf/imgservice** |
| **Requirements** | All requirements in Clause 8 |
| **Conformance tests** | Annex A.2 |
| **time** | **Title** | Export Image with support for time |
| **Standardization target type** | Web service |
| **Dependencies** | **conf/imgservice** |
| **Requirements** | All requirements in Clause 9 |
| **Conformance tests** | Annex A.3 |
| **identify** | **Title** | Query/Identify |
| **Standardization target type** | Web service |
| **Dependencies** | **conf/imgservice** |
| **Requirements** | All requirements in Clause 10 |
| **Conformance tests** | Annex A.4 |
| **query** | **Title** | Query |
| **Standardization target type** | Web service |
| **Dependencies** | **conf/imgeservice****http://www.opengis.net/spec/gsr/1.0/conf/feature** |
| **Requirements** | All requirements in Clause 11 |
| **Conformance tests** | Annex A.5 |
| **querytime** | **Title** | Temporal Query  |
| **Standardization target type** | Web service |
| **Dependencies** | **conf/query** |
| **Requirements** | All requirements in Clause 12 |
| **Conformance tests** | Annex A.6 |
| **catalog** | **Title** | Raster Catalog |
| **Standardization target type** | Web service |
| **Dependencies** | **conf/imgservice****http://www.opengis.net/spec/gsr/1.0/conf/feature** |
| **Requirements** | All requirements in Clause 13 |
| **Conformance tests** | Annex A.7 |

Figure 1 – Conformance class overview

# References

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

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

# Terms and Definitions

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

# Conventions

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

# Image Service overview

An image service provides access to published imagery. Image services support two views of the published imagery: a mosaicked image view and a raster catalog view. A raster catalog is a collection of raster datasets defined in a table format in which each record represents an individual raster dataset in the catalog.

Use an image service to do the following:

* Get image service information, including its native spatial reference, extent, pixel size, pixel type, number of bands, and band statistics.
* Generate an image.
* Query the raster catalog.
* Download rasters.

The Image Service Root resource returns information about imagery exposed through a Web service, such as the imagery's extent, pixel sizes, and band counts. This resource also returns the accessible fields of the image service.

The Image Service resource supports the following operations:

* Export Image: Returns a seamless mosaicked image for the specified area
* Identify: Identifies the content of an image service
* Query: Queries the image service
* Download: Downloads raw raster files

The Query and Download operations are not available if the service does not include an accessible image catalog.

The following figure provides an overview of the resources in a Image 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

# Image Service Core

## Overview

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

Table 2 – Image Service Core overview

|  |  |  |
| --- | --- | --- |
| **Resource** | **Parameters** | **Resource representation** |
| Image Service Root | f=json | JSON representation validAll JSON schema elements supported (except timeInfo) |
| Export Image | f=jsonf=imagebboxsizeformatpixelTypenoDatainterpolationcompressionQualitybandIdsmosaicRulerenderingRule | JSON representation validAll JSON schema elements supportedImage has correct format and size |

## Image Service Root

### Image Service Root URI

In the following URI templates, these variables are used:

* imgServiceRootURI: the URL of the service

If the Image Service is referenced from a Catalog Service, the URI template of variable is

{catServiceRootURI}/{imgServiceName}/ImageServer

where imgServiceName is the name of a image service referenced in the catalogue.

Table 3 – Image Service Root reference

|  |  |
| --- | --- |
| **URI template** | {+imgServiceRootURI}{?f} |
| **HTTP methods** | GET |
| **Parent Resource Type** | - |
| **Child Resource Types**  | Export ImageIdentify (only if conformance class "identify" is supported)Query (only if conformance class "query" is supported)Raster Catalog Item (only if conformance class "catalog" is supported)Download Rasters (only if conformance class "catalog" is supported)Raster File (only if conformance class "catalog" is supported) |

Table 4 – Image Service Root parameters

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

**Request Requirements**

|  |
| --- |
| * + 1. The Image 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.

imgservice/request |

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

imgservice/parameters |

### Image Service Root resources

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

imgservice/valid |

|  |
| --- |
| * + 1. Each spatial reference in the JSON representation of a Image Service Root resource SHALL conform to the requirements for geometry objects (see requirements class **http://www.opengis.net/spec/gsr/1.0/req/geometry**).

imgservice/spatialReference |

|  |
| --- |
| * + 1. The envelope in property extent SHALL contain all locations supported by the image service.

imgservice/extent |

|  |
| --- |
| * + 1. The property bandCount SHALL reflect that number of bands in the raster files.

imgservice/bandCount |

|  |
| --- |
| * + 1. The property pixelType SHALL specify the data type for each band in each raster cell.

imgservice/pixelType |

The property serviceDataType SHOULD describe the characteristic of the raster data:

* ImageServiceDataTypeGeneric: no information about the data
* ImageServiceDataTypeRGB: images with RGB color values
* ImageServiceDataTypeElevation: elevation data (height/depth values)
* ImageServiceDataTypeThematic: thematic data
* ImageServiceDataTypeProcessed: processed data

The properties minValues, maxValues, meanValues and stdvValues SHOULD provide information about the minimum, maxmimum, mean and standard deviation values for each band.

|  |
| --- |
| * + 1. If the raster catalog is accessible, the fields (attributes) of the raster features SHALL be specified in the property fields and the object identifier property SHALL be specified in the objectIdField property.

imgservice/fields |

### Example

URL for the MyImage service on example.com:

http://example.com/rest/services/MyImage/ImageServer?f=json

**Request**

GET /rest/services/MyImage/ImageServer?f=json HTTP/1.1

Host: example.com

**Response**

HTTP/1.1 200 OK

Content-Type: application/json

Content-Length: nnn

{

"serviceDescription" : "Test Image Service Description",

"name" : "wsiearth.tif",

"description" : "wsiearth.tif",

"extent" : {"xmin" : -180, "ymin" : -90, "xmax" : 180, "ymax" : 90, "spatialReference" : {"wkid" : 4326}},

"timeInfo" : {"timeExtent" : [1106822673000,1125907321000], "timeReference" : null},

"pixelSizeX" : 30.386,

"pixelSizeY" : 30.386,

"bandCount" : 3,

"pixelType" : "U8",

"minPixelSize" : 0.0,

"maxPixelSize" : 0.0,

"copyrightText" : "",

"serviceDataType" : "ImageServiceDataTypeRGB",

"minValues" : [0.0, 0.0, 0.0],

"maxValues" : [255.0, 254.0, 255.0],

"meanValues" : [82.707, 107.448, 60.118],

"stdvValues" : [39.838, 37.735, 36.466],

"objectIdField":"OBJECTID",

"fields":[{"name":"OBJECTID","type":"FieldTypeOID","alias":"OBJECTID"},{"name":"Shape","type":"FieldTypeGeometry","alias":"Shape"},{"name":"Name","type":"FieldTypeString","alias":"Name"},{"name":"CloudCover","type":"FieldTypeDouble","alias":"Cloud Cover"}]}

## Export Image

### Overview

The Export Image operation is performed on a controller resource of the image service. The result of this operation provides an Image resource with information about the exported image such as its URL, width, height, and extent. The Image 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. No other information about the image is returned with this option.

Users provide arguments to the operation as query parameters. These parameters include the request extent, size information, interpolation, and pixel type.

