An Esri® White Paper • September 2010

GeoServices REST Specification
Version 1.0

Esri, 380 New York St., Redlands, CA 92373-8100 USA
TEL 909-793-2853 • FAX 909-793-5953 • E-MAIL info@esri.com • WEB www.esri.com

GeoServices REST Specification Version 1.0

An Esri White Paper

Contents **Page**

Contents **Page**

Contents **Page**

Contents **Page**

Contents **Page**

Contents **Page**

Contents **Page**

Contents **Page**

Contents **Page**

Appendixes

GeoServices REST Specification Version 1.0

1.0 INTRODUCTION

The GeoServices REST Specification provides a standard way for web clients to communicate with geographic information system (GIS) servers through Representational State Transfer (REST) technology. Clients issue requests to the server through structured URLs. The server responds with map images, text-based geographic information, or other resources that satisfy the request. Although the GeoServices REST Specification was originally built to communicate with Esri's ArcGIS® Server product, the specification has been opened such that developers can expose the GeoServices REST Specification request structure from other back-end GIS servers or processes.

1.0.1 Why Implement the GeoServices REST Specification?

The GeoServices REST Specification offers a simple way for applications to request map, feature, attribute, and image information from a GIS server. Developers who adopt the GeoServices REST Specification are choosing a proven implementation that has been widely deployed and exercised in the field and that exposes server-side resources to a broad range of clients and applications. They are also choosing a JSON-based, REST-ful specification that will make the server instantly usable by thousands of developers working in popular client-side development environments with the ArcGIS web mapping APIs for JavaScript™, Flex™, Silverlight®, iOS®, and Android™, all of which are powered by the GeoServices REST Specification.

1.0.2 How Do I Implement the GeoServices REST Specification?

To implement the GeoServices REST Specification, developers architect the back-end server to respond to specifically structured REST requests in an expected way. For example, if someone issues a request to the server to export a map image, such as http://<mapservice-url>/export?bbox=-127.8,15.4,-63.5,60.5, the server should return a map image with a lower left coordinate of (-127.8, 15.4) and an upper right coordinate of (-63.5, 60.5). *How* the server generated the image is not so important as the fact that *it responded in an expected way* when issued a URL whose structure followed the GeoServices REST Specification.

The full GeoServices REST Specification is described in this document. Developers can choose how much or how little to implement. For example, if geocoding or geoprocessing operations are not exposed by the GIS server, developers may not need to implement the Geocode Service or GP Service piece of the specification.

2.0 OVERVIEW OF THE GEOSERVICES REST SPECIFICATION

All resources and operations exposed by the GeoServices REST Specification are accessible through a hierarchy of endpoints or uniform resource locators (URLs) for each available GIS service. When using the GeoServices REST Specification, users typically start from a well-known endpoint, which represents the server catalog. From the catalog, different types of resources are available as child nodes. These resources comprise services for mapping, geocoding, and so on.

The GeoServices REST Specification is stateless because REST does not keep track of transactions from one request to the next. Each request must contain all the information necessary for successful processing.

2.0.1 Resources and Operations

The GeoServices REST Specification works with a hierarchy of resources. Each service type recognized by the GeoServices REST Specification (map, geocode, and so on) is a resource and has a unique URL. Although a REST system always returns only representations of resources to client, for the sake of simplicity, the resources of the GeoServices REST Specification are divided into two types: resources and operations.

2.0.2 The Role of Services

The GeoServices REST Specification describes a catalog of web services that are designed for different GIS functions (map, geocode, and so on). Esri® ArcGIS Server can be considered a reference implementation of this specification, meaning that it implements all the service types in the catalog. The GeoServices REST Specification can be implemented with non-Esri GIS servers, but the back-end web services must respond to the URL formats described in the specification.

2.0.3 Response Formats

Many resources in the GeoServices REST Specification have a parameter, f, that denotes the response format. Developers can program resources to respond to REST requests with various data formats, including JSON, HTML, and KMZ.

At the least, the JSON response format should be implemented, and examples for doing so are provided in this specification. Other formats are optional, and they can be exposed through the f parameter; however, formats other than JSON are not detailed in this specification.

3.0 CATALOG

The catalog resource is the root node and initial entry point into a GIS server. This resource represents a catalog of folders and services published on the host.

The response optionally includes the specVersion and currentVersion properties. The specVersion is the version of the GeoServices REST Specification through which the catalog is implemented. If specVersion is not included in the response, its value is assumed to be 1.0.

The currentVersion property can be used to specify a version of the implementer's software.

3.0.1 Catalog Reference

- **URL:** http://<host>/.../<optional root folder >. Services may optionally be organized into folders, yielding a URL such as http://<host>/…/<optional root folder>/<folderName>.
- **Child Resources:** Map Service, Geocode Service, GP Service, Geometry Service, Image Service, Feature Service.

3.0.2 Catalog Parameters

Parameter	Details
f	Description: The response format Values: json | (other formats)

3.0.3 Catalog Example

Example

URL for the root directory of a GIS server:

```
http://myserver/rest/services
```

JSON Response Syntax

```
{
"specVersion": <versionOfGeoServicesRestSpecification>,
"currentVersion": <versionOfImplementerSoftware>,
"folders":["<folderName1>", "<folderName2>"],
"services":[
    {"name" : "<serviceName1>", "type" : "<serviceType1>"},
    {"name" : "<serviceName2>", "type" : "<serviceType2>"}
]
}
```

JSON Response Example

```
{
"specVersion": 1.0,
"currentVersion": 10,
"folders":["Editing", "USA"],
"services":[
    {"name" : "Anaheim", "type" : "MapServer"},
    {"name" : "Switzerland", "type" : "MapServer"},
    {"name" : "USALocator", "type" : "GeocodeServer"}
]
}
```

4.0 MAP SERVICE

Map services offer access to map and layer content. A map service can either fulfill requests with 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 also expose tabular data, whether this is associated with geographic features or not. The REST response from a map service 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 supports querying and exporting maps based on time, the REST response includes a timeInfo property, which includes information about the map's time extent and the map's native time reference.

The GeoServices REST Specification Map Service 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 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.

In addition to the above operations, a Query operation is available on a layer/table. It returns a subset of features in a layer or records in a table based on query criteria.

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

4.0.1 Map Service Reference

- **URL:** http://<catalog-url>/<serviceName>/MapServer
- **Supported Operations:** Export Map, Identify, Find
- **Parent Resource:** Catalog
- **Child Resources:** Map Tile, Layer/Table, All Layers/Tables

4.0.2 Map Service Parameters

Parameter	Details
f	Description: The response format Values: json | (other formats)

4.0.3 Map Service Example

Example

URL for the StateCityHighway service on myserver:

```
http://myserver/rest/services/StateCityHighway/MapServer
```

JSON Response Syntax

```
{
"serviceDescription" : "<serviceDescription>",
"mapName" : "<mapName>"
"description": "<description>",
"copyrightText" : "<copyrightText>",
"layers": [ //the spatial layers exposed by this service
    {
    "id" : <layerId1>,
    "name" : "<layerName1>",
    "defaultVisibility" : <true|false>,
    "parentLayerId" : <parentLayerId1>,
    "subLayerIds" : [<subLayerId11>, <subLayerId12>]
    },
```

J-9948

```
    {
    "id" : <layerId2>,
    "name" : "<layerName2>",
    "defaultVisibility" : <true|false>,
    "parentLayerId" : <parentLayerId2>,
    "subLayerIds" : [<subLayerId21>, <subLayerId22>]
    }
],
"tables": [ //the tables exposed by this service
    {
    "id" : <tableId1>,
    "name" : "<tableName1>"
    },
    {
    "id" : <tableId2>,
    "name" : "<tableName2>"
    }
],
"spatialReference" : {<spatialReference>},
"singleFusedMapCache" : <true | false>,
"tileInfo": {
    "rows" : <rows>, "cols" : <cols>, "dpi" : <dpi>, "format" :
<format>, "compressionQuality" : <quality>,
    "origin" : {<point>},
    "spatialReference" : {<spatialReference>},
    "lods": [
        {"level" : <level1>, "resolution" : <resolution1>, "scale"
: <scale1> },
        {"level" : <level2>, "resolution" : <resolution2>, "scale"
: <scale2> }
    ]
},
"initialExtent" : {<envelope>},
"fullExtent" : {<envelope>},
// if the map supports querying and exporting maps based on time
"timeInfo" : {
  "timeExtent" : [<startTime>, <endTime>],
  "timeReference" : {
    "timeZone" : "<timeZone>",
    "respectsDaylightSaving" : <true | false>
  }
},
"units" : "<units>",
"supportedImageFormatTypes" : "<supportedImageFormatTypes>",
"documentInfo": {
    "<key1>" : "<value1>",
    "<key2>" : "<value2>"
} ,
//comma-separated list of supported capabilities - e.g.
"Map,Query,Data"
"capabilities" : "<capabilities>"
}
```

JSON Response Example


```
{
"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" : 8.46, "scale" : 32000.0 },
        {"level" : 1, "resolution" : 4.23, "scale" : 16000.0 },
        {"level" : 2, "resolution" : 2.11, "scale" : 8000.0 },
        {"level" : 3, "resolution" : 1.05, "scale" : 4000.0 },
        {"level" : 4, "resolution" : 0.52, "scale" : 2000.0 }
    ]
},
"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" : "esriDecimalDegrees",
"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"
}
```

4.0.4 Export Map Operation

The Export Map operation is performed on a Map Service resource. The result of this operation provides information about the exported map image such as its URL, width and height, extent, and scale.

Apart from the usual 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.

Note that the extent displayed in the exported map image may 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 should be resized to prevent map images from appearing stretched. The exported map's extent is sent along with the JSON response and may be used in client-side calculations, so it is important that the client-side code update its extent based on the response.

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.

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

4.0.4.1 Export Map Reference

- **URL:** http://<mapservice-url>/export
- **Parent Resource:** Map Service

4.0.4.2 Export Map Parameters

Parameter	Details
f	Description: The response format. If the format is image, the image is streamed to the client. Values: json | image | (other formats)

bbox	**Required**
	Description: The extent (bounding box) of the exported image. This parameter is required. The bbox is assumed to be in the spatial reference of the map unless the bboxSR parameter has been specified.
	
	Syntax:
	
	`<xmin>, <ymin>, <xmax>, <ymax>`
	
	Example:
	
	`bbox=-104,35.6,-94.32,41`

Parameter	Details
	The bbox coordinates should always use a period as the decimal separator even in countries where traditionally a comma is used.

size	Description: The size (width * height) of the exported image in pixels. If the size is not specified, an image with a default size of 400 * 400 is exported.
	
	Syntax:
	
	`<width>, <height>`
	
	Example:
	
	`size=600,550`

dpi	Description: The device resolution of the exported image (dots per inch). If the dpi is not specified, an image with a default dpi of 96 is exported. Example: `dpi=200`
imageSR	Description: The spatial reference of the exported image The spatial reference can be specified as either a well-known ID (WKID) or a spatial reference JSON object. See the Spatial References section of this document for more information about WKID. See the Geometry Objects section of this document for more information about spatial reference JSON objects. If the imageSR parameter is not specified, the image is exported in the spatial reference of the map.
bboxSR	Description: The spatial reference of the bbox. The spatial reference can be specified as either a WKID or a spatial reference JSON object. If the bboxSR parameter is not specified, the bbox is assumed to be in the spatial reference of the map.
format	Description: The format of the exported image. The default format value is png. Values: png | png8 | png24 | jpg | pdf | bmp | gif | svg | png32
layerDefs	Description: Allows you to filter the features of individual layers in the exported map by specifying definition expressions for those layers **Simple Syntax** Syntax: `layerId1:layerDef1;layerId2:layerDef2` where layerId1 and layerId2 are the layer IDs returned by the Map Service resource

Parameter	Details

	Example:
	
	`0:POP2000 > 1000000;5:AREA > 100000`
	
	JSON Syntax
	
	You can also use a JSON representation for layer definitions.
	
	Syntax:
	
	`{ "<layerId1>" : "<layerDef1>" , "<layerId2>" : "<layerDef2>" }`
	
	where layerId1 and layerId2 are the layer IDs returned by the Map Service resource
	
	Example:
	
	`{"0":"POP2000 > 1000000","5":"AREA > 100000"}`

layers	Description: 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.
	
	Syntax:
	
	`[show | hide | include | exclude]:layerId1,layerId2`
	
	In the above example, layerId1 and layerId2 are the layer IDs returned by the Map Service resource.
	
	Example:
	
	`layers=show:2,4,7`

transparent	Description: If true, the image is exported with the background color of the map set as its transparent color. The default is false. Values: true | false

Parameter	Details

time	Description: The time instant or the time extent of the exported map image
	
	Time Instant
	
	Syntax:
	
	`time=<timeInstant>`
	
	Example:
	
	`time=1199145600000`
	
	(1 Jan. 2008 00:00:00 UTC)
	
	Time Extent
	
	Syntax:
	
	`time=<startTime>, <endTime>`
	
	Example:
	
	`time=1199145600000, 1230768000000`
	
	(1 Jan. 2008 00:00:00 UTC to 1 Jan. 2009 00:00:00 UTC)

layerTimeOptions	Description: The time options per layer. Users can indicate whether the layer should 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.
	
	Syntax:
	
	`{`
	`  "<layerId1>" : {`
	`    //If true, use the time extent specified by the time parameter`
	`    "useTime" : < true | false >,`
	`    //If true, draw all the features from the beginning of time for that data`
	`    "timeDataCumulative" : < true | false >,`
	`    //Time offset for this layer so that it can be overlaid on the top of a previous or future time period`
	`    "timeOffset" : <timeOffset1>,`
	`    "timeOffsetUnits" : "<esriTimeUnitsCenturies | esriTimeUnitsDays | esriTimeUnitsDecades | esriTimeUnitsHours | esriTimeUnitsMilliseconds | esriTimeUnitsMinutes | esriTimeUnitsMonths | esriTimeUnitsSeconds | esriTimeUnitsWeeks | esriTimeUnitsYears |`

Parameter	Details

	esriTimeUnitsUnknown>"
	},
	"<layerId2>" : {
	"useTime" : < true | false >,
	"timeDataCumulative" : < true | false >,
	"timeOffsetOffset" : <timeOffset2>,
	"timeOffsetUnits" : "<timeOffsetUnits2>"
	}
	}
	
	Example:
	
	{
	"0" : {
	"useTime" : true,
	"timeDataCumulative" : false,
	"timeOffset" : 1,
	"timeOffsetUnits" : "esriTimeUnitsYears"
	},
	"3" : {
	"useTime" : false
	}
	}

4.0.4.3 Export Map Example

Example 1

Export a map. Include only the bounding box:

```
http://myserver/rest/services/StateCityHighway/MapServer/export?bb
ox=-127.8,15.4,-63.5,60.5
```

Example 2

Export a map and change the imageSR value to WKID 102004 (USA_Contiguous_Lambert_Conformal_Conic projection):

```
http://myserver/rest/services/StateCityHighway/MapServer/export?bb
ox=-127.8,15.4,-63.5,60.5&imageSR=102004
```

Example 3

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://myserver/rest/services/StateCityHighway
/MapServer/export?bbox=-115.8,30.4,-
85.5,50.5&size=800,600&imageSR=102004&format=gif&transparent=false
&dpi=&f=html
```

Example 4

Export the same map as above but change the output format to pretty JSON (f=pjson):

```
http://myserver/rest/services/StateCityHighway/MapServer/export?bb
ox=-115.8,30.4,-85.5,50.5&size=800,600&imageSR=102004&format=
gif&transparent=false&f=pjson
```

JSON Response Syntax

```
{
"href" : "<href>",
"width" : <width>,
"height" : <height>,
"extent" : {<envelope>},
"scale" : <scale>
}
```

JSON Response Example

```
{
"href" :
"http://myserver/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
}
```

4.0.5 Identify Operation

The Identify operation is performed on a Map Service resource 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.

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

4.0.5.1 Identify Reference

- **URL:** http://<mapservice-url>/identify
- **Parent Resource:** Map Service

4.0.5.2 Identify Parameters

Parameter	Details
f	Description: The response format Values: json | (other formats)
geometry	**Required** Description: The geometry to identify on. The type of the geometry is specified by the geometryType parameter. In

Parameter	Details

	addition to the JSON structures, for points and envelopes, users can specify the geometries with a simpler comma-separated syntax.
	
	Syntax:
	
	■ JSON structures:
	`geometryType=<geometryType>&geometry={geometry}`
	
	■ Point simple:
	`geometryType=esriGeometryPoint&geometry=<x>,<y>`
	
	■ Envelope simple:
	`geometryType=esriGeometryEnvelope&geometry=<xmin>,<ymin>,<xmax>,<ymax>`
	
	Examples:
	
	■ `geometryType=esriGeometryPoint&geometry={x: -104, y: 35.6}`
	
	■ `geometryType=esriGeometryPoint&geometry=-104,35.6`
	
	■ `geometryType=esriGeometryEnvelope&geometry=-104,35.6,-94.32,41`
	
	The coordinates must always use a period as the decimal separator even in countries where a comma is traditionally used.

geometryType	Description: The type of geometry specified by the geometry parameter. The geometry type could be a point, line, polygon, or envelope. The default geometry type is a point (esriGeometryPoint). Values: esriGeometryPoint | esriGeometryMultipoint | esriGeometryPolyline | esriGeometryPolygon | esriGeometryEnvelope ■ esriGeometryPoint: A zero-dimensional object that represents a specific x,y location in the two-dimensional x,y plane. ■ esriGeometryPoint: A collection of points. A multipoint is a one-dimensional geometry object. ■ esriGeometryPolyline: A collection of one or many paths of points. The paths do not have to be connected to each other.

Parameter	Details
	■ esriGeometryPolygon: A collection of one or many exterior and interior rings. The rings do not need to be connected to or contained by other rings in the polygon. However, all rings are considered to be part of a single polygon regardless of their location. Rings can be embedded in the interior of other rings. Embedded rings define interior boundaries or holes within the polygon. Exterior rings are oriented in a clockwise direction, while interior rings are oriented counterclockwise. ■ esriGeometryEnvelope: A rectangular window specified by XMin, XMax, YMin, and YMax coordinate values. All geometries have a containing envelope specifying the bounds of the geometry.
sr	Description: The well-known ID of the spatial reference of the input and output geometries as well as the map extent. If a value for sr is not specified, the geometry and the map extent are assumed to be in the spatial reference of the map, and the output geometries are also in the spatial reference of the map.

layerDefs	Description: Allows filtering of the features of individual layers in the exported map by specifying definition expressions for those layers. Definition expressions for a layer that is published with the service will always be honored.
	
	Simple Syntax
	
	Syntax:
	
	`layerId1:layerDef1;layerId2:layerDef2`
	
	where layerId1 and layerId2 are the layer IDs returned by the Map Service resource
	
	Example:
	
	`0:POP2000 > 1000000;5:AREA > 100000`
	
	JSON Syntax
	
	Users can also use a JSON representation for layer definitions.
	
	Syntax:
	
	`{ "<layerId1>" : "<layerDef1>" , "<layerId2>" : "<layerDef2>" }`
	
	where layerId1 and layerId2 are the layer IDs returned by the Map Service resource
	
	Example:
	
	`{"0":"POP2000 > 1000000","5":"AREA > 100000"}`

Parameter	Details

time	Description: The time instant or the time extent of the features to be identified
	
	Time Instant
	
	Syntax:
	
	`time=<timeInstant>`
	
	Example:
	
	`time=1199145600000`
	
	(1 Jan. 2008 00:00:00 GMT)
	
	Time Extent
	
	Syntax:
	
	`time=<startTime>, <endTime>`
	
	Example:
	
	`time=1199145600000, 1230768000000`
	
	(1 Jan. 2008 00:00:00 GMT to 1 Jan. 2009 00:00:00 GMT)
	
	A null value specified for start time or end time represents infinity for start or end time, respectively.

layerTimeOptions	Description: The time options per layer. Users can indicate whether the layer should 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.
	
	Syntax:
	
	`{`
	`  "<layerId1>" : {`
	`  //If true, use the time extent specified by the time parameter`
	`  "useTime" : < true | false >,`
	`  //If true, draw all the features from the beginning of time for that data`
	`  "timeDataCumulative" : < true | false >,`
	`  //Time offset for this layer so that it can be overlaid on the top of a previous or future time period`
	`  "timeOffset" : <timeOffset1>,`
	`  "timeOffsetUnits" : "<esriTimeUnitsCenturies | esriTimeUnitsDays | esriTimeUnitsDecades | esriTimeUnitsHours | esriTimeUnitsMilliseconds | esriTimeUnitsMinutes | esriTimeUnitsMonths | esriTimeUnitsSeconds | esriTimeUnitsWeeks | esriTimeUnitsYears | esriTimeUnitsUnknown>"`

Parameter	Details

	<pre> },
	  "<layerId2>" : {
	    "useTime" : < true | false >,
	    "timeDataCumulative" : < true | false >,
	    "timeOffsetOffset" : <timeOffset2>,
	    "timeOffsetUnits" : "<timeOffsetUnits2>"
	  }
	}</pre>
	Example:
	<pre>{
	  "0" : {
	    "useTime" : true,
	    "timeDataCumulative" : false,
	    "timeOffset" : 1,
	    "timeOffsetUnits" : "esriTimeUnitsYears"
	  },
	  "3" : {
	    "useTime" : false
	  }
	}</pre>

layers	Description: 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 location ■ visible: All visible layers at the specified location ■ all: All layers at the specified location Example: <pre>layers=all</pre> Default: By default, the Identify operation identifies the topmost layer (i.e., layers=top) Users can 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. Syntax: <pre>[top | visible | all]:layerId1,layerId2</pre> where layerId1 and layerId2 are the layer IDs returned by the Map Service resource Example: <pre>layers=visible:2,5</pre>

Parameter	Details

imageDisplay	**Required**
	
	Description: 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.
	
	Syntax:
	
	`<width>, <height>, <dpi>`
	
	Example:
	
	`imageDisplay=600,550,96`

returnGeometry	Description: If true, the result set includes the geometries associated with each result. The default is true. Values: true | false
maxAllowableOffset	Description: Specifies the maximum allowable offset to be used for generalizing geometries returned by the Identify 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. Example: `maxAllowableOffset=2`

4.0.5.3 Identify Example

Example 1

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

```
http://myserver/rest/services/StateCityHighway/MapServer/identify?
geometryType=esriGeometryPoint&geometry=-
120,40&tolerance=10&mapExtent=-119,38,-
121,41&imageDisplay=400,300,96
```

Example 2

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

```
http://myserver/rest/services/StateCityHighway/MapServer/identify?
geometryType=esriGeometryPoint&geometry={“x”: -
120,”y”:40}&tolerance=10&mapExtent=-119,38,-
121,41&imageDisplay=400,300,96&f=json
```

Example 3

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://myserver/rest/services/StateCityHighway/MapServer/identify?
geometryType=esriGeometryPoint&geometry={“x”: -
120,”y”:40}&layers=all:2&tolerance=10&mapExtent=-119,38,-
121,41&imageDisplay=400,300,96&returnGeometry=true
```

JSON Response Syntax

```
{
"results" : [
 {
 "layerId" : <layerId1>,
 "layerName" : "<layerName1>",
 "value" : "<value1>",
 "displayFieldName" : "<displayFieldName1>",
 "attributes" : {
  "<fieldName11>" : <fieldValue11>,
  "<fieldName12>" : <fieldValue12>
 },
 "geometryType" : "<geometryType1>",
 "geometry" : {<geometry1>}
 },
 {
 "layerId" : <layerId2>,
 "layerName" : "<layerName2>",
 "value" : "<value2>",
 "displayFieldName" : "<displayFieldName1>",
 "attributes" : {
  "<fieldName21>" : <fieldValue21>,
  "<fieldName22>" : <fieldValue22>
 },
 "geometryType" : "<geometryType2>",
 "geometry" : {<geometry2>}
 }
]
}
```

JSON Response Example

```
 {
"results" : [
 {
 "layerId" : 3,
 "layerName" : "Cities",
 "value" : "Joe City",
 "displayFieldName" : "City Name",
 "attributes" : {
  "City Name" : "Joe City",
  "CLASS" : "city",
  "ST" : "CA"
 },
 "geometryType" : "esriGeometryPoint",
```

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

4.0.6 Find Operation

The Find operation is performed on a Map Service 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.

You can provide arguments to the Find operation as query parameters as defined in the parameters table below.

4.0.6.1 Find Reference

- **URL:** http://<mapservice-url>/find
- **Parent Resource:** Map Service

4.0.6.2 Find Parameters

Parameter	Details
f	Description: The response format Values: json | (other formats)
searchText	**Required** Description: The search string. This is the text that is searched across the layers and the fields that the user specifies. Example: `searchText=Los`
contains	Description: 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. The default is true. Values: true | false

Parameter	Details

searchFields	Description: The names of the fields to search. The fields are specified as a comma-separated list of field names. If this parameter is not specified, all fields are searched.
	
	Syntax:
	
	`searchFields=<fieldName1>,<fieldName2>`
	
	where fieldName1 and fieldName2 are the field names returned by the Layer resource
	
	Example:
	
	`searchFields=AREANAME,SUB_REGION`

sr	Description: The well-known ID of the spatial reference of the output geometries. If a value for sr is not specified, the output geometries are returned in the spatial reference of the map.

layerDefs	Description: Allows users to filter the features of individual layers in the exported map by specifying definition expressions for those layers. Definition expressions for a layer that is published with the service will always be honored.
	
	Simple Syntax
	
	Syntax:
	
	`layerId1:layerDef1;layerId2:layerDef2`
	
	where layerId1 and layerId2 are the layer IDs returned by the Map Service resource
	
	Example:
	
	`0:POP2000 > 1000000;5:AREA > 100000`
	
	JSON Syntax
	
	Users can also use a JSON representation for layer definitions.
	
	Syntax:
	
	`{ "<layerId1>" : "<layerDef1>" , "<layerId2>" : "<layerDef2>" }`
	
	where layerId1 and layerId2 are the layer IDs returned by the Map Service resource
	
	Example:
	
	`{"0":"POP2000 > 1000000","5":"AREA > 100000"}`

Parameter	Details

layers	**Required**
	
	Description: The layers to perform the Find operation on. The layers are specified as a comma-separated list of layer IDs.
	
	Syntax:
	
	`layers=<layerId1>,<layerId2>`
	
	where layerId1 and layerId2 are the layer IDs returned by the Map Service resource
	
	Example:
	
	`layers=2,4,7`

returnGeometry	Description: If true, the result set includes the geometries associated with each result. The default is true. Values: true | false
maxAllowableOffset	Description: 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. Example: `maxAllowableOffset=2`

4.0.6.3 Find Example

Example

Find operation that includes search text and a layer:

```
http://myserver/rest/services/StatesCitiesRivers/MapServer/find?se
archText=island&contains=true&layers=0,2&returnGeometry=true
```

JSON Response Syntax

```
{
"results" : [
 {
 "layerId" : <layerId1>,
 "layerName" : "<layerName1>",
 "displayFieldName" : "<displayFieldName1>"
 "foundFieldName" : "<foundFieldName1>",
 "value" : "<value1>",
 "attributes" : {
 "<fieldName11>" : <fieldValue11>,
 "<fieldName12>" : <fieldValue12>
 },
 "geometryType" : "<geometryType1>",
 "geometry" : {<geometry1>}
```

```
 },
 { "layerId" : <layerId2>,
 "layerName" : "<layerName2>",
 "displayFieldName" : "<displayFieldName2>"
 "foundFieldName" : "<foundFieldName2>",
 "value" : "<value2>",
 "attributes" : {
 "<fieldName21>" : <fieldValue21>,
 "<fieldName22>" : <fieldValue22>
  },
 "geometryType" : "<geometryType2>",
 "geometry" : {<geometry2>}
 }
]
}
```

JSON Response Example

```
{
"results" : [
 { "layerId" : 3,
 "layerName" : "Cities",
 "displayFieldName" : "City Name"
 "foundFieldName" : "City Name",
 "value" : "Joe City",
 "attributes" : {
 "City Name" : "Joe City",
 "CLASS" : "city",
 "ST" : "CA"
 },
 "geometryType" : "esriGeometryPoint",
 "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" : "esriGeometryPolygon",
 "geometry" : { "spatialReference" : {"wkid" : 4326}, "rings" :
[[[-118.35,32.81],[-118.42.806],[-118.511,32.892],[-
118.35,32.81]]]}
 }
]
}
```

4.1 Map Tile

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.

4.1.1 Map Tile Reference

- **URL:** http://<mapservice-url>/tile/<level>/<row>/<column>
- **Parent Resource:** Map Service

4.1.2 Map Tile Example

Example

Request a map tile for the Chicago service on myserver:

```
http://myserver/rest/services/Chicago/MapServer/tile/5/7/10
```

4.2 Layer/Table

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.

If a layer/table supports querying and exporting maps based on time, the response should include a timeInfo property with 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.

4.2.1 Layer/Table Reference

- **URL:** http://<mapservice-url>/<layerOrTableId>
- **Supported Operations:** Query, Query Related Records
- **Parent Resource:** Map Service
- **Child Resource:** Feature

4.2.2 Layer/Table Parameters

Parameter	Details
f	Description: The response format Values: json | (other formats)

4.2.3 Layer/Table Example

Example

Get information about the layer with table of contents index position 2 in the StateCityHighway service on myserver:

```
http://myserver/rest/services/StateCityHighway/MapServer/2
```

JSON Response Syntax

```
{
"id" : <layerOrTableId>,
"name" : "<layerOrTableName>",
"type" : "<layerOrTableType>", //for tables, the type will be
"Table"
"description" : "<description>",
"definitionExpression" : "<definitionExpression>",
```

```
//properties specific to layers only
"geometryType" : "<geometryType>",
"copyrightText" : "<copyrightText>",
"parentLayer" : {"id" : <parentLayerId>, "name" :
"<parentLayerName>"},
"subLayers" : [
    {"id" : <subLayerId1>, "name" : "<subLayerName1>"},
    {"id" : <subLayerId2>, "name" : "<subLayerName2>"}
],
"minScale" : <minScale>,
"maxScale" : <maxScale>,
"extent" :  <envelope>,

// if the layer / table supports querying and exporting maps based
on time
"timeInfo" : {
  "startTimeField" : "<startTimeFieldName>",
  "endTimeField" : "<endTimeFieldName>",
  "trackIdField" : "<trackIdFieldName>",
  "timeExtent" : [<startTime>, <endTime>],
  "timeReference" : {
    "timeZone" : "<timeZone>",
    "respectsDaylightSaving" : <true | false>
  },
  "timeInterval" : <timeInterval>,
  "timeIntervalUnits" : "<timeIntervalUnits>",
  //the default time-related export options for this layer
  "exportOptions" : {
    //If true, use the time extent specified by the time parameter
    "useTime" : < true | false >,
    //If true, draw all the features from the beginning of time
for that data
    "timeDataCumulative" : < true | false >,
    //Time offset for this layer so that it can be overlaid on the
top of a previous or future time period
    "timeOffset" : <timeOffset1>,
    "timeOffsetUnits" : "<esriTimeUnitsCenturies |
esriTimeUnitsDays | esriTimeUnitsDecades |
                              esriTimeUnitsHours |
esriTimeUnitsMilliseconds | esriTimeUnitsMinutes |
                              esriTimeUnitsMonths |
esriTimeUnitsSeconds | esriTimeUnitsWeeks | esriTimeUnitsYears |
                              esriTimeUnitsUnknown>"
  }
},

//for feature layers only
"drawingInfo" : {
  "renderer" : <renderer>,
  "transparency" : <transparency>,
  "labelingInfo" : <labelingInfo>
},

//indicates whether the layer / table has attachments or not
"hasAttachments" : <true | false>,
```

J-9948

```
//indicates whether the layer / table has htmlPopups
"htmlPopupType" : "<esriServerHTMLPopupTypeNone |
esriServerHTMLPopupTypeAsURL |
esriServerHTMLPopupTypeAsHTMLText>",

//layer / table field information
"displayField" : "<displayFieldName>",
"typeIdField" : "<typeIFieldName>",
"fields" : [
    {"name" : "<fieldName1>", "type" : "<fieldType1>", "alias" :
"<fieldAlias1>", "length" : "<length1>", "domain" : <domain1>},
    {"name" : "<fieldName2>", "type" : "<fieldType2>", "alias" :
"<fieldAlias2>", "length" : "<length2>", "domain" : <domain2>}
],

//if the layer has sub-types, they are included
"types" : [
  {
    "id" : <typeId1>,
    "name" : "<typeName1>",
    "domains" : {
      "<domainField11>" : <domain11>,
      "<domainField12>" : <domain12>
    }
  },
  {
    "id" : <typeId2>,
    "name" : "<typeName2>",
    "domains" : {
      "<domainField11>" : <domain21>,
      "<domainField12>" : <domain22>
    }
  }
],

//if the layer / table participates in relationships with other
layers / tables
"relationships" : [
  {
    "id" : <relationshipId1>,
    "name" : "<relationshipName1>",
    "relatedTableId" : <relatedTableId1>,
  },
  {
    "id" : <relationshipId2>,
    "name" : "<relationshipName2>",
    "relatedTableId" : <relatedTableId2>,
  }
],
//comma separated list of supported capabilities - e.g.
"Map,Query,Data"
"capabilities" : "<capabilities>"
}
```

JSON Response Example


```
{
      "id" : 0,
      "name" : "Wells",
      "type" : "Feature Layer",
      "description" : "",
      "definitionExpression" : "",
      "geometryType" : "esriGeometryPoint",
      "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" : "esriTimeUnitsYears",
        "exportOptions" : {
          "useTime" : true,
          "timeDataCumulative" : false, "timeOffset" : null,
"timeOffsetUnits" : null
        }
      },
      "drawingInfo" : {"renderer" :
        {
          "type" : "simple",
          "symbol" :
          {
            "type" : "esriSMS",
            "style" : "esriSMSCircle",

            "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" : "esriFieldTypeOID",
          "alias" : "OBJECTID"},
        {
          "name" : "Shape",
          "type" : "esriFieldTypeGeometry",
          "alias" : "Shape"},
        {
          "name" : "KID",
          "type" : "esriFieldTypeDouble",
          "alias" : "KID"},
        {
          "name" : "STATE_CODE",
          "type" : "esriFieldTypeSmallInteger",
          "alias" : "STATE_CODE"}
      ],
      "relationships" : [
        {
          "id" : 3,
          "name" : "Well 2 Tops",
          "relatedTableId" : 2},
        {
          "id" : 2,
          "name" : "Wells 2 Field",
          "relatedTableId" : 1}
      ],
    "capabilities" : "Map,Query,Data"
}
```


4.2.4 Query Operation (Map Service)

The Query operation is performed on a Layer/Table resource. The result of this operation is a feature set, which is an informational structure for storing multiple feature objects (described in Section 11.0 Feature Object). The result feature set contains feature objects, including the values for the fields requested by the user. For layers, if the user requests geometry information, the geometry of each feature is also returned in the feature set. For tables, the feature set does not include geometries.

Note that all parameters related to geometry are ignored when querying tables.

For time-aware layers, users can leverage the time parameter to specify the time instant or the time extent to query.

The returnIdsOnly parameter determines what is included in the response. If set to false (default), the response is a feature set. If true, the response is an array of ObjectIDs.

Users can provide arguments to the Query operation as query parameters defined in the parameters table below.

