

INTRODUCTION

ArcGIS for Server web services are great for sharing GIS data and functionality, but present some interesting challenges for GIS developers, administrators, and other GIS professionals. These common challenges include:

- Finding out what services are available on a server.
- Discovering the properties and capabilities of a service.
- Testing whether a service is functioning as expected.
- Troubleshooting problems that occur when working with a service.

The ArcGIS for Server REST API can help you meet those challenges.

- [LEARNING OBJECTIVES](#)
- [KEY TERMS](#)
- [DATA](#)

After completing this course, you will be able to:

- Describe the benefits of RESTful client-server communications.
- Describe the methods and formats supported by the REST API.
- Use the REST API to discover available GIS services.
- Use the REST API to communicate with GIS servers.
- Use the REST API to help solve common client-server communication issues.

Introducing the REST API

GIS administrators are publishing more and more GIS services to the web. GIS developers and desktop GIS users want to consume those services in their applications and maps. In order to do this, the GIS services must be findable and understandable, and must behave in a consistent way.

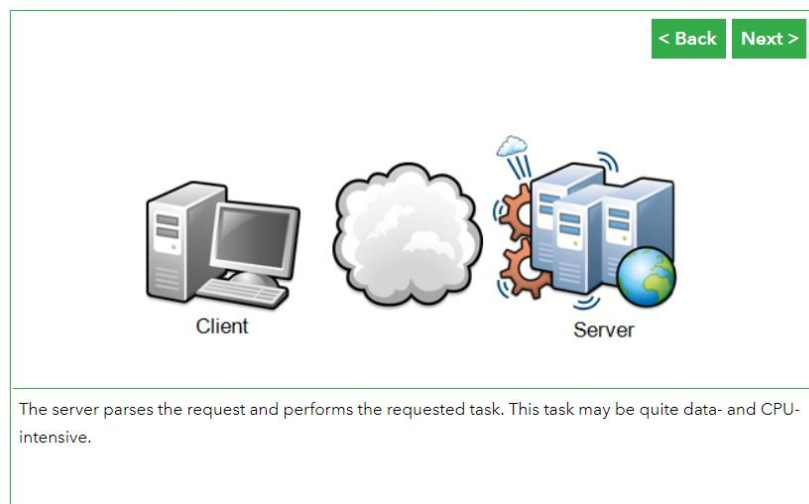
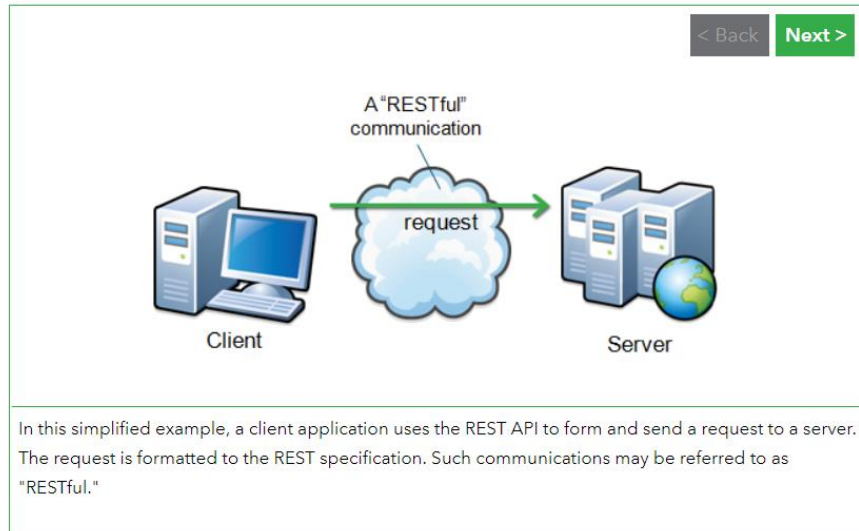
The ArcGIS for Server REST API provides a standard, foundational solution for finding, understanding, and working with GIS services. Whether you're a GIS server administrator, a GIS developer, or a desktop GIS user, you will benefit from knowing about the REST API. Watch this short introductory video to learn more. [Transcript below]