### Export Image URI

In the following URI templates, these variables are used:

* imgServiceURI: URL of a Image Service Root resource without any parameter

Table 5 – Export Image reference

|  |  |
| --- | --- |
| **URI template** | {+imgServiceURI}/exportImage{?f,bbox,size,format,pixelType,noData, interpolation,compressionQuality,bandIds,mosaicRule,renderingRule} |
| **HTTP methods** | GETPOST (application/x-www-form-urlencoded) |
| **Parent Resource** | Image Service Root |

Table 6 – Export Image 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 image service. 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 |
| format | The format of the exported image.Pre-defined values:* png: image as specified by media type "image/png"
* png8: image as specified by media type "image/png" with 8-bit color depth
* png24: image as specified by media type "image/png" with 24-bit color depth
* jpg: image as specified by media type "image/jpeg
* bmp: file according to the Device Independent Bitmap (DIB) file format as specified by Microsoft
* gif: image as specified by media type "image/gif"
* tiff: image as specified by media type "image/tiff"
* jpgpng: uses "jpg", if there are no transparent pixels in the requested extent; otherwise "png" is used
 |
| Required | No. Default: "jpgpng" |
| Syntax | "jpgpng" / "png" / "png8" / "png24" / "jpg" / "bmp" / "gif" / "tiff"  |
| Example | format=tiff |
| pixelType | The pixel type, also known as data type, pertains to the type of values stored in a band of the raster, such as signed integer ("S"), unsigned integer ("U"), or floating point ("F"). The number following the letter is the number of bits. Integers are whole numbers, whereas floating points have decimals. |
| Required | No. Default: The value specified in the Image Service Root resource is used. |
| Syntax | "C128" / "C64" / "F32" / "F64" / "S16" / "S32" / "S8" / "U1" / "U16" / "U2" / "U32" / "U4" / "U8" / "UNKNOWN" |
| Example | pixelType=U8 |
| noData | The pixel value representing "no information".In the case of multiple bands, the value applies to all bands. |
| Required | No. Default: no value is treated as "no information". |
| Syntax | NUMBER / STRING |
| Example | noData=0 |
| interpolation | The resampling process of extrapolating the pixel values while transforming the raster dataset, e.g. when it undergoes warping or changes the coordinate reference system.Pre-defined values:* RSP\_BilinearInterpolation: resample pixel by bilinear interpolation
* RSP\_CubicConvolution: resample pixel by cubic convolution
* RSP\_Majority: resample pixel by majority value
* RSP\_NearestNeighbor: resample pixel by nearest neighbor
 |
| Required | No. Default: RSP\_NearestNeighbor. |
| Syntax | "RSP\_BilinearInterpolation" / "RSP\_CubicConvolution" / "RSP\_Majority" / "RSP\_NearestNeighbor" |
| Example | noData=0 |
| compressionQuality | Controls how much loss the image will be subjected to by the compression algorithm. Valid value ranges of compression quality are from 0 to 100. Larger numbers mean less compression occurs, resulting in higher-quality images. |
| Required | No. |
| Syntax | INTEGER |
| Example | compressionQuality=75 |
| bandIds | If there are multiple bands, users can specify a single band to export, or they can change the band combination (red, green, blue) by specifying the band number. Band number is 0 based. |
| Required | No. Default: all bands in the default sequence. |
| Syntax | INTEGER \*("," INTEGER) |
| Example | bandIds=2,1,0 |
| mosaicRule | Specifies the mosaic rule when defining how individual images should be mosaicked. The rule defines selection, mosaic method, sort order, overlapping pixel resolution, and so forth. Pre-defined mosaic methods are:MosaicNone: Images have no special ordering.MosaicCenter: Sorts rasters based on their proximity to the view center or the center of view extent.MosaicNadir: Sorts rasters based on the distance between the nadir position and view center.MosaicViewpoint: Sorts rasters based on a user-defined viewpoint location and nadir location for the raster Mosaic operations that apply: first (default), last, min, max, mean, and blend.MosaicAttribute: Sorts rasters based on an attribute field and its difference from a base value.MosaicLockRaster: Selects only the rasters in a given list of raster IDs to participate in the mosaic.MosaicNorthwest: Sorts rasters in a view-independent way, where rasters with their centers most northwest are displayed on top.MosaicSeamline: Cuts the raster using a predefined seamline shape for each raster using optional feathering along the seams.Mosaicking determines which cell value is used in the case of overlapping rasters (the first raster specified, the last raster specified, the minimum value, the maximum value, the mean, or a blend). |
| Required | No. Default: MosaicNone. |
| Syntax | JSONValidates against http://schemas.opengis.net/gsr-is/1.0/mosaicRule.json  |
| Example | mosaicRule={"mosaicMethod":"MosaicLockRaster", "lockRasterIds":[32,454,14]} |
| renderingRule | Specifies the rendering rule for how the requested image will be rendered.The syntax of the rendering rule varies based on the specified rasterFunction property value:The following lists the pre-defined raster function names and the corresponding arguments supported:**Aspect**The Aspect raster function takes no arguments. Hence, specifying only the rasterFunction property suffices in this case.**Colormap**The arguments for the Colormap function are either the name of a colormap or the specification of a colormap:* ColorMapName: Random
* Colormap: array of integer arrays; the integer arrays have four values, the first the value and the other three the associated red/green/blue value

**Hillshade**The arguments for the Hillshade function are Azimuth, Altitude and ZFactor (all double values).**NDVI**The arguments for the NDVI function are VisibleBandID and InfraredBandID (all integer values).**ShadedRelief**The arguments for the ShadedRelief function are Azimuth, Altitude and ZFactor (all double values) as well as Colormap (array of integer arrays; the integer arrays have four values, the first the value and the other three the associated red/green/blue value).**Slope**The arguments for the Slope function is ZFactor (double value).**Statistics**The arguments for the Statistics function are Type (Min | Max | Mean | StandardDeviation), KernelColumns and KernelRows (both integers).**Stretch**The arguments for the Stretch function are * StrechType: integer value: 0 = None, 3 = StandardDeviation, 4 = Histogram Equalization, 5 = MinMax
* NumberOfStandardDeviations: integer value
* Statistics: array of array of four doubles (min value, max value, mean value, standard deviation)
* Gamma: array of doubles
 |
| Required | No. |
| Syntax | JSONValidates against http://schemas.opengis.net/gsr-is/1.0/renderingRule.json  |
| Examples | **Aspect:**renderingRule={"rasterFunction" : "Aspect"}**Colormap:**renderingRule={ "rasterFunction" : "Colormap", "rasterFunctionArguments" : { "ColormapName" : "Random" }, "variableName" : "Raster"}renderingRule={ "rasterFunction" : "Colormap", "rasterFunctionArguments" : { "Colormap" : [ [0, 1, 2, 3], [2, 45, 52, 13] ] }, "variableName" : "Raster"}**Hillshade**renderingRule={ "rasterFunction" : "Hillshade", "rasterFunctionArguments" : { "Azimuth" : 215.0, "Altitude" : 75.0, "ZFactor" : 0.3 }, "variableName" : "DEM"}**NDVI**renderingRule={ "rasterFunction" : "NDVI", "rasterFunctionArguments" : { "VisibleBandID" : 2, "InfraredBandID" : 1 }, "variableName" : "Raster"}**ShadedRelief**renderingRule={ "rasterFunction" : "ShadedRelief", "rasterFunctionArguments" : { "Azimuth" : 215.0, "Altitude" : 75.0, "ZFactor" : 0.3, "Colormap" : [ [0, 1, 2, 3], [2, 45, 52, 13] ] }, "variableName" : "Raster"}**Slope**renderingRule={ "rasterFunction" : "Slope", "rasterFunctionArguments" : { "ZFactor" : 0.3 }, "variableName" : "DEM"}**Statistics**renderingRule={ "rasterFunction" : "Statistics", "rasterFunctionArguments" : { "Type" : "Mean", "KernelColumns" : 3, "KernelRows" : 3 }, "variableName" : "Raster"}**Stretch**renderingRule={ "rasterFunction" : "Stretch", "rasterFunctionArguments" : { "StretchType" : 3, "NumberOfStandardDeviations" : 2, "Statistics" : [ [0.2, 222.46, 99.35, 1.64], [5.56, 100.345, 45.4, 3.96], [0, 352.37, 172.284, 2] ], "Gamma" : [1.25, 2, 3.95] }, "variableName" : "Raster"} |

**Request Requirements**

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

imgservice/imgRequest |

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

imgservice/imgParameters |

### Image 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-is/1.0/image.json** or in case of an exception against JSON Schema http://schemas.opengis.net/gsr/1.0/exception.json.

imgservice/imgValid |

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

imgservice/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 Image resource is returned, the bbox SHALL be the adjusted envelope.

imgservice/aspectRatio |

|  |
| --- |
| * + 1. The image SHALL be in the requested format.

imgservice/imageFormat |

### Examples

Export an image with the bounding box [[-117, 34] - [-116, 35]] in the JSON representation:

http://example.com/rest/services/MyImage/ImageServer/exportImage?bbox=-117,34,-116,35&f=json

**Request**

GET /rest/services/MyImage/ImageServer/exportImage?bbox=-117,34,-116,35&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/\_301413356.jpg",

 "width" : 400,

 "height" : 400,

 "extent" : {

 "xmin" : -117,

 "ymin" : 34,

 "xmax" : -116,

 "ymax" : 35,

 "spatialReference" : {

 "wkid" : 4326

 }

 },

 "scale" : 0

}

Export the same image, but directly request the image file.