4.2.4.1 Query Reference

- **URL:** http://<layer-url>/query
- **Parent Resource:** Layer/Table

4.2.4.2 Query Parameters

Parameter	Details
f	Description: The response format Values: json | (other formats)

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. This parameter is ignored if the where parameter is specified.
	
	Example:
	
	`text = Los`

geometry	Description: The geometry to apply as the spatial filter. The structure of the geometry is the same as the structure of the JSON geometry objects described elsewhere in this specification. In addition to the JSON structures, for envelopes and points, a user can specify the geometry with a simpler comma-separated syntax.
	
	Syntax:
	
	■ JSON structures:
	`geometryType=<geometryType>&geometry={geometry}`
	
	■ Envelope simple:
	`geometryType=esriGeometryEnvelope&geometry=<xmin>, <ymin>,<xmax>,<ymax>`
	
	■ Point simple:
	`geometryType=esriGeometryPoint&geometry=<x>,<y>`

J-9948

Parameter	Details
	Examples: ■ `geometryType=esriGeometryEnvelope&geometry={ xmin: -104, ymin: 35.6, xmax: -94.32, ymax: 41}` ■ `geometryType=esriGeometryEnvelope&geometry=-104,35.6,-94.32,41` ■ `geometryType=esriGeometryPoint &geometry=-104,35.6`
geometryType	Description: The type of geometry specified by the geometry parameter. The geometry type can be an envelope, point, line, or polygon. The default geometry type is an envelope. Values: esriGeometryPoint | esriGeometryMultipoint | esriGeometryPolyline | esriGeometryPolygon | esriGeometryEnvelope
inSR	Description: The spatial reference of the input geometry. This can be specified as either a well-known ID or a spatial reference JSON object. If a value for inSR is not specified, the geometry is assumed to be in the spatial reference of the map.
spatialRel	Description: The spatial relationship to be applied on the input geometry while performing the query. The supported spatial relationships include intersects, contains, envelope intersects, and within. The default spatial relationship is intersects (esriSpatialRelIntersects). Values: esriSpatialRelIntersects | esriSpatialRelContains | esriSpatialRelCrosses | esriSpatialRelEnvelopeIntersects | esriSpatialRelIndexIntersects | esriSpatialRelOverlaps | esriSpatialRelTouches | esriSpatialRelWithin | esriSpatialRelRelation ■ esriSpatialRelIntersects: Returns a feature if any spatial relationship is found. Applies to all shape type combinations. ■ esriSpatialRelContains: Returns a feature if its shape is wholly contained within the search geometry. Valid for all shape type combinations. ■ esriSpatialRelCrosses: 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.

<table>
<tr><th>Parameter</th><th>Details</th></tr>
<tr><td></td><td>

esriSpatialRelEnvelopeIntersects: Returns a feature if the envelope of the two shapes intersects.
esriSpatialRelIndexIntersects: Returns a feature if the envelope of the query geometry intersects the index entry for the target geometry.
esriSpatialRelOverlaps: 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.
esriSpatialRelTouches: 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.
esriSpatialRelWithin: Returns a feature if its shape wholly contains the search geometry. Valid for all shape type combinations.
esriSpatialRelRelation: Defines a custom spatial relationship as specified by the relationParam parameter.

</td></tr>
<tr><td>relationParam</td><td>
Description: 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 shape (G1) can be tested against each element of the second shape (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 previous 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:
<table>
<tr><td>1</td><td>G1.interior</td><td>G2.interior</td></tr>
<tr><td>2</td><td>G1.interior</td><td>G2.boundary</td></tr>
<tr><td>3</td><td>G1.interior</td><td>G2.exterior</td></tr>
<tr><td>4</td><td>G1.boundary</td><td>G2.interior</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Parameter</th><th>Details</th></tr>
<tr><td></td><td>
<table>
<tr><td>5</td><td>G1.boundary</td><td>G2.boundary</td></tr>
<tr><td>6</td><td>G1.boundary</td><td>G2.exterior</td></tr>
<tr><td>7</td><td>G1.exterior</td><td>G2.interior</td></tr>
<tr><td>8</td><td>G1.exterior</td><td>G2.boundary</td></tr>
<tr><td>9</td><td>G1.exterior</td><td>G2.exterior</td></tr>
</table>

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 shape element relationship is that the dimension of intersection between the shape elements is not null. This function does not evaluate the dimension of intersection between the shape elements, whether or not the intersection exists.

Any two shapes' exteriors always intersect, and the dimension of intersection is 2 (area).</td></tr>
<tr><td>where</td><td>Description: A WHERE clause for the query filter. Any legal SQL WHERE clause operating on the fields in the layer is allowed.

Example:

<code>where=POP2000 > 350000</code></td></tr>
<tr><td>objectIds</td><td>Description: The ObjectIDs of this layer/table to be queried

Syntax:

<code>objectIds=<objectId1>, <objectId2></code>

Example:

<code>objectIds=37, 462</code></td></tr>
<tr><td>time</td><td>Description: The time instant or the time extent to query

Time Instant

Syntax:

<code>time=<timeInstant></code>

Example:

<code>time=1199145600000</code>

(1 Jan. 2008 00:00:00 GMT)

Time Extent

Syntax:

<code>time=<startTime>, <endTime></code></td></tr>
</table>

Parameter	Details
	Example: `time=1199145600000, 1230768000000` (1 Jan. 2008 00:00:00 GMT to 1 Jan. 2009 00:00:00 GMT)

outFields	Description: 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.
	
	Example:
	
	`outFields=AREANAME,ST,POP2000`
	
	Example (wildcard usage):
	
	`outFields=*`

returnGeometry	Description: If true, the result set includes the geometry associated with each result. The default is true. If the outFields parameter is set to the wildcard, it implies returnGeometry=true, and setting returnGeometry to false has no effect. Values: true | false
maxAllowableOffset	Description: This option specifies the maximum allowable offset to be used for generalizing geometries returned by the Query operation. Example: `maxAllowableOffset=2`
outSR	Description: The spatial reference of the returned geometry. The spatial reference can be specified as either a well-known ID or a spatial reference JSON object. If a value for outSR is not specified, the geometry is returned in the spatial reference of the map.
returnIdsOnly	Description: If true, the response only includes an array of ObjectIDs. Otherwise, the response is a feature set. The default is false. Values: false | true

4.2.4.3 Query Examples

Example 1

Query using the text parameter on the states layer of the StateCityHighway service on myserver:

```
http://myserver/rest/services/StateCityHighway/MapServer/1/query?text=Texas
```

Example 2

Query using a where statement on the same layer. The output is JSON format:

```
http://myserver/rest/services/StateCityHighway/MapServer/1/query?where=STATE_NAME='Florida'&f=json
```

Example 3

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

```
http://myserver/rest/services/StateCityHighway/MapServer/1/query?where=UPPER(STATE_NAME)=UPPER('colorado')
```

Example 4

Query the same states layer using geometry (envelope):

```
http://myserver/rest/services/StateCityHighway/MapServer/1/query?geometry=-125.4,35.2,-118.7,43.8&geometryType=esriGeometryEnvelope
```

Example 5

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

```
http://myserver/rest/services/StateCityHighway/MapServer/1/query?geometry=-125.4,35.2,-
118.7,43.8&geometryType=esriGeometryEnvelope&where=POP1999>5000000
```

Example 6

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

```
http://myserver/rest/services/StateCityHighway/MapServer/1/query?where=POP1999>15000000&returnGeometry=false&outFields=STATE_NAME,MALES,FEMALES,POP1999
```

Example 7

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

```
http://myserver/rest/services/StateCityHighway/MapServer/1/query?t
ext=New+York&outSR=102113
```

Example 8

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

```
http://myserver/rest/services/311Incidents/MapServer/1/
query?objectIds=&where=agree_with_incident+%3D+1&geometryType=esri
GeometryEnvelope&spatialRel=esriSpatialRelIntersects&returnGeometr
y=true&returnIdsOnly=true
```

JSON Response Syntax (when returnIdsOnly=false)

```
{
"displayFieldName" : "<displayFieldName>",
"fields" : [
    {"name" : "<fieldName1>", "type" : "<fieldType1>", "alias" :
"<fieldAlias1>", "length" : "<length1>"},
    {"name" : "<fieldName2>", "type" : "<fieldType2>", "alias" :
"<fieldAlias2>", "length" : "<length2>"}
],
"geometryType" : "<geometryType>", //for layers only
"spatialReference" : <spatialReference>, //for layers only
"features" : [ //features may include geometry for layers only
  <feature1>, <feature2>
]
}
```

JSON Response Syntax (when returnldsOnly=true)

```
{
"objectIdFieldName" : "<objectIdFieldName>",
"objectIds" : [ <objectId1>, <objectId2> ]
}
```

JSON Response Example (when returnIdsOnly=false)

```
{
"displayFieldName" : "AREANAME",
"fields" : [
{
    "name" : "ST",
    "alias" : "ST",
    "type" : "esriFieldTypeString",
    "length" : 2
},
{
```

```
    "name" : "POP2000",
    "alias" : "Population - 2000",
        "type" : "esriFieldTypeInteger"
},

{
    "name" : "AREANAME",
    "alias" : "City Name",
    "type" : "esriFieldTypeString",
    "length" : 255
}
],
"geometryType" : "esriGeometryPoint",
"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 }
    }
]
}
```

JSON Response Example (when returnIdsOnly=true)

```
{
"objectIdFieldName":"objectid",
"objectIds":[1,2,3,4,5,7]
}
```

4.2.5 Query Related Records Operation

The Query Related Records operation is performed on a Layer/Table resource. The result of this operation is one or more feature sets grouped by source layer/table ObjectIDs. Each feature set contains feature objects that include the values for the fields requested by the user. For related layers, if the 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.

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.

Note that all parameters related to geometry are ignored when querying related tables.

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

4.2.5.1 Query Related Records Reference

- **URL:** http://<layer-url>/queryRelatedRecords
- **Parent Resource:** Layer/Table

4.2.5.2 Query Related Records Parameters

Parameter	Details
f	Description: The response format Values: json | (other formats)

objectIds	Description: The ObjectIDs of the layer/table to be queried. Records related to these ObjectIDs will be queried.
	
	Syntax:
	
	`objectIds=<objectId1>, <objectId2>`
	
	Example:
	
	`objectIds=37, 462`

relationshipId	Description: 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. Example: `relationshipId=4`

outFields	Description: 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.
	
	Example:
	
	`outFields=AREANAME,ST,POP2000`
	
	Example (wildcard usage):
	
	`outFields=*`

Parameter	Details
definitionExpression	Description: The definition expression to be applied to the related layer/table. From the list of records that are related to the specified ObjectIDs, only those records that conform to this expression are returned. Example: `definitionExpression=POP2000 > 100000`
returnGeometry	Description: If true, the feature set includes the geometry associated with each feature. The default is true. Note that this parameter only applies to related layers. It will be ignored for related tables. Values: true | false
maxAllowableOffset	Description: This option can be used to specify 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 output 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. Example: `maxAllowableOffset=2`
outSR	Description: The spatial reference of the returned geometry. The spatial reference can be specified as either a well-known ID or as a spatial reference JSON object. If a value for outSR is not specified, the geometry is returned in the spatial reference of the map. Note that this parameter only applies to related layers. It is ignored for related tables.

4.2.5.3 Query Related Records Example

Example

Query related records:

```
http://myserver/rest/services/KSPetro/MapServer/0/queryRelatedReco
rds?objectIds=3,4,5&relationshipId=2&returnGeometry=true&outFields
=*
```

JSON Response Syntax

```
{
"geometryType" : "<geometryType>", //if records include geometry
"spatialReference" : <spatialReference>, //if records include
geometry
```

```
"fields" : [
    {"name" : "<fieldName1>", "type" : "<fieldType1>", "alias" :
"<fieldAlias1>", "length" : "<length1>"},
    {"name" : "<fieldName2>", "type" : "<fieldType2>", "alias" :
"<fieldAlias2>", "length" : "<length2>"}
],
"relatedRecordGroups" : [
  {
    "objectId" : <objectId1>,
    "relatedRecords" : [ //features may include geometry for
related layers only
      <relatedFeature11>, <relatedFeature12>
    ]
  },
  {
    "objectId" : <objectId2>,
    "relatedRecords" : [
      <relatedFeature21>, <relatedFeature22>
    ]
  }
]
}
```

JSON Response Example

```
{
  "geometryType" : "esriGeometryPolygon",
  "spatialReference" : {
    "wkid" : 4267
  },
  "fields" : [
    {
      "name" : "OBJECTID",
      "type" : "esriFieldTypeOID",
      "alias" : "OBJECTID"},
    {
      "name" : "FIELD_KID",
      "type" : "esriFieldTypeString",
      "alias" : "FIELD_KID",
      "length" : 25},
    {
      "name" : "APPROXACRE",
      "type" : "esriFieldTypeDouble",
      "alias" : "APPROXACRE"},
    {
      "name" : "FIELD_NAME",
      "type" : "esriFieldTypeString",
      "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
                ]
              ]
            ]
          }
        }
      ]
    }
  ]
}
```


4.3 Feature (Map Service)

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

The Feature resource has two child resources:

- Attachment Infos: Returns information about attachments associated with this feature. This resource is available only if the layer has advertised that it has attachments.
- HTML Popup: Returns information about this feature that is intended for display in an HTML pop-up balloon.

4.3.1 Feature Reference

- **URL:** http://<layer-url>/<featureId>
- **Parent Resource:** Layer
- **Child Resources:** Attachment Infos, HTML Popup

4.3.2 Feature Parameters

Parameter	Details
f	Description: The response format Values: json | (other formats)

4.3.3 Feature Example

Example

```
http://myserver/rest/services/Hurricanes/tracks/ MapServer/0/1
```

JSON Response Syntax

```
{ "feature" : <feature> }
```

JSON Response Example

```
{
  "feature" :
  {
    "attributes" : {
      "OBJECTID" : 1,
      "BTID" : 1,
      "NAME" : "NOTNAMED",
      "LAT" : 28,
      "LONG" : -94.8,
      "WIND_KTS" : 80,
      "PRESSURE" : 0,
      "CAT" : "H1",
      "BASIN" : "North Atlantic",
      "TRACK_DATE" : -3740169600000,
      "Shape_Length" : 0.600000000000023
    }
    ,
    "geometry" :
    {
      "paths" :
      [
        [
          [-94.7999999999999, 28.0000000000001],
          [-95.3999999999999, 28.0000000000001]
        ]
      ]
    }
  }
}
```

4.4 Attachment Infos (Map Service)

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:

- Attachment: Streams the content of an individual attachment

4.4.1 Attachment Infos Reference

- **URL:** http://<feature-url>/attachments
- **Parent Resource:** Feature
- **Child Resource:** Attachment

4.4.2 Attachment Infos Parameters

Parameter	Details
f	Description: The response format Values: json | (other formats)

4.4.3 Attachment Infos Example

Example

```
http://myserver/rest/services/311Incidents/MapServer/0/
818654/attachments
```

JSON Response Syntax

```
{
 "attachmentInfos": [
  {
   "id": <attachmentId1>,
   "contentType": "<contentType1>",
   "size": <size1>,
   "name": "<name1>"
  },
  {
   "id": <attachmentId2>,
   "contentType": "<contentType2>",
   "size": <size2>,
   "name": "<name2>"
  }
 ]
}
```

JSON Response Example

```
{
 "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"
  }
 ]
}
```

4.5 Attachment (Map Service)

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.

The contents of the attachment are streamed to the client. If the attachment is not found, an HTTP status code of 404 (not found) is returned.

4.5.1 Attachment Reference

- **URL:** http://<attachmentinfos-url>/<attachmentId>
- **Parent Resource:** Attachment Infos

4.5.2 Attachment Example

Example

```
http://myserver/rest/services/311Incidents/MapServer/0/818654/
attachments/1
```

4.6 HTML Popup (Map Service)

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 esriServerHTMLPopupTypeNone.

4.6.1 HTML Popup Reference

URL: http://<feature-url>/htmlPopup

4.6.2 HTML Popup Parameters

Parameter	Details
f	Description: The response format Values: json | (other formats)

4.6.3 HTML Popup Example

Example

```
http://myserver/rest/services/StantonCounty/MapServer/0/1/
htmlPopup
```

JSON Response Syntax

```
{
  "htmlPopupType" : "<esriServerHTMLPopupTypeNone |
esriServerHTMLPopupTypeAsURL |
esriServerHTMLPopupTypeAsHTMLText>",
  "content": "<htmlContent>"
}
```

JSON Response Example

```
{
  "htmlPopupType" : "esriServerHTMLPopupTypeAsHTMLText",
  "content": "A <b>Sample HTML</b> pop up."
}
```

4.7 Image

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.

The image bytes are directly streamed to the client.

4.7.1 Image Reference

- **URL:** http://<layer-url>/images/<imageId>
- **Parent Resource:** Layer

4.7.2 Image Example

Example

```
http://myserver/rest/services/311Incidents/MapServer/
0/images/1DD4FC53
```

4.8 All Layers and Tables

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.

4.8.1 All Layers and Tables Reference

- **URL:** http:// <mapservice-url>/layers

4.8.2 All Layers and Tables Parameters

Parameter	Details
f	Description: The response format Values: json | (other formats)

4.8.3 All Layers and Tables Example

Example

```
http://myserver/rest/services/KSPetro/MapServer/layers
```

JSON Response Syntax

```
{
"layers" : [
   {
     <layer1>
   },
   {
     <layer2>
   }
 ],
 "tables" : [
   {
     <table1>
   },
   {
     <table2>
   }
 ]
}
```

JSON Response Example

```
{
  "layers" : [
    {
      "id" : 0,
      "name" : "Daily fish passage",
      "type" : "Feature Layer",
      "description" : "",
      "definitionExpression" : "",
      "geometryType" : "esriGeometryPoint",
      "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" : "esriServerHTMLPopupTypeAsHTMLText",
      "drawingInfo" : {"renderer" :
        {
          "type" : "simple",
          "symbol" :
          {
            "type" : "esriSMS",
            "style" : "esriSMSCircle",
```

J-9948


```
            "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" : "esriFieldTypeOID",
          "alias" : "OBJECTID"},
        {
          "name" : "SHAPE",
          "type" : "esriFieldTypeGeometry",
          "alias" : "SHAPE"},
        {
          "name" : "DATE",
          "type" : "esriFieldTypeDate",
          "alias" : "DAY",
          "length" : 8},
        {
          "name" : "STEELHEAD",
          "type" : "esriFieldTypeInteger",
          "alias" : "STEELHEAD"},
        {
          "name" : "SOCKEYE",
          "type" : "esriFieldTypeInteger",
          "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" : "esriServerHTMLPopupTypeNone",
      "displayField" : "NAME",
      "fields" : [
        {
          "name" : "OBJECTID",
          "type" : "esriFieldTypeInteger",
          "alias" : "OBJECTID"},
        {
          "name" : "NAME",
          "type" : "esriFieldTypeString",
          "alias" : "NAME",
          "length" : 50},
        {
          "name" : "X",
          "type" : "esriFieldTypeDouble",
          "alias" : "X"},
        {
          "name" : "Y",
          "type" : "esriFieldTypeDouble",
          "alias" : "Y"}
      ],
      "typeIdField" : null,
      "types" : null,
      "relationships" : [],
      "capabilities" : "Map,Query,Data"
    }
  ]
}
```


5.0 GEOCODE SERVICE

Geocoding is the process of assigning a location, usually in the form of coordinate values (points), to an address by comparing the descriptive location elements in the address to those present in the reference material. Addresses come in many forms, ranging from the common address format of a house number followed by the street name to other location descriptions such as postal zone or census tract. An address includes any type of information that distinguishes a place.

The GeoServices REST Specification Geocode Service resource accesses a web service that performs geocoding. The resource provides basic information associated with the service such as the service description, the address fields, spatial reference, and locator properties.

The Geocode Service resource supports two operations:

- Find Address Candidates: Returns a list of candidates based on address and location
- Reverse Geocode: Returns information about all the address fields pertaining to the reverse geocoded address, as well as the address's exact location

5.0.1 Geocode Service Reference

- **URL:** http://<catalog-url>/<serviceName>/GeocodeServer
- **Supported Operations:** Find Address Candidates, Reverse Geocode
- **Parent Resource:** Catalog

5.0.2 Geocode Service Parameters

Parameter	Details
f	Description: The response format Values: json | (other formats)

5.0.3 Geocode Service Example

Example

URL for the geocoding web service GeocodeUSA running on myserver:

```
http://myserver/rest/services/GeocodeUSA/GeocodeServer
```

JSON Response Syntax

```
{
"serviceDescription" : "<serviceDescription>",
"addressFields": [
  {"name" : "<fieldName1>", "alias" : "<fieldAlias1>", "required"
: <true | false>, "type" : "<fieldType1>"},
  {"name" : "<fieldName2>", "alias" : "<fieldAlias2>", "required"
: <true | false>, "type" : "<fieldType2>"}
],
//if the locator supports geocoding using single line address
"singleLineAddressField" :
{"name" : "<fieldName1>", "alias" : "<fieldAlias1>", "required" :
<true | false>, "type" : "<fieldType1>"},
"candidateFields": [
  {"name" : "<fieldName1>", "alias" : "<fieldAlias1>", "type" :
"<fieldType1>"},
  {"name" : "<fieldName2>", "alias" : "<fieldAlias2>", "type" :
"<fieldType2>"}
],
"intersectionCandidateFields": [
  {"name" : "<fieldName1>", "alias" : "<fieldAlias1>", "type" :
"<fieldType1>"},
  {"name" : "<fieldName2>", "alias" : "<fieldAlias2>", "type" :
"<fieldType2>"}
],
"spatialReference" : <spatialReference>,
"locatorProperties": {
  "<key1>" : "<value1>",
  "<key2>" : "<value2>"
}
}
```

JSON Response Example

```
{
"serviceDescription" : "Test Geocode Service Description",
"addressFields": [
  {"name" : "Street", "alias" : "Street or Intersection",
"required" : true, "type" : "esriFieldTypeString"},
  {"name" : "Zone", "alias" : "Zip Code", "required" : false,
"type" : "esriFieldTypeString"}
],
"singleLineAddressField" : {"name" : "Single Line Input","type" :
"esriFieldTypeString","alias" : "Full Address","required" : false}
,
"candidateFields": [
  {"name" : "Score", "alias" : "", "type" :
"esriFieldTypeSmallInteger"},
  {"name" : "StreetName", "alias" : "", "type" :
"esriFieldTypeString"}
],
"intersectionCandidateFields": [
  {"name" : "Score", "alias" : "", "type" :
"esriFieldTypeSmallInteger"},
  {"name" : "StreetName1", "alias" : "", "type" :
"esriFieldTypeString"}
],
"spatialReference" : {"wkid" : 4326},
"locatorProperties": {
  "MinimumCandidateScore" : "10",
  "SideOffsetUnits" : "ReferenceDataUnits",
  "SpellingSensitivity" : "80",
  "MinimumMatchScore" : "60",
  "IntersectionConnectors" : "& | @"
}
}
```

5.0.4 Find Address Candidates Operation

The Find Address Candidates operation is performed on a Geocode Service resource. The result of this operation is a resource representing the list of address candidates. This resource provides information about candidates, including the address, location, and score. Users can provide arguments to the Find Address Candidates operation as query parameters.

5.0.4.1 Find Address Candidates Reference

- **URL:** http://<geocodeservice-url>/findAddressCandidates
- **Parent Resource:** Geocode Service

5.0.4.2 Find Address Candidates Parameters

Parameter	Details
f	Description: The response format Values: json | (other formats, for example, kmz)

Parameter	Details
<addressField1>, <addressField2>, ...	Description: The various address fields accepted by the corresponding geocode service. These fields are listed in the addressFields property of the JSON representation associated Geocode Service resource. Example: Suppose that addressFields of a Geocode Service resource includes fields with the following names: Street, City, State, and Zone. To perform the Find Address Candidates operation by providing values for Street and Zone, set the query parameters as such: `Street=380+New+York+St&Zone=92373.`
outfields	Description: The list of fields to be included in the returned result set. This is a comma-delimited list of field names. If the shape field is specified in the list of return fields, it is ignored. For nonintersection addresses, specify the candidate fields from the Geocode Service resource. For intersection addresses, specify the intersection candidate fields from the Geocode Service resource; outFields=* returns all fields. Example: `outFields=StreetName,StreetType`
outSR	Description: The well-known ID of the spatial reference or a spatial reference JSON object for the returned address candidates

5.0.4.3 Find Address Candidates Example

Example

Geocode an address (380 New York Street, Redlands, CA 92373):

```
http://myserver/rest/services/GeocodeUSA/GeocodeServer/findAddress
Candidates?Address=380+New+York+Street&City=
Redlands&State=CA&Zip=92373
```

JSON Response Syntax

```
{
"spatialReference" : <spatialReference>,
"candidates" : [
  {
  "address" : "<address1>",
  "location" : { <point1> },
  "score" : <score1>,
  "attributes" : {<fieldName1> : <value11>, <fieldName2> :
<value12>}
  },
  {
  "address" : "<address2>",
  "location" : { <point2> },
  "score" : <score2>,
```

```
  "attributes" : {<fieldName1> : <value21>, <fieldName2> :
<value22>}
  }
]
}
```

JSON Response Example

```
{
"spatialReference": {"wkid" : 4326},
"candidates" : [
  {
  "address" : "1 MASON ST",
  "location" : { "x" : -122.408951, "y" : 37.783206 },
  "score" : 75,
  "attributes" : {"StreetName" : "MASON", "StreetType" : "ST"}
  },
  {
  "address" : "49 MASON ST",
  "location" : { "x" : -122.408986, "y" : 37.783460 },
  "score" : 27,
  "attributes" : {"StreetName" : "MASON", "StreetType" : "ST"}
  }
]
}
```

5.0.5 Reverse Geocode Operation

The Reverse Geocode operation is performed on a Geocode Service resource. The result of this operation provides information about all the address fields pertaining to the reverse geocoded address, as well as its exact location. Users can provide arguments to the Reverse Geocode operation as query parameters.

5.0.5.1 Reverse Geocode Reference

- **URL:** http://<geocodeservice-url>/reverseGeocode
- **Parent Resource:** Geocode Service

5.0.5.2 Reverse Geocode Parameters

Parameter	Details
f	Description: The response format Values: json | (other formats, for example, kmz)

location	Description: The point at which to search for the closest address. The structure of the point is the same as the structure of the JSON point object returned by the GeoServices REST Specification. In addition to the JSON structure, the location can be specified with a simpler comma-separated syntax.
	
	Syntax:
	
	■ JSON structure:
	`location={point}`
	
	■ Simple syntax:
	`location=<x>,<y>`

Parameter	Details
	Examples: ■ `location={x: -122.4, y: 37.7}` ■ `location=-122.4,37.7` If not specified in the JSON object or if using the simple comma-separated syntax, the location is assumed to be in the same spatial reference as that of the geocode service.
distance	Description: The distance in meters from the given location within which a matching address should be searched. If this parameter is not provided or an invalid value is provided, a default value of 0 meters is used. Example: `distance=100`
outSR	Description: The well-known ID of the spatial reference or a spatial reference JSON object for the returned address candidates

5.0.5.3 Reverse Geocode Example

Example

Reverse geocode based on a point:

```
http://myserver/rest/services/GeocodeUSA/GeocodeServer/reverseGeoc
ode?location=-117.195681386,34.057517097&distance=0
```

JSON Response Syntax

```
{
"address" : {
  "<fieldName1>" : "<fieldValue1>",
  "<fieldName2>" : "<fieldValue2>"
},
"location" :  { <point> }
}
```

JSON Response Example

```
{
"address" : {
  "Street" : "771 TUNNEL AVE",
  "Zone" : "94005"
},
"location" :  { "x" : -122.400260954336, "y" :
37.7000445053795,"spatialReference": {"wkid": 4269} }
}
```

6.0 GP Service

Geoprocessing (GP) is a fundamental part of enterprise GIS operations. Geoprocessing provides the data analysis, management, and conversion tools necessary for all GIS users.

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

Use a GP service to do the following:

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

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

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

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

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

6.0.1 GP Service Reference

- **URL:** http://<catalog-url>/<serviceName>/GPServer
- **Parent Resource:** Catalog
- **Child Resource:** Task

6.0.2 GP Service Parameters

Parameter	Details
f	Description: The response format Values: json | (other formats)

6.0.3 GP Service Example

Example

URL for a geoprocessing service:

```
http://myserver/rest/services/MyCustomTools/GPServer
```

JSON Response Syntax

```
{
"serviceDescription" : "<serviceDescription>",
"tasks": ["<taskName1>", "<taskName2>"],
"executionType": "<executionType>",
"resultMapServerName": "<resultMapServerName>"
}
```

JSON Response Example

```
{
"serviceDescription" : "Test Geoprocessing Service Description",
"tasks": [
"BufferPointsByRef",
"BufferLinesByRef",
"BufferPolygonsByRef"
],
"executionType": "esriExecutionTypeAsynchronous",
"resultMapServerName": "BufferByRef"
}
```

6.1 Task

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

The GP task supports two operations:

- Execute Task: Used for synchronous tasks. *Synchronous* means that the application will wait while the tool executes on the server. Because the end user must wait, it should be determined if the wait time is acceptable for the type of application.

- Submit Job: Used for *asynchronous* tasks. Asynchronous means that the application does not wait for the task to finish execution, and the end user can continue using the application.

6.1.1 Task Reference

- **URL:** http://<gpservice-url>/<taskName>
- **Supported Operations:** Execute Task, Submit Job
- **Parent Resource:** GP Service
- **Child Resource:** Job

6.1.2 Task Parameters

Parameter	Details
f	Description: The response format Values: json | (other formats)

6.1.3 Task Example

Example

URL for a geoprocessing task CreateDriveTimePolygons:

```
http://myserver/rest/services/MyCustomTools/GPServer/
CreateDriveTimePolygons
```

JSON Response Syntax

```
{
"name" : "<taskName>",
"displayName" : "<displayName>",
"category" : "<category>",
"helpUrl" : "<url>",
"executionType": "<executionType>",
"parameters": [
{
    "name" : "<paramName1>",
    "dataType" : "<dataType1>",
    "displayName" : "<displayName1>",
    "direction" : "<direction1>",
    "defaultValue" : {<defaultValue1>},
    "parameterType" : "<parameterType1>",
    "category" : "<paramCategory1>",
    "choiceList" : ["choice11", "choice12"]
},
{
    "name" : "<paramName2>",
    "dataType" : "<GPFeatureRecordSetLayer | GPRecordSet>",
    "displayName" : "<displayName2>",
    "direction" : "<direction2>",
    "defaultValue" : {
        "geometryType" : "<geometryType2>",
        "spatialReference" : {<spatialReference>},
        "fields" : [
            {"name" : "<fieldName21>", "type" : "<fieldType21>"},
            {"name" : "<fieldName22>", "type" : "<fieldType22>"}
        ]
    },
```

```
    "parameterType" : "<parameterType2>",
    "category" : "<paramCategory2>",
    "choiceList" : ["choice21", "choice22"]
}
]
}
```

JSON Response Example

```
{
"name" : "BufferPoints",
"displayName" : "BufferPoints",
"category" : "",
"helpUrl" : "http://myserver/output/BufferByVal/BufferPoints.htm",
"executionType": "esriExecutionTypeAsynchronous",
"parameters": [
{
    "name" : "Input_Points",
    "dataType" : "GPFeatureRecordSetLayer",
    "displayName" : "Input Points",
    "direction" : "esriGPParameterDirectionInput",
    "defaultValue" : {
        "geometryType" : "esriGeometryPoint",
        "spatialReference" : {"wkid" : 4326},
        "fields" : [
            {"name" : "FID", "type" : "esriFieldTypeOID"},
            {"name" : "Shape", "type" : "esriFieldTypeGeometry"},
            {"name" : "Id", "type" : "esriFieldTypeInteger"}
        ]
    },
    "parameterType" : "esriGPParameterTypeRequired",
    "category" : "",
    "choiceList" : []
},
{
    "name" : "Distance",
    "dataType" : "GPLinearUnit",
    "displayName" : "Distance",
    "direction" : "esriGPParameterDirectionInput",
    "defaultValue" : { "distance" : 100.0, "units" : "esriMiles"},
    "parameterType" : "esriGPParameterTypeRequired",
    "category" : "",
    "choiceList" : []
},
{
    "name" : "Input_Points_By_Ref",
    "dataType" : "GPString",
    "displayName" : "Input Points By Ref",
    "direction" : "esriGPParameterDirectionInput",
    "defaultValue" : "fourptsinacol",
    "parameterType" : "esriGPParameterTypeRequired",
    "category" : "",
    "choiceList" : [

"fourptsinacol","fourptsinarow","partiallines","line_shp","partial
polys","polygon_shp"
    ]
},
```

```
{
    "name" : "Output_Buffered_Points",
    "dataType" : "GPFeatureRecordSetLayer",
    "displayName" : "Output Buffered Points",
    "direction" : "esriGPParameterDirectionOutput",
    "defaultValue" : {
        "geometryType" : "esriGeometryPoint",
        "spatialReference" : {"wkid" : 4326},
        "fields" : [
            {"name" : "FID", "type" : "esriFieldTypeOID"},
            {"name" : "Shape", "type" : "esriFieldTypeGeometry"},
            {"name" : "Id", "type" : "esriFieldTypeInteger"}
            {"name" : "Name", "type" : "esriFieldTypeString"}
        ]
    },
    "parameterType" : "esriGPParameterTypeRequired",
    "category" : "",
    "choiceList" : []
}
]
}
```


6.1.4 Execute Task Operation

The Execute Task operation is performed on a Task resource for GP services when the execution type is synchronous. The result of this operation is a Result resource that contains an array of result parameters and the GP task execution messages. Each result parameter provides information such as the parameter name; the data type; and, most importantly, the value for that parameter.

The values have different structures based on the data types of the parameters. Details about values for every data type are included in Section 6.3.4 Result Parameter Values.

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

6.1.4.1 Execute Task Reference

- **URL:** http://<task-url>/execute
- **Parent Resource:** Task

6.1.4.2 Execute Task Parameters

Parameter	Details
f	Description: The response format Values: json | (other formats, for example, kmz or amf)
<gpParameter1>, <gpParameter2>, ...	Description: The various input parameters accepted by the corresponding GP task. These parameters are listed in the parameters property of the JSON representation associated with the Task resource. The valid values for the input parameters are dependent on the

Parameter	Details
	data type of the parameter. These values are discussed in more detail in Section 6.1.5 Submit Job Operation.
env:outSR	Description: The spatial reference of the output geometries. The spatial reference can be specified as either a well-known ID or a spatial reference JSON object. If a value for env:outSR is not specified, the output geometries are in the spatial reference of the input geometries. If a value for env:processSR is specified and one for env:outSR is not specified, the output geometries are in the spatial reference of the process spatial reference.
env:processSR	Description: The spatial reference that the model will use to perform geometry operations. The spatial reference can be specified as either a well-known ID or a spatial reference JSON object. If a value for env:processSR is specified but one for env:outSR is not specified, the output geometries are in the spatial reference of the process spatial reference.

6.1.4.3 Execute Task Examples

With this operation, the majority of the parameters included in the request are dependent on the input parameter types of the geoprocessing task being executed. Note that the GP Execute Task and Submit Job operations share the same syntax for all input parameter types, as described in Section 6.1.6 Input Parameter Values.