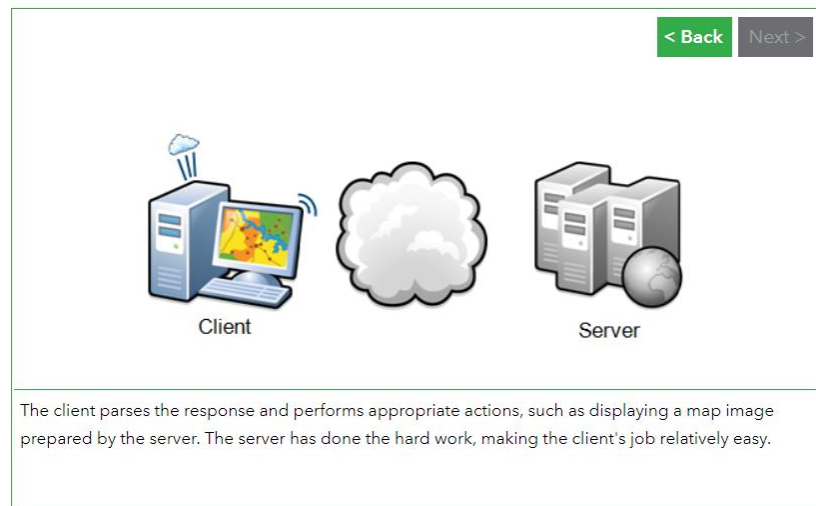
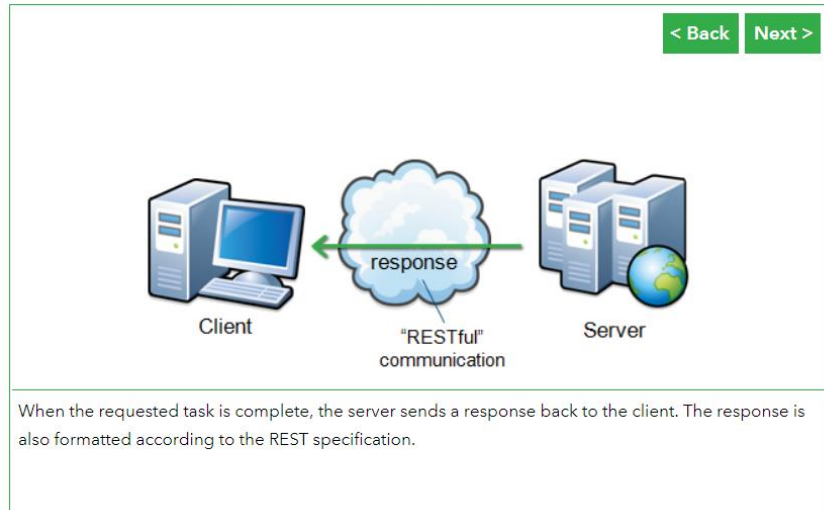
Time	Caption
0:01	As part of the larger ArcGIS system. ArcGIS for
0:05	Server publishes GIS data and functionality as
0:09	services. These GIS services can then be
0:13	consumed by Esri software such as ArcGIS
0:16	Online map viewer. as well as by custom GIS
0:19	applications built with Esri's Runtime and Web
0:22	developer toolkits. These solutions communicate
0:26	with GIS services using a specification known as
0:30	the ArcGIS for Server REST API. In this course.
0:34	you will learn about the REST API and how it can
0:38	help you discover available GIS services and their
0:41	capabilities. communicate with those GIS
0:45	services for testing purposes. and troubleshoot
0:49	and solve some common service communication
0:52	issues. The REST API skills and knowledge you'll
0:55	gain in this course will help you be successful
0:58	when using ArcGIS for Server services as part of
1:02	your GIS solutions. Let's get started.

What is REST?

Before you try to understand the ArcGIS for Server REST API, it will help to look at the architecture on which it is based.

REpresentational State Transfer, or [REST](#), is a way for software to work when distributed across a network. When software is being run across a network, a client sends a request to a server, the server processes the request, and then the server sends a response back to the client.





REST sets guidelines for communications to ensure that clients and servers can understand each other to get work done. A RESTful communication:

- Identifies a specific resource, such as a particular service running on a particular server. Specific resources are identified using URLs that end with the name of the resource being targeted.
- Usually includes well-formatted data that the targeted resource uses to perform a task.

A RESTful application programming interface (API) is one that's based on HTTP and the REST guidelines. By using the REST architecture, distributed applications gain the following benefits.

- **Consistency:** Clients and servers know what to expect when working in a RESTful architecture because resource names and request/response syntax follow a consistent structure.