**Request**

GET /rest/services/MyImage/ImageServer/exportImage?bbox=-117,34,-116,35&f=image HTTP/1.1

Host: example.com

**Response**

HTTP/1.1 200 OK

Content-Type: image/png

Content-Length: nnn

iVBORw0KGgoAAAANSUhEUgAAADIA…

# Export Image 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 7 – Export Image with support for coordinate transformation overview

|  |  |  |
| --- | --- | --- |
| **Resource** | **Parameters** | **Resource representation** |
| Export Image | imageSRbboxSR | - |

## Export Image

### Dependency

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

### Export Image request

Table 8 – Additonal Export Image 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 image service. |
| 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 image service. |
| Syntax | POSINT / JSON |
| Example | bboxSR=4326 |

**Request Requirements**

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

convert/parameters |

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

convert/validSR |

### Image resources

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

convert/imageSR |

### Examples

Export an image with the bounding box [[-117, 34] - [-116, 35]] in the JSON representation, and request that the image uses the Web Mercator projection

http://example.com/rest/services/MyImage/ImageServer/exportImage?bbox=-117,34,-116,35&f=json&bboxSR=4326&imageSR=3857

**Request**

GET /rest/services/MyImage/ImageServer/exportImage?bbox=-117,34,-116,35&f=json&bboxSR=4326&imageSR=3857 HTTP/1.1

Host: example.com

**Response**

HTTP/1.1 200 OK

Content-Type: application/json

Content-Length: nnn

{

 "href" : "http://example.com/output/\_757147.jpg",

 "width" : 400,

 "height" : 400,

 "extent" : {

 "xmin" : -13036260.2363813,

 "ymin" : 4028802.02613441,

 "xmax" : -12901181.1184514,

 "ymax" : 4163881.14406429,

 "spatialReference" : {

 "wkid" : 3857

 }

 },

 "scale" : 0

}

# Export Image with support for time

## Overview

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

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

Table 9 – Export Image with support for time

|  |  |  |
| --- | --- | --- |
| **Resource** | **Parameters** | **Resource representation** |
| Export Image | time | - |

## Image Service Root resources

|  |
| --- |
| * + 1. If an Export Image resource supports exporting Images based on time, the Image Service Root resource SHALL include a timeInfo property.

time/timeInfo |

The timeInfo property provides information such as the time extent supported.

**Example**

...

 "timeInfo" : {

 "timeExtent" : [1106822673000,1125907321000],

 "timeReference" : null

 },

...

## Export Image

### Dependency

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

### Export Image URI

Table 10 – Additonal Export Image parameters

|  |  |
| --- | --- |
| **Parameter** | **Details** |
| time | The time instant or the time extent of the exported 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 image. |
| 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 Export Image resource SHALL support all parameters and values specified in Table 10.

time/parameters |

### Image resources

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

# Identify

## Overview

The Identify operation is performed on a controller resource of the image service to discover content for a given location and a given mosaic rule. The location can be a point or a polygon.

The result of the Identify operation includes the pixel value of the mosaic for a given mosaic rule, a resolution (pixel size), and a set of catalog items that overlap the given geometry. The single pixel value is that of the mosaic at the centroid of the specified location. If there are multiple rasters overlapping the location, the visibility of a raster is determined by the order of the rasters defined in the mosaic rule.

The catalog items that overlap the given geometry are ordered based on the mosaic rule. A list of catalog item visibilities gives the percentage contribution of the item to the overall mosaic.

The identified result is 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 11 – Identify overview

|  |  |  |
| --- | --- | --- |
| **Resource** | **URI parameters** | **Resource representation** |
| Identify | f=jsongeometrygeometryTypemosaicRulepixelSize | JSON representation validAll JSON schema elements supported |

## Identify URI

In the following URI templates, these variables are used:

* imgServiceURI: URL of a Image Service Root resource without any parameter

Table 12 – Identify reference

|  |  |
| --- | --- |
| **URI template** | {+imgServiceURI}/identify(?f, geometry,geometryType, mosaicRule,pixelSize} |
| **HTTP methods** | GETPOST (application/x-www-form-urlencoded) |
| **Parent Resource Type** | Image Service Root |

Table 13 – Identify parameters

|  |  |
| --- | --- |
| **Parameter** | **Details** |
| f | The response format.  |
| Required | Yes |
| Syntax | "json" |
| 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 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.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 |
| Examples | **JSON:**geometryType=GeometryPoint&geometry={x:-104,y:35.6}**Simple syntax for point geometries:**geometryType=GeometryPoint&geometry=-104,35.6 |
| geometryType | The type of geometry specified by the geometry parameter. The well-known geometry types supported by this parameter are points ("GeometryPoint") and polygons ("GeometryPolygon"). See GeoServices REST API – Core, Clause 9 for additional information about these geometry types. |
| Required | No. Default: "GeometryPoint" |
| Syntax | "GeometryPoint" / "GeometryPolygon" |
| Example | geometryType=GeometryPolygon |
| mosaicRule | Specifies the mosaic rule defining the image sorting order. Mosaicking determines which cell value is used in the case of overlapping rasters (the first raster specified, the last raster specified, the minimum value, the maximum value, the mean, or a blend).See Table 6 for a description of the available mosaic rules.  |
| Required | No. Default: MosaicCenter. |
| Syntax | JSON |
| Example | mosaicRule={"mosaicMethod":"MosaicCenter", "sortField":"Category","ascending":false} |
| pixelSize | The pixel level being identified (or the resolution being looked at). I The structure of the pixelSize parameter is the same as the structure of the point object (see GeoServices REST API – Core, subclause 9.3). In addition to the JSON structure, users can specify the pixel size with a simpler comma-separated syntax.  |
| Required | No. Default: the base resolution of the dataset |
| Syntax | JSON / X "," Y |
| Example | pixelSize={"x":0.18,"y":0.18}pixelSize=0.18,0.18 |

**Request Requirements**

|  |
| --- |
| * + 1. The Identify resource SHALL accept requests that conform to the URI template in Table 12 and use any HTTP method identified in the same table.

identify/request |

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

identify/parameters |

## Identified Items resources

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

identify/valid |

## Examples

Identify a single raster image service using a point geometry:

http://example.com/rest/services/Earthquakes/SanAndreasLidar/ImageServer/identify?geometry={"x":575505.5,"y":3733770}&geometryType=GeometryPoint&pixelSize=0.5,0.5&f=json

**Request**

GET /rest/services/Earthquakes/SanAndreasLidar/ImageServer/identify? geometry={"x":575505.5,"y":3733770}&geometryType=GeometryPoint&pixelSize=0.5,0.5&f=json HTTP/1.1

Host: example.com

**Response**

HTTP/1.1 200 OK

Content-Type: application/json

Content-Length: nnn

{

 "objectId" : 0,

 "name" : "Pixel",

 "value" : "-17.5575",

 "location" : {

 "x" : 575505.5,

 "y" : 3733770,

 "spatialReference" : {

 "wkid" : 26911

 }

 },

 "properties" : null,

 "catalogItems" : null,

 "catalogItemVisibilities" : []

}

Identify a mosaicked image service using a polygon geometry and specify the mosaic rule using the MosaicAttribute method:

http://example.com/rest/services/Portland/CascadeLandsat/ImageServer/identify?geometry={"rings":[[[-13555360.4191,5911556.581],

[-13489311.5669,5898227.932],[-13423477.4153,5884426.3329],

[-13602646.9571,5717848.4135],[-13587119.9125,5781976.6214],

[-13571360.1713,5846543.2654],[-13555360.4191,5911556.581]]]}&

geometryType=GeometryPolygon&mosaicRule={"mosaicMethod":"MosaicAttribute","where":"Name+NOT+LIKE+'Ov%25'","sortField":"Name","mosaicOperation":"MT\_MAX"}&f=json

**Request**

GET /rest/services/Portland/CascadeLandsat/ImageServer/identify? geometry={"rings":[[[-13555360.4191,5911556.581],