Example 1

Execute a MessageInABottle task designed to find the location of a bottle thrown into the ocean at a particular coordinate (in this case, [0,0]) after a given number of days (in this case, 50):

```
http://myserver/rest/services/MyCustomTools/GPServer/
MessageInABottle/execute?Input_Point={"features":
[{"geometry":{"x":0,"y":0}}]}&Days=50
```

Example 2

Execute a task similar to example 1 but request the output spatial reference to be Web Mercator (WKID 102113):

```
http://myserver/rest/services/MyCustomTools/GPServer/
MessageInABottle/execute?Input_Point={"features":
[{"geometry":{"x":0,"y":0}}]}&Days=50&env:outSR=102113
```

JSON Response Syntax

```
  {
"results" : [
  {
  "paramName" : "<paramName1>",
  "dataType" : "<dataType1>",
```

```
  "value" : <valueLiteralOrObject1>
  },
  {
  "paramName" : "<paramName2>",
  "dataType" : "<dataType2>",
  "value" : <valueLiteralOrObject2>
  }
],
"messages" : [
  {"type" : "<type1>", "description" : "<description1>"},
  {"type" : "<type2>", "description" : "<description2>"}
]
}
```

JSON Response Example

```
{
"results" : [
  {
  "paramName" : "Output_String",
  "dataType" : "GPString",
  "value" : "Test String"
  },
  {
  "paramName" : "Output_Double",
  "dataType" : "GPDouble",
  "value" : 545.64
  },
  {
  "paramName" : "Output_Linear_Unit",
  "dataType" : "GPLinearUnit",
  "value" : { "distance" : 123.45, "units" : "esriKilometers" }
  }
],
"messages" : [
  {"type" : "esriJobMessageTypeInformative", "description" :
"Executing (TestTask)"},
  {"type" : "esriJobMessageTypeInformative", "description" :
"Start Time: Thu Jul 05 16:36:25 2007"},
  {"type" : "esriJobMessageTypeInformative", "description" :
"Executing Copy Features..."},
  {"type" : "esriJobMessageTypeInformative", "description" :
"Start Time: Thu Jul 05 16:36:25 2007"},
  {"type" : "esriJobMessageTypeInformative", "description" :
"Executed (Copy Features) successfully."},
  {"type" : "esriJobMessageTypeInformative", "description" : "End
Time: Thu Jul 05 16:36:26 2007 (Elapsed Time: 1.00 seconds)"},
  {"type" : "esriJobMessageTypeInformative", "description" :
"Executed (TestTask) successfully."},
  {"type" : "esriJobMessageTypeInformative", "description" : "End
Time: Thu Jul 05 16:36:26 2007 (Elapsed Time: 1.00 seconds)"}
]
}
```

6.1.5 Submit Job Operation

The Submit Job operation is performed on an asynchronous Task resource. The result of this operation is a Job resource. Users can provide arguments to the Submit Job operation as query parameters.

6.1.5.1 Submit Job Reference

- **URL:** http://<task-url>/submitJob
- **Parent Resource:** Task

6.1.5.2 Submit Job Parameters

Parameter	Details
f	Description: The response format Values: json | (other formats, for example kmz)
<gpParameter1>, <gpParameter2>, ...	Description: The various input parameters accepted by the corresponding GP task. These parameters are listed in the parameters property of the JSON representation associated with the Task resource. The valid values for the input parameters are dependent on the data type of the parameter.
env:outSR	Description: The spatial reference of the output geometries. The spatial reference can be specified as either a well-known ID or a spatial reference JSON object. If a value for env:outSR is not specified, the output geometries are in the spatial reference of the input geometries. If a value for env:outSR is not specified but one for env:processSR is specified, the output is in the spatial reference of the process spatial reference.
env:processSR	Description: The spatial reference that the model will use to perform geometry operations. The spatial reference can be specified as either a well-known ID or a spatial reference JSON object. If a value for env:outSR is not specified but one is specified for env:processSR, the output geometries are in the spatial reference of the process spatial reference.

6.1.5.3 Submit Job Examples

With this operation, the majority of the parameters included in the request are dependent on the input parameter types of the geoprocessing task being submitted. Note that the Execute Task and Submit Job operations share the same syntax for all input parameter types, as described in Section 6.1.6 Input Parameter Values.

Example 1

Submit a job to a MailingList task for parcel ID 1N1E34CC-06600 within a search distance of 100 feet:

```
http://myserver/rest/services/MyCustomTools/GPServer/MailingList/
submitJob?Parcel_ID=1N1E34CC  -06600&SearchDistance_ft=100
```

Example 2

Submit a job to the task similar to example 1 but request the output spatial reference to be Web Mercator (WKID 102113):

```
http://myserver/rest/services/MyCustomTools/GPServer/MailingList/s
ubmitJob?Parcel_ID=1N1E34CC -
06600&SearchDistance_ft=100&env:outSR=102113
```

JSON Response Syntax

See the JSON response syntax for the Job resource.

JSON Response Example

See the JSON response example for the Job resource.

6.1.6 Input Parameter Values

The values provided for task input parameters are dependent on the data type of the parameter. The following examples show the input parameter syntax based on the parameter data type.

6.1.6.1 GPBoolean, GPDouble, GPLong, and GPString Input Parameters

For the simple data types and the parameter values GPBoolean, GPDouble, GPLong, and GPString, use their literal values.

```
InputBoolean=true&InputDouble=345.678&InputLong=345&InputString=My
String
```

6.1.6.2 GPLinearUnit Input Parameter

The parameter value for GPLinearUnit is a JSON structure with the following fields:

- distance: A double value
- units: A string with unit values such as "esriMeters" or "esriMiles"

```
{ "distance" : 345.678, "units" : "esriMiles" }
```

6.1.6.3 GPFeatureRecordSet Layer and GPRecordSet Input Parameters

The parameter value for GPFeatureRecordSetLayer and GPRecordSet is a JSON structure containing either one of these fields: features or url.

The features field is an array of features. Each feature in turn contains the following fields:

- geometry: Can be points, lines, or polygons. The structure for the geometries is the same as the structure of the JSON geometry objects returned by the GeoServices REST Specification.

- attributes: Key-value pairs where the key is a field name in the list of fields of the record set and the value is the value for the corresponding field.

Note that values for GPFeatureRecordSetLayer contain the geometry field, while values for GPRecordSet do not.

```
{
"features"  : [
{
    "geometry" : {"x" : -104.44, "y" : 34.83},
    "attributes" : {"Id" : 43, "Name" : "Feature 1"}
},
{
    "geometry" : {"x" : -100.65, "y" : 33.69},
    "attributes" : {"Id" : 67, "Name" : "Feature 2"}
}
]
}
```

For "schema-less" input features—those feature sets where the geometry type or spatial reference is not specified by the model publisher—users must provide the geometry type and the spatial reference in addition to the features as specified below:

```
{
"geometryType" : "esriGeometryPoint",
"spatialReference" : {"wkid" : 4326},
"features"  : [
{
    "geometry" : {"x" : -104.44, "y" : 34.83},
    "attributes" : {"Id" : 43, "Name" : "Feature 1"}
},
{
    "geometry" : {"x" : -100.65, "y" : 33.69},
    "attributes" : {"Id" : 67, "Name" : "Feature 2"}
}
]
}
```

The geometry type can be esriGeometryPoint, esriGeometryPolyline, or esriGeometryPolygon. If the geometry type is not specified, it is assumed to be esriGeometryPoint. If the spatial reference is not specified, it defaults to an unknown coordinate system.

For a large set of geometries, users can specify a URL to the input geometries stored in a JSON structure in a file on a public server.

```
{ "url" : "http://myserver/myfeatures/afile.txt" }
```

6.1.6.4 GPDate Input Parameter

The parameter value for GPDate is a number that represents the number of milliseconds since epoch (January 1, 1970) in UTC.

```
InputDate=1199145600000 // 1 Jan 2008 00:00:00 GMT
```

6.1.6.5 GPDataFile Input Parameter

The parameter value for GPDataFile is a JSON structure with a url field. The value of the url field is a URL to the location of the input data file.

```
{ "url" : "http://myserver/myfile" }
```

6.1.6.6 GPRasterData and GPRasterDataLayer Input Parameters

The parameter value for GPRasterData and GPRasterDataLayer data types is a JSON structure with the following fields:

- url: URL to the location of the input raster data file.
- format: The format of the raster data—JPEG, TIFF, and so on

```
{ "url" : "http://myserver/lake.tif", "format" : "tif" }
```

6.1.6.7 GPMultiValue Input Parameter

The fully qualified data type for a GPMultiValue parameter is GPMultiValue:<memberDataType>, where memberDataType is one of the data types defined above, for example, GPMultiValue:GPString or GPMultiValue:GPLong.

The parameter value for GPMultiValue data types is a JSON array. Each element in this array is of the data type as defined by the memberDataType suffix of the fully qualified GPMultiValue data type name.

GPMultiValue: GPStringData Type Example

```
["Parcels", "Street Lights"]
```

6.2 Job

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

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

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

6.2.1 Job Reference

- **URL:** http://<task-url>/jobs/<job-id>
- **Parent Resource:** Task
- **Child Resources:** Result, Input

6.2.2 Job Parameters

Parameter	Details
f	Description: The response format Values: json | (other formats, for example, kmz)

6.2.3 Job Example

Example

When submitting a job to the Mailing List task as shown in the Submit Job example, the user will be redirected to the Job resource page whose URL might be similar to the following:

```
http://myserver/rest/services/MyCustomTools/GPServer/MailingList/
jobs/jdb4cce48c464424e913c15c4f419b32a
```

Note that jdb4cce48c464424e913c15c4f419b32a is an example job ID; job IDs might vary.

JSON Response Syntax

```
{
"jobId" : "<jobId>",
"jobStatus" : "<jobStatus>",
"results" : {
"<param1>" : {"paramUrl" : "<relativeUrlToParam1>"},
"<param2>" : {"paramUrl" : "<relativeUrlToParam2>"}
},
"messages" : [
{"type" : "<type1>", "description" : "<description1>"},
{"type" : "<type2>", "description" : "<description2>"}
]
}
```

JSON Response Example

```
{
"jobId" : "JE30155054C2C416EA3CF6C88A40B94FF",
"jobStatus" : "esriJobSucceeded",
"results" : {
"Output_Long" : {"paramUrl" : "results/Output_Long"},
"Output_Feature_Class" : {"paramUrl" :
"results/Output_Feature_Class"}
},
"inputs" : {
"Input_String" : {"paramUrl" : "inputs/Input_String"},
"Input_Feature_Class" : {"paramUrl" :
"inputs/Input_Feature_Class"}
},
"messages" : [
{"type" : "esriJobMessageTypeInformative", "description" :
"Executing (TestTask)"},
{"type" : "esriJobMessageTypeInformative", "description" : "Start
Time: Thu Jul 05 16:36:25 2007"},
{"type" : "esriJobMessageTypeInformative", "description" :
```

```
"Executing Copy Features..."},
{"type" : "esriJobMessageTypeInformative", "description" : "Start
Time: Thu Jul 05 16:36:25 2007"},
{"type" : "esriJobMessageTypeInformative", "description" :
"Executed (Copy Features) successfully."},
{"type" : "esriJobMessageTypeInformative", "description" : "End
Time: Thu Jul 05 16:36:26 2007 (Elapsed Time: 1.00 seconds)"},
{"type" : "esriJobMessageTypeInformative", "description" :
"Executed (TestTask) successfully."},
{"type" : "esriJobMessageTypeInformative", "description" : "End
Time: Thu Jul 05 16:36:26 2007 (Elapsed Time: 1.00 seconds)"}
]
}
```

6.3 Result

The Result resource represents a result parameter for a GP job. It provides information about the result parameter such as its name, data type, and value. The value is the most important piece of information provided by this resource. Based on the data type of the parameter, the values provide different types of information. Given this fact, the value will have different structures based on the data type as defined below.

6.3.1 Result Reference

- **URL:** http://<job-url>/results/<param-name>
- **Parent Resource:** Job

6.3.2 Result Parameters

Parameter	Details
f	Description: The response format Values: json | image | (other formats, for example, kmz or amf)
outSR	Description: The spatial reference of the output geometries. The spatial reference can be specified as either a well-known ID or a spatial reference JSON object. This parameter is applicable for result parameters that contain geometries. This parameter can be used to return the geometries in a spatial reference that is different from the spatial reference in which the outputs were initially created.
returnType	Description: If the GP service is configured to return map images to be displayed with a certain map service (as described by the resultMapServerName property on the GP Service resource), the default output for the GPRasterDataLayer and GPFeatureSetLayer parameters is a map image. However, users can explicitly request the raw raster data by using returnType and setting its value to data. Value: data

6.3.3 Result Example

Example

For a Mailing List task job as described in the Job Example section, the user gets access to its Report_html result parameter with the following URL:

```
http://myserver/rest/services/MyCustomTools/GPServer/MailingList/jobs/jdb4cce48c464424e913c15c4f419b32a/results/Report_html
```

Note that jdb4cce48c464424e913c15c4f419b32a is an example job ID; job IDs might vary.

JSON Response Syntax

```
{
"paramName" : "<paramName>",
"dataType" : "<dataType>",
"value" : <valueLiteralOrObject>
}
```

JSON Response Example

```
{
"paramName" : "Output_String",
"dataType" : "GPString",
"value" : "TestString"
}
```

6.3.4 Result Parameter Values

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

6.3.4.1 GPBoolean, GPDouble, GPLong, and GPString Result Parameters

These simple data types have parameter values that are their literal values.

JSON Response Syntax

```
{
"paramName" : "<paramName>",
"dataType" : "<GPBoolean | GPDouble | GPLong | GPString>",
"value" : <valueLiteral>
}
```

JSON Response Example

```
{
"paramName" : "Output_Double",
"dataType" : "GPDouble",
"value" : 1234.56
}
```

6.3.4.2 GPDate Result Parameter

The value for the GPDate data type is a number that represents the number of milliseconds since epoch (January 1, 1970) in UTC.

JSON Response Syntax

```
{
 "paramName": "<paramName>",
 "dataType": "GPDate",
 "value": <millisecondsSinceEpoch>
}
```

JSON Response Example

```
{
 "paramName": "Output_Date",
 "dataType": "GPDate",
 "value": 1199145600000 // 1 Jan 2008 00:00:00 GMT
}
```

6.3.4.3 GPLinearUnit Result Parameter

The parameter value for GPLinearUnit is a JSON structure with the following fields:

- distance: A double value
- units: A string whose values can be "esriMeters", "esriMiles", and so forth

JSON Response Syntax

```
{
"paramName" : "<paramName>",
"dataType" : "GPLinearUnit",
"value" : { "distance" : <distance>, "units" : "<units>" }
}
```

JSON Response Example

```
{
"paramName" : "Output_Linear_Unit",
"dataType" : "GPLinearUnit",
"value" : { "distance" : 1234.56, "units" : "esriMiles" }
}
```

6.3.4.4 GPDataFile Result Parameter

The parameter value for GPDataFile is a JSON structure with a url field. The value of the url field is a URL to the location of the data file.

JSON Response Syntax

```
{
"paramName" : "<paramName>",
"dataType" : "GPDataFile",
"value" : { "url" : "<url>" }
}
```

JSON Response Example

```
{
"paramName" : "Output_File",
"dataType" : "GPDataFile",
"value" :
 {
 "url" :
"http://myserver/jobs/ByValTools_GPServer/J1E7A1738AC054CDCBFC4A41
3DD9033CE/scratch/output.txt"
 }
}
```

6.3.4.5 GPRasterData Result Parameter

The parameter value for GPRasterData is a JSON structure with the following fields:

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

JSON Response Syntax

```
{
"paramName" : "<paramName>",
"dataType" : "GPRasterData",
"value" : { "url" : "<url>", "format" : "<format>" }
}
```

JSON Response Example

```
{
"paramName" : "Output_Raster",
"dataType" : "GPRasterData",
"value" :
 {
 "url" :
"http://myserver/jobs/ByValTools_GPServer/JD613584CA6AC462AB8229A9
A27B3DA79/scratch/slpgrd.tif",
 "format" : "tif"
 }
}
```

6.3.4.6 GPRecordSet Result Parameter

The parameter value for GPRecordSet is a JSON structure with a field called features.

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

The exceededTransferLimit property is true only if the number of records exceeds the maximum number configured by the server administrator. Otherwise, it is false.

JSON Response Syntax

```
{
"paramName" : "<paramName>",
"dataType" : "GPRecordSet",
"value" :
```

```
 {
 "features" : [
 {
 "attributes" :
 {
 "<field1>" : <value11>,
 "<field2>" : <value12>
  }
  },
 {
 "attributes" :
 {
 "<field1>" : <value21>,
 "<field2>" : <value22>
  }
 }
 ],
"exceededTransferLimit" : false | true
 }
}
```

JSON Response Example

```
{
"paramName" : "Output_Record_Set",
"dataType" : "GPRecordSet",
"value" :
 {
 "features" : [
 {
 "attributes" :
 {
 "TextField" : "a",
 "IntField" : 1234,
 "DoubleField" : 1234.56,
 "DateField" : "Sun Apr 10 16:00:00 PST 1977"
  }
  },
 {
 "attributes" :
 {
 "TextField" : "b",
 "IntField" : 5678,
 "DoubleField" : 5678.91,
 "DateField" : "Mon Apr 11 16:00:00 PST 1977"
  }
  }
 ],
"exceededTransferLimit" : false
 }
}
```

6.3.4.7 Map Images as Geoprocessing Results

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

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

JSON Response Syntax

```
{
"paramName" : "<paramName>",
"dataType" : "<GPRasterDataLayer | GPFeatureRecordSetLayer>",
"value" :
 {
 "mapImage" :
 {
 "href" : "<href>",
 "width" : <width>,
 "height" : <height>,
 "extent" : {<envelope>},
 "scale" : <scale>
  }
 }
}
```

JSON Response Example

```
{
"paramName" : "Output_Raster_Layer",
"dataType" : "GPRasterDataLayer",
"value" :
 {
 "mapImage" :
 {
 "href" :
"http://myserver/output/map40a7f57f31474933a94b5c672b7205f0.png",
 "width" : 400,
 "height" : 400,
 "extent" : {
 "xmin" : -109.55, "ymin" : 25.76, "xmax" : -86.39, "ymax" :
49.94,
 "spatialReference" : {"wkid" : 4326}
 },
```

```
 "scale" : 2.53E7
  }
 }
}
```

6.3.4.8 GPRasterDataLayer Result Parameter

If the GP service is configured to return map images for display with a certain map service (as described by the resultMapServerName property on the GP Service resource), the default output for GPRasterDataLayer parameters is a map image. However, users can explicitly request the raw raster data by using the returnType parameter in the URL and setting its value to data.

If the GP service is not configured to return map images, or if the returnType parameter is set to data, the parameter value for GPRasterDataLayer is a JSON structure with the following fields:

- url: A URL to the location of the raw raster data
- format: A string representing the format of the raster

JSON Response Syntax

```
{
"paramName" : "<paramName>",
"dataType" : "GPRasterDataLayer",
"value" : { "url" : "<url>", "format" : "<format>" }
}
```

JSON Response Example

```
{
"paramName" : "Output_Raster_Layer",
"dataType" : "GPRasterDataLayer",
"value" :
 {
 "url" :
"http://myserver/jobs/ByRefTools_GPServer/J3D1737BA4584441FACBD556
3AD1A47D5/scratch/outrast.tif",
 "format" : "tif"
 }
}
```

6.3.4.9 GPFeatureRecordSet Layer Result Parameter

If the GP service is configured to return map images for display with a certain map service (as described by the resultMapServerName property on the GP Service resource), the default output for GPFeatureRecordSetLayer parameters is a map image. However, users can explicitly request the raw raster data by using the returnType parameter in the URL and setting its value to data.

If the GP service is not configured to return map images, or if the returnType parameter is set to data, the parameter value for GPFeatureRecordSetLayer is a JSON structure with the following fields:

- features: An array of features. Each feature contains the following fields:
  - geometry: Points, lines, or polygons. The structure for the geometries is the same as the structure of the JSON geometry objects returned by the GeoServices REST Specification.
  - attributes: Key-value pairs where the key is a field name in the list of fields of the record set and the value is the value of the corresponding field.
- spatialReference: The well-known ID of a spatial reference
- geometryType:The geometry type of the layer

JSON Response Syntax

```
{
"paramName" : "<paramName>",
"dataType" : "GPFeatureRecordSetLayer",
"value" :
 {
 "geometryType" : "<geometryType>",
 "spatialReference" : {<spatialReference>},
 "features" : [
 {
 "geometry" : {<geometry1>},
 "attributes" :
 {
 "<field1>" : <value11>,
 "<field2>" : <value12>
  }
  },
 {
 "geometry" : {<geometry2>},
 "attributes" :
 {
 "<field1>" : <value21>,
 "<field2>" : <value22>
  }
  }
 ]
 }
}
```

JSON Response Example

```
{
"paramName" : "Output_Features",
"dataType" : "GPFeatureRecordSetLayer",
"value" :
 {
 "geometryType" : "esriGeometryPoint",
 "spatialReference" : {"wkid" : 4326},
 "features" : [
 {
 "geometry" : {"x" : -104.36, "y" : 34.657},
 "attributes" :
```

```
 {
 "TextField" : "a",
 "IntField" : 1234,
 "DoubleField" : 1234.56,
 "DateField" : "Sun Apr 10 16:00:00 PST 1977"
  }
 },
 {
 "geometry" : {"x" : -114.749, "y" : 31.439},
  "attributes" :
 {
 "TextField" : "b",
 "IntField" : 5678,
 "DoubleField" : 5678.91,
 "DateField" : "Mon Apr 11 16:00:00 PST 1977"
  }
 }
 ]
 }
}
```

6.3.4.10 GPMultiValue Result Parameter

The fully qualified data type for a GPMultiValue parameter is GPMultiValue:<memberDataType>, where memberDataType is one of the data types defined above, for example, GPMultiValue:GPString or GPMultiValue:GPLong.

The parameter value for GPMultiValue data types is a JSON array. Each element in this array is of the data type as defined by the memberDataType suffix of the fully-qualified GPMultiValue data type name.

GPMultiValue: GPString Example

```
{
 "paramName": "Output_Layers",
 "dataType": "GPMultiValue:GPString",
 "value": ["Parcels", "Street Lights"]
}
```

6.4 Input

The Input resource represents an input parameter for a GP job. It provides information about the input parameter such as its name, data type, and value. The value is the most important piece of information provided by this resource.

The information provided by each input parameter is identical to the information provided by the Result resource parameter. Based on the data type of the parameter, the values provide different types of information. Given this fact, the value will have different structures based on the data type. Details about values for every data type are included in Section 6.1.6 Input Parameter Values.

6.4.1 Input Reference

- **URL:** http://<job-url>/inputs/<param-name>
- **Parent Resource:** Job

6.4.2 Input Parameters

Parameter	Details
f	Description: The response format Values: json | (other formats, for example, kmz)

6.4.3 Input Example

Example

For a Mailing List task job as described in the Job example, the user gets access to its SearchDistance_ft input parameter with the following URL:

```
http://myserver/rest/services/MyCustomTools/GPServer/MailingList/jobs/jdb4cce48c464424e913c15c4f419b32a/inputs/SearchDistance_ft
```

Note that jdb4cce48c464424e913c15c4f419b32a is an example job ID; job IDs might vary.

JSON Response Syntax

```
{
"dataType" : "<dataType>",
"value" : <ValueLiteralOrObject>
}
```

JSON Response Example

```
{
"paramName" : "Input_String",
"dataType" : "GPString",
"value" : "TestString"
}
```

7.0 GEOMETRY SERVICE

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

- Project: Returns an array of projected geometries
- Simplify: Returns an array of simplified geometries
- Buffer: Returns an array of polygons at the specified distances for the input geometry (An option is available to union buffer polygons at each distance.)
- Areas and Lengths: Calculates areas and perimeter lengths for each polygon specified in the input array
- Lengths: Calculates the lengths of each polyline specified in the input array

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

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

Note that geometry input and output, where required, are always packaged as arrays.

7.0.1 Geometry Service Reference

- **URL:** http://<catalog-url>/<serviceName>/GeometryServer
- **Supported Operations:** Project, Simplify, Buffer, Areas and Lengths, Lengths, Relation, Label Points, Auto Complete, Convex Hull, Cut, Densify, Difference, Generalize, Intersect, Offset, Reshape, Trim/Extend, Union
- Parent Resource: Catalog

7.0.2 Geometry Service Parameters

Parameter	Details
f	Description: The response format Values: json | (other formats)

7.0.3 Geometry Service Example

Example

URL for a geometry service on myserver:

```
http://myserver/rest/services/Geometry/GeometryServer
```

JSON Response Syntax

```
{
"serviceDescription" : "<serviceDescription>"
}
```

JSON Response Example

```
{
"serviceDescription" : "Test Geometry Service Description"
}
```

7.0.4 Project Operation

The Project operation is performed on a Geometry Service resource. The result of this operation is an array of projected geometries. This resource projects an array of input geometries from an input spatial reference to an output spatial reference. Users can provide arguments to the Project operation as query parameters.

7.0.4.1 Project Reference

- **URL:** http://<geometryservice-url>/project
- **Parent Resource:** Geometry Service

7.0.4.2 Project Parameters

Parameter	Details
f	Description: The response format Values: json | (other formats)
geometries	Description: The array of geometries to be projected. The structure of each geometry in the array is the same as the structure of the JSON geometry objects returned by the GeoServices REST Specification. **JSON Structures** Syntax: `{ "geometryType" : "<esriGeometryPoint | esriGeometryMultipoint | esriGeometryPolyline |`

Parameter	Details

	<pre>esriGeometryPolygon | esriGeometryEnvelope>",
	  "geometries" : [<geometry1>, <geometry2>]
	}</pre>
	The geometries property is an array of input geometries. All geometries in this array should be of the type defined by the geometryType property.
	
	Example:
	
	<pre>{"geometryType" :
	"esriGeometryPoint","geometries" : [{"x" : -
	104.53, "y" : 34.74}, {"x" : -63.53, "y" :
	10.23}]}</pre>
	Simple Syntax for Point Geometries
	
	When using points, in addition to the JSON structures, geometries can be specified with a simpler comma-separated syntax.
	
	Syntax:
	
	<pre>geometries=x1, y1, x2, y2, ..., xn, yn</pre>
	Example:
	
	<pre>geometries=-104.53, 34.74, -63.53, 10.23</pre>
	URL-Based Syntax
	
	For a large set of geometries, a URL can be specified to the input geometries stored in a JSON structure in a file on a public server.
	
	Syntax:
	
	<pre>geometries={ "url" : "<URL to file>" }</pre>
	Example:
	
	<pre>geometries={ "url" :
	"http://myserver/mygeometries/afile.txt" }</pre>

inSR	The well-known ID of the spatial reference or a spatial reference JSON object for the input geometries
outSR	The well-known ID of the spatial reference or a spatial reference JSON object for the returned geometries

7.0.4.3 Project Example

Example

Project the point [-117, 34] from WGS84 (WKID 4326) to Web Mercator (WKID 102113):

```
http://myserver/rest/services/Geometry/GeometryServer/project?inSR
=4326&outSR=102113&geometries={"geometryType":"esriGeometryPoint",
"geometries":[{"x":-117,"y":34}]}
```

JSON Response Syntax

```
{
"geometries" : [ <geometry1>, <geometry2> ]
}
```

JSON Response Example

```
{
"geometries" : [
 { "x" : -1.16362263726209E7, "y" : 4104255.01132978 },
 { "x" : -7072127.25009667, "y" : 1137314.06593893 }
]
}
```

7.0.5 Simplify Operation

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

7.0.5.1 Simplify Reference

- **URL:** http://<geometryservice-url>/simplify
- **Parent Resource:** Geometry Service

7.0.5.2 Simplify Parameters

Parameter	Details
f	Description: The response format Values: json | (other formats)

geometries	Description: The array of geometries to be simplified. The structure of each geometry in the array is the same as the structure of the JSON geometry objects returned by the GeoServices REST Specification.
	
	JSON Structures
	
	Syntax:
	
	`{`
	`"geometryType" : "<esriGeometryPoint | esriGeometryMultipoint | esriGeometryPolyline |`

Parameter	Details

	esriGeometryPolygon>"
	"geometries" : [<geometry1>, <geometry2>]
	}
	
	The geometries property is an array of input geometries. All geometries in this array should be of the type defined by the geometryType property.
	
	Example:
	
	{
	"geometryType" : "esriGeometryPoint",
	"geometries" : [
	{"x" : -104.53, "y" : 34.74},
	{"x" : -63.53, "y" : 10.23}
	]
	}
	
	Simple Syntax for Point Geometries
	
	When using points, in addition to the JSON structures, the geometries can be specified with a simpler comma-separated syntax.
	
	Syntax:
	
	geometries=x1, y1, x2, y2, ..., xn, yn
	
	Example:
	
	geometries=-104.53, 34.74, -63.53, 10.23
	
	URL-Based Syntax
	
	For a large set of geometries, a URL can be specified to the input geometries stored in a JSON structure in a file on a public server.
	
	Syntax:
	
	geometries={ "url" : "<URL to file>" }
	
	Example:
	
	geometries={ "url" :
	"http://myserver/mygeometries/afile.txt" }

sr	Description: The well-known ID of the spatial reference or a spatial reference JSON object for the input and output geometries

7.0.5.3 Simplify Example

Example

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

```
http://myserver/rest/services/Geometry/GeometryServer/simplify?sr=
4326&geometries={"geometryType":"esriGeometryPolygon","geometries"
:[{"rings":[[[-117,34],[-115,36],[-115,33],[-117,36],[-
117,34]]]}]}
```

JSON Response Syntax

```
{
"geometries" : [ <geometry1>, <geometry2> ]
}
```

JSON Response Example

```
{
"geometries" : [
 {
 "paths" : [
  [ [13,21], [13,62], [35,73] ],
  [ [23,23], [43,64], [45,52] ]
 ]
 },
 {
 "paths" : [
  [ [53.248,60.991], [-53,50] ],
  [ [53.248,60.991], [62.0,-44.999], [-64,40] ],
  [ [62.999,62], [53.248,60.991] ],
  [ [53,64], [53.248,60.991] ]
 ]
 }
]
}
```

7.0.6 Buffer Operation

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

The following are true when the bufferSR value is a geographic coordinate system:

- Points and Multipoints: If the unit is linear, such as feet or meters, geodesic buffering is performed.
- Polylines and Polygons: The unit must be angular, such as decimal degrees, for buffering to be performed.

7.0.6.1 Buffer Reference

- **URL:** http://<geometryservice-url>/buffer
- **Parent Resource:** Geometry Service

7.0.6.2 Buffer Parameters

Parameter	Details
f	Description: The response format Values: json | (other formats)

geometries	Description: The array of geometries to be buffered. The structure of each geometry in the array is the same as the structure of the JSON geometry objects returned by the GeoServices REST Specification.
	
	JSON Structures
	
	Syntax:
	
	`{`
	`"geometryType" : "<esriGeometryPoint | esriGeometryMultipoint | esriGeometryPolyline | esriGeometryPolygon>"`
	`"geometries" : [ <geometry1>, <geometry2> ]`
	`}`
	
	The "geometries" property is an array of input geometries. All geometries in this array should be of the type defined by "geometryType".
	
	Example:
	
	`{`
	`"geometryType" : "esriGeometryPoint",`
	`"geometries" : [`
	`{"x" : -104.53, "y" : 34.74},`
	`{"x" : -63.53, "y" : 10.23}`
	`]`
	`}`
	
	Simple Syntax for Point Geometries
	
	When using points, in addition to the JSON structures, the geometries can be specified with a simpler comma-separated syntax.
	
	Syntax:
	
	`geometries=x1, y1, x2, y2, ..., xn, yn`
	
	Example:
	
	`geometries=-104.53, 34.74, -63.53, 10.23`
	
	URL-Based Syntax
	
	For a large set of geometries, a URL can be specified to the input

J-9948

Parameter	Details

	geometries stored in a JSON structure in a file on a public server.
	
	Syntax:
	
	`geometries={ "url" : "<URL to file>" }`
	
	Example:
	
	`geometries={ "url" : "http://myserver/mygeometries/afile.txt" }`

inSR	Description: The well-known ID of the spatial reference or a spatial reference JSON object for the input geometries
outSR	Description: The well-known ID of the spatial reference or a spatial reference JSON object for the returned geometries. If a value for outSR is not specified, the output geometries are in the spatial reference specified by the bufferSR parameter. If no value for bufferSR is specified, they are in the spatial reference specified by the inSR parameter.
bufferSR	Description: The well-known ID of the spatial reference or a spatial reference JSON object in which the geometries are buffered. If a value for bufferSR is not specified, the geometries are buffered in the spatial reference set for outSR. If no value for outSR is specified, they are buffered in the spatial reference specified by the inSR parameter.

distances	Description: The distances the input geometries are buffered. The distance units are specified by the unit parameter.
	
	Syntax:
	
	`distances=<distance1>, <distance2>`
	
	Example:
	
	`distances=100, 123.45`

unit	The distance units to be applied to the calculation. This is specified as a numerical constant, as listed in the Esri documentation for esriSRUnitType constants (http://links.esri.com/esriSRUnitTypeConstants) and esriSRUnit2Type constants (http://links.esri.com/esriSRUnit2TypeConstants). For example, the value for meters is 9001.
unionResults	Description: If true, all geometries buffered at a given distance are combined into a single (possibly multipart) polygon, and the unioned geometry is placed in the output array. The default is false. Values: true | false

7.0.6.3 Buffer Example

Example

In this example, the point [-117, 34] is buffered in WGS84 (WKID 4326) at a distance of 1,000 meters. The geometry should be buffered using the Web Mercator projection (WKID 102113), and the output polygon should be returned in WGS84 (WKID 4326):

```
http://myserver/rest/services/Geometry/GeometryServer/
buffer?geometries=-117,34&inSR=4326&outSR=4326&bufferSR=
102113&distances=1000
```

JSON Response Syntax

```
{
"geometries" : [ <geometry1>, <geometry2> ]
}
```

JSON Response Example

```
{
"geometries" : [
 {
  "rings" : [
   [ [-97.06138,32.837], [-97.06133,32.836], [-97.06124,32.834],
[-97.06127,32.832], [-97.06138,32.837] ],
   [ [-97.06326,32.759], [-97.06298,32.755], [-97.06153,32.749],
[-97.06326,32.759] ]
  ]
 },
 {
  "rings" : [
   [ [-96.3575,28.423], [-96.3448,28.416], [-96.3399,28.408], [-
96.3575,28.423] ]
  ]
 }
]
}
```

7.0.7 Areas and Lengths Operation

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

7.0.7.1 Areas and Lengths Reference

- **URL:** http:// <geometryservice-url>/areasAndLengths
- **Parent Resource:** Geometry Service

7.0.7.2 Areas and Lengths Parameters

Parameter	Details
f	Description: The response format Values: json | (other formats)

J-9948

Parameter	Details

polygons	Description: The array of polygons whose areas and lengths are to be computed. The spatial reference of the polygons is specified by the sr parameter. The structure of each polygon in the array is the same as the structure of the JSON polygon objects returned by the GeoServices REST Specification.
	
	JSON Structures
	
	Syntax:
	
	`[ <polygon1>, <polygon2> ]`
	
	Example:
	
	`[ { "rings" : [ [[-117,34],[-116,34],[-117,33],[-117,34]], [[-115,44],[-114,43],[-115,43],[-115,44]] ] }, { "rings" : [ [[32.49,17.83],[31.96,17.59],[30.87,17.01],[30.11,16.86],[32.49,17.83]] ] }]`
	
	URL-Based Syntax
	
	For a large set of geometries, a URL can be specified to the input geometries stored in a JSON structure in a file on a public server.
	