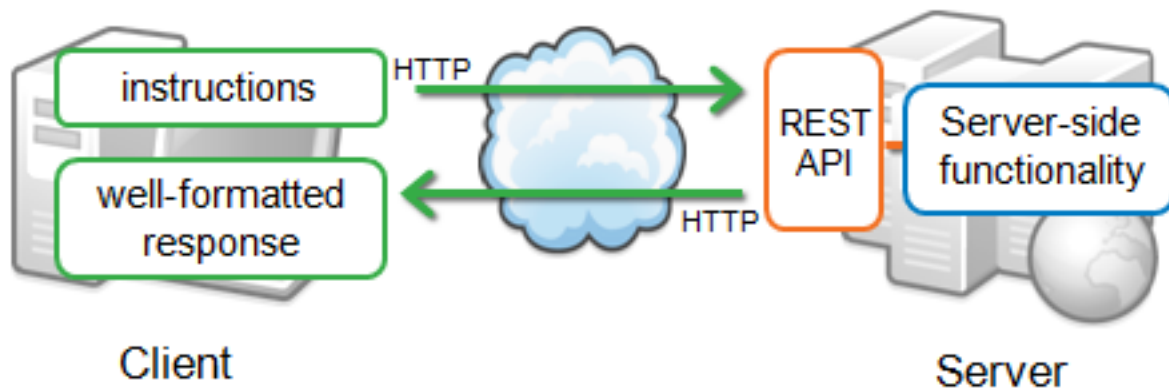
- **Readability:** REST uses formats that are easy for both people and search engines to read.
- **Efficiency:** RESTful communications are compact and take advantage of cached content to reduce bandwidth requirements and wait times.

What is the REST API?

The ArcGIS for Server REST API is an *interface* that simplifies interactions between GIS applications (clients) and ArcGIS for Server services.

The REST API:

- Allows the server to interpret instructions received from the client via HTTP.
- Enables the server to return well-formatted responses to the client via HTTP.

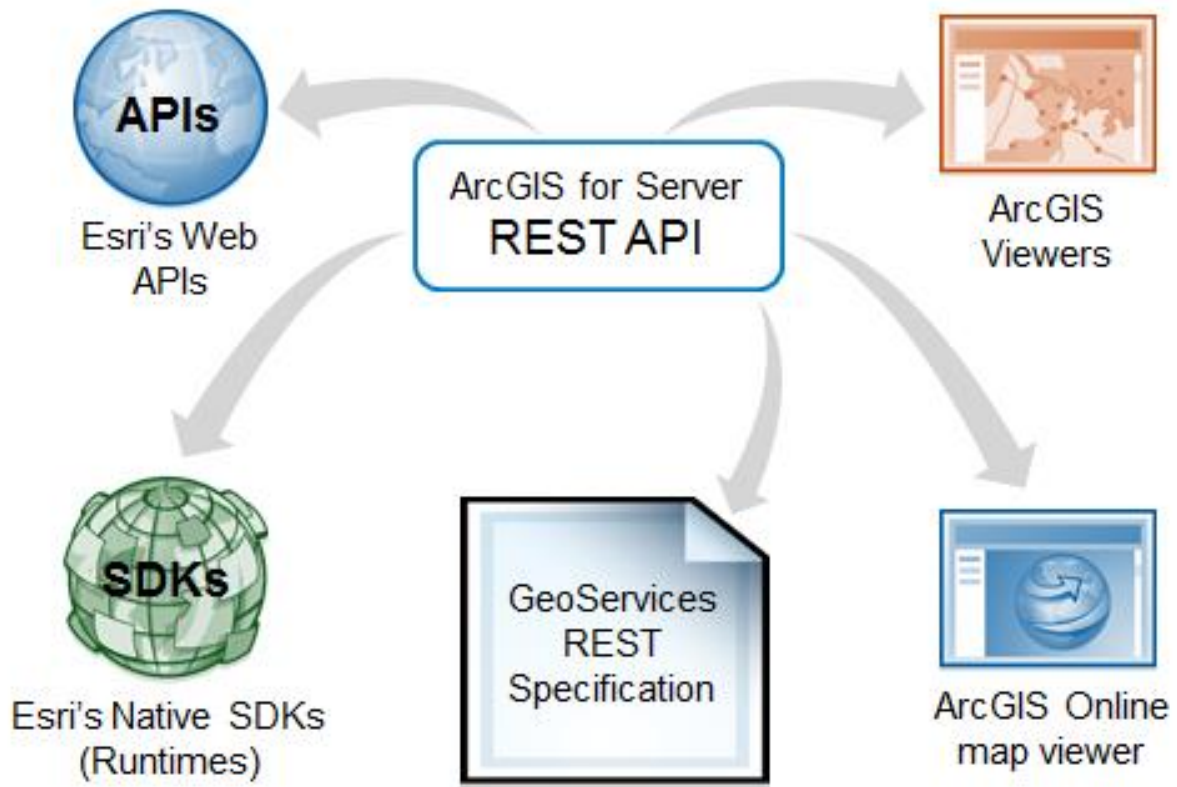


Using the REST API, a GIS application (client) interacts with a GIS server via HTTP requests and responses.