[-13489311.5669,5898227.932],[-13423477.4153,5884426.3329],

[-13602646.9571,5717848.4135],[-13587119.9125,5781976.6214],

[-13571360.1713,5846543.2654],[-13555360.4191,5911556.581]]]}& geometryType=GeometryPolygon&mosaicRule={"mosaicMethod":"MosaicAttribute","where":"Name+NOT+LIKE+'Ov%25'","sortField":"Name","mosaicOperation":"MT\_MAX"}&f=json HTTP/1.1

Host: example.com

**Response**

HTTP/1.1 200 OK

Content-Type: application/json

Content-Length: nnn

{

 "objectId" : 0,

 "name" : "Pixel",

 "value" : "17, 22, 39, 45",

 "location" :

 {

 "x" : -13527177.6374152,

 "y" : 5837991.41167063,

 "spatialReference" : {

 "wkid" : 54004

 }

 },

 "properties" :

 {

 "Values" : [

 "10 18 34 43",

 "22 27 44 50",

 "22 30 46 55",

 "17 22 39 45"

 ]

 },

 "catalogItems" :

 {

 "objectIdFieldName" : "OBJECTID",

 "spatialReference" : {

 "wkid" : 54004

 },

 "geometryType" : "GeometryPolygon",

 "features" : [

 {

 "geometry" :

 {

 "rings" :

 [

 [

 [-13460551.7089, 5854521.5319],

 [-13478287.1495, 5790460.0595],

 …

 [-13525950.7118, 5869297.4578],

 [-13460551.7089, 5854521.5319]

 ]

 ]

 },

 "attributes" :

 {

 "OBJECTID" : 6,

 "Name" : "p046r028\_7t19990907.met;p046r028\_7t19990907.met",

 "MinPS" : 0,

 "MaxPS" : 28.5,

 "LowPS" : 14.25,

 "HighPS" : 114,

 "Category" : 1,

 "Tag" : "Pansharpened",

 "GroupName" : "p046r028\_7t19990907",

 "ProductName" : "Level1",

 "CenterX" : -13624980.3112093,

 "CenterY" : 5756154.02144619,

 "ZOrder" : null,

 "SOrder" : null,

 "StereoID" : "",

 "SensorName" : "Landsat-7-ETM+",

 "AcquisitionDate" : 936662400000,

 "SunAzimuth" : 150.8831799,

 "SunElevation" : 46.5205819,

 "CloudCover" : 0

 }

 },

 {

 "geometry" :

 {

 "rings" :

 [

 [

 [-13555360.4191, 5911556.581],

 [-13489311.5669, 5898227.932],

 …

 [-13571360.1713, 5846543.2654],

 [-13555360.4191, 5911556.581]

 ]

 ]

 },

 "attributes" :

 {

 "OBJECTID" : 1,

 "Name" : "p045r028\_7t19991002.met",

 "MinPS" : 28.5,

 "MaxPS" : 342,

 "LowPS" : 28.5,

 "HighPS" : 114,

 "Category" : 1,

 "Tag" : "MS",

 "GroupName" : "p045r028\_7t19991002",

 "ProductName" : "Level1",

 "CenterX" : -13456998.9817332,

 "CenterY" : 5756986.51347787,

 "ZOrder" : null,

 "SOrder" : null,

 "StereoID" : "",

 "SensorName" : "Landsat-7-ETM+",

 "AcquisitionDate" : 938822400000,

 "SunAzimuth" : 157.6031865,

 "SunElevation" : 37.975699,

 "CloudCover" : 50

 }

 },

 {

 "geometry" :

 {

 "rings" :

 [

 [

 [-13723464.45, 5910823.3172],

 [-13657416.2669, 5897455.1875],

 …

 [-13739408.9372, 5845800.1358],

 [-13723464.45, 5910823.3172]

 ]

 ]

 },

 "attributes" :

 {

 "OBJECTID" : 5,

 "Name" : "p046r028\_7t19990907.met",

 "MinPS" : 28.5,

 "MaxPS" : 342,

 "LowPS" : 28.5,

 "HighPS" : 114,

 "Category" : 1,

 "Tag" : "MS",

 "GroupName" : "p046r028\_7t19990907",

 "ProductName" : "Level1",

 "CenterX" : -13624980.3112093,

 "CenterY" : 5756154.02144619,

 "ZOrder" : null,

 "SOrder" : null,

 "StereoID" : "",

 "SensorName" : "Landsat-7-ETM+",

 "AcquisitionDate" : 936662400000,

 "SunAzimuth" : 150.8831799,

 "SunElevation" : 46.5205819,

 "CloudCover" : 0

 }

 },

 {

 "geometry" :

 {

 "rings" :

 [

 [

 [-13292489.9099, 5855431.779],

 [-13310286.7337, 5791381.4753],

 …

 [-13357867.1993, 5870158.6064],

 [-13292489.9099, 5855431.779]

 ]

 ]

 },

 "attributes" :

 {

 "OBJECTID" : 2,

 "Name" : "p045r028\_7t19991002.met;p045r028\_7t19991002.met",

 "MinPS" : 0,

 "MaxPS" : 28.5,

 "LowPS" : 14.25,

 "HighPS" : 114,

 "Category" : 1,

 "Tag" : "Pansharpened",

 "GroupName" : "p045r028\_7t19991002",

 "ProductName" : "Level1",

 "CenterX" : -13456998.9817332,

 "CenterY" : 5756986.51347787,

 "ZOrder" : null,

 "SOrder" : null,

 "StereoID" : "",

 "SensorName" : "Landsat-7-ETM+",

 "AcquisitionDate" : 938822400000,

 "SunAzimuth" : 157.6031865,

 "SunElevation" : 37.975699,

 "CloudCover" : 50

 }

 }

 ]

 },

 "catalogItemVisibilities" : [

 0.671180049953907,

 0.328819950035319,

 0,

 0

 ]

}

# Query

## Overview

The Query operation is performed on a controller resource of the image service.

It queries a raster catalog by applying the filter specified by the user. The result of this operation is either a set of features in the raster catalog or an array of raster IDs (if returnIdsOnly is set to true).

The result is not stored on the server and is returned in the response to the request.

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

Table 14 – Query overview

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

## Query URI

In the following URI templates, these variables are used:

* imgServiceURI: URL of a Image Service Root resource without any parameter

Table 15 – Query reference

|  |  |
| --- | --- |
| **URI template** | {+imgServiceURI}/query{?f,objectIds,where,geometry,geometryType,returnGeometry,inSR,outSR,spatialRel,outFields,returnIdsOnly} |
| **HTTP methods** | GETPOST (application/x-www-form-urlencoded) |
| **Parent Resource** | Image Service Root |