	Syntax:
	
	`polygons={ "url" : "<URL to file>" }`
	
	Example:
	
	`polygons={ "url" : "http://myserver/mygeometries/afile.txt" }`

sr	Description: The well-known ID of the spatial reference or a spatial reference JSON object for the input polygons
lengthUnit	Description: The length unit in which perimeters of polygons will be calculated. This is specified as a numerical constant, as listed in the Esri documentation for esriSRUnitType constants (http://links.esri.com/esriSRUnitTypeConstants) and esriSRUnit2Type constants (http://links.esri.com/esriSRUnit2TypeConstants). For example, the value for meters is 9001.

Parameter	Details

areaUnit	Description: The area unit in which areas of polygons will be calculated. It can be specified using either a numerical constant or a string.
	
	If the area unit is specified using a numerical constant, values should be used from the Esri documentation for esriSRUnitType constants (http://links.esri.com/esriSRUnitTypeConstants) and esriSRUnit2Type constants (http://links.esri.com/esriSRUnit2TypeConstants). Units are in square units. For example, the value for square meters is 9001.
	
	If the area unit is specified using a string, the following may be used:
	
	esriSquareInches | esriSquareFeet | esriSquareYards | esriAcres | esriSquareMiles | esriSquareMillimeters | esriSquareCentimeters | esriSquareDecimeters | esriSquareMeters | esriAres | esriHectares | esriSquareKilometers
	
	If units are not specified, the units are derived from the spatial reference.
	
	JSON Structures (when using a string to specify the area unit)
	
	Syntax:
	
	`{`
	`"areaUnit" : "<esriAreaUnits>"`
	`}`
	
	Example:
	
	`{`
	`"areaUnit" : "esriHectares"`
	`}`

7.0.7.3 Areas and Lengths Example

Example

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

```
http://myserver/rest/services/Geometry/GeometryServer/areasAndLeng
ths?sr=102009&polygons=[{"rings" : [[[-
628833.344099998,206205.236200001],[-
630269.659900002,192298.906100001],[-
631848.233800001,173991.394400001],[-
616471.690300003,341822.557500001],[-
620213.661300004,301450.162799999],[-
625923.431999996,237538.0579],[-
628833.344099998,206205.236200001]]]}
]&lengthUnit=9035&areaUnit={"areaUnit" : "esriAcres"}
```

JSON Response Syntax

```
{
"areas" : [ <area1>, <area2> ],
"lengths" : [ <length1>, <length2> ]
}
```

JSON Response Example

```
{
"areas" : [ 1.0, 0.167199999999997 ],
"lengths" : [ 6.82842712474619, 5.16125300726341 ]
}
```

7.0.8 Lengths Operation

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

7.0.8.1 Lengths Reference

- **URL:** http://<geometryservice-url>/lengths
- **Parent Resource:** Geometry Service

7.0.8.2 Lengths Parameters

Parameter	Details
f	Description: The response format Values: json | (other formats)

polylines	Description: The array of polylines whose lengths are to be computed. The spatial reference of the polylines is specified by the sr parameter. The structure of each polyline in the array is the same as the structure of the JSON polyline objects returned by the GeoServices REST Specification.
	
	JSON Structures
	
	Syntax:
	
	`[ <polyline1>, <polyline2> ]`
	
	Example:
	
	`[ { "paths" : [ [[-117,34],[-116,34],[-117,33]], [[-115,44],[-114,43],[-115,43]] ] }, { "paths" : [ [[32.49,17.83],[31.96,17.59],[30.87,17.01],[30.11,16.86]]`

Parameter	Details

	]
	}
	]
	
	URL-Based Syntax
	
	For a large set of geometries, a URL can be specified to the input geometries stored in a JSON structure in a file on a public server.
	
	Syntax:
	
	`polylines={ "url" : "<URL to file>" }`
	
	Example:
	
	`polylines={ "url" : "http://myserver/mygeometries/afile.txt" }`

sr	Description: The well-known ID of the spatial reference or a spatial reference JSON object for the input polylines
lengthUnit	Description: The length unit in which perimeters of polygons will be calculated. This is specified as a numerical constant, as listed in the Esri documentation for esriSRUnitType constants (http://links.esri.com/esriSRUnitTypeConstants) and esriSRUnit2Type constants (http://links.esri.com/esriSRUnit2TypeConstants). For example, the value for meters is 9001.
geodesic	Description: If polylines are in a geographic coordinate system, geodesic should be set to true to calculate the ellipsoidal shortest path distance between each pair of vertices in the polylines. If a lengthUnit value is not specified, the output is always returned in meters.

7.0.8.3 Lengths Example

Example

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

```
http://myserver/rest/services/Geometry/GeometryServer/lengths?sr=4
269&polylines=[{"paths":[[[-117,34],[-116,34],[-117,33]],[[-
115,44],[-114,43],[-
115,43]]]},{"paths":[[[32.49,17.83],[31.96,17.59],[30.87,17.01],[3
0.11,16.86]]]}]&lengthUnit=9036&geodesic=true
```

JSON Response Syntax

```
{
"lengths" : [ <length1>, <length2> ]
}
```

JSON Response Example

```
{
"lengths" : [ 456.036465954783, 277.294288451794 ]
}
```

7.0.9 Relation Operation

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

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

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

7.0.9.1 Relation Reference

- **URL:** http://<geometryservice-url>/relation
- **Parent Resource:** Geometry Service

7.0.9.2 Relation Parameters

Parameter	Details
f	Description: The response format Values: json | (other formats)

geometries1	Description: The first array of geometries for computing the relations. The structure of each geometry in the array is the same as the structure of the JSON geometry objects returned by the GeoServices REST Specification.
	
	JSON Structures
	
	Syntax:
	
	`{`
	`"geometryType" : "<esriGeometryPoint | esriGeometryMultipoint | esriGeometryPolyline | esriGeometryPolygon | esriGeometryEnvelope>"`
	`"geometries" : [ <geometry1>, <geometry2> ]`
	`}`
	
	The geometries1 property is an array of input geometries. All geometries in this array should be of the type defined by the geometryType property.
	
	Example:
	
	`{`
	`"geometryType" : "esriGeometryPoint",`
	`"geometries" : [`
	`{"x" : -104.53, "y" : 34.74},`
	`{"x" : -63.53, "y" : 10.23}`

Parameter	Details

	]
	}
	
	URL-Based Syntax
	
	For a large set of geometries, you can specify a URL to the input geometries stored in a JSON structure in a file on a public server.
	
	Syntax:
	
	`geometries1={ "url" : "<URL to file>" }`
	
	Example:
	
	`geometries1={ "url" : "http://myserver/mygeometries/afile. txt" }`

geometries2	The second array of geometries for computing the relations. The structure of each geometry in the array is the same as the structure of the JSON geometry objects returned by the GeoServices REST Specification. The syntax and examples for these geometries are the same as the ones described for the geometries1 parameter above.
sr	Description: The well-known ID of the spatial reference or a spatial reference JSON object for the input geometries
relation	Description: The spatial relationship to be tested between the two input geometry arrays Values: esriGeometryRelationCross | esriGeometryRelationDisjoint | esriGeometryRelationIn | esriGeometryRelationInteriorIntersection | esriGeometryRelationIntersection | esriGeometryRelationLineCoincidence | esriGeometryRelationLineTouch | esriGeometryRelationOverlap | esriGeometryRelationPointTouch | esriGeometryRelationTouch | esriGeometryRelationWithin | esriGeometryRelationRelation Note that if the relation is specified as esriGeometryRelationRelation, the relationParam parameter describes the spatial relationship and must be specified.
relationParam	Description: The Shape Comparison Language string to be evaluated. Strings such as 'FFFTTT***' are accepted, in addition to other kinds of strings. This string describes the spatial relationship to be tested when the relation parameter is specified as esriGeometryRelationRelation. For additional information on how to construct this string, see the relationParam description in Section 4.2.4.2 Query Parameters.

7.0.9.3 Relation Example

Example

This example determines which of the two input points lie within the input polygon:

```
http://myserver/rest/services/Geometry/GeometryServer/
relation?sr=4326&relation=esriGeometryRelationWithin&geometries1={
"geometryType" : "esriGeometryPoint","geometries" : [{"x":-
104.53,"y":34.74},{"x":-
63.53,"y":10.23}]}&geometries2={"geometryType" :
"esriGeometryPolygon","geometries" : [{"rings":[[[-105,34],[-
104,34],[-104,35],[-105,35],[-105,34]]]}]}
```

JSON Response Syntax

```
{
  "relations" : [
    { "geometry1Index" : <geometry1Index1>, "geometry2Index" :
<geometry2Index1> },
    { "geometry1Index" : <geometry1Index2>, "geometry2Index" :
<geometry2Index2> }
  ]
}
```

JSON Response Example

```
{
  "relations" : [
    { "geometry1Index" : 0, "geometry2Index" : 1 },
    { "geometry1Index" : 1, "geometry2Index" : 1 }
  ]
}
```

7.0.10 Label Points Operation

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

7.0.10.1 Label Points Reference

- **URL:** http://<geometryservice-url>/labelPoints
- **Parent Resource:** Geometry Service

7.0.10.2 Label Points Parameters

Parameter	Details
f	Description: The response format Values: json | (other formats)
polygons	Description: The array of polygons whose label points are to be computed. The spatial reference of the polygons is specified by the value set for sr. The structure of each polygon in the array is the same as the structure of the JSON polygon objects returned by the GeoServices REST Specification.

Parameter	Details

	JSON Structures
	
	Syntax:
	
	`[ <polygon1>, <polygon2> ]`
	
	Example:
	
	`[ { "rings" : [ [[-117,34],[-116,34],[-117,33],[-117,34]], [[-115,44],[-114,43],[-115,43],[-115,44]] ] }, { "rings" : [ [[32.49,17.83],[31.96,17.59],[30.87,17.01],[30.11,16.86],[32.49,17.83]] ] } ]`
	
	URL-Based Syntax
	
	For a large set of geometries, a URL can be specified to the input geometries stored in a JSON structure in a file on a public server.
	
	Syntax:
	
	`polygons={ "url" : "<URL to file>" }`
	
	Example:
	
	`polygons={ "url" : "http://myserver/mygeometries/afile.txt" }`

sr	Description: The well-known ID of the spatial reference or a spatial reference JSON object for the input polygons

7.0.10.3 Label Points Example

Example

Compute a label point within the input polygon:

```
http://myserver/rest/services/Geometry/GeometryServer/labelPoints?
sr=4326&polygons=[{"rings":[[[-105,34],[-104,34],[-104,35],
[-105,35],[-105,34]]]}]
```

JSON Response Syntax

```
{
"labelPoints": [ <point1>, <point2> ]
}
```

JSON Response Example

```
{
"labelPoints": [
  { "x": -116.5, "y": 33.75 },
  { "x": 31.3, "y": 17.29 }
]
}
```

7.0.11 Distance Operation

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

7.0.11.1 Distance Reference

- **URL:** http://<geometryservice-url>/distance
- **Parent Resource:** Geometry Service

7.0.11.2 Distance Parameters

Parameter	Details
f	Description: The response format Values: json | (other formats)

geometry1	Description: The geometry from which the distance is to be measured. The structure of the geometry is the same as the structure of the JSON geometry objects returned by the GeoServices REST Specification. The use of simple syntax is not supported.
	
	Syntax:
	
	`{`
	`"geometryType" : "<esriGeometryPoint | esriGeometryPolyline | esriGeometryEnvelope | esriGeometryPolygon | esriGeometryMultipoint>",`
	`"geometry" : <geometry1>`
	`}`
	
	Example:
	
	`{`
	`"geometryType" : "esriGeometryPoint",`
	`"geometry" : {`
	`"x" : -118.15, "y" : 33.80,`
	`"spatialReference" : {"wkid" : 4326}`
	`}`
	`}`

Parameter	Details
geometry2	Description: The geometry from which the distance is to be measured. The structure of the geometry is the same as the structure of the JSON geometry objects returned by the GeoServices REST Specification. The use of simple syntax is not supported. The syntax and examples for this geometry are the same as those described for the parameter geometry1.
sr	Description: The well-known ID or a spatial reference JSON object for input geometries. If the planar distance is being measured, the spatial reference can be a projected coordinate system. If the geodesic distance is being measured, the spatial reference can be either a projected coordinate system or a geographic coordinate system.
distanceUnit	Description: Specifies the units for measuring distance between the geometry1 and geometry2 geometries. This is specified as a numerical constant, as listed in the Esri documentation for esriSRUnitType constants (http://links.esri.com/esriSRUnitTypeConstants) and esriSRUnit2Type constants (http://links.esri.com/esriSRUnit2TypeConstants). For example, the value for meters is 9001. If a unit is not specified, the units are derived from the spatial reference.
geodesic	Description: If true, the geodetic distance is measured between the geometries geometry1 and geometry2. If no value is specified for this option, it is treated as false.

7.0.11.3 Distance Example

Example

In this example, the distance in miles is computed between two points:

```
http://myserver/rest/services/Geometry/GeometryServer/distance?sr=
4326&geodesic=true&distanceUnit=9035&geometry1={"geometryType":"es
riGeometryPoint","geometry":{"x":-
117.47697998046874,"y":34.121858211839566,"spatialReference":{"wki
d":4326}}}&geometry2={"geometryType":"esriGeometryPoint","geometry
":{"x":-
117.41586853027343,"y":34.108125301683316,"spatialReference":{"wki
d":4326}}}
```

JSON Response Syntax

```
{
"distance" : <distance>
}
```

JSON Response Example

```
{
 "distance" : 10
}
```

7.0.12 Densify Operation

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

7.0.12.1 Densify Reference

- **URL:** http://<geometryservice-url>/densify
- **Parent Resource:** Geometry Service

7.0.12.2 Densify Parameters

Parameter	Details
f	Description: The response format Values: json | (other formats)

geometries	Description: The array of geometries to be densified. The structure of each geometry in the array is the same as the structure of the JSON geometry objects returned by the GeoServices REST Specification.
	
	Syntax:
	
	`{`
	`"geometryType" : "<esriGeometryPolyline | esriGeometryPolygon>"`
	`"geometries" : [ <geometry1>, <geometry2> ]`
	`}`
	
	The geometries property is an array of input geometries. All geometries in this array should be of the type defined by the geometryType property.
	
	Example:
	
	`{`
	`"geometryType" : "esriGeometryPolyline",`
	`"geometries" : [`
	`{`
	`"paths" : [`
	`[[-117,34],[-116,34],[-117,33]],`
	`[[-115,44],[-114,43],[-115,43]]`
	`]`
	`},`
	`{`
	`"paths" : [`
	
	`[[32.49,17.83],[31.96,17.59],[30.87,17.01],[30.11,16.86]]`
	`]`

Parameter	Details
	}] }
sr	Description: The well-known ID or a spatial reference JSON object for the input polylines
maxSegmentLength	Description: All segments longer than the value specified for maxSegmentLength are replaced with sequences of lines no longer than the maximum segment length.
geodesic	Description: A flag that can be set to true if a geographic coordinate system is used
lengthUnit (optional)	Description: The length unit of the maxSegmentLength value. This is specified as a numerical constant, as listed in the Esri documentation for esriSRUnitType constants (http://links.esri.com/esriSRUnitTypeConstants) and esriSRUnit2Type constants (http://links.esri.com/esriSRUnit2TypeConstants). For example, the value for meters is 9001. If this parameter is not specified, and if the sr is a projected coordinate system, the unit is determined by the unit of the spatial reference; otherwise, it is meter.

7.0.12.3 Densify Example

Example

Densify a polyline:

```
http://myserver/rest/services/Geometry/GeometryServer/densify?sr=3
395&geometries={"geometryType" : "esriGeometryPolyline",
"geometries" : [{"paths" : [[[-17313284.793,2209625.866],[-
17312808.186926104, 2210504.3164105085],[-
17308518.732261017,2218410.3701050845],
[-17260185.82890302,2310809.9320710143],
[-17307752.671522036,2223194.8742101695],
[-14501308.957,7392483.288],[-13773503.446,6003036.405
]]]}]}&maxSegmentLength=10000&useDeviationDensification=false&dens
ificationParameter=0&geodesic=false&lengthUnit=&f=json
```

JSON Response Syntax

```
{
"geometryType" : "<esriGeometryType>",
"geometries" : [<geometry1>, <geometry2>]
}
```

JSON Response Example

```
{
"geometryType" : "esriGeometryPolyline",
"geometries" : [
 {
   "paths" : [
    [[-117,34],[-116,34],[-117,33]],
```

```
    [[-115,44],[-114,43],[-115,43]]
   ]
 },
{
  "paths" : [
  [[32.49,17.83],[31.96,17.59],[30.87,17.01],[30.11,16.86]]
 ]
}
]
}
```

7.0.13 Generalize Operation

The Generalize operation is performed on a Geometry Service resource. It returns generalized versions of the input geometries, derived using the Douglas-Peucker algorithm. Users can provide arguments to the Generalize operation as query parameters.

7.0.13.1 Generalize Reference

- **URL:** http://<geometryservice-url>/generalize
- **Parent Resource:** Geometry Service

7.0.13.2 Generalize Parameters

Parameter	Details
f	Description: The response format Values: json | (other formats)

geometries	Description: The array of geometries to be generalized. The spatial reference of the geometries is specified by the value for sr. The structure of each geometry in the array is the same as the structure of the JSON geometry objects returned by the GeoServices REST Specification.
	
	Syntax:
	
	`{`
	`"geometryType" : "<esriGeometryPolyline | esriGeometryPolygon>",`
	`"geometries" : [ <geometry1>, <geometry2> ]`
	`}`
	
	The geometries property is an array of input geometries. All geometries in this array should be of the type defined by the geometryType property.
	
	Example:
	
	`{`
	`"geometryType" : "esriGeometryPolyline",`
	`"geometries" : [`
	`{`
	`"paths" : [`
	`[[-117,34],[-116,34],[-117,33]],`
	`[[-115,44],[-114,43],[-115,43]]`
	`]`
	`},`

Parameter	Details

	<pre>{
	  "paths" : [
	
	[[32.49,17.83],[31.96,17.59],[30.87,17.01],[30.11
	 16.86]]
	]
	}
	]
	}</pre>

sr	Description: The well-known ID or a spatial reference JSON object for the input geometries
maxDeviation	Description: Specifies the maximum deviation for constructing a generalized geometry based on the input geometries
deviationUnit (optional)	Description: A unit for maximum deviation. This is specified as a numerical constant, as listed in the Esri documentation for esriSRUnitType constants (http://links.esri.com/esriSRUnitTypeConstants) and esriSRUnit2Type constants (http://links.esri.com/esriSRUnit2TypeConstants). For example, the value for meters is 9001. If a unit is not specified, the units are derived from the value set for sr.

7.0.13.3 Generalize Example

Example

Generalize a polygon:

```
http://myserver/rest/services/Geometry/GeometryServer/generalize?s
r=4326&geometries={"geometryType":"esriGeometryPolygon","spatialRe
ference":{"wkid":4326},"geometries":[{"rings":[[[-
87.099342820011174,40.228084543758385],[-
87.100223146960218,40.381421968321966],[-
87.101720814594017,40.480793699969276],[-
87.100295810761097,40.57694817663144],[-
86.777024960686575,40.576769365423601],[-
86.700217232694484,40.574640906530412],[-
86.700551416568786,40.443071019286172],[-
86.704228938064603,40.225843915639118],[-
87.099342820011174,40.228084543758385]]]}]}&maxDeviation=0.01&devi
ationUnit=&f=json
```

JSON Response Syntax

```
{
  "geometryType" : "<esriGeometryPolyline | esriGeometryPolygon>"
  "geometries" : [ <geometry1>, <geometry2> ]
}
```

JSON Response Example

```
{
  "geometryType" : "esriGeometryPolygon",

  "geometries" :
  [
    {
      "rings" :
      [
        [
          [-87.0993428200112, 40.2280845437584],
          [-87.1002958107611, 40.5769481766314],
          [-86.7002172326945, 40.5746409065304],
          [-86.7042289380646, 40.2258439156391],
          [-87.0993428200112, 40.2280845437584]
        ]
      ]
    }
  ]
}
```

7.0.14 Convex Hull Operation

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

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

7.0.14.1 Convex Hull Reference

- **URL:** http://<geometryservice-url>/convexHull
- **Parent Resource:** Geometry Service

7.0.14.2 Convex Hull Parameters

Parameter	Details
f	Description: The response format Values: json | (other formats)

geometries	Description: The geometries whose convex hull is to be created. The structure of the geometry is the same as the structure of the JSON geometry objects returned by the GeoServices REST Specification.
	
	Syntax:
	
	`{`
	`"geometryType" : "<esriGeometryPoint |`
	`esriGeometryPolyline |`
	`esriGeometryPolygon |`
	`esriGeometryMultipoint>",`
	`"geometries" : [ <geometry1>, <geometry2> ]`
	`}`

Parameter	Details

	Example:
	
	`{`
	`"geometryType" : "esriGeometryPolyline",`
	`"geometries" : [`
	`{`
	`"paths" : [`
	`[[-117,34],[-116,34],[-117,33]],`
	`[[-115,44],[-114,43],[-115,43]]`
	`]`
	`},`
	`{`
	`"paths" : [`
	`[[32.49,17.83],[31.96,17.59],[30.87,17.01],`
	`[30.11,16.86]]`
	`]`
	`}`
	`]`
	`}`

sr	Description: The well-known ID or a spatial reference JSON object for the output geometry

7.0.14.3 Convex Hull Example

Example

Compute the convex hull for three points:

```
http://myserver/rest/services/Geometry/GeometryServer/convexHull?s
r=4326&geometries={"geometryType":"esriGeometryPoint","geometries"
:[{"type":"point","x":-
117.2332208251953,"y":34.086152645433316,"spatialReference":{"wkid
":4326}},{"type":"point","x":-
117.21536804199218,"y":34.0854659999255,"spatialReference":{"wkid"
:4326}},{"type":"point","x":-
117.22498107910155,"y":34.06623992570675,"spatialReference":{"wkid
":4326}}]}
```

JSON Response Syntax

```
{
"geometryType" : "<esriGeometryPoint | esriGeometryPolyline |
esriGeometryPolygon >"
"geometry" : { <geometry1> }
}
```

JSON Response Example

```
{
  "geometryType" : "esriGeometryPolygon",
  "geometry" :
  {
    "rings" :
    [
      [
        [-117.224981079102, 34.0662399257068],
        [-117.233220825195, 34.0861526454333],
```

```
        [-117.215368041992, 34.0854659999255],
        [-117.224981079102, 34.0662399257068]
      ]
    ]
  }
}
```

7.0.15 Offset Operation

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

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

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

7.0.15.1 Offset Reference

- **URL:** http://<geometryservice-url>/offset
- **Parent Resource:** Geometry Service

7.0.15.2 Offset Parameters

Parameter	Details
f	Description: The response format Values: json | (other formats)

geometries	Description: The array of geometries to be offset. The spatial reference of the geometries is specified by the value for sr. The structure of each geometry in the array is the same as the structure of the JSON geometry objects returned by the GeoServices REST Specification.
	
	Syntax:
	
	`{`
	`"geometryType" : "<esriGeometryPolyline | esriGeometryPolygon>",`
	`"geometries" : [ <geometry1>, <geometry2> ]`
	`}`
	
	The geometries property is an array of input geometries. All geometries in this array should be of the type defined by the geometryType property.
	
	Example:
	
	`{`
	`"geometryType" : "esriGeometryPolyline ",`
	`"geometries" : [`
	`{`

Parameter	Details

	<pre>"paths" : [
	[[-97.06138,32.837], [-97.06133,32.836], [-97.06124,32.834], [-97.06127,32.832]],
	[[-97.06326,32.759], [-97.06298,32.755]]
	],
	"spatialReference" : {"wkid" : 4326}
	}
	]
	}</pre>

sr	Description: The well-known ID or a spatial reference JSON object for the input geometries
offsetDistance	Description: Specifies the distance for constructing an offset based on the input geometries. If the offsetDistance parameter is positive, the constructed offset is on the right side of the curve. Left side offsets are constructed with negative values.
offsetUnit (optional)	Description: A unit for offset distance. This is specified as a numerical constant, as listed in the Esri documentation for esriSRUnitType constants (http://links.esri.com/esriSRUnitTypeConstants) and esriSRUnit2Type constants (http://links.esri.com/esriSRUnit2TypeConstants). For example, the value for meters is 9001. If a unit is not specified, the units are derived from the value for sr.
offsetHow	Description: This parameter determines how outer corners between segments are handled. Rounded offset rounds the corner between extended offsets. Beveled offset squares off the corner after a given ratio distance. Mitered offset attempts to allow extended offsets to naturally intersect, but if that intersection occurs too far from the corner, the corner is eventually beveled at a fixed distance. Choose from esriGeometryOffsetMitered, esriGeometryOffsetBevelled, and esriGeometryOffsetRounded.
bevelRatio	Description: The bevelRatio value is multiplied by the offset distance, and the result determines how far a mitered offset intersection can be located before it is beveled. When mitered is specified, the input bevel ratio is ignored and 10 is used internally. When beveled is specified, 1.1 is used if bevelRatio is not specified. The bevelRatio parameter is ignored for rounded offsets.

7.0.15.3 Offset Example

Example

Offset multiple polylines by one meter:

```
http://myserver/rest/services/Geometry/GeometryServer/offset?sr=22
29&geometries={"geometryType":"esriGeometryPolyline","spatialRefer
ence":{"wkid":2229},"geometries":[{"paths":[[[6805566.1574656013,1
846311.2154481949],[6805527.5463925907,1847577.0219133438],[680556
7.9057296626,1846206.4173309559],[6805566.1574656013,1846311.21544
81949],[6805489.2513970956,1845286.8614968264],[6805567.9057296626
,1846206.4173309559]]]},{"paths":[[[6805493.9062097641,1845121.674
091635],[6805489.2513970956,1845286.8614968264],[6805496.388459288
5,1844963.0200417505],[6805493.9062097641,1845121.674091635],[6805
512.6586404499,1843725.7847297059],[6805496.3884592885,1844963.020
0417505],[6805514.211678369,1843607.5195617655],[6805512.658640449
9,1843725.7847297059],[6805523.8066700343,1842901.206113206],[6805
514.211678369,1843607.5195617655],[6805532.3821443468,1842246.5626
597235],[6805523.8066700343,1842901.206113206]]]}]}&offsetDistance
=1&offsetUnit=9001&offsetHow=esriGeometryOffsetMitered&bevelRatio=
2&simplifyResult=false
```

JSON Response Syntax

```
{
  "geometryType" : "<esriGeometryPolyline> |
<esriGeometryPolygon>"
  "geometries" : [ <geometry1>, <geometry2> ]
}
```

JSON Response Example

```
{
  "geometryType" : "esriGeometryPolyline",

  "geometries" :
  [
    {
      "paths" :
      [
        [
          [6805569.43677369, 1846311.31547739],
          [6805530.62569506, 1847583.67882845],
          [6805524.07389809, 1847583.48243998],
          [6805564.81883934, 1846199.78272534],
          [6805571.29520759, 1846199.93209211],
          [6805569.3307883, 1846317.68744017],
          [6805563.36634781, 1846317.86122269],
          [6805562.87287955, 1846311.28844923]
        ]
      ]
    },
```

```
    {
      "paths" :
      [
        [
          [6805497.18574129, 1845121.76650549],
          [6805492.34638359, 1845293.50294242],
          [6805485.82700549, 1845293.33924976],
          [6805493.25276803, 1844956.39812766],
          [6805499.7713768, 1844956.52095142],
          [6805497.08404916, 1845128.28265713],
          [6805490.53757939, 1845128.18747252],
          [6805509.46623861, 1843719.18004924],
          [6805516.02546532, 1843719.26723603],
          [6805499.58272771, 1844969.62427911],
          [6805493.02163891, 1844969.53800327],
          [6805511.0173989, 1843600.91535478],
          [6805511.01872677, 1843600.91537222],
          [6805529.18919647, 1842239.95819138],
          [6805535.74863772, 1842240.04494229],
          [6805527.08722191, 1842901.24908666]
        ]
      ]
    }
  ]
}
```

7.0.16 Trim/Extend Operation

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

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

7.0.16.1 Trim/Extend Reference

- **URL:** http://<geometryservice-url>/trimExtend
- **Parent Resource:** Geometry Service

7.0.16.2 Trim/Extend Parameters

Parameter	Details
f	Description: The response format Values: json | (other formats)
polylines	Description: An array of polylines to be trimmed or extended. The spatial reference of the polylines is specified by the sr parameter. The structure of each polyline in the array is the same as the structure of the JSON polyline objects returned by the GeoServices REST Specification. Syntax: `[ <polyline1>, <polyline2> ]`

Parameter	Details
	Example: `[ { "paths" : [ [[-117,34],[-116,34],[-117,33]], [[-115,44],[-114,43],[-115,43]] ] }, { "paths" : [ [[32.49,17.83],[31.96,17.59],[30.87,17.01],[30.11,16.86]] ] } ]`

trimExtendTo	Description: A polyline that is used as a guide for trimming and extending input polylines. The spatial reference of the polylines is specified by the sr parameter. The structure of each polyline is the same as the structure of the JSON polyline objects returned by the GeoServices REST Specification.
	
	Syntax:
	
	`<polyline>`
	
	Example:
	
	`{ "paths" : [ [[-117,34],[-116,34],[-117,33]], [[-115,44],[-114,43],[-115,43]] ] }`

sr	Description: The well-known ID of the spatial reference or a spatial reference JSON object for the input polylines
extendHow (optional)	Description: A flag that is used with the Trim/Extend operation ■ 0—By default, extension considers both ends of paths. The old ends remain and new points are added to the extended ends. The new points have attributes that are extrapolated from adjacent existing segments. ■ 1—If an extension is performed at an end, relocate the end point to the new position instead of leaving the old point and adding a new point at the new position. ■ 2—If an extension is performed at an end, do not extrapolate the end segment's attributes for the new point. Instead, make its attributes the same as the current end.

Parameter	Details
	■ 4—If an extension is performed at an end, do not extrapolate the end segment's attributes for the new point. Instead, make its attributes empty. ■ 8—Do not extend the from end of any path. ■ 16—Do not extend the to end of any path.

7.0.16.3 Trim/Extend Example

Example

Trims/Extend two polyline segments

```
http://myserver/rest/services/Geometry/GeometryServer/trimExtend?s
r=2229&polylines=[{"paths" :
[[[6805512.658537939,1843725.7846097648],[6805496.38855736,1844963
.0199961811]]]},{"paths" :
[[[6805532.382251769,1842246.5625026077],[6805523.806809604,184290
1.206206441]]]}]&trimExtendTo={"paths" :
[[[6804206.368171528,1843554.492957607],[6805395.769992188,1843570
.1779655963],[6805514.211684436,1843607.5194263458],[6805740.68892
1779,1843619.888168022]]]}&extendHow=2
```

JSON Response Syntax

```
{
  "geometryType" : "<esriGeometryPolyline>",
  "geometries" : [ <geometry1>, <geometry2> ]
}
```

JSON Response Example

```
{
  "geometries" :
  [
    {
      "paths" :
      [
        [
          [6805514.21365289, 1843607.51942632],
          [6805512.65853788, 1843725.78460972],
          [6805496.3885573, 1844963.01999615]
        ]
      ]
    },
    {
      "paths" :
      [
        [
          [6805532.38225172, 1842246.56250256],
          [6805523.80680956, 1842901.2062064],
          [6805514.55420339, 1843607.53812706]
        ]
      ]
    }
  ]
}
```

7.0.17 Auto Complete Operation

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

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

7.0.17.1 Auto Complete Reference

- **URL:** http://<geometryservice-url>/autoComplete
- **Parent Resource:** Geometry Service

7.0.17.2 Auto Complete Parameters

Parameter	Details
f	Description: The response format Values: json | (other formats)

polygons	Description: The array of polygons that will provide some boundaries for new polygons. The spatial reference of the polygons is specified by the sr parameter. The structure of the polygon in the array is the same as the structure of the JSON polygon objects returned by the GeoServices REST Specification.
	
	Syntax:
	
	`[ <polygon1>, <polygon2> ]`
	
	Example:
	
	`[ { "rings" : [ [[-117,34],[-116,34],[-117,33],[-117,34]], [[-115,44],[-114,43],[-115,43],[-115,44]] ] }, { "rings" : [ [[32.49,17.83],[31.96,17.59],[30.87,17.01], [30.11,16.86],[32.49,17.83]] ] } ]`

polylines	Description: An array of polylines that will provide the remaining boundaries for new polygons. The spatial reference of the polylines is specified by the sr parameter. The structure of polylines in the array is the same as the structure of the JSON polyline objects returned by the GeoServices REST Specification. Syntax: `[ <polyline1>, <polyline2> ]`

Parameter	Details
	Example: `[ { "paths" : [ [[-117,34],[-116,34],[-117,33]], [[-115,44],[-114,43],[-115,43]] ] }, { "paths" : [ [[32.49,17.83],[31.96,17.59],[30.87,17.01],[30.11,16.86]] ] } ]`
sr	Description: The well-known ID or a spatial reference JSON object for the input polygons and polylines

7.0.17.3 Auto Complete Example

Example

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

JSON Response Syntax

```
{
"geometries" : [ <polygon1>, <polygon2> ]
}
```

JSON Response Example

```
{
  "geometries" :
  [
 {
  "rings" : [
   [[-117,34],[-116,34],[-117,33],[-117,34]],
   [[-115,44],[-114,43],[-115,43],[-115,44]]
  ]
 },
 {
  "rings" : [

[[32.49,17.83],[31.96,17.59],[30.87,17.01],[30.11,16.86],[32.49,17
.83]]
  ]
 }
]
}
```

7.0.18 Cut Operation

The Cut operation is performed on a Geometry Service resource. This operation splits the input polyline or polygon where it crosses a cutting polyline. Users can provide arguments to the Cut operation as query parameters.

7.0.18.1 Cut Reference

- **URL:** http://<geometryservice-url>/cut
- **Parent Resource:** Geometry Service

7.0.18.2 Cut Parameters

Parameter	Details
f	Description: The response format Values: json | (other formats)

cutter	Description: The polyline that will be used to divide the target into pieces where it crosses the target. The spatial reference of the polylines is specified by the sr parameter. The structure of the polyline is the same as the structure of the JSON polyline objects returned by the GeoServices REST Specification.
	
	Syntax:
	
	`<polyline1>`
	
	Example:
	
	`{ "paths" : [ [[-117,34],[-116,34],[-117,33]], [[-115,44],[-114,43],[-115,43]] ] }`

target	Description: The array of polylines/polygons to be cut. The structure of the geometry is the same as the structure of the JSON geometry objects returned by the GeoServices REST Specification.
	
	Syntax:
	
	`{ "geometryType" : "<esriGeometryPolyline | esriGeometryPolygon>" "geometries" : [ <geometry1>, <geometry2> ] }`
	
	The target property is a geometry of the type defined by geometryType.
	