The REST API provides the foundation for client-server communications in many Esri products. Esri applications communicate with Esri's GIS services through the REST API. The REST API is also the basis for the GeoServices REST Specification, which describes a standard that developers can adopt when architecting their own custom GIS servers. To learn more about the many ways the REST API is used, you can go

to <http://developers.arcgis.com/en/documentation> help.

to access the ArcGIS REST API




The REST API is used by a growing number of Esri APIs, SDKs, and applications.

Understanding the REST API and how you can use it will help you be successful when working with these products.

What can I do with the REST API?

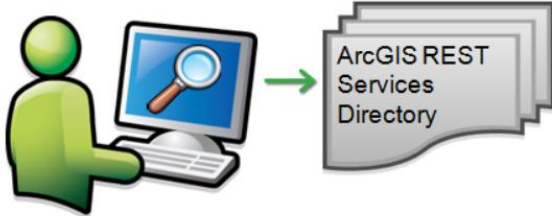
Whether you're a GIS application developer, a GIS server administrator, or a GIS professional, the REST API can help you when working with web services.

[< Back](#) [Next >](#)



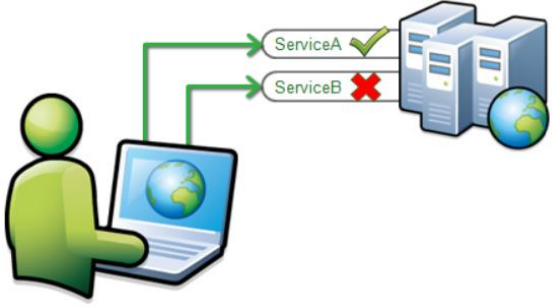
The REST API is more than just an interface that works behind the scenes. It's also a useful tool for anyone working with web services.

[< Back](#) [Next >](#)



The REST API can be used to display a server's Services Directory, which is a browsable, searchable, interactive listing of the RESTful GIS services available on the server. You can use the Services Directory to discover a service's properties and capabilities.

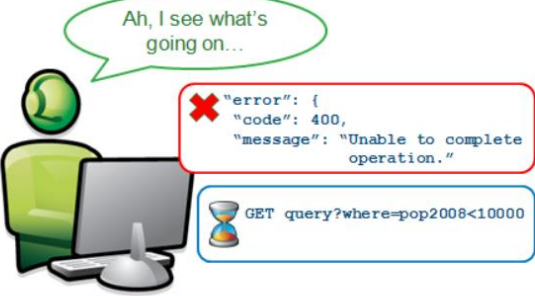
[< Back](#) [Next >](#)



The diagram illustrates a user at a laptop sending two requests to a server. The first request, labeled 'ServiceA', is marked with a green checkmark, indicating a successful operation. The second request, labeled 'ServiceB', is marked with a red 'X', indicating a failure. The server is represented by a stack of blue server racks and a globe.

The REST API allows you to verify that new services are operational, by accessing and testing them in a browser. You can also verify that data updates have taken effect by viewing the service in one of the suggested viewers.

[< Back](#) [Next >](#)



A user is shown at a computer. A speech bubble above them says, "Ah, I see what's going on...". A red-bordered box contains an error message:

```
"error": {  
  "code": 400,  
  "message": "Unable to complete operation."  
}
```

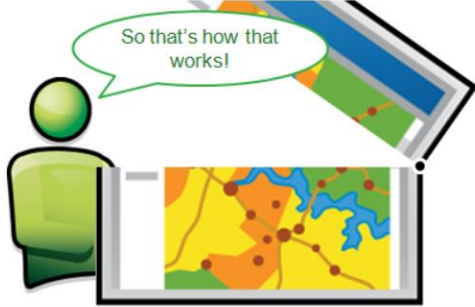
 Below this, a blue-bordered box contains a query:

```
GET query?where=pop2008<10000
```

 An hourglass icon is next to the query.

The REST API can help you reproduce, evaluate, and test client-server communications issues, including errors that may occur client-side or server-side. The REST API can also help you evaluate and test performance on a request-by-request basis.

[< Back](#) [Next >](#)



A user is shown at a computer. A speech bubble above them says, "So that's how that works!". The computer screen displays a map with various colored regions and a network of lines.

By gaining an understanding of the REST API, you can see how all Esri RESTful services and applications work under the hood. You can use that insight to use GIS services more effectively to successfully complete more GIS projects.