Table 16 – Query parameters

|  |  |
| --- | --- |
| **Parameter** | **Details** |
| f | The response format.  |
| Required | Yes |
| Syntax | "json" |
| Example | f=json |
| objectIds | The object IDs of the raster catalog to be queried. When this parameter is specified, * any other filter parameters (including where) are ignored,
* setting returnIdsOnly=true is invalid.
 |
| Required | No. |
| Syntax | POSINT \*("," POSINT) |
| Example | objectIds=5,6 |
| where | A WHERE clause for the query filter. Any legal SQL WHERE clause operating on the fields in the raster catalog is allowed. |
| Required | No. Default: no filter |
| Syntax | STRING |
| Example | where=CLOUDCOVER < 50 |
| geometry | The geometry to query 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 |
| returnGeometry | If true, the result set includes the geometry associated with each result.If the outFields parameter is set to the wildcard "\*", it implies returnGeometry=true, and setting returnGeometry to false has no effect. |
| Required | No. Default: "true". |
| Syntax | BOOLEAN |
| Example | returnGeometry=false |
| inSR | The spatial reference of the input geometry. The spatial reference is specified as either a well-known ID (WKID) or a spatial reference JSON object. See Geoservices REST API Core standard for more requirements related to spatial references. |
| Required | No. Default: The geometry is assumed to be in the spatial reference of the map. |
| Syntax | POSINT / JSON |
| Example | sr=4326 |
| outSR | The spatial reference of the output geometry. The spatial reference is specified as either a well-known ID (WKID) or a spatial reference JSON object. See Geoservices REST API Core standard for more requirements related to spatial references. |
| Required | No. Default: The output geometry is exported in the spatial reference of the map. |
| Syntax | POSINT / JSON |
| Example | sr=4326 |
| spatialRel | The spatial relationship to be applied on the input geometry while performing the query. The supported spatial relationships include intersects, contains, envelope intersects, and within.Pre-defined values: SpatialRelIntersects: Returns a feature if any spatial relationship is found. Applies to all shape type combinations.SpatialRelContains: Returns a feature if its shape is wholly contained within the search geometry. Valid for all shape type combinations.SpatialRelCrosses: Returns a feature if the intersection of the interiors of the two shapes is not empty and has a lower dimension than the maximum dimension of the two shapes. Two lines that share an endpoint in common do not cross. Valid for line/line, line/area, multipoint/area, and multipoint/line shape type combinations.SpatialRelEnvelopeIntersects: Returns a feature if the envelope of the two shapes intersects.SpatialRelIndexIntersects: Returns a feature if the envelope of the query geometry intersects the index entry for the target geometry.SpatialRelOverlaps: Returns a feature if the intersection of the two shapes results in an object of the same dimension but different from both of the shapes. Applies to area/area, line/line, and multipoint/multipoint shape type combinations.SpatialRelTouches: Returns a feature if the two shapes share a common boundary. However, the intersection of the interiors of the two shapes must be empty. In the point/line case, the point may touch an endpoint only of the line. Applies to all combinations except point/point.SpatialRelWithin: Returns a feature if its shape wholly contains the search geometry. Valid for all shape type combinations. |
| Required | No. Default: "SpatialRelIntersects" |
| Syntax | "SpatialRelIntersects" / "SpatialRelContains" / "SpatialRelCrosses" / "SpatialRelEnvelopeIntersects" / "SpatialRelIndexIntersects" / "SpatialRelOverlaps" / "SpatialRelTouches" / "SpatialRelWithin"  |
| Example | spatialRel=SpatialRelContains |
| 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. Note that the wildcard also implicitly implies returnGeometry=true and setting returnGeometry to false will have no effect. |
| 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 15 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 16.

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 |

## Examples

Query using a WHERE clause and return specific output fields of the raster features:

http://example.com/rest/services/Landsat/ImageServer/query?where=Name LIKE 'p045r028%' AND Name NOT LIKE 'Ovr%'&

outFields=Name,MinPS,MaxPS,LowPS,HighPS&returnGeometry=true&returnIdsOnly=false&f=json

**Request**

GET rest/services/Landsat/ImageServer/query? where=Name LIKE 'p045r028%' AND Name NOT LIKE 'Ovr%'&outFields=Name,MinPS,MaxPS,LowPS,HighPS&

returnGeometry=true&returnIdsOnly=false&f=json HTTP/1.1

Host: example.com

**Response**

HTTP/1.1 200 OK

Content-Type: application/json

Content-Length: nnn

{

 "objectIdFieldName" : "OBJECTID",

 "geometryType" : "GeometryPolygon",

 "fields" : [

 {

 "name" : "OBJECTID",

 "type" : "FieldTypeOID",

 "alias" : "OBJECTID"},

 {

 "name" : "Name",

 "type" : "FieldTypeString",

 "alias" : "Name",

 "length" : 50},

 {

 "name" : "MinPS",

 "type" : "FieldTypeDouble",

 "alias" : "MinPS"},

 {

 "name" : "MaxPS",

 "type" : "FieldTypeDouble",

 "alias" : "MaxPS"},

 {

 "name" : "LowPS",

 "type" : "FieldTypeDouble",

 "alias" : "LowPS"},

 {

 "name" : "HighPS",

 "type" : "FieldTypeDouble",

 "alias" : "HighPS"},

 {

 "name" : "Shape",

 "type" : "FieldTypeGeometry",

 "alias" : "Shape"}

 ],

 "features" : [

 {

 "geometry" :

 {

 "rings" :

 [

 [

 [-13555360.4191, 5911556.581],

 [-13489311.5669, 5898227.932],

 [-13423477.4153, 5884426.3329],

 [-13357867.1993, 5870158.6064],

 [-13292489.9099, 5855431.779],

 [-13310286.7337, 5791381.4753],

 [-13327810.7467, 5727748.1244],

 [-13345069.9061, 5664524.4905],

 [-13362071.9117, 5601703.4638],

 [-13425742.0169, 5615460.9307],

 [-13489617.0175, 5628791.9699],

 [-13553688.6321, 5641690.4875],

 [-13617948.3761, 5654150.574],

 [-13602646.9571, 5717848.4135],

 [-13587119.9125, 5781976.6214],

 [-13571360.1713, 5846543.2654],

 [-13555360.4191, 5911556.581]

 ]

 ]

 },

 "attributes" :

 {

 "OBJECTID" : 1,

 "Name" : "p045r028\_7t19991002.met",

 "MinPS" : 28.5,

 "MaxPS" : 342,

 "LowPS" : 28.5,

 "HighPS" : 114

 }

 },

 {

 "geometry" :

 {

 "rings" :

 [

 [

 [-13292489.9099, 5855431.779],

 [-13310286.7337, 5791381.4753],

 [-13327810.7467, 5727748.1244],

 [-13345069.9061, 5664524.4905],

 [-13362071.9117, 5601703.4638],

 [-13425742.0169, 5615460.9307],

 [-13489617.0175, 5628791.9699],

 [-13553688.6321, 5641690.4875],

 [-13617948.3761, 5654150.574],

 [-13602646.9571, 5717848.4135],

 [-13587119.9125, 5781976.6214],

 [-13571360.1713, 5846543.2654],

 [-13555360.4191, 5911556.581],

 [-13489311.5669, 5898227.932],

 [-13423477.4153, 5884426.3329],

 [-13357867.1993, 5870158.6064],

 [-13292489.9099, 5855431.779]

 ]

 ]

 },

 "attributes" :

 {

 "OBJECTID" : 2,

 "Name" : "p045r028\_7t19991002.met;p045r028\_7t19991002.met",

 "MinPS" : 0,

 "MaxPS" : 28.5,

 "LowPS" : 14.25,

 "HighPS" : 114

 }

 }

 ]

}

Query using a point geometry and a where clause. Return only OBJECTIDs:

http://example.com/rest/services/Landsat/ImageServer/query?where=NAME+NOT+LIKE+'Ov\_%25'&geometry={"x":-122.895114,"y":45.558214,"spatialReference": {"wkid":4269}}&geometryType=GeometryPoint&inSR=4326&spatialRel=SpatialRelIntersects&outFields=\*&returnGeometry=false&outSR=&returnIdsOnly=true&f=json

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

**Request**

GET /rest/services/Landsat/ImageServer/query?where=NAME+NOT+LIKE+'Ov\_%25'&geometry={"x":-122.895114,"y":45.558214,"spatialReference":{"wkid":4269}}& geometryType=GeometryPoint&inSR=4326&spatialRel=SpatialRelIntersects&outFields=\*&returnGeometry=false&outSR=&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" : [5,6]

}

# 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 17 – 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 11.

### Query request

Table 18 – 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 18.

querytime/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.

# Raster Catalog

## Overview

This conformance class adds support for querying the raster catalog and downloading raw raster files.

A raster catalog is a collection of raster datasets defined in a table.

Table 19 – Raster Catalog overview

|  |  |  |
| --- | --- | --- |
| **Resource** | **URI parameters** | **Resource representation** |
| Raster Catalog Item  | f=json | JSON representation validAll JSON schema elements supported |
| Raster Image | f=jsonf=imagebboxsizeimageSRbboxSRformatpixelTypenoDatainterpolationcompressionQuality | JSON representation validAll JSON schema elements supported |
| Raster Thumbnail | - | Thumbnail image, format determined by service |
| Raster Info  | f=json | JSON representation validAll JSON schema elements supported |
| Download Rasters | f=jsonrasterIdsgeometrygeometryTypeformat | JSON representation validAll JSON schema elements supported |
| Raster File  | id | Raster file, format determined by service |

## Raster Catalog Item

### Overview

The Raster Catalog Item resource represents one record, or feature, in the raster catalog. Each such feature has an associated raster.

The ObjectID of the raster catalog item is the same as the ID of the associated raster (the raster ID). The attributes of the raster catalog item are the attributes of the raster. The geometry of the raster catalog item is the footprint of the raster.