	Example:
	
	`{ "geometryType" : "esriGeometryPolyline", "geometries" : [`

Parameter	Details

	<pre>{
	  "paths" : [
	   [[-117,34],[-116,34],[-117,33]],
	   [[-115,44],[-114,43],[-115,43]]
	  ]
	},
	{
	  "paths" : [
	
	[[32.49,17.83],[31.96,17.59],[30.87,17.01],[30.1
	1,16.86]]
	  ]
	}
	]
	}</pre>

sr	Description: The well-known ID of the spatial reference or a spatial reference JSON object for the input geometry

7.0.18.3 Cut Example

Example

Cut a polyline geometry

```
http://myserver/rest/services/Geometry/GeometryServer/cut?sr=2229&
target={"geometryType" : "esriGeometryPolyline","spatialReference"
:{"wkid" : 2229},"geometries" :[{"paths" :
[[[6805743.810634688,1843230.507057026],
[6805740.688921779,1843619.888168022],
[6802621.935316771,1843581.5805018544],[6805496.38855736,1844963.0
199961811]]]}]}&cutter={"paths" :
[[[6805210,1843869],[6805842,1843529],[6805259,1843173]]]}
```

JSON Response Syntax

The Cut operation returns a cutIndexes array along with an array of cut geometries. The cutIndexes array can be used to determine which of the input geometries were cut to get the resultant geometries.

```
{
"geometryType" : "<esriGeometryPolyline | esriGeometryPolygon>",
"geometries" : [ <geometry1>, <geometry2> ],
"cutIndexes:" : [ integer1, integer2 ]
}
```

JSON Response Example

```
{
  "geometryType" : "esriGeometryPolyline",

  "geometries" :
  [
    {
      "paths" :
      [
        [
          [6805743.81063464, 1843230.50705698],
          [6805741.90775131, 1843467.87994181]
        ],
        [
          [6805740.98190014, 1843583.34526207],
          [6805740.68892172, 1843619.88816798],
          [6805674.56471014, 1843619.07583365]
        ]
      ]
    },
    {
      "paths" :
      [
        [
          [6805741.90775131, 1843467.87994181],
          [6805740.98190014, 1843583.34526207]
        ],
        [
          [6805674.56471014, 1843619.07583365],
          [6802621.93531673, 1843581.58050181],
          [6805496.3885573, 1844963.01999615]
        ]
      ]
    }
  ],
  "cutIndexes" : [
    0,
    0
  ]
}
```

7.0.19 Difference Operation

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

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

7.0.19.1 Difference Reference

- **URL:** http://<geometryservice-url>/difference
- **Parent Resource:** Geometry Service

7.0.19.2 Difference Parameters

Parameter	Details
f	Description: The response format Values: json | (other formats)

geometries	Description: An array of points, multipoints, polylines, or polygons. The structure of each geometry in the array is the same as the structure of the JSON geometry objects returned by the GeoServices REST Specification.
	
	Syntax:
	
	`{`
	`"geometryType" : "<esriGeometryPoint | esriGeometryMultipoint | esriGeometryPolyline | esriGeometryPolygon >"`
	`"geometries" : [ <geometry1>, <geometry2> ]`
	`}`
	
	Example:
	
	`{`
	`"geometryType" : "esriGeometryPolyline",`
	`"geometries" : [`
	`{`
	`"paths" : [`
	`[[-117,34],[-116,34],[-117,33]],`
	`[[-115,44],[-114,43],[-115,43]]`
	`]`
	`},`
	`{`
	`"paths" : [`
	`[[32.49,17.83],[31.96,17.59],[30.87,17.01],[30.11,16.86]]`
	`]`
	`}`
	`]`
	`}`

geometry	Description: A single geometry of any type of dimension equal to or greater than the elements of the geometries property. The structure of the geometry is the same as the structure of the JSON geometry objects returned by the GeoServices REST Specification. The use of simple syntax is not supported.
	
	Syntax:
	
	`{`
	`"geometryType" : "<esriGeometryPoint | esriGeometryPolyline | esriGeometryPolygon | esriGeometryMultipoint>",`
	`"geometry" : <geometry1>`
	`}`

Parameter	Details

	Example:
	
	`{`
	`"geometryType" : "esriGeometryPoint",`
	`"geometry" : {`
	`  "rings" : [`
	`   [[-117,34],[-116,34],[-117,33],[-117,34]],`
	`   [[-115,44],[-114,43],[-115,43],[-115,44]]`
	`  ]`
	` }`
	`}`

sr	Description: The well-known ID of the spatial reference or a spatial reference JSON object for the input geometries

7.0.19.3 Difference Example

Example

Calculate the difference between two polygons:

```
http://myserver/rest/services/Geometry/GeometryServer/difference?s
r=4269&geometries={"geometryType" :
"esriGeometryPolygon","spatialReference" :{"wkid" :
4269},"geometries" :[{"rings" : [[[-
75.48928066099995,39.714858219000064],[-
75.4759742679999,39.720084384000074],[-
75.47476845699993,39.741832093000085],[-
75.46039411899994,39.763362027000085],[-
74.73882472699995,40.17772564400008],[-
74.9166543419999,39.17063854200006],[-
75.01440707699993,39.198363837000045],[-
75.11995811199995,39.18469178100008],[-
75.4156722749999,39.374971842000036],[-
75.55276303999995,39.49051430700007],[-
75.5166888839999,39.56656841600005],[-
75.57023418699993,39.61773496300009],[-
75.48928066099995,39.714858219000064]]]}]}
&geometry={"geometryType" :
"esriGeometryPolygon","spatialReference" :{"wkid" :
4269},"geometry" :{"rings" : [[[-
75.46039411899994,39.763362027000085],[-
74.73882472699995,40.17772564400008],[-
75.46039411899994,39.763362027000085]]]}}
```

JSON Response Syntax

```
{
"geometryType" : "<esriGeometryPoint |
esriGeometryMultipoint | esriGeometryPolyline |
esriGeometryPolygon >"
"geometries" : [ <geometry1>, <geometry2> ]
}
```

JSON Response Example

```
{
  "geometryType" : "esriGeometryPolygon",
  "geometries" :
  [
    {
      "rings" :
      [
        [
          [-74.738824727, 40.1777256440001],
          [-74.916654342, 39.1706385420001],
          [-75.014407077, 39.198363837],
          [-75.1199581119999, 39.184691781],
          [-75.415672275, 39.374971842],
          [-75.5527630399999, 39.4905143070001],
          [-75.516688884, 39.5665684160001],
          [-75.570234187, 39.617734963],
          [-75.489280661, 39.7148582190001],
          [-75.475974268, 39.7200843840001],
          [-75.474768457, 39.741832093],
          [-75.4603941189999, 39.763362027],
          [-74.738824727, 40.1777256440001]
        ]
      ]
    }
  ]
}
```

7.0.20 Intersect Operation

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

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

7.0.20.1 Intersect Reference

- **URL:** http://<geometryservice-url>/intersect
- **Parent Resource:** Geometry Service

7.0.20.2 Intersect Parameters

Parameter	Details
f	Description: The response format Values: json | (other formats)

geometries	Description: An array of points, multipoints, polylines, or polygons. The structure of each geometry in the array is the same as the structure of the JSON geometry objects returned by the GeoServices REST Specification.
	
	Syntax:
	
	`{`
	`"geometryType" : "<esriGeometryPoint | esriGeometryMultipoint | esriGeometryPolyline |`

J-9948

Parameter	Details

	esriGeometryPolygon>"
	"geometries" : [<geometry1>, <geometry2>]
	}
	
	Example:
	
	{
	"geometryType" : "esriGeometryPolyline",
	"geometries" : [
	{
	"paths" : [
	[[-117,34],[-116,34],[-117,33]],
	[[-115,44],[-114,43],[-115,43]]
	]
	},
	{
	"paths" : [
	[[32.49,17.83],[31.96,17.59],[30.87,17.01],
	[30.11,16.86]]
	]
	}
	]
	}

geometry	Description: A single geometry of any type of dimension equal to or greater than the elements of the geometries property. The structure of the geometry is the same as the structure of the JSON geometry objects returned by the GeoServices REST Specification. The use of simple syntax is not supported.
	
	Syntax:
	
	{
	"geometryType" : "<esriGeometryPoint |
	esriGeometryMultipoint |esriGeometryPolyline |
	esriGeometryPolygon | esriGeometryMultipoint>",
	"geometry" : <geometry1>
	}
	
	Example:
	
	{
	"geometryType" : "esriGeometryPoint",
	"geometry" : {
	"rings" : [
	[[-117,34],[-116,34],[-117,33],[-117,34]],
	[[-115,44],[-114,43],[-115,43],[-115,44]]
	]
	}
	}

sr	Description: The well-known ID or a spatial reference JSON object for the input geometries

7.0.20.3 Intersect Example

Example

Calculate the intersection of two polygons:

```
http://myserver/rest/services/Geometry/GeometryServer/intersect?sr
=4269&geometries={"geometryType" :
"esriGeometryPolygon","spatialReference" :{
"wkid" : 4269},"geometries" :[{"rings" : [[
[-75.48928066099995,39.714858219000064],
[-75.4759742679999,39.720084384000074],
[-75.47476845699993,39.741832093000085],
[-75.46039411899994,39.763362027000085],
[-74.73882472699995,40.17772564400008],
[-74.9166543419999,39.17063854200006],
[-75.01440707699993,39.198363837000045],
[-75.11995811199995,39.18469178100008],
[-75.4156722749999,39.374971842000036],
[-75.55276303999995,39.49051430700007],
[-75.5166888839999,39.56656841600005],
[-75.57023418699993,39.61773496300009],
[-75.48928066099995,39.714858219000064]]]}]}&geometry={
"geometryType" : "esriGeometryPolygon","spatialReference" :{
"wkid" : 4269},"geometry" :{"rings" : [[
[-75.48928066099995,39.714858219000064],
[-75.4759742679999,39.720084384000074],
[-75.47476845699993,39.741832093000085],
[-75.46039411899994,39.763362027000085],
[-74.73882472699995,40.17772564400008],
[-75.48928066099995,39.714858219000064]]]}}
```

JSON Response Syntax

```
{
"geometryType" : "<esriGeometryPoint | esriGeometryMultipoint |
esriGeometryPolyline | esriGeometryPolygon>"
"geometries" : [ <geometry1>, <geometry2> ]
}
```

JSON Response Example

```
{
  "geometryType" : "esriGeometryPolygon",

  "geometries" :
  [
    {
      "rings" :
      [
        [
          [-75.475803134, 39.7231709100001],
          [-75.474768457, 39.741832093],
          [-75.4603941189999, 39.763362027],
          [-74.738824727, 40.1777256440001],
          [-75.475803134, 39.7231709100001]
```

```
        ]
      ]
    }
  ]
}
```

7.0.21 Reshape Operation

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

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

7.0.21.1 Reshape Reference

- **URL:** http://<geometryservice-url>/reshape
- **Parent Resource:** Geometry Service

7.0.21.2 Reshape Parameters

Parameter	Details
f	Description: The response format Values: json | (other formats)

target	Description: The polyline or polygon to be reshaped
	
	Syntax:
	
	`{`
	`"geometryType" : "<esriGeometryPolyline | esriGeometryPolygon>"`
	`"geometry" : <geometry>`
	`}`
	
	Example:
	
	`{`
	`"geometryType" : "esriGeometryPoint",`
	`"geometry" : {`
	`"rings" : [`
	`[[-117,34],[-116,34],[-117,33],[-117,34]],`
	`[[-115,44],[-114,43],[-115,43],[-115,44]]`
	`]`
	`}`
	`}`

reshaper	Description: The single-part polyline that does the reshaping
	
	Syntax:
	
	`{`
	`"paths" : <polyline>`
	`}`
	
	Example:
	
	`{`
	`"paths" : [`

Parameter	Details

	<pre>[[-117,34],[-116,34],[-117,33]],
	[[-115,44],[-114,43],[-115,43]]
	]
	}</pre>

sr	Description: The well-known ID of the spatial reference or a spatial reference JSON object for the input geometry

7.0.21.3 Reshape Example

Example

Reshape a polygon using a polyline reshaper:

```
http://myserver/rest/services/Geometry/GeometryServer/reshape?sr=2
229&target={"geometryType" :
"esriGeometryPolygon","spatialReference" :{"wkid" :
2229},"geometry" :{"rings" :
[[[6807691.607592106,1841423.2521413565],[6807403.895241022,184122
6.589476943],[6807330.383577019,1841176.5013225228],[6807144.49746
5864,1841049.4458023459],[6806716.6797519475,1840757.4020951092],[
6806479.870514274,1840595.731486693],[6806183.101422772,1840393.18
46476793],[6806075.24238652,1840319.1683915257],[6805779.309579447
,1840125.002769351],[6805524.879970193,1839930.444431439],[6804797
.212462276,1841029.5203172714],[6804973.141244277,1841152.84257718
92],[6805745.21122244,1841695.4566494375],[6806023.85141319,184189
0.9388700128],[6806319.65692395,1842098.571656853],[6806763.038583
115,1842410.1881596],[6806913.661969528,1842516.0485121906],[68071
50.782230198,1842187.7000875175],[6807308.85212402,1841967.8783486
933],[6807691.258839533,1841423.802665189]]]}}&reshaper={"paths" :
[[[6804973.141244277,1841152.8425771892],[6804797.212462276,184102
9.5203172714],[6804463.906370357,1841533.088863939],[6804224.75560
1943,1841930.77901344],[6804233.406175196,1842251.835129857],[6804
206.368171528,1843554.492957607],[6805395.769992188,1843570.177965
5963],[6805514.211684436,1843607.5194263458],[6805740.688921779,18
43619.888168022],[6806080.253859445,1843657.1859936863],[6806290.2
70171687,1843380.829262942],[6806717.911376774,1842787.4210009426]
,[6806913.661969528,1842516.0485121906],[6806763.038583115,1842410
.1881596],[6806319.65692395,1842098.571656853],[6806027.585329607,
1841887.9188629389],[6805745.21122244,1841695.4566494375],[6804969
.741316691,1841150.3071491867]]]}
```

JSON Response Syntax

```
{
"geometryType" : "<esriGeometryPolyline |
esriGeometryPolygon>"
"geometry" : <geometry>
}
```

JSON Response Example

```
{
  "geometryType" : "esriGeometryPolygon",
  "geometry" :
  {
    "rings" :
    [
      [
        [6804973.14124422, 1841152.84257714],
        [6804797.21246222, 1841029.52031723],
        [6804463.9063703, 1841533.08886389],
        [6804224.75560188, 1841930.7790134],
        [6804233.40617514, 1842251.83512981],
        [6804206.36817147, 1843554.49295756],
        [6805395.76999214, 1843570.17796557],
        [6805514.21168439, 1843607.51942632],
        [6805740.68892172, 1843619.88816798],
        [6806080.25385939, 1843657.18599364],
        [6806290.27017164, 1843380.8292629],
        [6806717.91137671, 1842787.4210009],
        [6806913.66196947, 1842516.04851215],
        [6806763.03858306, 1842410.18815956],
        [6806319.65692389, 1842098.57165681],
        [6806027.58532955, 1841887.91886289],
        [6805745.21122238, 1841695.45664939],
        [6804973.14124422, 1841152.84257714]
      ]
    ]
  }
}
```

7.0.22 Union Operation

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

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

7.0.22.1 Union Reference

- **URL:** http://<geometryservice-url>/union
- **Parent Resource:** Geometry Service

7.0.22.2 Union Parameters

Parameter	Details
f	Description: The response format Values: json | (other formats)

Parameter	Details

geometries	Description: The array of geometries on which the Union operation will be performed. The structure of each geometry in the array is the same as the structure of the JSON geometry objects returned by the GeoServices REST Specification.
	
	Syntax:
	
	`{`
	`"geometryType" : "<esriGeometryPoint | esriGeometryMultipiont | esriGeometryPolyline | esriGeometryPolygon>"`
	`"geometries" : [ <geometry1>, <geometry2> ]`
	`}`
	
	The geometries property is an array of input geometries. All geometries in this array should be of the type defined by the geometryType property.
	
	Example:
	
	`{`
	`  "geometryType":"esriGeometryPolygon",`
	`  "geometries":[`
	`{`
	`  "rings" : [`
	`  [[-117,34],[-116,34],[-117,33],[-117,34]],`
	`  [[-115,44],[-114,43],[-115,43],[-115,44]]`
	`  ]`
	`},`
	`{`
	`  "rings" : [`
	
	`[[32.49,17.83],[31.96,17.59],[30.87,17.01],[30.11,16.86],[32.49,17.83]]`
	`  ]`
	`}`
	`]`
	`}`

sr	Description: The well-known ID of the spatial reference or a spatial reference JSON object for the input geometries

7.0.22.3 Union Example

Example

Make a union of two multipoint geometries:

```
http://myserver/rest/services/Geometry/GeometryServer/union?sr=102113&geometries={%22geometryType%22%3A+%22esriGeometryMultipoint%22%2C%0D%0A+%22spatialReference%22%3A+{%22wkid%22%3A+102113}%2C%0D%0A+++%22geometries%22%3A+[{%22points%22%3A+[%0D%0A+++[%0D%0A++++-
8418433.3989%2C%0D%0A++++5262954.0867%0D%0A+++]%2C%0D%0A+++[%0D%0A++++-
8353363.717800001%2C%0D%0A++++5381714.5528%0D%0A+++]%2C%0D%0A+++[%0D%0A++++-
8250336.7228%2C%0D%0A++++5337002.1307%0D%0A+++]%2C%0D%0A+++[%0D%0A++++-
```

J-9948

```
8250336.7228%2C%0D%0A++++5509587.154700004%0D%0A+++]%0D%0A++]}%2C%
0D%0A++{%22points%22%3A+[%0D%0A+++++[%0D%0A++++++-
1.37211748048E7%2C%0D%0A++++++4991906.4582%0D%0A+++++]%2C%0D%0A+++
++[%0D%0A++++++-
1.35126474057E7%2C%0D%0A++++++4800770.159900002%0D%0A+++++]%2C%0D%
0A+++++[%0D%0A++++++-
1.35126474057E7%2C%0D%0A++++++4991906.4582%0D%0A+++++]%2C%0D%0A+++
++[%0D%0A++++++-
1.3470941925900001E7%2C%0D%0A++++++4468209.699000001%0D%0A+++++]%2
C%0D%0A+++++[%0D%0A++++++-
1.32936936367E7%2C%0D%0A++++++4494424.728500001%0D%0A+++++]%2C%0D%
0A+++++[%0D%0A++++++-
1.31998563072E7%2C%0D%0A++++++4133116.2506000027%0D%0A+++++]%2C%0D
%0A+++++[%0D%0A++++++-
1.3126871717500001E7%2C%0D%0A++++++4260779.642999999%0D%0A+++++]%2
C%0D%0A+++++[%0D%0A++++++-
1.29391970583E7%2C%0D%0A++++++4044613.1774000004%0D%0A+++++]%2C%0D
%0A+++++[%0D%0A++++++-
1.29183443184E7%2C%0D%0A++++++4222325.328299999%0D%0A+++++]%2C%0D%
0A+++++[%0D%0A++++++-
1.28870652086E7%2C%0D%0A++++++3906909.467299998%0D%0A+++++]%0D%0A+
+++]}]%0D%0A%0D%0A}
```

JSON Response Syntax

```
{
"geometryType" : "<esriGeometryPoint | esriGeometryMultipiont |
esriGeometryPolyline | esriGeometryPolygon>"
"geometry" :  <geometry>
}
```

JSON Response Example

```
{
  "geometryType" : "esriGeometryMultipoint",
  "geometry" :
  {
    "points" :
    [
      [-13721174.8048, 4991906.4582],
      [-13512647.4057, 4800770.1599],
      [-13512647.4057, 4991906.4582],
      [-13470941.9259, 4468209.699],
      [-13293693.6367, 4494424.7285],
      [-13199856.3072, 4133116.2506],
      [-13126871.7175, 4260779.643],
      [-12939197.0583, 4044613.1774],
      [-12918344.3184, 4222325.3283],
      [-12887065.2086, 3906909.4673],
      [-8418433.3989, 5262954.0867],
      [-8353363.7178, 5381714.5528],
      [-8250336.7228, 5337002.1307],
      [-8250336.7228, 5509587.1547]
    ]
  }
}
```

8.0 IMAGE SERVICE

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 GeoServices REST Specification Image Service 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.

8.0.1 Image Service Reference

- **URL:** http://<catalog-url>/<serviceName>/ImageServer
- **Supported Operations:** Export Image, Identify, Query, Download
- **Parent Resource:** Catalog
- **Child Resources:** Raster Catalog Item, Raster File

8.0.2 Image Service Parameters

Parameter	Details
f	Description: The response format Values: json | (other formats, for example, kmz)

8.0.3 Image Service Example

Example

URL to the MyImage image service:

```
http://myserver/rest/services/MyImage/ImageServer
```

JSON Response Syntax

```
{
"serviceDescription" : "<serviceDescription>",
"name" : "<name>",
"description" : "<description>",
"extent" : {<extent>},
//if the Image Service supports query/exportImage based on time
"timeInfo" :
{
  "timeExtent" : [<startTime>, <endTime>],
  "timeReference" : null
 },
"pixelSizeX" : <pixelSizeX>,
"pixelSizeY" : <pixelSizeY>,
"bandCount" : <bandCount>,
"pixelType" : "<pixelType>",
"minPixelSize" : <minPixelSize>,
"maxPixelSize" : <maxPixelSize>,
"copyrightText" : "<copyrightText>",
"serviceDataType" : "<serviceDataType>",
"minValues" : [<minValue1>, <minValue2>],
"maxValues" : [<maxValue1>, <maxValue2>],
"meanValues" : [<meanValue1>, <meanValue2>],
"stdvValues" : [<stdvValue1>, <stdvValue2>],

//accessible fields
"objectIdField" : "<objectIdFieldName>",
//Query and Download operations and Raster Catalog Item and Raster
File resources
//are available only if fields is not null and not empty
"fields" : [
  {"name" : "<fieldName1>", "type" : "<fieldType1>", "alias" :
"<fieldAlias1>"},
  {"name" : "<fieldName2>", "type" : "<fieldType2>", "alias" :
"<fieldAlias2>"}
]
}
```

JSON Response Example

```
{
"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" : "ESRI",
"serviceDataType" : "esriImageServiceDataTypeRGB",
"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":"esriFieldTypeOID","alias":"OB
JECTID"},{"name":"Shape","type":"esriFieldTypeGeometry","alias":"S
hape"},{"name":"Name","type":"esriFieldTypeString","alias":"Name"}
]}
```


8.0.4 Export Image Operation

The Export Image operation is performed on an Image Service resource. The result of this operation provides information about the exported image, such as its URL, width and height, and extent.

Apart from the usual JSON response format, users can also request a format called image while performing this operation. When users perform an 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 can provide arguments to the Export Image operation as query parameters. These parameters include the request extent, size information, interpolation, and pixel type.

8.0.4.1 Export Image Reference

- **URL:** http://<imageservice-url>/exportImage
- **Parent Resource:** Image Service

8.0.4.2 Export Image Parameters

Parameter	Details
f	Description: The response format Values: json | image | (other formats, for example, kmz)

bbox	Description: The extent (bounding box) of the exported image. Unless the bboxSR parameter has a value specified, the bbox is assumed to be in the spatial reference of the image service.
	
	Syntax:
	
	`<xmin>, <ymin>, <xmax>, <ymax>`
	
	Example:
	
	`bbox=-104,35.6,-94.32,41`
	
	The bbox coordinates should always use a period as the decimal separator even in countries where traditionally a comma is used.

size	Description: The size (width * height) of the exported image in pixels. If the size is not specified, an image with a default size of 400 * 400 will be exported. Syntax: `<width>, <height>`

Parameter	Details
	Example: `size=600,550`
imageSR	Description: The spatial reference of the exported image. The spatial reference can be specified as either a well-known ID or a spatial reference JSON object. If a value for imageSR is not specified, the image is exported in the spatial reference of the image service.
bboxSR	Description: The spatial reference of the bbox. The spatial reference can be specified as either a well-known ID or a spatial reference JSON object. If a value for bboxSR is not specified, the bbox is assumed to be in the spatial reference of the image service.

time	Description: The time instant or the time extent of the exported image
	
	Time Instant
	
	Syntax:
	
	`time=<timeInstant>`
	
	Example:
	
	`time=1199145600000`
	
	(1 Jan. 2008 00:00:00 GMT)
	
	Time Extent (For time extents, one of <startTime> or <endTime> could be null.)
	
	Syntax:
	
	`time=<startTime>, <endTime>`
	
	Example:
	
	`time=1199145600000, 1230768000000`
	
	(1 Jan. 2008 00:00:00 GMT to 1 Jan. 2009 00:00:00 GMT)
	
	A null value specified for start time or end time will represent infinity for start or end time, respectively.

format	Description: The format of the exported image. The default value is jpgpng. This format returns a JPEG if there are no transparent pixels in the requested extent; otherwise it returns a PNG. Values: jpgpng | png | png8 | png24 | jpg | bmp | gif | tiff

Parameter	Details
pixelType	Description: The pixel type, also known as data type, pertains to the type of values stored in the raster, such as signed integer, unsigned integer, or floating point. Integers are whole numbers, whereas floating points have decimals. Values: C128 | C64 | F32 | F64 | S16 | S32 | S8 | U1 | U16 | U2 | U32 | U4 | U8 | UNKNOWN
noData	Description: The pixel value representing no information Example: `noData=0`
interpolation	Description: The resampling process of extrapolating the pixel values while transforming the raster dataset when it undergoes warping or changes coordinate space Values: RSP_BilinearInterpolation | RSP_CubicConvolution | RSP_Majority | RSP_NearestNeighbor
compressionQuality	Description: 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. Example: `compressionQuality=75`
bandIds	Description: 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. Example: `bandIds=2,1,0`
mosaicRule	Description: 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. When a mosaic rule is not specified, the value defaults to esriMosaicNone. Mosaic rules are as follows: ■ esriMosaicNone: Images have no special ordering. ■ esriMosaicCenter: Sorts rasters based on their proximity to the view center or the center of view extent. ■ esriMosaicNadir: Sorts rasters based on the distance between the nadir position and view center. ■ esriMosaicViewpoint: Sorts rasters based on a user-defined viewpoint location and nadir location for the raster Mosaic operations that apply: first (default), last, min,

Parameter	Details
	max, mean, and blend. ■ esriMosaicAttribute: Sorts rasters based on an attribute field and its difference from a base value. ■ esriMosaicLockRaster: Selects only the rasters in a given list of raster IDs to participate in the mosaic. ■ esriMosaicNorthwest: Sorts rasters in a view-independent way, where rasters with their centers most northwest are displayed on top. ■ esriMosaicSeamline: 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). Syntax: `{` ` "mosaicMethod" : "<esriMosaicNone	esriMosaicCenter	esriMosaicNadir	esriMosaicViewpoint	esriMosaicAttribute	esriMosaicLockRaster	esriMosaicNorthwest	esriMosaicSeamline>",` ` "where" : "<where>",` ` "sortField" : "<sortFieldName>",` ` "sortValue" : <sortValue>,` ` "ascending" : <true	false>,` ` "lockRasterIds" : [<rasterId1>, <rasterId2>],` ` "viewpoint" : <point>,` ` "fids" : [<fid1>, <fid2>],` ` "mosaicOperation" : "<MT_FIRST	MT_LAST	MT_MIN	MT_MAX	MT_MEAN	MT_BLEND>"` `}`  Example:  `{` ` "mosaicMethod" : "esriMosaicLockRaster",` ` "lockRasterIds" : [32, 454, 14]` `}`

renderingRule	Description: Specifies the rendering rule for how the requested image should be rendered
	
	Syntax:
	
	`{`
	`  "rasterFunction" : "<rasterFunctionName>",`
	`  "rasterFunctionArguments" : {<rasterFunctionArguments>},`

Parameter	Details
	<pre> //optional parameter "variableName" : "<variableName>" }</pre> For information on default variable names, see Section 8.0.4.4 Raster Functions for Use with Image Exports. The syntax of the rasterFunctionArguments property varies based on the specified rasterFunction value. Refer to Section 8.0.4.4 Raster Functions for Use with Image Exports for more details.

8.0.4.3 Export Image Examples

Example 1

Export an image with the bounding box [[-117, 34] - [-116, 35]] in WGS84 (WKID 4326):

```
http://myserver/rest/services/MyImage/ImageServer/exportImage?bbox
=-117,34,-116,35&bboxSR=4326
```

Example 2

Export an image similar to example 1, but request the image in Web Mercator (WKID 102113):

```
http://myserver/rest/services/MyImage/ImageServer/exportImage?bbox
=-117,34,-116,35&bboxSR=4326&imageSR=102113
```

JSON Response Syntax

```
{
"href" : "<href>",
"width" : <width>,
"height" : <height>,
"extent" : {<envelope>}
}
```

JSON Response Example

```
{
"href" :
"http://myserver/output/map42ef5eae899942a9b564138e184a55c9.png",
"width" : 400,
"height" : 400,
"extent" : {
 "xmin" : -109.55, "ymin" : 25.76, "xmax" : -86.39, "ymax" :
49.94,
 "spatialReference" : {"wkid" : 4326}
}
}
```

8.0.4.4 Raster Functions for Use with Image Exports

The image service's Export Image operation supports a renderingRule parameter. This parameter has the following JSON syntax:

```
{
  "rasterFunction" : "<rasterFunctionName>",
  "rasterFunctionArguments" : {<rasterFunctionArguments>},
  "variableName" : "<variableName>"
}
```

The structure of the rasterFunctionArguments object varies based on the rasterFunction value. This document lists the raster function names and the corresponding arguments supported by the GeoServices REST Specification.

Aspect

The Aspect raster function takes no arguments. Hence, specifying only the rasterFunction property suffices in this case.

```
{
  "rasterFunction" : "Aspect"
}
```

Colormap

The arguments for the Colormap function are shown below:

```
{
  "rasterFunction" : "Colormap",
  "rasterFunctionArguments" : {
    "ColormapName" : "<Random | NDVI | Elevation | Gray>",
    "Colormap" : [
      [<value1>, <red1>, <green1>, <blue1>], //[int, int, int,
int]
      [<value2>, <red2>, <green2>, <blue2>]
    ]
  },
  "variableName" : "Raster"
}
```

Example 1

```
{
  "rasterFunction" : "Colormap",
  "rasterFunctionArguments" : {
    "ColormapName" : "Random"
  },
  "variableName" : "Raster"
}
```

Example 2

```
{
  "rasterFunction" : "Colormap",
  "rasterFunctionArguments" : {
    "Colormap" : [
      [0, 1, 2, 3],
      [2, 45, 52, 13]
    ]
  },
  "variableName" : "Raster"
}
```

Hillshade

The arguments for the Hillshade function are shown below:

```
{
  "rasterFunction" : "Hillshade",
  "rasterFunctionArguments" : {
    "Azimuth" : <Azimuth>, //double (e.g. 215.0)
    "Altitude" : <Altitude>, //double (e.g. 75.0)
    "ZFactor" : <ZFactor> //double (e.g. 0.3)
  },
  "variableName" : "DEM"
}
```

Example

```
{
  "rasterFunction" : "Hillshade",
  "rasterFunctionArguments" : {
    "Azimuth" : 215.0,
    "Altitude" : 75.0,
    "ZFactor" : 0.3
  },
  "variableName" : "DEM"
}
```

NDVI

The arguments for the NDVI function are shown below:

```
{
  "rasterFunction" : "NDVI",
  "rasterFunctionArguments" : {
    "VisibleBandID" : <VisibleBandID>, //int (zero-based band id,
e.g. 2)
    "InfraredBandID" : <InfraredBandID> //int (zero-based band id,
e.g. 1)
  },
  "variableName" : "Raster"
}
```

Example

```
{
  "rasterFunction" : "NDVI",
  "rasterFunctionArguments" : {
    "VisibleBandID" : 2,
    "InfraredBandID" : 1
  },
  "variableName" : "Raster"
}
```

ShadedRelief

The arguments for the ShadedRelief function are shown below:

```
{
  "rasterFunction" : "ShadedRelief",
  "rasterFunctionArguments" : {
    "Azimuth" : <Azimuth>, //double (e.g. 215.0)
    "Altitude" : <Altitude>, //double (e.g. 75.0)
    "ZFactor" : <ZFactor>, //double (e.g. 0.3)
    "Colormap" : [
      [<value1>, <red1>, <green1>, <blue1>], //[int, int, int,
int]
      [<value2>, <red2>, <green2>, <blue2>]
    ]
  },
  "variableName" : "Raster"
}
```

Example

```
{
  "rasterFunction" : "ShadedRelief",
  "rasterFunctionArguments" : {
    "Azimuth" : 215.0,
    "Altitude" : 75.0,
    "ZFactor" : 0.3,
    "Colormap" : [
      [0, 1, 2, 3],
      [2, 45, 52, 13]
    ]
  },
  "variableName" : "Raster"
}
```

Slope

The arguments for the Slope function are shown below:

```
{
  "rasterFunction" : "Slope",
  "rasterFunctionArguments" : {
    "ZFactor" : <ZFactor> //double (e.g. 0.3)
  },
  "variableName" : "DEM"
}
```

Example

```
{
  "rasterFunction" : "Slope",
  "rasterFunctionArguments" : {
    "ZFactor" : 0.3
  },
  "variableName" : "DEM"
}
```

Statistics

The arguments for the Statistics function are shown below:

```
{
  "rasterFunction" : "Statistics",
  "rasterFunctionArguments" : {
    "Type" : "<Min | Max | Mean | StandardDeviation>",
    "KernelColumns" : <KernelColumns>, //int (e.g. 3)
    "KernelRows" : <KernelRows> //int (e.g. 3)
  },
  "variableName" : "Raster"
}
```

Example

```
{
  "rasterFunction" : "Statistics",
  "rasterFunctionArguments" : {
    "Type" : "Mean",
    "KernelColumns" : 3,
    "KernelRows" : 3
  },
  "variableName" : "Raster"
}
```

Stretch

The arguments for the Stretch function are shown below:

```
{
  "rasterFunction" : "Stretch",
  "rasterFunctionArguments" : {
    "StretchType" : <StretchType>, //int (0 = None, 3 =
StandardDeviation, 4 = Histogram Equalization, 5 = MinMax)
    "NumberOfStandardDeviations" : <NumberOfStandardDeviations>,
//int (e.g. 2)
    "Statistics" : [
      [<min1>, <max1>, <mean1>, <standardDeviation1>], //[double,
double, double, double]
      [<min2>, <max2>, <mean2>, <standardDeviation2>]
    ],
    "Gamma" : [<gamma1>, <gamma2>] //array of doubles
  },
  "variableName" : "Raster"
}
```

Example

```
{
  "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"
}
```

8.0.5 Query Operation (Image Services)

The Query operation is performed on an Image Service resource. 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). Users can provide arguments to the Query operation as query parameters.

8.0.5.1 Query Reference

- **URL:** http://<imageservice-url>/query
- **Parent Resource:** Image Service

8.0.5.2 Query Parameters

Parameter	Details
f	Description: The response format Values: json | (other formats)

Parameter	Details

objectIds	Description: The ObjectIDs of the raster catalog to be queried. Note that when this parameter is specified, any other filter parameters (including where) are ignored.
	
	When this parameter is specified, setting returnIdsOnly=true is invalid.
	
	Syntax:
	
	`objectIds=<objectId1>, <objectId2>`
	
	Example:
	
	`objectIds=37, 462`

where	Description: A WHERE clause for the query filter. Any legal SQL WHERE clause operating on the fields in the raster catalog is allowed. Example: `where=POP2000 > 350000`

time	Description: The time instant or the time extent to query
	
	Time Instant
	
	Syntax:
	
	`time=<timeInstant>`
	
	Example:
	
	`time=1199145600000`
	
	(1 Jan. 2008 00:00:00 GMT)
	
	Time Extent (For time extents, one of <startTime> or <endTime> could be null.)
	