In this course, you'll use the REST API to help you with a variety of common tasks.

Discovering available services

In the next exercise, you will use the REST API to access the [Services Directory](#) for one of the ArcGIS Online public servers. To access the Services Directory, send a request to the REST API in the form of the following URL:

<http://services.arcgis.com/arcgis/rest/services>

This URL is an example of a [REST endpoint](#). When a valid REST endpoint URL is passed to the REST API, the response is an HTML-formatted view of the Services Directory page for that endpoint.

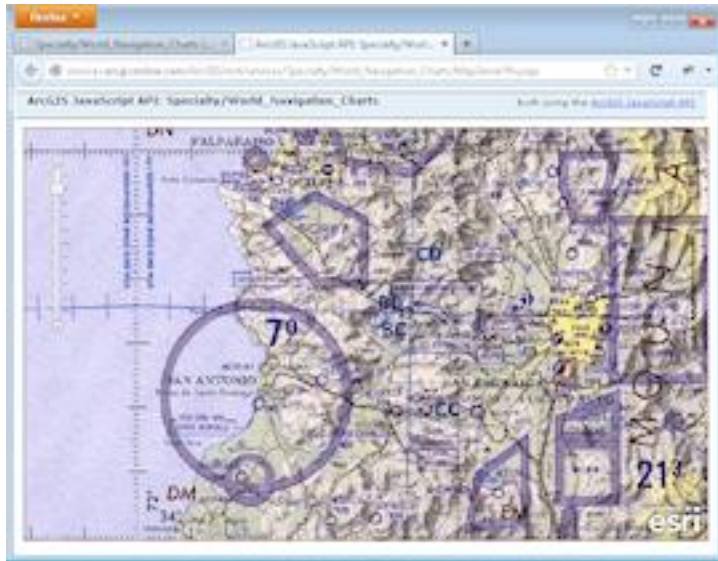
Note: Some services in a Services Directory may have been hidden by the ArcGIS Server's administrator. You can still use hidden services if you know their endpoint URL.



Use the REST API to display the Services Directory for a GIS server, such as ArcGIS Online.

The preceding REST endpoint URL is often called the *services endpoint* because the URL ends with the services (root) folder. It is also referred to as the *well-known endpoint* because it is a stable URL that is unlikely to change. This course uses the term *well-known endpoint* for clarity.

After you have accessed the Services Directory, you can browse available services by following the links provided. Each service has its own endpoint. A service endpoint returns a page describing the service's properties and capabilities. The service endpoint also provides links you can use to view the service's data. You can use this feature to verify changes after a data update.



You can view services to help determine whether a service has the information needed for your project.

You can also search for RESTful GIS services using your preferred search engine. HTTP allows search engines to discover, crawl, and index the information in the Services Directory.

Note: The Services Directory will be disabled for ArcGIS Online services that require a subscription. To learn more, visit the hosting server's home page URL; for example, the ArcGIS Online Geocoding Service at <http://geocode.arcgis.com>

Searching techniques

The following video demonstrates the browsing and searching techniques you'll use to find and learn about GIS services in the first exercise. [Transcript below]

Time	Caption
0:01	Suppose you've received a specification for a GIS
0:04	mapping application from a customer. The
0:07	application needs a basemap showing
0:09	topographic features. and some editable
0:12	Colorado wildfire data that can be identified and
0:16	queried. The REST API can help you find GIS
0:20	services that meet your customer's needs. One
0:24	technique is to use the REST API's Services
0:27	Directory to browse available services. To do this.
0:31	enter into a browser the URL for a REST services
0:35	endpoint. such as the one shown here for ArcGIS
0:39	Online. The REST API returns the Services
0:44	Directory showing the services available at that
0:46	endpoint. You can browse the services to see
0:50	descriptive information and properties. You also
0:53	have options for viewing the service's data to help
0:57	you determine whether it meets your needs.
0:61	Another technique for finding services is to use a
1:03	search engine such as Google. You can use
1:07	advanced search techniques to target specific
1:10	ArcGIS for Server instances and to limit results to
1:14	only certain types of services. The search returns
1:18	REST service endpoints that you can then visit
1:21	and browse to learn more about each service.
1:25	Using these browsing and searching techniques.
1:28	you can use the REST API's Services Directory
1:31	to look for available GIS services that best meet

1:35 the requirements for your projects. Also, don't
1:38 forget that ArcGIS.com is a great source for
1:43 finding online maps, services, GIS tools and
1:46 more.