The Raster Catalog Item resource has three child resources:

■ Raster Image: Returns a composite image of the associated raster

■ Raster Thumbnail: Returns a thumbnail for the associated raster

■ Raster Info: Returns the info for the associated raster (such as its width, height, number of bands, and pixel type)

### Raster Catalog Item URI

In the following URI templates, these variables are used:

* imgServiceURI: URL of a Image Service Root resource without any parameter
* rasterId: identifier of the raster catalog item

Table 20 – Raster Catalog Item reference

|  |  |
| --- | --- |
| **URI template** | {+imgServiceURI}/{+rasterId}(?f} |
| **HTTP methods** | GETPOST (application/x-www-form-urlencoded) |
| **Parent Resource Type** | Image Service Root |
| **Child Resource Types** | Raster ImageRaster ThumbnailRaster Info |

Table 21 – Raster Catalog Item parameters

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

**Request Requirements**

|  |
| --- |
| * + 1. The Raster Catalog Item resource SHALL accept requests that conform to the URI template in Table 20 and use any HTTP method identified in the same table.

catalog/catItemRequest |

|  |
| --- |
| * + 1. The Download Rasters resource SHALL support all parameters and values specified in Table 21.

catalog/catItemParameters |

### Raster Catalog Item resources

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

catalog/catItemValid |

### Example

Return the raster catalog item for raster ID 8:

http://example.com/rest/services/Landsat/ImageServer/8?f=json

**Request**

GET /rest/services/Landsat/ImageServer/8?f=json HTTP/1.1

Host: example.com

**Response**

HTTP/1.1 200 OK

Content-Type: application/json

Content-Length: nnn

{

 "geometry" : {

 "rings" : [

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

 ]

 },

 "attributes" : {

 "IMAGEID" : 37,

 "OWNER" : "Joe Smith",

 "VALUE" : 94820.37,

 "APPROVED" : true,

 "LASTUPDATE" : 1227663551096

 }

}

## Raster Image

### Overview

The Raster Image resource returns a composite image for a single raster catalog item. This resource can also provide information about the exported image, such as its URL, width, height, and extent.

Apart from the usual JSON response format, users can also request a format called image. When this format is requested, the server responds by directly streaming the image bytes to the client. No other information about the image is returned.

### Raster Catalog Item URI

In the following URI templates, these variables are used:

* imgServiceURI: URL of a Image Service Root resource without any parameter
* rasterId: identifier of the raster catalog item

Table 23 – Raster Image reference

|  |  |
| --- | --- |
| **URI template** | {+imgServiceURI}/{+rasterId}/image(?f,bbox,size,imageSR,bboxSR,format, pixelType,noData,interpolation,compressionQuality} |
| **HTTP methods** | GETPOST (application/x-www-form-urlencoded) |
| **Parent Resource Type** | Raster Catalog Item |

Table 24 – Raster Image parameters

|  |  |
| --- | --- |
| **Parameter** | **Details** |
| f | The response format.  |
| Required | Yes |
| Syntax | "json" / "image" |
| Example | f=json |
| bbox | The extent (bounding box) of the exported image.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 |
| format | The format of the exported image.Pre-defined values:* png: image as specified by media type "image/png"
* png8: image as specified by media type "image/png" with 8-bit color depth
* png24: image as specified by media type "image/png" with 24-bit color depth
* jpg: image as specified by media type "image/jpeg
* bmp: file according to the Device Independent Bitmap (DIB) file format as specified by Microsoft
* gif: image as specified by media type "image/gif"
* tiff: image as specified by media type "image/tiff"
* jpgpng: uses "jpg", if there are no transparent pixels in the requested extent; otherwise "png" is used
 |
| Required | No. Default: "jpgpng" |
| Syntax | "jpgpng" / "png" / "png8" / "png24" / "jpg" / "bmp" / "gif" / "tiff"  |
| Example | format=tiff |
| pixelType | The pixel type, also known as data type, pertains to the type of values stored in the raster, such as characters ("C"), signed integer ("S"), unsigned integer ("U"), or floating point ("F"). The number is the number of bits. Integers are whole numbers, whereas floating points have decimals. |
| Required | No. Default: The value specified in the Image Service Root resource is used. |
| Syntax | "C128" / "C64" / "F32" / "F64" / "S16" / "S32" / "S8" / "U1" / "U16" / "U2" / "U32" / "U4" / "U8" / "UNKNOWN" |
| Example | pixelType=U8 |
| noData | The pixel value representing "no information".In the case of multiple bands, the value applies to all bands. |
| Required | No. Default: no value is treated as "no information". |
| Syntax | NUMBER / STRING |
| Example | noData=0 |
| interpolation | The resampling process of extrapolating the pixel values while transforming the raster dataset, e.g. when it undergoes warping or changes the coordinate reference system.Pre-defined values:* RSP\_BilinearInterpolation: resample pixel by bilinear interpolation
* RSP\_CubicConvolution: resample pixel by cubic convolution
* RSP\_Majority: resample pixel by majority value
* RSP\_NearestNeighbor: resample pixel by nearest neighbor
 |
| Required | No. Default: RSP\_NearestNeighbor. |
| Syntax | "RSP\_BilinearInterpolation" / "RSP\_CubicConvolution" / "RSP\_Majority" / "RSP\_NearestNeighbor" |
| Example | noData=0 |
| compressionQuality | Controls how much loss the image will be subjected to by the compression algorithm. Valid value ranges of compression quality are from 0 to 100. Larger numbers mean less compression occurs, resulting in higher-quality images. |
| Required | No. |
| Syntax | INTEGER |
| Example | compressionQuality=75 |
| 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 image service. |
| 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 image service. |
| Syntax | POSINT / JSON |
| Example | bboxSR=4326 |

**Request Requirements**

|  |
| --- |
| * + 1. The Raster Image resource SHALL accept requests that conform to the URI template in Table 23 and use any HTTP method identified in the same table.

catalog/rasterImgRequest |

|  |
| --- |
| * + 1. The Raster Image resource SHALL support all parameters and values specified in Table 24.

catalog/rasterImgParameters |

### Raster Image resources

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

catalog/rasterImgValid |

### Example

Return the raster image for raster ID 8 in PNG 8-bit format:

http://example.com/rest/services/Landsat/ImageServer/8/image?bbox=378501.375,4825171.125,634687.875,5051974.125&f=json&format=png8

**Request**

GET /rest/services/Landsat/ImageServer/8/image?bbox=378501.375,4825171.125,634687.875,5051974.125&f=json&format=png8 HTTP/1.1

Host: example.com

**Response**

HTTP/1.1 200 OK

Content-Type: application/json

Content-Length: nnn

{

 "href" : "http://example.com/output/\_336971124.png",

 "width" : 400,

 "height" : 400,

 "extent" : {

 "xmin" : 7585040,

 "ymin" : 695086,

 "xmax" : 7590710,

 "ymax" : 700756,

 "spatialReference" : {

 "wkt" : "PROJCS[\"NAD\_1983\_HARN\_StatePlane\_Oregon\_North\_FIPS\_3601\",GEOGCS[\"GCS\_North\_American\_1983\_HARN\",DATUM[\"D\_unknown\",SPHEROID[\"North\_American\_1983\_HARN\",6378137.0,298.257222101]],PRIMEM[\"Greenwich\",0.0],UNIT[\"Degree\",0.0174532925199433]],PROJECTION[\"Lambert\_Conformal\_Conic\"],PARAMETER[\"false\_easting\",8202099.737532808],PARAMETER[\"false\_northing\",0.0], PARAMETER[\"central\_meridian\",-120.5],PARAMETER[\"standard\_parallel\_1\", 44.33333333333334],PARAMETER[\"standard\_parallel\_2\",46.0],PARAMETER[\"latitude\_of\_origin\",43.66666666666666],UNIT[\"Foot\",0.3048]]"

 }

 },

 "scale" : 0

}

## Raster Thumbnail

### Overview

The Raster Thumbnail resource returns a reduced-size thumbnail image for a single raster catalog item. This resource streams the thumbnail contents to the client.