	Syntax:
	
	`time=<startTime>, <endTime>`
	
	Example:
	
	`time=1199145600000, 1230768000000`
	
	(1 Jan. 2008 00:00:00 GMT to 1 Jan. 2009 00:00:00 GMT)
	
	A null value specified for start time or end time will represent infinity for start or end time, respectively.

geometry	Description: The geometry to apply as the spatial filter. The structure of the geometry is the same as the structure of the JSON geometry objects returned by the GeoServices REST Specification. In addition to the JSON structures, for envelopes

Parameter	Details

	and points, a user can specify the geometry with a simpler comma-separated syntax.
	
	Syntax:
	
	■ JSON structures:
	`geometryType=<geometryType>&geometry={geometry}`
	
	■ Envelope simple:
	`geometryType=esriGeometryEnvelope&geometry=<xmin>,<ymin>,<xmax>,<ymax>`
	
	■ Point simple:
	`geometryType=esriGeometryPoint&geometry=<x>,<y>`
	
	Examples:
	
	■ `geometryType=esriGeometryEnvelope&geometry={xmin: -104, ymin: 35.6, xmax: -94.32, ymax: 41}`
	
	■ `geometryType=esriGeometryEnvelope&geometry=-104,35.6,-94.32,41`
	
	■ `geometryType=esriGeometryPoint&geometry=-104,35.6`

geometryType	Description: The type of geometry specified by the geometry parameter. The geometry type can be an envelope, point, line, or polygon. The default geometry type is an envelope. Values: esriGeometryPoint | esriGeometryMultipoint | esriGeometryPolyline | esriGeometryPolygon | esriGeometryEnvelope
inSR	Description: The spatial reference of the input geometry. The spatial reference can be specified as either a well-known ID or a spatial reference JSON object. If a value for inSR is not specified, the geometry is assumed to be in the spatial reference of the map.
spatialRel	Description: The spatial relationship to be applied on the input geometry while performing the query. The supported spatial relationships include intersects, contains, envelope intersects, and within. The default spatial relationship is intersects (esriSpatialRelIntersects). Values: esriSpatialRelIntersects | esriSpatialRelContains | esriSpatialRelCrosses | esriSpatialRelEnvelopeIntersects | esriSpatialRelIndexIntersects | esriSpatialRelOverlaps | esriSpatialRelTouches | esriSpatialRelWithin

Parameter	Details

outFields	Description: The list of fields to be included in the returned result set. This is a comma-delimited list of field names. To request geometry, returnGeometry should be set to true.
	
	If the wildcard (*) is specified as the value of this parameter, the query results include all the field values. Note that the wildcard also implies returnGeometry=true, and setting returnGeometry to false has no effect.
	
	Example:
	
	`outFields=AREANAME,ST,POP2000`
	
	Example (wildcard usage):
	
	`outFields=*`

returnGeometry	Description: If true, the result set includes the geometry associated with each result. The default is true. Note that if the outFields parameter is set to the wildcard, it implies returnGeometry=true, and setting returnGeometry to false has no effect. Values: true | false
outSR	Description: The spatial reference of the returned geometry. The spatial reference can be specified as either a well-known ID or a spatial reference JSON object. If a value for outSR is not specified, the geometry is returned in the spatial reference of the image service.
returnIdsOnly	Description: If true, the response includes only an array of raster IDs. Otherwise, the response is a raster set. The default is false. Values: false | true

8.0.5.3 Query Examples

Example 1

Query using a WHERE clause and return specific output fields:

```
http://myserver/rest/services/Landsat/ImageServer/query?
where=Name LIKE 'p045r028%' AND Name NOT LIKE 'Ovr%'
&outFields=Name,MinPS,MaxPS,LowPS,HighPS&returnGeometry=true&retur
nIdsOnly=false&f=json
```

Example 2

Query using a point geometry and a WHERE clause. Return only ObjectIDs:

```
http://myserver/rest/services/Landsat/ImageServer/query?
where=NAME NOT LIKE 'Ov_%'&time=&geometry={"x" : -122.895114,"y" :
45.558214,"spatialReference" : {"wkid" :
4269}}&geometryType=esriGeometryPoint&inSR=4326&spatialRel=esriSpa
tialRelIntersects&outFields=*&returnGeometry=false&returnIdsOnly=t
rue&f=json
```

JSON Response Syntax (when returnIdsOnly=false)

```
{
"objectIdFieldName" : "<objectIdFieldName>",
"spatialReference" : <spatialReference>,
"fields" : [
    {"name" : "<fieldName1>", "type" : "<fieldType1>", "alias" :
"<fieldAlias1>", "length" : "<length1>"},
    {"name" : "<fieldName2>", "type" : "<fieldType2>", "alias" :
"<fieldAlias2>", "length" : "<length2>"}
],
"features" : [
  <feature1>, <feature2>
]
}
```

JSON Response Syntax (when returnIdsOnly=true)

```
{
"objectIdFieldName" : "<objectIdFieldName>",
"objectIds" : [ <objectId1>, <objectId2> ]
}
```

JSON Response Example

```
{
"objectIdFieldName" : "IMAGEID",
"spatialReference" : {"wkid" : 4326},

"fields" : [
{
    "name" : "ST",
    "alias" : "ST",
    "type" : "esriFieldTypeString",
    "length" : 2
},
{

    "name" : "OBJECTID",
    "alias" : "OBJECTID",
    "type" : "esriFieldTypeOID"
},
```

```
{
    "name" : "AREANAME",
    "alias" : "City Name",
    "type" : "esriFieldTypeString",
    "length" : 255
}
],
"geometryType" : "esriGeometryPolygon",
"features" : [
  {
    "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
    }
  },
  {
    "geometry" : {
      "rings" : [
         [ [-97.06326,32.759], [-97.06298,32.755], [-
97.06153,32.749], [-97.06326,32.759] ]
      ]
    },
    "attributes" : {
      "IMAGEID" : 462,
      "OWNER" : "John Doe",
      "VALUE" : 17325.90,
      "APPROVED" : false,
      "LASTUPDATE" : 1227628579430
    }
  }
]
}
```


8.0.6 Identify Operation (Image Services)

The Identify operation is performed on an Image Service resource. It identifies the content of an image service 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.

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

8.0.6.1 Identify Reference

- **URL:** http://<imageservice-url>/identify
- **Parent Resource:** Image Service

8.0.6.2 Identify Parameters

Parameter	Details
f	Description: The response format Values: json | (other formats)

geometry	Description: A geometry that defines the location to be identified. The location can be a point or a polygon. The structure of the geometry is the same as the structure of the JSON geometry objects returned by the GeoServices REST Specification. In addition to the JSON structures, for points, users can specify the geometry with a simpler comma-separated syntax.
	
	This is a required parameter. The default geometry type is a point. By default, the geometry is assumed to be in the spatial reference of the image service. Users can specify a different spatial reference by using the JSON structure syntax for geometries.
	
	Syntax:
	
	JSON structures:
	
	`geometryType=<geometryType>&geometry={geometry}`
	
	Point simple:
	
	`geometryType=esriGeometryPoint&geometry=<x>,<y>`
	
	Examples:
	
	`geometryType=esriGeometryPoint&geometry={x: -104, y: 35.6}`
	
	`geometryType=esriGeometryPoint&geometry=-104,35.6`

geometryType	Description: The type of geometry specified by the geometry parameter. The geometry type can be a point or polygon. Values: esriGeometryPoint | esriGeometryPolygon
mosaicRule	Description: Specifies the mosaic rule defining the image sorting order. When a mosaic rule is not specified, the mosaicRule value defaults to esriMosaicCenter. See Section 8.0.4.2 Export Image Parameters for descriptions of

Parameter	Details

	the available mosaic rules.
	
	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).
	
	Syntax:
	
	`{`
	`  "mosaicMethod" : "<esriMosaicNone | esriMosaicCenter | esriMosaicNadir | esriMosaicViewpoint | esriMosaicAttribute | esriMosaicLockRaster | esriMosaicNorthwest | esriMosaicSeamline>",`
	`  "where" : "<where>",`
	`  "sortField" : "<sortFieldName>",`
	`  "sortValue" : <sortValue>,`
	`  "ascending" : <true | false>,`
	`  "lockRasterIds" : [<rasterId1>, <rasterId2>],`
	`  "viewpoint" : <point>,`
	`  "fids" : [<fid1>, <fid2>],`
	`  "mosaicOperation" : "<MT_FIRST | MT_LAST | MT_MIN | MT_MAX | MT_MEAN | MT_BLEND>"`
	`}`
	
	Example:
	
	`{`
	`  "mosaicMethod" : "esriMosaicCenter",`
	`  "sortField" : "Category",`
	`  "ascending" : false`
	`}`

pixelSize	Description: The pixel level being identified (or the resolution being looked at). If pixel size is not specified, then the pixelSize value defaults to the base resolution of the dataset. The raster at the specified pixel size in the mosaic will be used to identify.
	
	The structure of the pixelSize parameter is the same as the structure of the point object returned by the GeoServices REST Specification. In addition to the JSON structure, users can specify the pixel size with a simpler comma-separated syntax.
	
	Syntax:
	
	JSON structures:
	
	`pixelSize={point}`
	
	Point simple:
	
	`pixelSize=<x>,<y>`

Parameter	Details

	Examples:
	
	`pixelSize={x: 0.18, y: 0.18}`
	
	`pixelSize=0.18,0.18`

8.0.6.3 Identify Examples

Example 1

Identify a single raster image service using a point geometry:

```
http://myserver/rest/services/SanAndreasLidar/ImageServer/identify
?geometry={"x":575505.5,"y":3733770}&geometryType=esriGeometryPoin
t&mosaicRule=&pixelSize=0.5,0.5&f=pjson
```

Example 2

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

```
http://myserver/rest/services/Landsat
/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=esriGeometryPolygon&mos
aicRule={"mosaicMethod" : "esriMosaicAttribute","where":"Name NOT
LIKE 'Ov%'","sortField":"Name","mosaicOperation":"MT_MAX"}
```

JSON Response Syntax

```
{
"objectId" : <objectId>,
"name" : "<name>",
"value" : "<pixelValue>",
"location" : <point>, //the identified location
"properties" : { //the properties (attributes) of the identified
object. (returned only when the image service source is a
mosaicked dataset)
  "name1" : <value1>,
  "name2" : <value2>
},
//catalogItems are returned only when the image service source is
a mosaicked dataset.
"catalogItems" : {
  "objectIdFieldName" : "<objectIdFieldName>",
  "spatialReference" : <spatialReference>,
  "geometryType" : "<geometryType>",
  "features" : [
    <feature1>, <feature2>
  ]
},
```

```
//catalogItemVisibilities are returned only when the image service
source is a mosaicked dataset.
"catalogItemVisibilities" : [ <catalogItem1Visibility>,
<catalogItem2Visibility> ]
}
```

JSON Response Example

```
{
"objectId" : 22,
"name" : "John Snow",
"value" : "0,1,2",
"location" : {x: -104, y: 35.6},
"properties" : {
   "Value: "10,22,33"
},
"catalogItems" : {
  "objectIdFieldName" : "IMAGEID",
  "spatialReference" : {"wkid" : 4326},
  "geometryType" : "esriGeometryPolygon",
  "features" : [
    {
      "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
      }
    },
    {
      "geometry" : {
        "rings" : [
           [ [-97.06326,32.759], [-97.06298,32.755], [-
97.06153,32.749], [-97.06326,32.759] ]
        ]
      },
      "attributes" : {
        "IMAGEID" : 462,
        "OWNER" : "John Doe",
        "VALUE" : 17325.90,
        "APPROVED" : false,
        "LASTUPDATE" : 1227628579430
      }
    }
  ]
},
"catalogItemVisibilities" : [ 0.7, 0.5 ]
}
```

8.0.7 Download Rasters Operation

The Download Rasters operation is performed on an Image Service resource. It returns information (the file ID) that can be used to download the raw raster files associated with a specified set of rasters in the raster catalog.

The file IDs returned by this operation can be used to download individual files using the Raster File resource.

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

8.0.7.1 Download Rasters Reference

- **URL:** http://<imageservice-url>/download
- **Parent Resource:** Image Service

8.0.7.2 Download Rasters Parameters

Parameter	Details
f	Description: The response format Values: json | (other formats)
rasterIds	Description: A comma-separated list of raster IDs whose files are to be downloaded Example: `rasterIds=37, 462`

geometry	Description: The geometry to apply for clipping. If specified, the selected rasters will be clipped on the server. The structure of the geometry is the same as the structure of the JSON geometry objects returned by the GeoServices REST Specification. In addition to the JSON structures, for envelopes, users can specify the geometry with a simpler comma-separated syntax.
	
	By default, the geometry is assumed to be in the spatial reference of the image service. A client can specify a different spatial reference by using the JSON structure syntax for geometries.
	
	Syntax:
	
	JSON structures:
	
	`geometryType=<geometryType>&geometry={geometry}`
	
	Envelope simple:
	
	`geometryType=esriGeometryEnvelope&geometry=<xmin>,<ymin>,<xmax>,<ymax>`
	
	Examples:
	
	`geometryType=esriGeometryEnvelope&geometry={xmin: -104, ymin: 35.6, xmax: -94.32, ymax: 41}`
	
	`geometryType=esriGeometryEnvelope&geometry=-104,35.6,-94.32,41`

Parameter	Details
geometryType	Description: The type of geometry specified by the geometry parameter. The geometry type can be an envelope or polygon. The default geometry type is an envelope. Values: esriGeometryPolygon | esriGeometryEnvelope
format	Description: The format of the rasters returned. If not specified, the rasters will be in their native format. The format applies only if the clip geometry is also specified, and the format is honored only when the raster is clipped. Valid formats include TIFF, IMAGINE Image, JPEG, BIL, BSQ, BIP, ENVI, JP2, GIF, BMP, and PNG. Example: `format=TIFF`

8.0.7.3 Download Rasters Example

Example

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

```
http://myserver/rest/services/SeattleImagery/ImageServer/download?
rasterIds=5,6,10,11,12&geometry={"xmin": -1949594.8286481365,
"ymin": 882737.0181116117,"xmax": -1946926.2791246006,"ymax":
884828.2021675818
,"spatialReference":
{"wkid":102009}}&geometryType=esriGeometryEnvelope&format=TIFF
```

JSON Response Syntax

```
{
"rasterFiles" : [ //the list of files that make up the rasters to
be downloaded
  { //info pertaining to a single file
    //use this id to download the file using the Raster File
resource
    "id" : "<fileId1>",
    "size" : <fileSize1>,
    //an array of IDs of rasters that include this file
    "rasterIds" : [ <rasterId11>, <rasterId12> ]
  },
  {
    "id" : "<fileId2>",
    "size" : <fileSize2>,
    "rasterIds" : [ <rasterId21>, <rasterId22> ]
  }
]
}
```

JSON Response Example


```
{
"rasterFiles" : [

    {
      "id" :
"http://myserver/output/507978500/md/data/2w21w_5_s6c.tif",
      "size" : 390431,
      "rasterIds" : [
        5
      ]
    },
    {
      "id" :
"http://myserver/output/507978500/md/data/2w21w_5_s6c.tfw",
      "size" : 90,
      "rasterIds" : [
        5
      ]
    },
    {
      "id" : "http://myserver/data/2w22w.jpg",
      "size" : 1913965,
      "rasterIds" : [
        6
      ]
    },
    {
      "id" : "http://myserver/data/2w22w.aux",
      "size" : 18049,
      "rasterIds" : [
        6
      ]
    },
    {
      "id" : "http://myserver/data/2w22w.rrd",
      "size" : 2339130,
      "rasterIds" : [
        6
      ]
    }
  ]
}
```


8.1 Raster Catalog Item

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

8.1.1 Raster Catalog Item Reference

- **URL:** http://<imageservice-url>/<rasterId>
- **Child Resources:** Raster Image, Raster Thumbnail, Raster Info
- **Parent Resource:** Image Service

8.1.2 Raster Catalog Item Parameters

Parameter	Details
f	Description: The response format Values: json | (other formats)

8.1.3 Raster Catalog Item Example

Example

Return the raster catalog item for raster ID 8:

```
http://myserver/rest/services/Landsat/ImageServer/8
```

JSON Response Syntax

The response uses the feature object syntax as described in Section 11.0 Feature Object.

JSON Response Example

```
{
  "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
  }
}
```

8.2 Raster Image

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.

8.2.1 Raster Image Reference

- **URL:** http://<rastercatalogitem-url>/image
- **Parent Resource:** Raster Catalog Item

8.2.2 Raster Image Parameters

Parameter	Details
f	Description: The response format. If the format is image, the image bytes are directly streamed to the client. Values: json | image | (other formats)

bbox	Description: The extent (bounding box) of the exported image. Unless the bboxSR parameter has been specified, the bbox is assumed to be in the spatial reference of the image service.
	
	Syntax:
	
	`<xmin>, <ymin>, <xmax>, <ymax>`
	
	Example:
	
	`bbox=-104,35.6,-94.32,41`
	
	The bbox coordinates should always use a period as the decimal separator, even in countries where traditionally a comma is used.

size	Description: The size (width * height) of the exported image in pixels. If the size is not specified, an image with a default size of 400 * 400 is exported.
	
	Syntax:
	
	`<width>, <height>`
	
	Example:
	
	`size=600,550`

imageSR	Description: The spatial reference of the exported image. The spatial reference can be specified as either a well-known ID or a spatial reference JSON object. If a value for imageSR is not specified, the image is exported in the spatial reference of the image service.

Parameter	Details
bboxSR	Description: The spatial reference of the bbox. The spatial reference can be specified as either a well-known ID or a spatial reference JSON object. If a value for bboxSR is not specified, the bbox is assumed to be in the spatial reference of the image service.
format	Description: The format of the exported image. The default format value is png. Values: png | png8 | png24 | jpg | bmp | gif
pixelType	The pixel type, also known as data type, pertains to the type of values stored in the raster, such as signed integer, unsigned integer, or floating point. Integers are whole numbers, whereas floating points have decimals. Values: C128 | C64 | F32 | F64 | S16 | S32 | S8 | U1 | U16 | U2 | U32 | U4 | U8 | UNKNOWN
noData	The pixel value representing no information Example: `noData=0`
interpolation	The resampling process of extrapolating the pixel values while transforming the raster dataset when it undergoes warping or changes coordinate space. Values: RSP_BilinearInterpolation | RSP_CubicConvolution | RSP_Majority | RSP_NearestNeighbor
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. Example: `compressionQuality=75`

8.2.3 Raster Image Examples

Example

Return a raster image for raster ID 8:

```
http://myserver/rest/services/Landsat/ImageServer/8/image?bbox=378
501.375,4825171.125,634687.875,5051974.125&bandIds=0,1,2
```

Example 2

Return a raster image (f=image) for raster ID 8 in PNG 8 format:

```
http://myserver/rest/services/Landsat/ImageServer/8/image?bbox=378
501.375,4825171.125,634687.875,5051974.125&bandIds=0,1,2&f=image&f
ormat=png8
```

JSON Response Syntax

```
{
"href" : "<href>",
"width" : <width>,
"height" : <height>,
"extent" : <envelope>
}
```

JSON Response Example

```
{
  "href" : "http://myserver/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[\"N
orth_American_1983_HARN\",6378137.0,298.257222101]],PRIMEM[\"Green
wich\",0.0],UNIT[\"Degree\",0.0174532925199433]],PROJECTION[\"Lamb
ert_Conformal_Conic\"],PARAMETER[\"false_easting\",8202099.7375328
08],PARAMETER[\"false_northing\",0.0],PARAMETER[\"central_meridian
\",-
120.5],PARAMETER[\"standard_parallel_1\",44.33333333333334],PARAME
TER[\"standard_parallel_2\",46.0],PARAMETER[\"latitude_of_origin\"
,43.66666666666666],UNIT[\"Foot\",0.3048]]"
    }
  },
  "scale" : 0
}
```

8.3 Raster Thumbnail

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.

8.3.1 Raster Thumbnail Reference

- **URL:** http://<rastercatalogitem-url>/thumbnail
- **Parent Resource:** Raster Catalog Item

8.3.2 Raster Thumbnail Example

Example

Return a raster thumbnail for raster ID 8:

```
http://myserver/rest/services/Landsat/ImageServer/8/thumbnail
```

8.4 Raster Info

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

8.4.1 Raster Info Reference

- **URL:** http://<rastercatalogitem-url>/info
- **Parent Resource:** Raster Catalog Item

8.4.2 Raster Info Parameters

Parameter	Details
f	Description: The response format Values: json | (other formats)

8.4.3 Raster Info Example

Example

Return the raster info for the raster with ID 8:

```
http://myserver/rest/services/Landsat/ImageServer/8/info
```

JSON Response Syntax

```
{
  "origin": <origin>,
  "blockWidth": <blockWidth>,
  "blockHeight": <blockHeight>,
  "pixelSizeX": <pixelSizeX>,
  "pixelSizeY": <pixelSizeY>,
  "extent": <extent>,
  "bandCount": <bandCount>,
  "pixelType": "< C128 | C64 | F32 | F64 | S16 | S32 | S8 | U1 |
U16 | U2 | U32 | U4 | U8 | UNKNOWN >",
  "firstPyramidLevel": <firstPyramidLevel>,
  "maxPyramidLevel": <maxPyramidLevel>
}
```

JSON Response Example

```
{
  "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
}
```

8.5 Raster File

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.

8.5.1 Raster File Reference

- **URL:** http://<imageservice-url>/file
- **Parent Resource:** Image Service

8.5.2 Raster File Parameters

Parameter	Details
id	Description: The ID of the raster file. This ID is obtained by using the Download Rasters operation. Example: Suppose the Download Rasters operation returned the following response: `{ "rasterFiles": [ { "id": "t1923.png", "rasterIds": [54], "size": 65417 }, { "id": "t1923.pgw", "rasterIds": [54], "size": 765368 }, { "id": "t1923.png.aux.xml", "rasterIds": [54], "size": 853 } ] }` Users can download the second file from above by specifying its ID as the query parameter for this resource: id=t1923.pgw.

8.5.3 Raster File Example

Example

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

```
http://myserver/rest/services/Landsat/ImageServer/file?id=http://servername/1n2w13w.jpg
```

9.0 FEATURE SERVICE

A feature service allows clients to query and edit features. Features include geometry, attributes, and symbology and are organized into layers and subtypes within a layer.

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

9.0.1 Feature Service Reference

- **URL:** http://<catalog-url>/<serviceName>/FeatureServer
- **Parent Resource:** Catalog
- **Child Resource:** Layer

9.0.2 Feature Service Parameters

Parameter	Details
f	Description: The response format Values: json | (other formats)

9.0.3 Feature Service Example

Example

URL to the 311Incidents feature service running on myserver:

```
http://myserver/rest/services/311Incidents/FeatureServer?f=json
```

JSON Response Syntax

```
{
  "serviceDescription": "<serviceDescription>",
  //the feature layers published by this service
  "layers": [
    { "id": <layerId1>, "name": "<layerName1>" },
    { "id": <layerId2>, "name": "<layerName2>" }
  ],
  //the non-spatial tables published by this service
  "tables": [
    { "id": <tableId1>, "name": "<tableName1>" },
    { "id": <tableId2>, "name": "<tableName2>" }
  ]
}
```

JSON Response Example

```
{
  "serviceDescription": "Edit parcels, buildings and owner
information.",
  "layers": [
    { "id": 0, "name": "Parcels" }
    { "id": 1, "name": "Buildings" }
  ],
  "tables": [
    { "id": 3, "name": "Owners" }
  ]
}
```

9.1 Layer

The Layer resource represents a single editable layer or nonspatial table in a feature service.

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

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

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

Each feature template includes a template name, description, and a prototypical feature.

If a layer supports querying based on time, the response includes a timeInfo property. This gives 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.

If a layer has attachments, its hasAttachments property is set to true.

9.1.1 Layer Reference

- **URL:** http://<featureservice-url>/<layerId>
- **Supported Operations:** Query, Query Related Records, Add Features, Update Features, Delete Features, Apply Edits
- **Parent Resource:** Feature Service
- **Child Resource:** Feature

9.1.2 Layer Parameters

Parameter	Details
f	Description: The response format Values: json | (other formats)

9.1.3 Layer Example

Example

Get information about layer 0 in a feature service (311Incidents) running on myserver:

```
http://myserver/rest/services/311Incidents/FeatureServer/0?f=json
```

JSON Response Syntax

```
{

//properties applicable to both feature layers and tables
"id" : <layerOrTableId>,
"name" : "<layerOrTableName>",
"type" : "<layerOrTableType>", //"Feature Layer" or "Table"
"description" : "<description>",
"copyrightText" : "<copyrightText>",
"relationships" : [
  {
    "id" : <relationshipId1>,
    "name" : "<relationshipName1>",
    "relatedTableId" : <relatedTableId1>,
  },
```

J-9948

```
  {
    "id" : <relationshipId2>,
    "name" : "<relationshipName2>",
    "relatedTableId" : <relatedTableId2>,
  }
],

//properties applicable to feature layers only
"geometryType" : "<geometryType>",
"minScale" : <minScale>,
"maxScale" : <maxScale>,
"extent" :  <envelope>,

//for feature layers only
"drawingInfo" : {
  "renderer" : <renderer>,
  "transparency" : <transparency>,
  "labelingInfo" : <labelingInfo>
},

//if the layer / table supports querying based on time
"timeInfo" : {
  "startTimeField" : "<startTimeFieldName>",
  "endTimeField" : "<endTimeFieldName>",
  "trackIdField" : "<trackIdFieldName>",
  "timeExtent" : [<startTime>, <endTime>],
  "timeReference" : {
    "timeZone" : "<timeZone>",
    "respectsDaylightSaving" : <true | false>
  },
  "timeInterval" : <timeInterval>,
  "timeIntervalUnits" : "<timeIntervalUnits>"
},

//if the layer / table has attachments, the hasAttachments
property will be true
"hasAttachments" : <true | false>

//indicates whether the layer / table has htmlPopups
"htmlPopupType" : "<esriServerHTMLPopupTypeNone |
esriServerHTMLPopupTypeAsURL |
esriServerHTMLPopupTypeAsHTMLText>",

//layer / table fields
"objectIdField" : "<objectIdFieldName>",
"globalIdField" : "<globalIdFieldName>",
"displayField" : "<displayField>",
"typeIdField" : "<typeIFieldName>",
"fields" : [
  {"name" : "<fieldName1>", "type" : "<fieldType1>", "length" :
"<length1>", "editable" : "<true | false>", "domain" : <domain1>},
  {"name" : "<fieldName2>", "type" : "<fieldType2>", "length" :
"<length2>", "editable" : "<true | false>", "domain" : <domain2>}
],
//layer / table sub-types
"types" : [
```

J-9948

```
  {
    "id" : <typeId1>,
    "name" : "<typeName1>",
    "domains" : {
      "<domainField11>" : <domain11>,
      "<domainField12>" : <domain12>
    },
    "templates" : [
      {
        "name" : "<templateName11>",
        "description" : "<templateDescription11>",
        "prototype" : <prototypicalFeature11>
      },
      {
        "name" : "<templateName12>",
        "description" : "<templateDescription12>",
        "prototype" : <prototypicalFeature12>
      }
    ]
  },
  {
    "id" : <typeId2>,
    "name" : "<typeName2>",
    "domains" : {
      "<domainField11>" : <domain21>,
      "<domainField12>" : <domain22>
    },
    "templates" : [
      {
        "name" : "<templateName21>",
        "description" : "<templateDescription21>",
        "prototype" : <prototypicalFeature21>,
        "drawingTool": "esriFeatureEditToolNone |
esriFeatureEditToolPoint | esriFeatureEditToolLine |
esriFeatureEditToolPolygon |
                        esriFeatureEditToolAutoCompletePolygon |
esriFeatureEditToolCircle | esriFeatureEditToolEllipse |
esriFeatureEditToolRectangle |
                        esriFeatureEditToolFreehand"
      },
      {
        "name" : "<templateName22>",
        "description" : "<templateDescription22>",
        "prototype" : <prototypicalFeature22>,
        "drawingTool": "esriFeatureEditToolNone |
esriFeatureEditToolPoint | esriFeatureEditToolLine |
esriFeatureEditToolPolygon |
                        esriFeatureEditToolAutoCompletePolygon |
esriFeatureEditToolCircle | esriFeatureEditToolEllipse |
esriFeatureEditToolRectangle |
                        esriFeatureEditToolFreehand"
      }
    ]
  }
],
```

```
//layer / table templates - usually present when the layer / table
has no sub-types
"templates" : [
  {
    "name" : "<templateName1>",
    "description" : "<templateDescription1>",
    "prototype" : <prototypicalFeature1>
  },
  {
    "name" : "<templateName2>",
    "description" : "<templateDescription2>",
    "prototype" : <prototypicalFeature2>
  }
],

//comma separated list of supported capabilities - e.g.
"Query,Editing"
"capabilities" : "<capabilities>"

}
```

JSON Response Example

```
{
  "id" : 0,
  "name" : "Incidents",
  "type" : "Feature Layer",
  "displayField" : "req_id",
  "description" : "",
  "copyrightText" : "",
  "relationships" : [
    {
      "id" : 1,
      "name" : "ServiceRequest_IncidentPriority",
      "relatedTableId" : 1
    }
  ],
  "geometryType" : "esriGeometryPoint",
  "minScale" : 0,
  "maxScale" : 0,
  "extent" : {
    "xmin" : -122.514435102,
    "ymin" : 5.6843418860808E-14,
    "xmax" : 138.625776397,
    "ymax" : 67.1577965990001,
    "spatialReference" : {
      "wkid" : 4326
    }
  },
  "drawingInfo" : {"renderer" :
    {
      "type" : "uniqueValue",
      "field1" : "req_type",
      "field2" : null,
      "field3" : null,
      "fieldDelimiter" : ", ",
      "defaultSymbol" : null,
      "defaultLabel" : "\u003call other values\u003e",
```

J-9948


```
      "uniqueValueInfos" : [
        {
          "value" : "Blocked Street or Sidewalk",
          "label" : "Blocked Street or Sidewalk",
          "description" : "",
          "symbol" :
          {
            "type" : "esriPMS",
            "url" : "1DD4FC53",
            "imageData" :
"iVBORw0KGgoAAAANSUhEUgAAABoAAAAaCAYAAACpSkzOAAAAAXNSR0IB2cksfwAAA
AlwSFlzAAAOxAAADsQBlSsOGwAAAsVJREFUSInt001IVFEUwPH/a2bewwyCCZEZMI2
gBoRKSdtEROBKooyiDwqt8QNpYQmjYuIHDZGOVDM5RJAwUvRBizYFLmrXoo+F0iKkl
ROkSOQmyrnqvNNinLHRmdcY5Kqzuvfde8/vnnN5dtYp7P+hlbFnp0uq9isCw7PaP4O
qq1zy/PY0KGDRKYGR7NhfQ2mIgoGGWSq3O+VEd2bsT5AYGChU2uHqKpcUbJpG80CkB
2qrwFQGx/b+4EmnU05eW41ZQTIz85XCwoKMlWgeSK6fO2AkKpuHmt3w+LJLTt2cTsM
sK6qpOYJhGCilAPA1OmWgdaldgGmaicESElf5yDwcLYVHTS45fXcZywqJCObDLchCP
qZyiqNxKrW2EHKjBiBUv5Xz3nqYv59CRBlIDDpG9bR8lhVtOPON+B03xED1uzHaE5h
s3oX5+QO+F4rGJgfPXsPh3cuIJ6ATjUZzap0kB7bmKeLX3cg8zHWVkOefRK8dZa6rB
ACHXef4oCJ2z5kVsaxI9boxehMV6B1TxDrdSAzUAx/G2QB5/knqvlxAz0u0yArJCpU
6DUwFP1tK2BicBMD8rsMCLLx6inE2AMC2bdsJBm9Qd96LJ/AyK5IVevbmI83XrjISi
WiENCkuLiYajS5viKTnKyoqskSyQWKacUYikdQHb0MT3V2dABogExOf8Hh2pOYOXYf
Eu66tolAo+PtUc9hsAtDX55eeni7iS/9PX59fABw2m1Ux2aFweIhweCh5SwA6OtoA6
O6+kkRS+212x9ohEcm0Ly3Gx8fT5m2+Vm4Fw1xquZg7lEuUlZXB8nsIgK4blmdWQjI
2Nk55eVoiAIaHR8TrreXgwUMrc2iA6Lo9ieb2Hy3GM7fO663VABkIDFJZUb4Kq/fWZ
TyXEervDyAST95yVbx9995yPWeovd1nmWBfZcWagYzQv4x1g34BI70ZygdCoCgAAAA
ASUVORK5CYII=",
            "contentType" : "image/png",
            "color" : null,
            "width" : 19,
            "height" : 19,
            "angle" : 0,
            "xoffset" : 0,
            "yoffset" : 0
          }
        },
        {
          "value" : "Damaged Property",
          "label" : "Damaged Property",
          "description" : "",
          "symbol" :
          {
            "type" : "esriPMS",
            "url" : "DF3100A6",
            "imageData" :
"iVBORw0KGgoAAAANSUhEUgAAABQAAAAMCAYAAABiDJ37AAAAAXNSR0IB2cksfwAAA
AlwSFlzAAAOxAAADsQBlSsOGwAAAY1JREFUKJGl0j+Iz2EYAPDPw+lJZ+CY/B2kbCK
kDEZMN8hoMBoMFmVSSqKUWIyXSThlQAZ/LgMZZFNncUhRvq6L8nTyGu6H3P3uDJ56h
+d9n/fzvs/bO+A/ozGICr7DwKzFDTI34YfBwc+6biLoZtUslrlT1WpsbqyLzOuq7s8
BZW5UdR6vVB3AWGNT41Zkjqra0tgWVUvxSOZYVI1H1cdfxF9gVD1oLMfalnlK1apgT
3BE1X68DUZwI5hSNecJBubMkI1xDATvMY0lLfNaVB3v5Q8x1e9Nf4NtaGiHrnuJp8G
h3umfcFLmYVXLeqXf8LixO5joCzYWta9fDwa7Gq+DN1hppv1hVcPB3cZIsAJrsN28Y
ObpqNraGA9GW+Y7VctlTqsaDJ7IHI6qC1jSuBLcnrflqDqLfYu4Cq3qOZ6pWo8vOKN
qFBeDU/2gv0Em9bBe/qENDQ3rupst81ZUnWjckbm0VR0NLi0I9ovouheNc1F1EpPBs
ah63zL39vsu/wR7N71sZvyZq7q30J6fOkKkEEs/eqsAAAAASUVORK5CYII=",
            "contentType" : "image/png",
            "color" : null,
            "width" : 15,
            "height" : 9,
```