See Exercise1.pdf: Find data for a prototype map

Working with GIS services

The REST Services Directory is more than just a static document describing available services; it is also a tool that developers, administrators, and GIS professionals can use to interact with services. The Services Directory lets you work with services at the most basic level of the REST API, without other APIs involved. This makes the Services Directory a useful tool for testing GIS services, because you can build specific test requests and see whether the service returns the expected response.

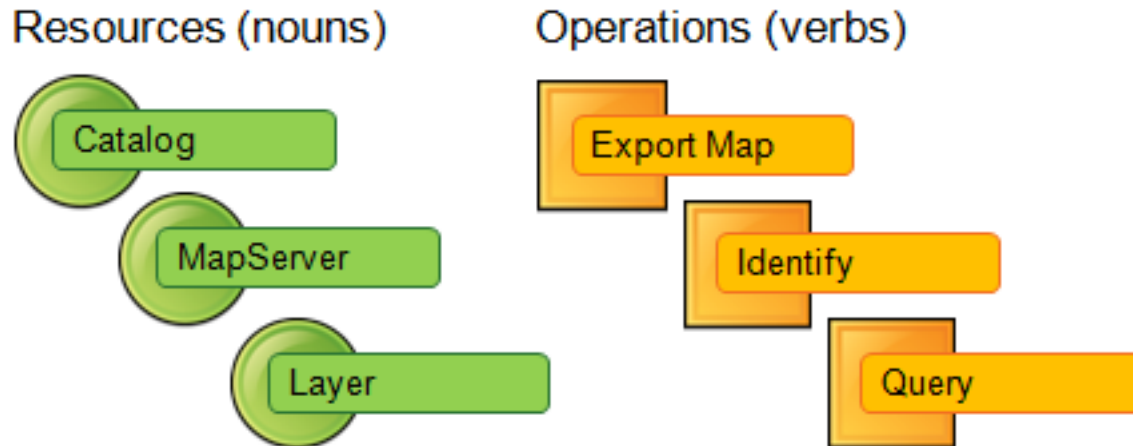
Watch the following video for an overview of what you will be learning about in this topic. [Transcript below]

Time	Caption
0:01	You've seen how to use the REST API to display
0:04	the Services Directory. but what about actually
0:07	working with services? The REST API makes it
0:10	easy to perform comprehensive testing of ArcGIS
0:14	for Server services. In this topic. you'll learn how
0:18	to use the REST API to perform service
0:21	operations. such as exporting a map or
0:24	performing a query. You'll use the REST API
0:28	Reference to discover the correct syntax for
0:31	specifying operation parameters. You'll see how
0:35	operational requests are structured. and how you
0:37	can manually adjust the request URI to return
0:41	responses in different formats. Finally. you'll use
0:45	the REST API to verify that a service is working
0:49	as expected. by sending specific operational
0:52	requests and checking for specific responses.
0:56	Let's get started.

GIS resources and operations

Working with GIS services involves communicating with them. RESTful GIS services use URLs to identify specific server resources, but how can you make those resources do specific tasks?

The REST API uses operations to identify tasks that a resource can do. If resources are the "nouns" of the Services Directory, operations are the "verbs."



Examples of REST API resources and operations.

A RESTful URL points to a resource, such as a map service. For example:

http://services.arcgisonline.com/arcgis/rest/services/World_Topo_Map/MapServer

It is helpful to refer to RESTful URLs (such as the preceding example) in a generic way, as follows:

`http://<service-url>`

To request that a service perform some task, you can add an operation to the URL. Operations appear at the end of the URL, followed by any parameters. For example:

`http://<service-url>/<operation>?<parameter-list>`

Throughout this course, the following symbology will be used to help differentiate the resource and operation portions of RESTful URLs:



Generic representation of a RESTful URL that includes an operation with parameters.

Note: If no operation is explicitly given, a default operation may be implied. For the REST API, the default operation is to return the Services Directory page (HTML) for the targeted resource.