### Raster Thumbnail URI

In the following URI templates, these variables are used:

* imgServiceURI: URL of a Image Service Root resource without any parameter
* rasterId: identifier of the raster catalog item

Table 26 – Raster Thumbnail reference

|  |  |
| --- | --- |
| **URI template** | {+imgServiceURI}/{+rasterId}/thumbnail |
| **HTTP methods** | GET |
| **Parent Resource Type** | Raster Catalog Item |

**Request Requirements**

|  |
| --- |
| * + 1. The Raster Thumbnail resource SHALL accept requests that conform to the URI template in Table 26 and use any HTTP method identified in the same table.

catalog/rasterThumbRequest |

### Example

Return the raster thumbnail for raster ID 8:

http://example.com/rest/services/Landsat/ImageServer/8/thumbnail

**Request**

GET /rest/services/Landsat/ImageServer/8/thumbnail HTTP/1.1

Host: example.com

**Response**

HTTP/1.1 200 OK

Content-Type: image/png

Content-Length: nnn

…

## Raster Info

### Overview

The Raster Info resource returns information about the associated raster (such as its width, height, number of bands, and pixel type).

### Raster Info URI

In the following URI templates, these variables are used:

* imgServiceURI: URL of a Image Service Root resource without any parameter
* rasterId: identifier of the raster catalog item

Table 28 – Raster Catalog Item reference

|  |  |
| --- | --- |
| **URI template** | {+imgServiceURI}/{+rasterId}/info(?f} |
| **HTTP methods** | GETPOST (application/x-www-form-urlencoded) |
| **Parent Resource Type** | Raster Catalog Item |
| **Child Resource Types** | Raster File |

Table 29 – Raster Info parameters

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

**Request Requirements**

|  |
| --- |
| * + 1. The Raster Info resource SHALL accept requests that conform to the URI template in Table 20 and use any HTTP method identified in the same table.

catalog/rasterInfoRequest |

|  |
| --- |
| * + 1. The Raster Info resource SHALL support all parameters and values specified in Table 21.

catalog/rasterInfoParameters |

### Raster Info resources

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

catalog/rasterInfoValid |

### Example

Return the raster info for raster ID 8:

http://example.com/rest/services/Landsat/ImageServer/8/info?f=json

**Request**

GET /rest/services/Landsat/ImageServer/8/info?f=json HTTP/1.1

Host: example.com

**Response**

HTTP/1.1 200 OK

Content-Type: application/json

Content-Length: nnn

{

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

 "blockWidth": 2726,

 "blockHeight": 1,

 "pixelSizeX": 30.386,

 "pixelSizeY": 30.386,

 "extent" : {"xmin" : -119.56, "ymin" : 33.54, "xmax" : -117.37, "ymax" : 36.71},

 "bandCount": 3,

 "pixelType": "U8",

 "firstPyramidLevel": 1,

 "maxPyramidLevel": 9

}

## Download Rasters

### Overview

The Download Rasters operation is performed on a controller resource of the image service and returns information (the file id) that can be used to download the raw raster files that are associated with a specified set of rasters in the raster catalog.

The file ids returned by this operation can be used to download individual Raster File resources.

Users can provide arguments to the Download Rasters operation as query parameters.

### Download Rasters URI

In the following URI templates, these variables are used:

* imgServiceURI: URL of a Image Service Root resource without any parameter

Table 31 – Download Rasters reference

|  |  |
| --- | --- |
| **URI template** | {+imgServiceURI}/download(?f, geometry,geometryType,rasterIds,format} |
| **HTTP methods** | GETPOST (application/x-www-form-urlencoded) |
| **Parent Resource Type** | Image Service Root |

Table 32 – Download Rasters parameters

|  |  |
| --- | --- |
| **Parameter** | **Details** |
| f | The response format.  |
| Required | Yes |
| Syntax | "json" |
| Example | f=json |
| geometry | The geometry to apply for clipping. The type of the geometry is specified by the geometryType parameter. In addition to the JSON structures, for 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 envelopes:**When using envelopes, the geometries may alternatively be specified with a simpler comma-separated syntax.NOTE Coordinates always use a period as the decimal separator even in countries where a comma is traditionally used. |
| Required | No. Default: No clipping. |
| Syntax | JSON / XMIN "," YMIN "," XMAX "," YMAX |
| Examples | **JSON:**geometryType=GeometryEnvelope&geometry= {"xmin":-104,"ymin":35,"xmax":-94,"ymax":41}**Simple syntax for envelopes:**geometryType=GeometryEnvelope&geometry=-104,35,-94,41 |
| geometryType | The type of geometry specified by the geometry parameter. The well-known geometry types supported by this parameter are envelopes ("GeometryEnvelope") and polygons ("GeometryPolygon"). See GeoServices REST API – Core, Clause 9 for additional information about these geometry types. |
| Required | No. Default: "GeometryEnvelope" |
| Syntax | "GeometryEnvelope" / "GeometryPolygon" |
| Example | geometryType=GeometryPolygon |
| rasterIds | A comma separated list of raster IDs whose files are to be downloaded. |
| Required | Yes |
| Syntax | JSON |
| Example | rasterIds=5,6,10,11,12 |
| format | The format of the rasters returned. The format applies only, if the clip geometry is also specified and the format will be honored only when the raster is clipped. Pre-defined formats are: * TIFF: image as specified by media type image/tiff
* Imagine: file according to the "IMAGINE" format
* JPEG: image as specified by media type image/jpeg
* BIL: file according to the format "Band Interleaved by Line"
* BSQ: file according to the format "Band Sequential"
* BIP: file according to the format "Band Interleaved by Pixel"
* ENVI: file according to the "ENVI-Header-Format"
* JP2: image as specified by media type image/jp2
* GIF: image as specified by media type image/gif
* BMP: file according to the Device Independent Bitmap (DIB) file format as specified by Microsoft
* PNG: image as specified by media type image/png
 |
| Required | No. Default: the rasters will be in their native format.  |
| Syntax | STRING |
| Example | format=TIFF |

**Request Requirements**

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

catalog/downloadRequest |

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

catalog/downloadParameters |

### Raster Files resources

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

catalog/downloadValid |

### Example

Download raster in TIFF output format, clipped to a specified envelope:

http://example.com/rest/services/MyImage/ImageServer/download?

rasterIds=5,6&

geometry={"xmin":-1949594.8286481365,"ymin":882737.0181116117,

"xmax":-1946926.2791246006,"ymax":884828.2021675818, "spatialReference":{"wkid":102009}}&geometryType=GeometryEnvelope& format=TIFF&f=json

**Request**

GET /rest/services/MyImage/ImageServer/download?rasterIds=5,6&

geometry={"xmin":-1949594.8286481365,"ymin":882737.0181116117,

"xmax":-1946926.2791246006,"ymax":884828.2021675818, "spatialReference":{"wkid":102009}}&geometryType=GeometryEnvelope& format=TIFF&f=json HTTP/1.1

Host: example.com

**Response**

HTTP/1.1 200 OK

Content-Type: application/json

Content-Length: nnn

{

"rasterFiles" : [

 {

 "id" : "http://example.com/output/\_507978500/md/data/2w21w\_5\_s6c.tif",

 "size" : 390431,

 "rasterIds" : [

 5

 ]

 },

 {

 "id" : "http://example.com/output/\_507978500/md/data/2w21w\_5\_s6c.tfw",

 "size" : 90,

 "rasterIds" : [

 5

 ]

 },

 {

 "id" : "http://example.com/data/2w22w.jpg",

 "size" : 1913965,

 "rasterIds" : [

 6

 ]

 },

 {

 "id" : "http://example.com/data/2w22w.aux",

 "size" : 18049,

 "rasterIds" : [

 6

 ]

 },

 {

 "id" : "http://example.com/data/2w22w.rrd",

 "size" : 2339130,

 "rasterIds" : [

 6

 ]

 }

 ]

}

## Raster File

### Overview

The Raster File resource represents a single raw raster file. The ID required to request the file can be obtained by using the Download Rasters operation.

This resource streams the file contents to the client.