J-9948


```
            "angle" : 0,
            "xoffset" : 0,
            "yoffset" : 0
          }
        },
        {
          "value" : "Graffiti Complaint - Public Property",
          "label" : "Graffiti Complaint",
          "description" : "",
          "symbol" :
          {
            "type" : "esriPMS",
            "url" : "B2E6E7A0",
            "imageData" :
"iVBORw0KGgoAAAANSUhEUgAAABoAAAAaCAYAAACpSkzOAAAAAXNSR0IB2cksfwAAA
AlwSFlzAAAOxAAADsQBlSsOGwAAAiVJREFUSInt1EFIVFEUxvH/4DgnmUR8MTpjUJQ
PImgTIQbWoqCFixSEgoTaJFJZibawjNJCkiQIF5GImxaGGGIiZFiEBSa1ikLIGpLpW
UNGb8J8OWfMsUVGFGZag5D4re4958KPe+BeN4sU99KFdovkpUBFu2rlTkjLEGmYUO/
LLuzjCYW6VfuBF4Dcg2eo5teu9Wd3hexEOD+Nbn2jafLUtms+Oo5vDO4OhkK3ikX6J
g2jKVM14w1s6bTtWmD4r6EKwzj6yLLWxWHqs/qvFpjJqwYsq6hVdceZcNiogw9NhpG
TZRjbr9j2MOACphcMXbbtSoA74OkjlHksiFUiUlMgcrEXbqL6eMBxzo9DFLhdLnJgB
LY9VD0ZhufzhopFuiYhvkv1EGABvPJ6R9c4Tvp+r7esRdW6pppfDys6INqoeqnZNDs
iwaAdXsiNrqsW/tos9Pkin0T871XH9gQCeTfC4bZTEC0RuRCBzaXB4MFSkRPpsHUEa
rMgr1O1FXj7W2i2lA8Ntc1Wb1GtroGUDphoVj1bBoHVIvtGoe6wSNoAPHii2jtvaK6
cg4mZZWqOaXZmJ0lu9Nv+9N4pihqCg1U90P3P0PdUS6DdnyS5MSBp6gsuIA4bNxlGf
Y9tJw4Sg/QMdzIA0+5kXDN1Q1IFfjz2hP51rjl6S/j3XoaWof8H8nk89/vtdxtSPZ6
VLnBPQzwWi407qq8TCh0JhaqAqj+dW7TRfQVgAs+jjDbsggAAAABJRU5ErkJggg=="
,
            "contentType" : "image/png",
            "color" : null,
            "width" : 19,
            "height" : 19,
            "angle" : 0,
            "xoffset" : 0,
            "yoffset" : 0
          }
        }
      ]
    },
    "transparency" : 0,
    "labelingInfo" : null},
  "hasAttachments" : true,
  "htmlPopupType" : "esriServerHTMLPopupTypeAsHTMLText",
  "objectIdField" : "objectid",
  "globalIdField" : "",
  "typeIdField" : "req_type",
  "fields" : [
    {
      "name" : "objectid",
      "type" : "esriFieldTypeOID",
      "alias" : "Object ID",
      "editable" : false,
      "domain" : null
    },
    {
      "name" : "req_id",
      "type" : "esriFieldTypeString",
      "alias" : "Request ID",
      "editable" : true,
```

J-9948

```
      "length" : 20,
      "domain" : null
    },
    {
      "name" : "req_type",
      "type" : "esriFieldTypeString",
      "alias" : "Request Type",
      "editable" : true,
      "length" : 40,
      "domain" : null
    },
    {
      "name" : "req_date",
      "type" : "esriFieldTypeString",
      "alias" : "Request Date",
      "editable" : true,
      "length" : 30,
      "domain" : null
    },
    {
      "name" : "req_time",
      "type" : "esriFieldTypeString",
      "alias" : "Request Time",
      "editable" : true,
      "length" : 20,
      "domain" : null
    },
    {
      "name" : "address",
      "type" : "esriFieldTypeString",
      "alias" : "Address",
      "editable" : true,
      "length" : 60,
      "domain" : null
    }
,
    {
      "name" : "status",
      "type" : "esriFieldTypeSmallInteger",
      "alias" : "Status",
      "editable" : true,
      "domain" :
      {
        "type" : "codedValue",
        "name" : "StatusCodes",
        "codedValues" : [
          {
            "name" : "New",
            "code" : 1
          },
          {
            "name" : "Open",
            "code" : 2
          },
          {
            "name" : "Closed",
            "code" : 3
          }
        ]
```

J-9948


```
      }
    }
  ],
  "types" : [
    {
      "id" : "Blocked Street or Sidewalk",
      "name" : "Blocked Street or Sidewalk",
      "domains" :
      {
      },
      "templates" : [
        {
          "name" : "Blocked Street or Sidewalk",
          "description" : "",
          "drawingTool" : "esriFeatureEditToolPoint",
          "prototype" : {
            "attributes" : {
              "status" : 1,
              "req_id" : null,
              "req_type" : "Blocked Street or Sidewalk",
              "req_date" : null,
              "req_time" : null,
              "address" : null,
              "x_coord" : null,
              "y_coord" : null,
              "district" : null
            }
          }
        }
      ]
    },
    {
      "id" : "Damaged Property",
      "name" : "Damaged Property",
      "domains" :
      {
      },
      "templates" : [
        {
          "name" : "Damaged Property",
          "description" : "",
          "drawingTool" : "esriFeatureEditToolPoint",
          "prototype" : {
            "attributes" : {
              "status" : 1,
              "req_id" : null,
              "req_type" : "Damaged Property",
              "req_date" : null,
              "req_time" : null,
              "address" : null,
              "x_coord" : null,
              "y_coord" : null,
              "district" : null
            }
          }
        }
      ]
    },
```

```
    {
      "id" : "Graffiti Complaint - Public Property",
      "name" : "Graffiti Complaint",
      "domains" :
      {
      },
      "templates" : [
        {
          "name" : "Graffiti Complaint",
          "description" : "",
          "drawingTool" : "esriFeatureEditToolPoint",
          "prototype" : {
            "attributes" : {
              "status" : 1,
              "req_id" : null,
              "req_type" : "Graffiti Complaint - Public Property",
              "req_date" : null,
              "req_time" : null,
              "address" : null,
              "x_coord" : null,
              "y_coord" : null,
              "district" : null
            }
          }
        }
      ]
    }
  ],
  "templates" : [
  ],
  "capabilities" : "Query,Editing"
}
```

9.1.4 Query Operation (Feature Services)

The Query operation is performed on a feature service Layer resource. The result of this operation is either a feature set or an array of feature IDs (if returnIdsOnly is set to true).

In the feature set response, the layer features include their geometries. The records for tables do not. For time-aware layers, the time parameter can be used to specify the time instant or the time extent to query.

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

9.1.4.1 Query Reference

- **URL:** http://<featurelayer-url>/query
- **Parent Resource:** Layer

9.1.4.2 Query Parameters

Parameter	Details
f	Description: The response format Values: json | (other formats, for example, amf)

Parameter	Details

objectIds	Description: The ObjectIDs of this layer/table to be queried
	
	Syntax:
	
	`objectIds=<objectId1>, <objectId2>`
	
	Example:
	
	`objectIds=37, 462`

where	Description: A WHERE clause for the query filter. Any legal SQL WHERE clause operating on the fields in the layer is allowed. Note that this parameter will be ignored if a value for ObjectIDs is specified. Example: `where=POP2000 > 350000`

geometry	Description: The geometry to apply as the spatial filter. The structure of the geometry is the same as the structure of the JSON geometry objects returned by the GeoServices REST Specification. In addition to the JSON structures, for envelopes and points, users can specify the geometry with a simpler comma-separated syntax.
	
	Syntax:
	
	■ JSON structures:
	`geometryType=<geometryType>&geometry={geometry}`
	
	■ Envelope simple:
	`geometryType=esriGeometryEnvelope&geometry=<xmin>,<ymin>,<xmax>,<ymax>`
	
	■ Point simple:
	`geometryType=esriGeometryPoint&geometry=<x>,<y>`
	
	Examples:
	
	■ `geometryType=esriGeometryEnvelope&geometry={xmin: -104, ymin: 35.6, xmax: -94.32, ymax: 41}`
	
	■ `geometryType=esriGeometryEnvelope&geometry=-104,35.6,-94.32,41`
	
	■ `geometryType=esriGeometryPoint&geometry=-104,35.6`

Parameter	Details
geometryType	Description: The type of geometry specified by the geometry parameter. The geometry type can be an envelope, point, line, or polygon. The default geometry type is an envelope. Values: esriGeometryPoint | esriGeometryMultipoint | esriGeometryPolyline | esriGeometryPolygon | esriGeometryEnvelope
inSR	Description: The spatial reference of the input geometry. The spatial reference can be specified as either a well-known ID or a spatial reference JSON object. If a value for inSR is not specified, the geometry is assumed to be in the spatial reference of the map.
spatialRel	Description: The spatial relationship to be applied on the input geometry while performing the query. The supported spatial relationships include intersects, contains, envelope intersects, and within. The default spatial relationship is intersects (esriSpatialRelIntersects). Values: esriSpatialRelIntersects | esriSpatialRelContains | esriSpatialRelCrosses | esriSpatialRelEnvelopeIntersects | esriSpatialRelIndexIntersects | esriSpatialRelOverlaps | esriSpatialRelTouches | esriSpatialRelWithin | esriSpatialRelRelation
relationParam	Description: The spatial relate function that can be applied while performing the Query operation. An example for this spatial relate function is 'FFFTTT***'. For additional information on how to construct this string, see the relationParam description in Section 4.2.4.2 Query Parameters.

time	Description: The time instant or the time extent to query
	
	Time Instant
	
	Syntax:
	
	`time=<timeInstant>`
	
	Example:
	
	`time=1199145600000`
	
	(1 Jan. 2008 00:00:00 GMT)
	
	Time Extent
	
	Syntax:
	
	`time=<startTime>, <endTime>`
	
	Example:
	
	`time=1199145600000, 1230768000000`

Parameter	Details
	(1 Jan. 2008 00:00:00 GMT to 1 Jan. 2009 00:00:00 GMT) A null value specified for start time or end time represents infinity for start or end time, respectively.

outFields	Description: The list of fields to be included in the returned result set. This is a comma-delimited list of field names.
	
	If the wildcard "*" is specified as the value of this parameter, the query results include all the field values.
	
	Note that the wildcard also implies returnGeometry=true, and setting returnGeometry to false will have no effect.
	
	Example:
	
	`outfields=AREANAME,ST,POP2000`
	
	Example (wildcard usage):
	
	`outfields=*`

returnGeometry	Description: If true, the result set includes the geometry associated with each result. The default is true. Note that if the outfields parameter is set to the wildcard, it implies returnGeometry=true, and setting returnGeometry to false has no effect. Values: true | false
outSR	Description: The spatial reference of the returned geometry. The spatial reference can be specified as either a well-known ID or a spatial reference JSON object. If a value for outSR is not specified, the geometry is returned in the spatial reference of the map.
returnIdsOnly	Description: If true, the response includes only an array of ObjectIDs. Otherwise, the response is a feature set. The default is false. Note that when a value for objectIds is specified, setting this parameter to true is invalid. Values: false | true

9.1.4.3 Query Examples

Example 1

Make a query using a WHERE clause:

```
http://myserver/rest/services/Earthquakes/FeatureServer/0/query?wh
ere=magnitude+%3E+4.5&outFields=*&returnGeometry=true&returnIdsOnl
y=false&f=json
```

Example 2

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

```
http://myserver/rest/services/311Incidents/FeatureServer/1/query?w
here=agree_with_incident+%3D+1
&returnGeometry=true&returnIdsOnly=true&f=json
```

JSON Response Syntax (when returnIdsOnly=false)

```
{
"objectIdFieldName" : "<objectIdFieldName>",
"globalIdFieldName" : "<globalIdFieldName>",
"geometryType" : "<geometryType>",
"spatialReference" : <spatialReference>,
"fields" : [
    {"name" : "<fieldName1>", "type" : "<fieldType1>", "alias" :
"<fieldAlias1>", "length" : "<length1>"},
    {"name" : "<fieldName2>", "type" : "<fieldType2>", "alias" :
"<fieldAlias2>", "length" : "<length2>"}
],
"features" : [
  <feature1>, <feature2>
]
}
```

JSON Response Syntax (when returnIdsOnly=true)

```
{
"objectIdFieldName" : "<objectIdFieldName>",
"objectIds" : [ <objectId1>, <objectId2> ]
}
```

JSON Response Example (when returnIdsOnly=false)

```
{
  "objectIdFieldName" : "objectid",
  "globalIdFieldName" : "",
  "geometryType" : "esriGeometryPoint",
  "spatialReference" : {
    "wkid" : 4326
  },
    "fields" : [
    {
      "name" : "objectid",
      "type" : "esriFieldTypeOID",
      "alias" : "Object ID"
    },
    {
      "name" : "datetime",
      "type" : "esriFieldTypeDate",
      "alias" : "Earthquake Date",
      "length" : 36
    },
```

J-9948


```
    {
      "name" : "depth",
      "type" : "esriFieldTypeDouble",
      "alias" : "Depth"
    },
    {
      "name" : "eqid",
      "type" : "esriFieldTypeString",
      "alias" : "Earthquake ID",
      "length" : 50
    },
    {
      "name" : "latitude",
      "type" : "esriFieldTypeDouble",
      "alias" : "Latitude"
    },
    {
      "name" : "longitude",
      "type" : "esriFieldTypeDouble",
      "alias" : "Longitude"
    },
    {
      "name" : "magnitude",
      "type" : "esriFieldTypeDouble",
      "alias" : "Magnitude"
    },
    {
      "name" : "numstations",
      "type" : "esriFieldTypeInteger",
      "alias" : "Number of Stations"
    },
    {
      "name" : "region",
      "type" : "esriFieldTypeString",
      "alias" : "Region",
      "length" : 200
    },
    {
      "name" : "source",
      "type" : "esriFieldTypeString",
      "alias" : "Source",
      "length" : 50
    },
    {
      "name" : "version",
      "type" : "esriFieldTypeString",
      "alias" : "Version",
      "length" : 50
    }
  ],
  "features" : [
    {
      "geometry" : {
        "x" : -178.24479999999991,
        "y" : 50.012500000000045
      },
      "attributes" : {
        "objectid" : 3745682,
        "datetime" : 1272210710000,
```

```
        "depth" : 31.100000000000001,
        "eqid" : "2010vma5",
        "latitude" : 50.012500000000003,
        "longitude" : -178.2448,
        "magnitude" : 4.7999999999999998,
        "numstations" : 112,
        "region" : "Andreanof Islands, Aleutian Islands, Alaska",
        "source" : "us",
        "version" : "Q"
      }
    },
    {
      "geometry" : {
        "x" : -72.865099999999927,
        "y" : -37.486599999999953
      },
      "attributes" : {
        "objectid" : 3745685,
        "datetime" : 1272210142999,
        "depth" : 40.600000000000001,
        "eqid" : "2010vma4",
        "latitude" : -37.486600000000003,
        "longitude" : -72.865099999999998,
        "magnitude" : 4.9000000000000004,
        "numstations" : 58,
        "region" : "Bio-Bio, Chile",
        "source" : "us",
        "version" : "7"
      }
    }
  ]
}
```

JSON Response Example (when returnIdsOnly=true)

```
{
"objectIdFieldName" : "objectid",
 "objectIds" : [1, 2, 3, 4, 5, 7]
}
```

9.1.5 Query Related Records Operation

The Query Related Records operation is performed on a feature service Layer resource. The result of this operation is feature sets grouped by source layer/table ObjectIDs. Each feature set contains GeoServices REST Specification feature objects including the values for the fields requested by the user.

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. Note that all parameters related to geometry are ignored when querying related tables.

9.1.5.1 Query Related Records Reference

- **URL:** http://<featurelayer-url>/queryRelatedRecords
- **Parent Resource:** Layer

9.1.5.2 Query Related Records Parameters

Parameter	Details
f	Description: The response format Values: json | (other formats, for example, amf)

objectIds	Description: The ObjectIDs of the layer/table to be queried. Records related to these ObjectIDs are queried.
	
	Syntax:
	
	`objectIds=<objectId1>, <objectId2>`
	
	Example:
	
	`objectIds=37, 462`

relationshipId	Description: The ID of the relationship to be queried. The relationships that this layer/table participates in are included in the feature service Layer resource response. Records in tables/layers corresponding to the related table/layer of the relationship are queried. Example: `relationshipId=4`

outFields	Description: The list of fields from the related table/layer to be included in the returned feature set. This is a comma-delimited list of field names. If a shape field is specified in the list of return fields, it is ignored. To request geometry, set returnGeometry to true.
	
	If the wildcard "*" is specified as the value of this parameter, the results include all the field values.
	
	Example:
	
	`outFields=AREANAME,ST,POP2000`
	
	Example (wildcard usage):
	
	`outFields=*`

definitionExpression	Description: The definition expression to be applied to the related table/layer. From the list of ObjectIDs, only those records that conform to this expression are returned. Example: `definitionExpression=POP2000 > 100000`

Parameter	Details
returnGeometry	Description: If true, the feature set includes the geometry associated with each feature. The default is true. Note that this parameter only applies to related layers. It is ignored for related tables. Also, note that if the outFields parameter is set to the wildcard, it implies returnGeometry=true, and setting returnGeometry to false has no effect. Values: true | false
outSR	Description: The spatial reference of the returned geometry. The spatial reference can be specified as either a well-known ID or a spatial reference JSON object. If a value for outSR is not specified, the geometry is returned in the spatial reference of the map. Note that this parameter only applies to related layers. It is ignored for related tables.

9.1.5.3 Query Related Records Example

Example

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

```
http://myserver/rest/services/Petroleum/FeatureServer/0/queryRelat
edRecords?objectIds=3,4,5&relationshipId=2&returnGeometry=true&out
Fields=*&f=json
```

JSON Response Syntax

```
{
"geometryType" : "<geometryType>", //if records include geometry
"spatialReference" : <spatialReference>, //if records include
geometry
"fields" : [
    {"name" : "<fieldName1>", "type" : "<fieldType1>", "alias" :
"<fieldAlias1>", "length" : "<length1>"},
    {"name" : "<fieldName2>", "type" : "<fieldType2>", "alias" :
"<fieldAlias2>", "length" : "<length2>"}
],
"relatedRecordGroups" : [
  {
    "objectId" : <objectId1>,
    "relatedRecords" : [ //features may include geometry for
related layers only
      <relatedFeature11>, <relatedFeature12>
    ]
  },
```

```
  {
    "objectId" : <objectId2>,
    "relatedRecords" : [
      <relatedFeature21>, <relatedFeature22>
    ]
  }
]
}
```

JSON Response Example

```
{
  "geometryType" : "esriGeometryPolygon",
  "spatialReference" : {
    "wkid" : 4267
  },
  "fields" : [
    {
      "name" : "OBJECTID",
      "type" : "esriFieldTypeOID",
      "alias" : "OBJECTID"},
    {
      "name" : "FIELD_KID",
      "type" : "esriFieldTypeString",
      "alias" : "FIELD_KID",
      "length" : 25},
    {
      "name" : "APPROXACRE",
      "type" : "esriFieldTypeDouble",
      "alias" : "APPROXACRE"},
    {
      "name" : "FIELD_NAME",
      "type" : "esriFieldTypeString",
      "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
                ]
              ]
            ]
          }
        }
      ]
    }
  ]
}
```

9.1.6 Add Features Operation

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

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

9.1.6.1 Add Features Reference

- **URL:** http://<featurelayer-url>/addFeatures (POST only)
- **Parent Resource:** Layer

9.1.6.2 Add Features Parameters

Parameter	Details
f	Description: The response format Values: json | (other formats)
features	Description: The array of features to be added. The structure of each feature in the array is the same as the structure of the JSON feature object returned by the GeoServices REST Specification. Features to be added to a feature layer should include the geometry. Records to be added to a table should not include the geometry. Syntax: `[ <feature1>, <feature2> ]`

Parameter	Details
	Example: `[{ "geometry" : {"x" : -118.15, "y" : 33.80}, "attributes" : { "OWNER" : "Joe Smith", "VALUE" : 94820.37, "APPROVED" : true, "LASTUPDATE" : 1227663551096 } }, { "geometry" : { "x" : -118.37, "y" : 34.086 }, "attributes" : { "OWNER" : "John Doe", "VALUE" : 17325.90, "APPROVED" : false, "LASTUPDATE" : 1227628579430 } }]`

9.1.6.3 Add Features Example

Example

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

```
http://myserver/rest/services/311Incidents/FeatureServer/0/
addFeatures
```

An input array of features might look like the following:

```
[
    {
      "attributes" : {
        "req_id" : "508389",
        "req_type" : "Graffiti Complaint - Public Property",
        "req_date" : "09\/19\/2009",
        "req_time" : "18:44",
        "address" : "11TH ST and HARRISON ST",
        "x_coord" : "6008925.0",
        "y_coord" : "2108713.8",
        "district" : "6",
        "status" : 1
      },
      "geometry" : {
        "x" : -122.41247978999991,
        "y" : 37.770630098000083
      }
    }
]
```

JSON Response Syntax

```
{
  "addResults" : [
    {
      "objectId" : <objectId1>,
      "globalId" : <globalId1>,
      "success" : <true | false>,
      "error" : { //only if success is false
        "code" : <code1>,
        "description" : "<description1>",
      }
    },
    {
      "objectId" : <objectId2>,
      "globalId" : <globalId2>,
      "success" : <true | false>,
      "error" : { //only if success is false
        "code" : <code2>,
        "description" : "<description2>",
      }
    }
  ]
}
```

JSON Response Example

```
{
  "addResults" : [
    {
      "objectId" : 37,
      "globalId" : null,
      "success" : true
    },
    {
      "objectId" : -1,
      "globalId" : null,
      "success" : false,
      "error" : {
        "code" : 50,
        "description" : "Cannot add unapproved parcels.",
      }
    }
  ]
}
```

9.1.7 Update Features Operation

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

Users can provide arguments to the Update Features operation as query parameters.

9.1.7.1 Update Features Reference

- **URL:** http://<featurelayer-url>/updateFeatures (POST only)
- **Parent Resource:** Layer

9.1.7.2 Update Features Parameters

Parameter	Details
f	Description: The response format Values: json | (other formats)

features	Description: The array of features to be updated. The structure of each feature in the array is the same as the structure of the JSON feature object returned by the GeoServices REST Specification.
	
	The attributes property of the feature should include the ObjectID (and the global ID, if available) of the feature along with the other attributes:
	
	`"attributes":`
	
	`{`
	`  "OBJECTID" : 37,`
	`  "OWNER" : "Joe Smith",`
	`  "VALUE" : 94820.37,`
	`  "APPROVED" : true,`
	`  "LASTUPDATE" : 1227667627940`
	`}`
	
	Features to be updated to a feature layer should include the geometry. Records to be added to a table should not include the geometry.
	
	Syntax:
	
	`[ <feature1>, <feature2> ]`
	
	Example:
	
	`[`
	`  {`
	`  "geometry" : {"x" : -118.15, "y" : 33.80},`
	`  "attributes" : {`
	`  "OBJECTID" : 37`
	`  "OWNER" : "Joe Smith",`
	`  "VALUE" : 94820.37,`
	`  "APPROVED" : true,`
	`  "LASTUPDATE" : 1227667627940`
	`  }`
	`  },`
	`  {`
	`  "geometry" : { "x" : -118.37, "y" : 34.086`
	`},`
	`  "attributes" : {`
	`  "OBJECTID" : 462`

Parameter	Details

	<pre> "OWNER" : "John Doe",
	      "VALUE" : 17325.90,
	      "APPROVED" : false,
	      "LASTUPDATE" : 9269154204840
	    }
	  }
	]</pre>

9.1.7.3 Update Features Example

Example

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

```
http://myserver/rest/services/311Incidents/FeatureServer/0/updateF
eatures
```

An input array of features might look like the following:

```
[
    {
      "attributes" : {
   "objectid": 1234567
        "req_id" : "508389",
        "req_type" : "Graffiti Complaint - Private Property",
        "req_date" : "09\/19\/2009",
        "req_time" : "18:44",
        "address" : "11TH ST and HARRISON ST",
        "x_coord" : "6008925.0",
        "y_coord" : "2108713.8",
        "district" : "6",
        "status" : 2
      },
      "geometry" : {
        "x" : -122.41247978999991,
        "y" : 37.770630098000083
      }
    }
]
```

JSON Response Syntax

```
{
  "updateResults" : [
    {
      "objectId" : <objectId1>,
      "globalId" : <globalId1>,
      "success" : <true | false>,
      "error" : { //only if success is false
        "code" : <code1>,
        "description" : "<description1>",
      }
    },
```

```
    {
      "objectId" : <objectId2>,
      "globalId" : <globalId2>,
      "success" : <true | false>,
      "error" : { //only if success is false
        "code" : <code2>,
        "description" : "<description2>",
      }
    }
  ]
}
```

JSON Response Example

```
{
  "updateResults" : [
    {
      "objectId" : 37,
      "globalId" : null,
      "success" : true
    },
    {
      "objectId" : 462,
      "globalId" : null,
      "success" : false,
      "error" : {
        "code" : 30,
        "description" : "'LASTUPDATED' date cannot be in the
future.",
      }
    }
  ]
}
```

9.1.8 Delete Features Operation

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

Users can provide arguments to the Delete Features operation as query parameters.

9.1.8.1 Delete Features Reference

- **URL:** http://<featurelayer-url>/deleteFeatures (POST only)
- **Parent Resource:** Layer

9.1.8.2 Delete Features Parameters

Parameter	Details
f	Description: The response format Values: json | (other formats)

Parameter	Details

objectIds	Description: The ObjectIDs of the layer/table to be deleted. Note that when this parameter is specified, any other filter parameters (including where) are ignored.
	
	Syntax:
	
	`objectIds=<objectId1>, <objectId2>`
	
	Example:
	
	`objectIds=37, 462`

where	Description: A WHERE clause for the query filter. Any legal SQL WHERE clause operating on the fields in the layer is allowed. Features conforming to the specified WHERE clause are deleted. Note that this parameter is ignored if a value for ObjectIDs is specified. Example: `where=POP2000 > 350000`

geometry	Description: The geometry to apply as the spatial filter. Features conforming to the spatial relationship (specified using the spatialRel parameter) of this geometry will be deleted.
	
	Note that this parameter is ignored if a value for ObjectIDs is specified.
	
	The structure of the geometry is the same as the structure of the JSON geometry objects returned by the GeoServices REST Specification. In addition to the JSON structures, for envelopes and points, users can specify the geometry with a simpler comma-separated syntax.
	
	Syntax:
	
	■ JSON structures:
	`geometryType=<geometryType>&geometry={geometry}`
	
	■ Envelope simple:
	`geometryType=esriGeometryEnvelope&geometry=<xmin>,<ymin>,<xmax>,<ymax>`
	
	■ Point simple:
	`geometryType=esriGeometryPoint&geometry=<x>,<y>`
	
	Examples:
	
	■ `geometryType=esriGeometryEnvelope&geometry={xmin: -104, ymin: 35.6, xmax: -94.32, ymax: 41}`

Parameter	Details
	■ `geometryType=esriGeometryEnvelope&geometry=-104,35.6,-94.32,41` ■ `geometryType=esriGeometryPoint&geometry=-104,35.6`
geometryType	Description: The type of geometry specified by the geometry parameter. The geometry type can be an envelope, point, line, or polygon. The default geometry type is an envelope. Values: esriGeometryPoint | esriGeometryMultipoint | esriGeometryPolyline | esriGeometryPolygon | esriGeometryEnvelope
inSR	Description: The spatial reference of the input geometry. The spatial reference can be specified as either a well-known ID or a spatial reference JSON object. If a value for inSR is not specified, the geometry is assumed to be in the spatial reference of the map.
spatialRel	Description: The spatial relationship to be applied on the input geometry while performing the query. The supported spatial relationships include intersects, contains, envelope intersects, and within. The default spatial relationship is intersects (esriSpatialRelIntersects). Values: esriSpatialRelIntersects | esriSpatialRelContains | esriSpatialRelCrosses | esriSpatialRelEnvelopeIntersects | esriSpatialRelIndexIntersects | esriSpatialRelOverlaps | esriSpatialRelTouches | esriSpatialRelWithin

9.1.8.3 Delete Features Example

Example

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

```
http://myserver/rest/services/311Incidents/FeatureServer/0/deleteFeatures
```

The input to the Delete Features operation can be a list of unique IDs, a WHERE clause to apply as an attribute filter, or a geometry to apply as a spatial filter.

JSON Response Syntax (when ObjectIDs are specified)

```
{
  "deleteResults" : [
    {
      "objectId" : <objectId1>,
      "globalId" : <globalId1>,
      "success" : <true | false>,
      "error" : { //only if success is false
        "code" : <code1>,
        "description" : "<description1>",
      }
    },
```

```
    {
      "objectId" : <objectId2>,
      "globalId" : <globalId2>,
      "success" : <true | false>,
      "error" : { //only if success is false
        "code" : <code2>,
        "description" : "<description2>",
      }
    }
  ]
}
```

JSON Response Syntax (when ObjectIDs are not specified)

```
{ "success" : true }
```

JSON Response Example

```
{
  "deleteResults" : [
    {
      "objectId" : 37,
      "globalId" : null,
      "success" : true
    },
    {
      "objectId" : 462,
      "globalId" : null,
      "success" : false,
      "error" : {
        "code" : 60,
        "description" : "Features whose last update was less than
2 days ago cannot be deleted.",
      }
    }
  ]
}
```

9.1.9 Apply Edits

The Apply Edits operation adds, updates, and deletes features to the associated feature layer or table in a single call (POST only). The Apply Edits operation is performed on a feature service Layer resource.

The results of this operation are three arrays of edit results (for additions, updates, and deletions, respectively). Each edit result identifies a single feature and indicates if the edits were successful or not. If not, it also includes an error code and an error description.

Users can provide arguments to the Apply Edits operation as query parameters.

9.1.9.1 Apply Edits Reference

- **URL:** http://<featurelayer-url>/applyEdits (POST only)
- **Parent Resource:** Layer

9.1.9.2 Apply Edits Parameters

Parameter	Details
f	Description: The response format Values: json | (other formats)

adds	Description: The array of features to be added. The structure of each feature in the array is the same as the structure of the JSON feature object returned by the GeoServices REST Specification.
	
	Features to be added to a feature layer should include the geometry. Records to be added to a table should not include the geometry.
	
	Syntax:
	
	`[ <feature1>, <feature2> ]`
	
	Example:
	
	`[{ "geometry" : {"x" : -118.15, "y" : 33.80}, "attributes" : { "OWNER" : "Joe Smith", "VALUE" : 94820.37, "APPROVED" : true, "LASTUPDATE" : 1227663551096 } }, { "geometry" : { "x" : -118.37, "y" : 34.086 }, "attributes" : { "OWNER" : "John Doe", "VALUE" : 17325.90, "APPROVED" : false, "LASTUPDATE" : 1227628579430 } }]`

updates	Description: The array of features to be updated. The structure of each feature in the array is the same as the structure of the JSON feature object returned by the GeoServices REST Specification. The attributes property of the feature should include the ObjectID (and the global ID, if available) of the feature along with the other attributes: `"attributes" : { "OBJECTID" : 37, "OWNER" : "Joe Smith", "VALUE" : 94820.37,`

J-9948

Parameter	Details

	<pre> "APPROVED" : true,
	  "LASTUPDATE" : 1227667627940
	}</pre>
	Features to be updated to a feature layer should include the geometry. Records to be added to a table should not include the geometry.
	
	Syntax:
	
	<pre>[<feature1>, <feature2>]</pre>
	Example:
	
	<pre>[
	  {
	    "geometry" : {"x" : -118.15, "y" : 33.80},
	    "attributes" : {
	      "OBJECTID" : 37
	      "OWNER" : "Joe Smith",
	      "VALUE" : 94820.37,
	      "APPROVED" : true,
	      "LASTUPDATE" : 1227667627940
	    }
	  },
	  {
	    "geometry" : { "x" : -118.37, "y" : 34.086
	},
	    "attributes" : {
	      "OBJECTID" : 462
	      "OWNER" : "John Doe",
	      "VALUE" : 17325.90,
	      "APPROVED" : false,
	      "LASTUPDATE" : 9269154204840
	    }
	  }
	]</pre>

deletes	Description: The ObjectIDs of the layer/table to be deleted Syntax: <pre>deletes=<objectId1>, <objectId2></pre> Example: <pre>deletes=37, 462</pre>

9.1.9.3 Apply Edits Example

Example

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

```
http://myserver/rest/services/311Incidents/FeatureServer/0/apply
Edits
```

Input for additions, represented by an array of features, might look like the following:

```
[
    {
      "attributes" : {
        "req_id" : "508389",
        "req_type" : "Graffiti Complaint - Public Property",
        "req_date" : "09\/19\/2009",
        "req_time" : "18:44",
        "address" : "11TH ST and HARRISON ST",
        "x_coord" : "6008925.0",
        "y_coord" : "2108713.8",
        "district" : "6",
        "status" : 1
      },
      "geometry" : {
        "x" : -122.41247978999991,
        "y" : 37.770630098000083
      }
    }
]
```

Example 2

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

```
http://myserver/rest/services/311Incidents/FeatureServer/0/applyEd
its
```

Input for updates, represented by an array of features, might look like the following:

```
[
    {
      "attributes" : {
   "objectid": 1234567
        "req_id" : "508389",
        "req_type" : "Graffiti Complaint - Private Property",
        "req_date" : "09\/19\/2009",
        "req_time" : "18:44",
        "address" : "11TH ST and HARRISON ST",
        "x_coord" : "6008925.0",
        "y_coord" : "2108713.8",
        "district" : "6",
        "status" : 2
      },
```

```
      "geometry" : {
        "x" : -122.41247978999991,
        "y" : 37.770630098000083
      }
    }
]
```

Example 3

Delete features using the Apply Edits operation on a feature service Layer resource:

```
http://myserver/rest/services/311Incidents/FeatureServer/0/apply
Edits
```

The input for deletions is a list of ObjectIDs of the features to be deleted.

JSON Response Syntax

```
{
  "addResults" : [
    {
      "objectId" : <objectId1>,
      "globalId" : <globalId1>,
      "success" : <true | false>,
      "error" : { //only if success is false
        "code" : <code1>,
        "description" : "<description1>",
      }
    },
    {
      "objectId" : <objectId2>,
      "globalId" : <globalId2>,
      "success" : <true | false>,
      "error" : { //only if success is false
        "code" : <code2>,
        "description" : "<description2>",
      }
    }
  ],
  "updateResults" : [
    {
      "objectId" : <objectId1>,
      "globalId" : <globalId1>,
      "success" : <true | false>,
      "error" : { //only if success is false
        "code" : <code1>,
        "description" : "<description1>",
      }
    },
    {
      "objectId" : <objectId2>,
      "globalId" : <globalId2>,
      "success" : <true | false>,
      "error" : { //only if success is false
        "code" : <code2>,
        "description" : "<description2>",
      }
```

```
    }
  ],
  "deleteResults" : [
    {
      "objectId" : <objectId1>,
      "globalId" : <globalId1>,
      "success" : <true | false>,
      "error" : { //only if success is false
        "code" : <code1>,
        "description" : "<description1>",
      }
    },
    {
      "objectId" : <objectId2>,
      "globalId" : <globalId2>,
      "success" : <true | false>,
      "error" : { //only if success is false
        "code" : <code2>,
        "description" : "<description2>",
      }
    }
  ]
}
```

JSON Response Example

```
{
  "addResults" : [
    {
      "objectId" : 37,
      "globalId" : null,
      "success" : true
    },
    {
      "objectId" : -1,
      "globalId" : null,
      "success" : false,
      "error" : {
        "code" : 50,
        "description" : "Cannot add unapproved parcels.",
      }
    }
  ],
  "updateResults" : [
    {
      "objectId" : 463,
      "globalId" : null,
      "success" : true
    },
    {
      "objectId" : 462,
      "globalId" : null,
      "success" : false,
      "error" : {
        "code" : 30,
        "description" : "'LASTUPDATED' date cannot be in the
future.",
      }
```

```
    }
  ],
  "deleteResults" : [
    {
      "objectId" : 9,
      "globalId" : null,
      "success" : true
    },
    {
      "objectId" : 625,
      "globalId" : null,
      "success" : false,
      "error" : {
        "code" : 60,
        "description" : "Features whose last update was less than
2 days ago cannot be deleted.",
      }
    }
  ]
}
```

9.2 Feature (Feature Service)

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

The Feature resource has two child resources:

- Attachment Infos: Returns information about attachments associated with this feature. This resource is available only if the layer has advertised that it has attachments.
- HTML Popup: Returns information about this feature that is intended for display in an HTML pop-up balloon.