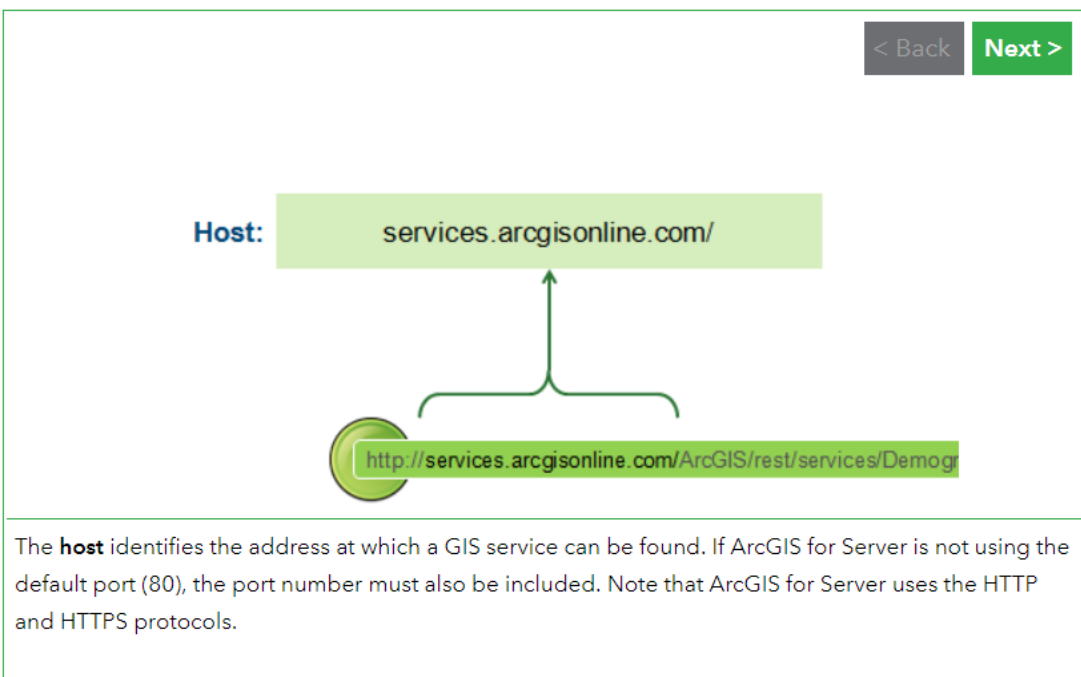
Anatomy of a RESTful URL

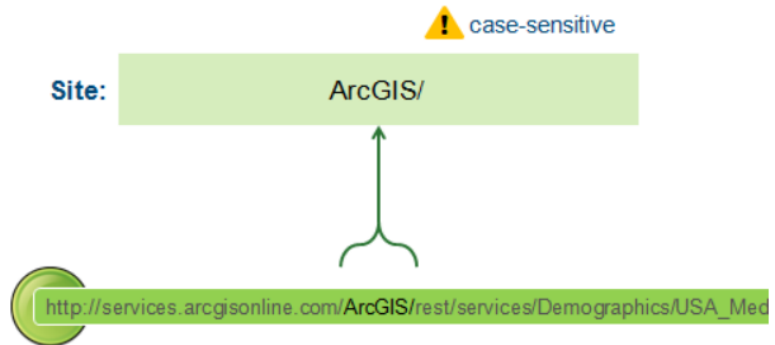
It can be helpful to think about RESTful URLs in terms of resources and operations, but these URLs actually consist of many pieces of information, and each piece of the URL plays an important role.

Suppose you have the following URL:

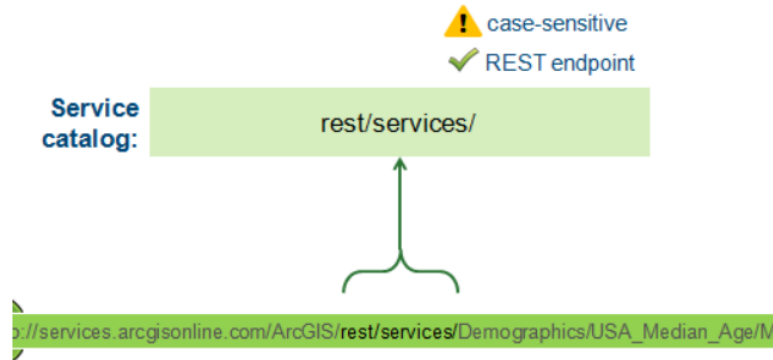
```
http://services.arcgisonline.com/ArcGIS/rest/services/Demographics/USA_Median_Age/MapServer/4/query?text=Vermont&returnGeometry=false
```

What are the pieces of this URL, and what is the purpose of each piece? The following graphic breaks down the preceding URL into its constituent pieces, and provides some tips for success when working with URLs like this.





ArcGIS for Server exposes GIS resources through an ArcGIS Server Site. The **site** indicates the ArcGIS Server Site where GIS resources are being exposed. This piece is case-sensitive: It should be noted as **arcgis** (lowercase). But the **ArcGIS** notation is also supported, and is used in this course for easier identification with the ArcGIS product name.



Add the **service catalog**, and you have the well-known endpoint URL. All lowercase is required here.