### Raster File URI

In the following URI templates, these variables are used:

* imgServiceURI: URL of a Image Service Root resource without any parameter

Table 34 – Raster File reference

|  |  |
| --- | --- |
| **URI template** | {+imgServiceURI}/file(?id} |
| **HTTP methods** | GETPOST (application/x-www-form-urlencoded) |
| **Parent Resource Type** | Image Service Root |

Table 35 – Raster File parameters

|  |  |
| --- | --- |
| **Parameter** | **Details** |
| id | The ID of the raster file. This ID is obtained by using the Download Rasters operation (the IDs are returned in the "id" properties). |
| Required | Yes |
| Syntax | STRING |
| Example | id=t1923.pgw |

**Request Requirements**

|  |
| --- |
| * + 1. The Raster File resource SHALL accept requests that conform to the URI template in Table 34 and use any HTTP method identified in the same table.

catalog/rasterFileRequest |

|  |
| --- |
| * + 1. The Raster File resource SHALL support all parameters and values specified in Table 35.

catalog/rasterFileParameters |

### Example

Access a raster file resource for ID 1n2w13w.jpg:

http://example.com/rest/services/Landsat/ImageServer/file?id=1n2w13w.jpg

**Request**

GET /rest/services/Landsat/ImageServer/file?id=1n2w13w.jpg HTTP/1.1

Host: example.com

**Response**

HTTP/1.1 200 OK

Content-Type: image/jpeg

Content-Length: nnn

…

Annex A
(normative)

Abstract Test Suite

Conformance class: imgservice

* 1. Test: imgservice/root

|  |  |
| --- | --- |
| Requirements | **imgservice/request, imgservice/parameters, imgservice/valid, imgservice/spatialReference, imgservice/extent, imgservice/bandCount, imgservice/pixelType, imgservice/fields** |
| Test purpose | Verify that the Image Service Root resource supports the request and response requirements. |
| Test method | Set up a test service. Construct valid requests for the Image Service Root resource.Inspect the responses and validate them against the JSON Schema http://schemas.opengis.net/gsr-is/1.0/root.json or for exceptions against http://schemas.opengis.net/gsr/1.0/exception.json. Inspect the Image Service Root resource and verify the correctness of the content. |
| Test type | Capability |

* 1. Test: imgservice/exportImage

|  |  |
| --- | --- |
| Requirements | **imgservice/imgRequest, imgservice/imgParameters, imgservice/imgValid, imgservice/imageSize, imgservice/aspectRatio, imgservice/imageFormat** |
| Test purpose | Verify that the Image Service Root resource supports the request and response requirements. |
| Test method | Set up a test service. Construct requests for the Export Image resource.Inspect the responses and validate them against the JSON Schema http://schemas.opengis.net/gsr-is/1.0/image.json or for exceptions against http://schemas.opengis.net/gsr/1.0/exception.json. Inspect the Image resource and verify the correctness of the content. |
| Test type | Capability |

Conformance class: convert

* 1. Test: convert/exportImage

|  |  |
| --- | --- |
| Requirements | **convert/parameters, convert/validSR, convert/imageSR** |
| Test purpose | Verify that the Export Image resource supports the request and response requirements. |
| Test method | Set up a test service. Construct requests for the Export Image resource with the additional parameters.Inspect the responses and validate them against the JSON Schema http://schemas.opengis.net/gsr-is/1.0/image.json or for exceptions against http://schemas.opengis.net/gsr/1.0/exception.json. Inspect the Image resource and verify the correctness of the content. |
| Test type | Capability |

Conformance class: time

* 1. Test: time/timeInfo

|  |  |
| --- | --- |
| Requirements | **time/timeInfo** |
| Test purpose | Verify that the Image Service Root resource supports the timeInfo property. |
| Test method | Set up a test service that supports time. Construct valid requests for the Image Service Root resource.Inspect the Image Service Root resource and verify the correctness of the timeInfo property. |
| Test type | Capability |

* 1. Test: time/exportImage

|  |  |
| --- | --- |
| Requirements | **time/parameters** |
| Test purpose | Verify that the Export Image resource supports the request and response requirements. |
| Test method | Set up a test service. Construct requests for the Export Image resource with the additional time parameter.Inspect the responses and validate them against the JSON Schema http://schemas.opengis.net/gsr-is/1.0/image.json or for exceptions against http://schemas.opengis.net/gsr/1.0/exception.json. Inspect the Image resource and verify the correctness of the content. |
| Test type | Capability |

Conformance class: identify

* 1. Test: identify/identify

|  |  |
| --- | --- |
| Requirements | **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. Construct requests for the Identify resource.Inspect the responses and validate them against the JSON Schema http://schemas.opengis.net/gsr-is/1.0/identifiedItems.json or for exceptions against http://schemas.opengis.net/gsr/1.0/exception.json.  |
| Test type | Capability |

Conformance class: query

* 1. Test: query/query

|  |  |
| --- | --- |
| Requirements | **query/request, query/parameters, query/valid** |
| Test purpose | Verify that the Query resource supports the request and response requirements. |
| Test method | Set up a test service. Construct requests for the Query resource.Inspect the responses and validate them against the JSON Schema http://schemas.opengis.net/gsr/1.0/feature(Id)Set.json or for exceptions against http://schemas.opengis.net/gsr/1.0/exception.json.  |
| Test type | Capability |

Conformance class: querytime

* 1. Test: querytime/query

|  |  |
| --- | --- |
| Requirements | **querytime/parameters** |
| Test purpose | Verify that the Query resource supports the request and response requirements. |
| Test method | Set up a test service. Construct requests for the Query resource with the additional time parameter.Inspect the responses and validate them against the JSON Schema http://schemas.opengis.net/gsr/1.0/feature(Id)Set.json or for exceptions against http://schemas.opengis.net/gsr/1.0/exception.json.  |
| Test type | Capability |

Conformance class: catalog

* 1. Test: catalog/catItem

|  |  |
| --- | --- |
| Requirements | **catalog/catItemRequest, catalog/catItemParameters, catalog/catItemValid** |
| Test purpose | Verify that the Raster Catalog Item resource supports the request and response requirements. |
| Test method | Set up a test service. Construct requests for the Raster Catalog Item resources.Inspect the responses and validate them against the JSON Schema http://schemas.opengis.net/gsr/1.0/feature.json or for exceptions against http://schemas.opengis.net/gsr/1.0/exception.json.  |
| Test type | Capability |

* 1. Test: catalog/rasterImg

|  |  |
| --- | --- |
| Requirements | **catalog/rasterImgRequest, catalog/rasterImgParameters, catalog/rasterImgValid** |
| Test purpose | Verify that the Raster Image resource supports the request and response requirements. |
| Test method | Set up a test service. Construct requests for the Raster Image resources.Inspect the responses and validate them against the JSON Schema http://schemas.opengis.net/gsr-is/1.0/image.json or for exceptions against http://schemas.opengis.net/gsr/1.0/exception.json.  |
| Test type | Capability |

* 1. Test: catalog/rasterThumb

|  |  |
| --- | --- |
| Requirements | **catalog/rasterThumbRequest** |
| Test purpose | Verify that the Raster Thumbnail resource supports the request and response requirements. |
| Test method | Set up a test service. Construct requests for the Raster Thumbnail resources and verify that a response is received.  |
| Test type | Capability |

* 1. Test: catalog/rasterInfo

|  |  |
| --- | --- |
| Requirements | **catalog/rasterInfoRequest, catalog/rasterInfoParameters, catalog/rasterInfoValid** |
| Test purpose | Verify that the Raster Info resource supports the request and response requirements. |
| Test method | Set up a test service. Construct requests for the Raster Info resources.Inspect the responses and validate them against the JSON Schema http://schemas.opengis.net/gsr-is/1.0/rasterInfo.json or for exceptions against http://schemas.opengis.net/gsr/1.0/exception.json.  |
| Test type | Capability |

* 1. Test: catalog/download

|  |  |
| --- | --- |
| Requirements | **catalog/downloadRequest, catalog/downloadParameters, catalog/downloadValid,** |
| Test purpose | Verify that the Download Rasters resource supports the request and response requirements. |
| Test method | Set up a test service. Construct requests for the Download Rasters resources.Inspect the responses and validate them against the JSON Schema http://schemas.opengis.net/gsr-is/1.0/rasterFiles.json or for exceptions against http://schemas.opengis.net/gsr/1.0/exception.json.  |
| Test type | Capability |

* 1. Test: catalog/rasterFile

|  |  |
| --- | --- |
| Requirements | **catalog/rasterFileRequest, catalog/rasterFileParameters** |
| Test purpose | Verify that the Raster File resource supports the request and response requirements. |
| Test method | Use the service and results from the catalog/download tests. Construct requests for the Raster File resources and verify that a response is received.  |
| Test type | Capability |

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