9.2.1 Feature Reference

- **URL:** http://<featurelayer-url>/<featureId>
- **Supported Operations:** Add Attachment, Update Attachment, Delete Attachments
- **Parent Resource:** Feature Layer
- **Child Resources:** Attachment Infos, HTML Popup

9.2.2 Feature Parameters

Parameter	Details
f	Description: The response format Values: json | (other formats)

9.2.3 Feature Example

Example

Retrieve a feature:

```
http://myserver/rest/services/Watershed/FeatureServer/1/1?f=json
```

JSON Response Syntax

```
{ "feature" : <feature> }
```

JSON Response Example

```
{
  "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
    }
  }
}
```

9.2.4 Add Attachment Operation

The Add Attachment operation adds an attachment to the associated feature (POST only). The Add Attachment operation is performed on a feature service Feature resource.

Since this request uploads a file, it must be a multipart request as per Internet Engineering Task Force (IETF) RFC1867.

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

This operation returns an array of results indicating whether the individual edits were successful or not. If not, the result includes an error code and an error description. If successful, the ObjectID of the result is the ID of the new attachment.

Users can provide arguments to the Add Attachment operation as query parameters.

9.2.4.1 Add Attachment Reference

- **URL:** http://<featureservicefeature-url>/addAttachment (POST only)
- **Parent Resource:** Feature (from a feature service)

9.2.4.2 Add Attachment Parameters

Parameter	Details
f	Description: The response format Values: json | (other formats)

Parameter	Details
attachment	Description: The file to be uploaded as a new feature attachment. The content type, size, and name of the attachment will be derived from the uploaded file.

9.2.4.3 Add Attachment Example

Example

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

```
http://myserver/rest/services/311Incidents/FeatureServer/0/818654/
addAttachment
```

JSON Response Syntax

```
{
  "addAttachmentResult": {
    "objectId" : <attachmentId>,
    "globalId" : <globalId>,
    "success" : <true | false>,
    "error" : { //only if success is false
      "code" : <code1>,
      "description" : "<description>",
    }
  }
}
```

JSON Response Example

```
{
  "addAttachmentResult": {
    "objectId" : 58,
    "globalId" : null,
    "success" : true
  }
}
```

9.2.5 Update Attachment Operation

The Update Attachment operation updates an attachment associated with a feature (POST only). The Update Attachment operation is performed on a feature service Feature resource.

Since this request uploads a file, it must be a multipart request as per IETF RFC1867.

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

This operation returns an array of results indicating whether the individual edits were successful or not. If not, the result includes an error code and an error description. If successful, the ObjectID of the result is the ID of the updated attachment.

9.2.5.1 Update Attachment Reference

- **URL:** http://<featureservicefeature-url>/updateAttachment (POST only)
- **Parent Resource:** Feature (from a feature service)

9.2.5.2 Update Attachment Parameters

Parameter	Details
f	Description: The response format Values: json | (other formats)
attachmentId	Description: The ID of the attachment to be updated Example: `attachmentId=58`
attachment	Description: The file to be uploaded as the updated feature attachment. The content type, size, and name of the attachment are derived from the uploaded file.

9.2.5.3 Update Attachment Example

Example

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

```
http://myserver/rest/services/311Incidents/FeatureServer/0/818654/
updateAttachment
```

The values for the input parameters attachmentIds and attachment to this operation are the ID of the attachment to be updated and a file, respectively.

JSON Response Syntax

```
{
  "updateAttachmentResult": {
    "objectId" : <attachmentId>,
    "globalId" : <globalId>,
    "success" : <true | false>,
    "error" : { //only if success is false
      "code" : <code1>,
      "description" : "<description>",
    }
  }
}
```

JSON Response Example

```
{
  "updateAttachmentResult": {
{
```

```
  "updateAttachmentResult": {
    "objectId" : 58,
    "globalId" : null,
    "success" : true
  }
}
```

9.2.6 Delete Attachments Operation

The Delete Attachments operation deletes attachments associated with a feature (POST only). The Delete Attachments operation is performed on a feature service Feature resource.

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

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

9.2.6.1 Delete Attachments Reference

- **URL:** http://<featureservicefeature-url>/deleteAttachments (POST only)
- **Parent Resource:** Feature (from a feature service)

9.2.6.2 Delete Attachments Parameters

Parameter	Details
f	Description: The response format Values: json | (other formats)

attachmentIds	Description: The IDs of the attachments to be deleted
	
	Syntax:
	
	`attachmentIds=<attachmentId1>, <attachmentId2>`
	
	Example:
	
	`attachmentIds=58, 4`

9.2.6.3 Delete Attachments Example

In this sample URL, the Delete Attachments operation is performed on feature ID 818654 belonging to layer 0 of the 311Incidents feature service:

```
http://myserver/rest/services/311Incidents/FeatureServer/0/818654/
deleteAttachments
```

The values for the input parameter attachmentIds to this operation are the IDs of attachments to be deleted from a Feature resource.

JSON Response Syntax

```
{
 "deleteAttachmentResults": [
  {
   "objectId": <attachmentId1>,
   "globalId": "<globalId1>",
   "success": <true | false>,
   "error" : { //only if success is false
      "code" : <code1>,
      "description" : "<description1>",
    }
  },
  {
   "objectId": <attachmentId2>,
   "globalId": "<globalId2>",
   "success": <true | false>,
   "error" : {
      "code" : <code2>,
      "description" : "<description2>",
    }
  }
 ]
}
```

JSON Response Example

```
{
 "deleteAttachmentResults": [
  {
   "objectId": 58,
   "globalId": null,
   "success": true
  },
  {
   "objectId": 4,
   "globalId": null,
   "success": false,
   "error": {
    "code": 50,
    "description": "Attachment not found"
   }
  }
 ]
}
```

9.3 Attachment Infos (Feature Service)

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 information about the attachment such as its ID, content type, size, and name.

The Attachment Infos resource has one child resource, Attachment, which streams the content of an individual attachment.

9.3.1 Attachment Infos Reference

- **URL:** http://<featureservicefeature-url>/attachments
- **Parent Resource:** Feature (from a feature service)
- **Child Resource:** Attachment

9.3.2 Attachment Infos Parameters

Parameter	Details
f	Description: The response format Values: json | (other formats)

9.3.3 Attachment Infos Example

Example

Link to attachment infos from the 311Incidents feature service:

```
http://myserver/rest/services/311Incidents/FeatureServer/0/818654/
attachments
```

JSON Response Syntax

```
{
 "attachmentInfos": [
  {
   "id": <attachmentId1>,
   "contentType": "<contentType1>",
   "size": <size1>,
   "name": "<name1>"
  },
  {
   "id": <attachmentId2>,
   "contentType": "<contentType2>",
   "size": <size2>,
   "name": "<name2>"
  }
 ]
}
```

JSON Response Example

```
{
 "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"
  }
 ]
}
```

9.4 Attachment (Feature Service)

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.

The contents of the attachment are streamed to the client. If the attachment is not found, an HTTP status code of 404 (not found) is returned.

9.4.1 Attachment Reference

- **URL:** http://<featureserviceattachtmentinfos-url>/<attachtmentId>
- **Parent Resource:** Attachment Infos (from a feature service)

9.4.2 Attachment Example

Example

Link to an attachment from the 311Incidents feature service:

```
http://myserver/rest/services/311Incidents/FeatureServer/0/818654/
attachments/1
```

9.5 HTML Popup

The HTML Popup resource provides details about the HTML content that should appear in association with each feature in a pop-up balloon.

This resource is available when a layer resource's htmlPopupType parameter is not set to esriServerHTMLPopupTypeNone.

9.5.1 HTML Popup Reference

- **URL:** http://<feature-url>/htmlPopup

9.5.2 HTML Popup Parameters

Parameter	Details
f	Description: The response format Values: json | (other formats)

9.5.3 HTML Popup Example

Example

Return HTML pop-up content for a feature in the Watershed feature service.

```
http://myserver/rest/services/Watershed/FeatureServer/1/1/
htmlPopUp
```

JSON Response Syntax

```
{
  "htmlPopupType" : "<esriServerHTMLPopupTypeNone |
esriServerHTMLPopupTypeAsURL |
esriServerHTMLPopupTypeAsHTMLText>",
  "content": "<htmlContent>"
}
```

JSON Response Example

```
{
  "htmlPopupType" : "esriServerHTMLPopupTypeAsHTMLText",
  "content": "A <b>Sample HTML</b> pop up."
}
```

9.6 Image (Feature Service)

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 value for imageId in the image URL.

The image bytes are directly streamed to the client.

9.6.1 Image Reference

- **URL:** http://<featurelayer-url>/images/<imageId>
- **Parent Resource:** Feature Layer

9.6.2 Image Example

Example

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

```
http://myserver/rest/services/311Incidents/FeatureServer/0/images/
1DD4FC53
```

10.0 GEOMETRY OBJECTS

This section discusses the JSON geometry and spatial reference objects as returned by the GeoServices REST Specification. The GeoServices REST Specification supports four geometry types: points, polylines, polygons, and envelopes.

10.1 Spatial Reference

The spatial reference can be specified using a well-known ID or well-known text (WKT).

JSON Syntax (WKID)

```
{"wkid" : <wkid>}
```

JSON Example (WKID)

```
{"wkid" : 4326}
```

JSON Syntax (WKT)

```
{"wkt" : "<wkt>"}
```

JSON Example (WKT)

```
{"wkt" :
"GEOGCS[\"GCS_WGS_1984\",DATUM[\"D_WGS_1984\",SPHEROID[\"WGS_1984\
",6378137,298.257223563]],PRIMEM[\"Greenwich\",0],UNIT[\"Degree\",
0.0174532925199433295]]"}
```

10.2 Point

A point contains x and y fields along with a spatialReference field.

JSON Syntax

```
{
"x" : <x>, "y" : <y>, "spatialReference" : {<spatialReference>}
}
```

JSON Example

```
{
"x" : -118.15, "y" : 33.80, "spatialReference" : {"wkid" : 4326}
}
```

10.3 Polyline

A polyline contains an array of paths and a spatial reference. Each path is represented as an array of points. Each point in the path is represented as a two-element array. The 0 index is the x-coordinate and the 1 index is the y-coordinate.

JSON Syntax

```
{
"paths" : [
 [ [<x11>, <y11>], [<x12>, <y12>] ],
 [ [<x21>, <y21>], [<x22>, <y22>] ]
],
"spatialReference" : {<spatialReference>}
}
```

JSON Example

```
{
"paths" : [
 [ [-97.06138,32.837], [-97.06133,32.836], [-97.06124,32.834], [-
97.06127,32.832] ],
 [ [-97.06326,32.759], [-97.06298,32.755] ]
],
"spatialReference" : {"wkid" : 4326}
}
```

10.4 Polygon

A polygon contains an array of rings and a spatial reference. Each ring is represented as an array of points. The first point of each ring is always the same as the last point. Each point in the ring is represented as a two-element array. The 0 index is the x-coordinate and the 1 index is the y-coordinate.

JSON Example

```
{
"rings" : [
 [ [<x11>, <y11>], [<x12>, <y12>], ..., [<x11>, <y11>] ],
 [ [<x21>, <y21>], [<x22>, <y22>], ..., [<x21>, <y21>] ]
],
"spatialReference" : {<spatialReference>}
}
```

JSON Example

```
{
"rings" : [
 [ [-97.06138,32.837], [-97.06133,32.836], [-97.06124,32.834], [-
97.06127,32.832], [-97.06138,32.837] ],
 [ [-97.06326,32.759], [-97.06298,32.755], [-97.06153,32.749], [-
97.06326,32.759] ]
],
"spatialReference" : {"wkid" : 4326}
}
```

10.5 Multipoint

A multipoint contains an array of points and a spatial reference. Each point is represented as a two-element array. The 0 index is the x-coordinate and the 1 index is the y-coordinate.

JSON Syntax

```
{
"points" : [ [<x1>, <y1>], [<x2>, <y2>] ],
"spatialReference" : {<spatialReference>}
}
```

JSON Example

```
{
"points" : [ [-97.06138,32.837], [-97.06133,32.836], [-
97.06124,32.834], [-97.06127,32.832] ],
"spatialReference" : {"wkid" : 4326}
}
```

10.6 Envelope

An envelope contains the corner points of an extent and is represented by xmin, ymin, xmax, and ymax, along with a spatial reference.

JSON Syntax

```
{
"xmin" : <xmin>, "ymin" : <ymin>, "xmax" : <xmax>, "ymax" :
<ymax>,
"spatialReference" : {<spatialReference>}
}
```

JSON Example

```
{
"xmin" : -109.55, "ymin" : 25.76, "xmax" : -86.39, "ymax" :
49.94,
"spatialReference" : {"wkid" : 4326}
}
```

11.0 FEATURE OBJECT

This section discusses the JSON feature object as returned by the GeoServices REST Specification.

A feature consists of two properties, geometry and attributes, which are both optional. The feature geometry can be any of the supported geometry types. It is a JSON object as defined in Section 10.0 Geometry Objects in this document.

Example geometry:

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

The feature attributes are in a JSON object that contains a dictionary of name-value pairs. The names are the feature field names. The values are the field values, and they can be any of the standard JSON types: string, number, and Boolean.

Note that date values are encoded as numbers. The number represents the number of milliseconds since epoch (January 1, 1970) in UTC.

Example attributes:

```
"attributes" : {
   "OWNER" : "Joe Smith", //string
   "VALUE" : 94820.37, //number
   "APPROVED" : true, //boolean
   "LASTUPDATE" : 1227663551096 //date
}
```

JSON Syntax

```
{
  "geometry" : <geometry>,

  "attributes" : {
    "name1" : <value1>,
    "name2" : <value2>,
  }
}
```

JSON Example

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

  "attributes" : {
    "OWNER" : "Joe Smith",
    "VALUE" : 94820.37,
```

```
    "APPROVED" : true,
    "LASTUPDATE" : 1227663551096
  }
}
```

12.0 SYMBOL OBJECTS

This section discusses the JSON symbol and supporting objects as returned by the GeoServices REST Specification.

12.1 Color

Color is represented as a four-element array. The four elements represent values for red, green, blue, and alpha, in that order. Values range from 0 through 255.

JSON Syntax

```
[ <red>, <green>, <blue>, <alpha> ]
```

JSON Example

```
[ 67, 0, 255, 40 ]
```

12.2 Simple Marker Symbol

Simple marker symbols can be used to symbolize point geometries. The type property for simple marker symbols is esriSMS.

JSON Syntax

```
{
"type" : "esriSMS",
"style" : "< esriSMSCircle | esriSMSCross | esriSMSDiamond |
esriSMSSquare | esriSMSX >",
"color" : <color>,
"size" : <size>,
"angle" : <angle>,
"xoffset" : <xoffset>,
"yoffset" : <yoffset>,
"outline" : { //if outline has been specified
  "color" : <color>,
  "width" : <width>
}
}
```

JSON Example

```
{
"type": "esriSMS",
 "style": "esriSMSSquare",
 "color": [76,115,0,255],
 "size": 8,
 "angle": 0,
 "xoffset": 0,
 "yoffset": 0,
 "outline":
  {
  "color": [152,230,0,255],
   "width": 1
  }
}
```

12.3 Simple Line Symbol

Simple line symbols can be used to symbolize polyline geometries or outlines for polygon fills. The type property for simple line symbols is esriSLS.

JSON Syntax

```
{
"type" : "esriSLS",
"style" : "< esriSLSDash | esriSLSDashDotDot | esriSLSDot |
esriSLSNull | esriSLSSolid >",
"color" : <color>,
"width" : <width>
}
```

JSON Example

```
{
"type": "esriSLS",
"style": "esriSLSDot",
"color": [115,76,0,255],
"width": 1
}
```

12.4 Simple Fill Symbol

Simple line symbols can be used to symbolize polygon geometries. The type property for simple line symbols is esriSFS.

JSON Syntax

```
{
"type" : "esriSFS",
"style" : "< esriSFSBackwardDiagonal | esriSFSCross |
esriSFSDiagonalCross | esriSFSForwardDiagonal | esriSFSHorizontal
| esriSFSNull | esriSFSSolid | esriSFSVertical >",
"color" : <color>,
"outline" : <simpleLineSymbol> //if outline has been specified
}
```

JSON Example

```
{
  "type": "esriSFS",
  "style": "esriSFSSolid",
  "color": [115,76,0,255],
    "outline": {
     "type": "esriSLS",
     "style": "esriSLSSolid",
     "color": [110,110,110,255],
     "width": 1
    }
}
```

12.5 Picture Marker Symbol

Picture marker symbols can be used to symbolize point geometries. The type property for picture marker symbols is esriPMS.

These symbols include the base64-encoded image data, as well as a URL that could be used to retrieve the image from the server. Note that this is a relative URL. It can become nonrelative by accessing the map layer Image resource or the feature layer Image resource.

JSON Syntax

```
{
"type" : "esriPMS",
"url" : "<imageUrl>", //relative URL
"imageData" : "<base64EncodedImageData>",
"contentType" : "<imageContentType>",
"color" : <color>,
"width" : <width>,
"height" : <height>,
"angle" : <angle>,
"xoffset" : <xoffset>,
"yoffset" : <yoffset>
}
```

JSON Example

```
{
   "type" : "esriPMS",
   "url" : "471E7E31",
   "imageData" :
"iVBORw0KGgoAAAANSUhEUgAAABoAAAAaCAYAAACpSkzOAAAAAXNSR0IB2cksfwAAA
AlwSFlzAAAOxAAADsQBlSsOGwAAAMNJREFUSIntlcENwyAMRZ+lSMyQFcI8rJA50jW
yQuahKzCDT+6h0EuL1BA1iip8Qg/Ex99fYuCkGv5bKK0EcB40YgSE7bnTxsa58LeOn
Md0QhwGXkxB3L0w0IDxPaMqpBFxjLMuaSVmRjurWIcRDHxaiWZuEbRcEhpZpSNhE9O
81GiMN5E0ZRt2M0iVjshek8UkTQfZy8JqGHYP/rJhODD4T6wehtbB9zD0MPQwlOpha
AxD/uPLK7Z8MB5gFet+WKcJPQDx29XkRhqr/AAAAABJRU5ErkJggg==",
   "contentType" : "image/png",
   "color" : null,
   "width" : 19.5,
   "height" : 19.5,
   "angle" : 0,
   "xoffset" : 0,
   "yoffset" : 0
}
```

12.6 Picture Fill Symbol

Picture fill symbols can be used to symbolize polygon geometries. The type property for picture fill symbols is esriPFS.

These symbols include the base64-encoded image data, as well as a URL that could be used to retrieve the image from the server. Note that this is a relative URL. It can become nonrelative by accessing the map layer Image resource or the feature layer Image resource.

JSON Syntax

```
{
"type" : "esriPFS",
"url" : "<imageUrl>", //relative URL
"imageData" : "<base64EncodedImageData>",
"contentType" : "<imageContentType>",
"color" : <color>,
"outline" : <simpleLineSymbol>, //if outline has been specified
"width" : <width>,
"height" : <height>,
"angle" : <angle>,
"xoffset" : <xoffset>,
"yoffset" : <yoffset>,
"xscale": <xscale>,
"yscale": <yscale>
}
```

JSON Example

```
{
   "type" : "esriPFS",
   "url" : "866880A0",
   "imageData" :
"iVBORw0KGgoAAAANSUhEUgAAAFQAAABUCAYAAAAcaxDBAAAAAXNSR0IB2cksfwAAA
AlwSFlzAAAOxAAADsQBlSsOGwAAAM9JREFUeJzt0EEJADAMwMA96l/zTBwUSk5ByLx
Qsx1wTUOxhmINxRqKNRRrKNZQrKFYQ7GGYg3FGoo1FGso1lCsoVhDsYZiDcUaijUUa
yjWUKyhWEOxhmINxRqKNRRrKNZQrKFYQ7GGYg3FGoo1FGso1lCsoVhDsYZiDcUaijU
UayjWUKyhWEOxhmINxRqKNRRrKNZQrKFYQ7GGYg3FGoo1FGso1lCsoVhDsYZiDcUai
jUUayjWUKyhWEOxhmINxRqKNRRrKNZQrKFYQ7GGYh/hIwFRFpnZNAAAAABJRU5ErkJ
ggg==",
   "contentType" : "image/png",
   "color" : null,
   "outline" :
   {
      "type" : "esriSLS",
      "style" : "esriSLSSolid",
      "color" : [110,110,110,255],
      "width" : 1
   },
   "width" : 63,
   "height" : 63,
   "angle" : 0,
   "xoffset" : 0,
   "yoffset" : 0,
   "xscale" : 1,
   "yscale" : 1
}
```

12.7 Text Symbol

Text symbols are used to add text to a feature (labeling). The type property for text symbols is esriTS.

JSON Syntax

```
{
  "type" : "esriTS",
  "color" : <color>,
  "backgroundColor" : <color>,
  "borderLineColor" : <color>,
  "verticalAlignment" : "<baseline | top | middle | bottom>",
  "horizontalAlignment" : "<left | right | center | justify>",
  "rightToLeft" : <true | false>,
  "angle" : <angle>,
  "xoffset" : <xoffset>,
  "yoffset" : <yoffset>,
  "kerning" : <true | false>,
  "font" : {
    "family" : "<fontFamily>",
    "size" : <fontSize>,
    "style" : "<italic | normal | oblique>",
    "weight" : "<bold | bolder | lighter | normal>",
    "decoration" : "<line-through | underline | none>"
  }
}
```

JSON Example

```
{
     "type": "esriTS",
     "color": [78,78,78,255],
     "backgroundColor": null,
     "borderLineColor": null,
     "verticalAlignment": "bottom",
     "horizontalAlignment": "left",
     "rightToLeft": false,
     "angle": 0,
     "xoffset": 0,
     "yoffset": 0,
     "font": {
      "family": "Arial",
      "size": 12,
      "style": "normal",
      "weight": "bold",
      "decoration": "none"
    }
}
```

13.0 DOMAIN OBJECTS

This section discusses the JSON domain objects as returned by the GeoServices REST Specification. Domains specify the set of valid values for a field.

13.1 Range Domain

Range domain specifies a range of valid values for a field. The type property for range domains is range.

JSON Syntax

```
{
"type" : "range",
"name" : "<domainName>",
"range" : [ <minValue>, <maxValue> ]
}
```

JSON Example

```
{
  "type": "range",
  "name": "Measured Length",
  "range": [1,10000]
}
```

13.2 Coded Value Domain

Coded value domain specifies an explicit set of valid values for a field. Each valid value is assigned a unique name. The type property for coded value domains is codedValue.

JSON Syntax

```
{
"type" : "codedValue",
"name" : "<domainName>",
"codedValues" :
[
  { "name" : "<codeName1>", "code" : <code1> },
  { "name" : "<codeName2>", "code" : <code2> }
]
}
```

JSON Example

```
{
 "type": "codedValue",
 "name": "Material",
 "codedValues":
 [
     {
      "name": "Aluminum",
      "code": "AL"
     },
     {
      "name": "Copper",
      "code": "CU"
     },
     {
      "name": "Steel",
      "code": "STEL"
     },
     {
      "name": "Not Applicable",
      "code": "NA"
     }
  ]
}
```

13.3 Inherited Domain

Inherited domains apply to domains on subtypes. An inherited domain implies that the domain for a field at the subtype level is the same as the domain for the field at the layer level.

JSON Syntax

```
{
"type" : "inherited"
}
```

JSON Example

```
{
"type" : "inherited"
}
```

14.0 LABEL OBJECTS

This section discusses the JSON label objects as returned by the GeoServices REST Specification.

14.1 Label Placement

Label placement is represented as a literal string. It specifies the placement of the label with respect to that of its feature. Below is a list of label placement values categorized by the geometry type of the feature.

- Label placement values for point features
  - esriServerPointLabelPlacementAboveCenter
  - esriServerPointLabelPlacementAboveLeft
  - esriServerPointLabelPlacementAboveRight
  - esriServerPointLabelPlacementBelowCenter
  - esriServerPointLabelPlacementBelowLeft
  - esriServerPointLabelPlacementBelowRight
  - esriServerPointLabelPlacementCenterCenter
  - esriServerPointLabelPlacementCenterLeft
  - esriServerPointLabelPlacementCenterRight
- Label placement values for line features
  - esriServerLinePlacementAboveAfter
  - esriServerLinePlacementAboveAlong
  - esriServerLinePlacementAboveBefore
  - esriServerLinePlacementAboveEnd
  - esriServerLinePlacementAboveStart
  - esriServerLinePlacementBelowAfter
  - esriServerLinePlacementBelowAlong
  - esriServerLinePlacementBelowBefore
  - esriServerLinePlacementBelowEnd
  - esriServerLinePlacementBelowStart
  - esriServerLinePlacementCenterAfter
  - esriServerLinePlacementCenterAlong

- esriServerLinePlacementCenterBefore
- esriServerLinePlacementCenterEnd
- esriServerLinePlacementCenterStart

■ Label placement values for polygon features

- esriServerPolygonPlacementAlwaysHorizontal

14.2 Label Class

A label class specifies the label definition for a given scale range.

JSON Syntax

```
{
  "labelPlacement" : "<labelPlacement>",
  "labelExpression" : "<labelExpression>",
  "useCodedValues": "<true | false>"
  "symbol" : "<textSymbol>",
  "minScale" : <minScale>,
  "maxScale" : <maxScale>
}
```

JSON Example

```
{
    "labelPlacement": "esriServerPointLabelPlacementAboveRight",
    "labelExpression": "[NAME]",
    "useCodedValues": false,
    "symbol": {
     "type": "esriTS",
     "color": [38,115,0,255],
     "backgroundColor": null,
     "borderLineColor": null,
     "verticalAlignment": "bottom",
     "horizontalAlignment": "left",
     "rightToLeft": false,
     "angle": 0,
     "xoffset": 0,
     "yoffset": 0,
     "font": {
      "family": "Arial",
      "size": 11,
      "style": "normal",
      "weight": "bold",
      "decoration": "none"
     }
    },
    "minScale": 0,
    "maxScale": 0
  }
```

14.3 Labeling Info

The labeling info object specifies the label definition for a layer. It is expressed as an array of label classes.

JSON Syntax

```
[ <labelClass1>, <labelClass2> ]
```

JSON Example

```
  [
   {
    "labelPlacement":
"esriServerPolygonPlacementAlwaysHorizontal",
    "labelExpression": "[TAG]",
    "useCodedValues": false,
    "symbol": {
     "type": "esriTS",
     "color": [78,78,78,255],
     "backgroundColor": null,
     "borderLineColor": null,
     "verticalAlignment": "bottom",
     "horizontalAlignment": "left",
     "rightToLeft": false,
     "angle": 0,
     "xoffset": 0,
     "yoffset": 0,
     "font": {
      "family": "Arial",
      "size": 12,
      "style": "normal",
      "weight": "bold",
      "decoration": "none"
     }
    },
    "minScale": 1999,
    "maxScale": 0
   },
   {
    "labelPlacement":
"esriServerPolygonPlacementAlwaysHorizontal",
    "labelExpression": "[TAG]",
    "useCodedValues": true,
    "symbol": {
     "type": "esriTS",
     "color": [78,78,78,255],
     "backgroundColor": null,
     "borderLineColor": null,
     "verticalAlignment": "bottom",
     "horizontalAlignment": "left",
     "rightToLeft": false,
     "angle": 0,
     "xoffset": 0,
     "yoffset": 0,
     "font": {
      "family": "Arial",
      "size": 12,
      "style": "normal",
      "weight": "bold",
```

```
      "decoration": "none"
     }
    },
    "minScale": 0,
    "maxScale": 7100
   }
  ]
```

15.0 RENDERER OBJECTS

This section discusses the JSON renderer objects as returned by the GeoServices REST Specification.

15.1 Simple Renderer

A simple renderer uses one symbol only. The type property for simple renderers is simple.

JSON Syntax

```
{
  "type" : "simple",
  "symbol" :  <symbol>,
  "label" : "<label>",
  "description" : "<description>"
}
```

JSON Example

```
{
   "type": "simple",
   "symbol":
   {
    "type": "esriSMS",
    "style": "esriSMSCircle",
    "color": [255,0,0,255],
    "size": 5,
    "angle": 0,
    "xoffset": 0,
    "yoffset": 0,
    "outline":
   {
     "color": [0,0,0,255],
     "width": 1
    }
   },
   "label": "",
   "description": ""
}
```

15.2 Unique Value Renderer

A unique value renderer symbolizes groups of features that have matching field values. The type property for unique value renderers is uniqueValue.

JSON Syntax

```
{
  "type" : "uniqueValue",
  "field1" : "<field1>",
  "field2" : "<field2>",
  "field3" : "<field3>",
```

```
  "fieldDelimiter" : "<fieldDelimiter>",
  "defaultSymbol" :  <symbol>,
  "defaultLabel" : "<defaultLabel>",
  "uniqueValueInfos" : [
    {
      "value" : "<value1>",
      "label" : "<label1>",
      "description" : "<description1>",
      "symbol" :  <symbol1>
    },
    {
      "value" : "<value2>",
      "label" : "<label2>",
      "description" : "<description2>",
      "symbol" :  <symbol2>
    }
  ]
}
```

JSON Example

```
{
      "type" : "uniqueValue",
      "field1" : "SubtypeCD",
      "field2" : null,
      "field3" : null,
      "fieldDelimiter" : ", ",
      "defaultSymbol" :
      {
        "type" : "esriSLS",
        "style" : "esriSLSSolid",

        "color" : [130,130,130,255],
        "width" : 1
      },
      "defaultLabel" : "\u003Other values\u003e",
      "uniqueValueInfos" : [
        {
          "value" : "1",
          "label" : "Duct Bank",
          "description" : "Duct Bank description",
          "symbol" :
          {
            "type" : "esriSLS",
            "style" : "esriSLSDash",

            "color" : [76,0,163,255],
            "width" : 1
          }
        },
        {
          "value" : "2",
          "label" : "Trench",
          "description" : "Trench description",
          "symbol" :
```

```
          {
            "type" : "esriSLS",
            "style" : "esriSLSDot",

            "color" : [115,76,0,255],
            "width" : 1
          }
        }
      ]
}
```

15.3 Class Breaks Renderer

A class breaks renderer symbolizes each feature based on the value of some numeric field. The type property for class breaks renderers is classBreaks.

JSON Syntax

```
{
  "type" : "classBreaks",
  "field" : "<field>",
  "minValue" : <minValue>,
  "classBreakInfos" : [
    {
      "classMaxValue" : <classMaxValue1>,
      "label" : "<label1>",
      "description" : "<description1>",
      "symbol" :  <symbol1>
    },
    {
      "classMaxValue" : <classMaxValue2>,
      "label" : "<label2>",
      "description" : "<description2>",
      "symbol" :  <symbol2>
    }
  ]
}
```

JSON Example

```
{
      "type" : "classBreaks",
      "field" : "Shape.area",
      "minValue" : 10.3906320193541,
      "classBreakInfos" : [
        {
          "classMaxValue" : 1000,
          "label" : "10.0 - 1000.000000",
          "description" : "10 to 1000",
          "symbol" :
          {
            "type" : "esriSFS",
            "style" : "esriSFSSolid",

            "color" : [236,252,204,255],
            "outline" :
```

```
            {
              "type" : "esriSLS",
              "style" : "esriSLSSolid",

              "color" : [110,110,110,255],
              "width" : 0.4
            }
          }
        },
        {
          "classMaxValue" : 5000,
          "label" : "1000.000001 - 5000.000000",
          "description" : "1000 to 5000",
          "symbol" :
          {
            "type" : "esriSFS",
            "style" : "esriSFSSolid",

            "color" : [218,240,158,255],
            "outline" :
            {
              "type" : "esriSLS",
              "style" : "esriSLSSolid",

              "color" : [110,110,110,255],
              "width" : 0.4
            }
          }
        }
      ]
}
```

16.0 SPATIAL REFERENCES

Features on a map refer to the actual locations of the objects they represent in the real world. The positions of objects on the earth's spherical surface are measured in degrees of latitude and longitude, also known as geographic coordinates. Though latitude and longitude can locate exact positions on the surface of the earth, they are not uniform units of measure; only along the equator does the distance represented by one degree of longitude approximate the distance represented by one degree of latitude.

To overcome measurement difficulties, data is often transformed from the three-dimensional geographic coordinate system to the two-dimensional planar surface in a projected coordinate system. Projected coordinate systems describe the distance from an origin (0,0) along two separate axes: a horizontal x-axis representing east–west and a vertical y-axis representing north–south.

Because the earth is round and maps are flat, getting information from the curved surface to a flat one involves a mathematical formula called a map projection. A map projection transforms latitude and longitude to x,y coordinates in a projected coordinate system.

The term *coordinate system,* which includes both geographic and projected coordinate systems, is used to describe the information about the projection, as well as other specifics such as datum, units, and meridians.

Coordinate systems are defined by WKIDs or well-known text (WKT) strings. See Section 10.1 Spatial Reference for an example of how WKIDs and WKT strings can be used.

For a list of WKIDs and WKT strings for projected coordinate systems, see http://links.esri.com/projectedcoordinatesystems.

For a list of WKIDs and WKT strings for geographic coordinate systems, see http://links.esri.com/geographiccoordinatesystems.

Appendix A: Errata

Section 4.0.5.2 Identify Parameters is missing two parameters: tolerance and mapExtent. These should be used as follows:

Parameter	Details
tolerance	**Required** Description: The distance in screen pixels from the specified geometry within which the Identify operation should be performed. The value for the tolerance is an integer. Example: `tolerance=2`

mapExtent	**Required**
	
	Description: The extent or bounding box of the map currently being viewed. Unless the sr parameter has been specified, mapExtent is assumed to be in the spatial reference of the map.
	
	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.
	
	Syntax:
	
	`<xmin>, <ymin>, <xmax>, <ymax>`
	
	Example:
	
	`mapExtent=-104, 35.6, -94.32, 41`

The example code in Section 4.0.5.3 is not affected by the above omission. The example contains the necessary parameters.

About Esri

Since 1969, Esri has been helping organizations map and model our world. Esri's GIS software tools and methodologies enable these organizations to effectively analyze and manage their geographic information and make better decisions. They are supported by our experienced and knowledgeable staff and extensive network of business partners and international distributors.

A full-service GIS company, Esri supports the implementation of GIS technology on desktops, servers, online services, and mobile devices. These GIS solutions are flexible, customizable, and easy to use.

Our Focus

Esri software is used by hundreds of thousands of organizations that apply GIS to solve problems and make our world a better place to live. We pay close attention to our users to ensure they have the best tools possible to accomplish their missions. A comprehensive suite of training options offered worldwide helps our users fully leverage their GIS applications.

Esri is a socially conscious business, actively supporting organizations involved in education, conservation, sustainable development, and humanitarian affairs.

Contact Esri

1 800 GIS XPRT (1 800 447 9778)
T 909 793 2853
F 909 793 5953
info@esri.com
esri.com

Offices worldwide
esri.com/locations

esri®
380 New York Street
Redlands, California 92373-8100 USA

Printed in USA