[< Back](#) [Next >](#)

⚠ case-sensitive
✓ REST endpoint

Folder: Demographics/

gisonline.com/ArcGIS/rest/services/Demographics/USA_Median_Age/MapServer/4/qu

A **folder** may be needed when a site's services have been organized into folders. Folder names are case-sensitive.

[< Back](#) [Next >](#)

⚠ case-sensitive

Service name: USA_Median_Age/

rcGIS/rest/services/Demographics/USA_Median_Age/MapServer/4/query?text=Vermont

The **service name** identifies the targeted service, and is case-sensitive. Note that the service name is not a valid REST endpoint. Trying to go here will return error 500. This and other errors will be covered later in this course.

! case-sensitive
✓ REST endpoint

Service type: MapServer/

ces/Demographics/USA_Median_Age/MapServer/4/query?text=Vermont&returnGeome

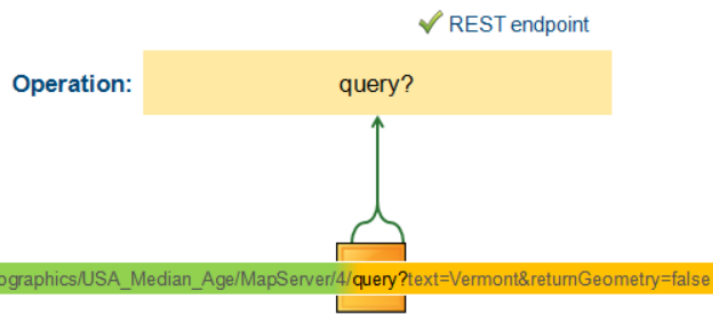
The **service type** indicates the kind of functionality provided by the service. The service type is always in camel-case: The first letter of each word is capitalized, and the remaining letters are lowercase.

✓ REST endpoint

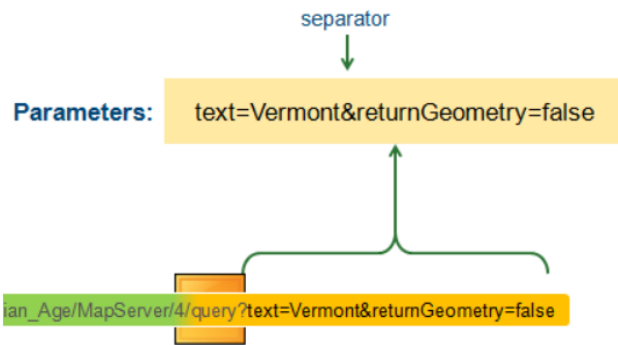
Layer: 4/

mographics/USA_Median_Age/MapServer/4/query?text=Vermont&returnGeometry=fal

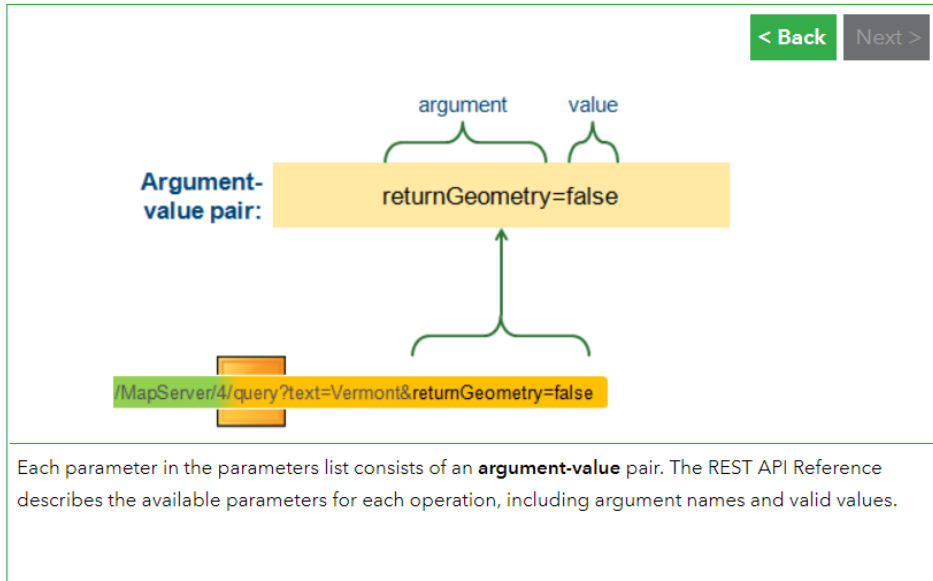
When the targeted resource is a **layer**, the layer is indicated by index number. Layers use a zero-based index, so index number 4 refers to the *fifth* layer in the service. Note that layers are valid REST endpoints.



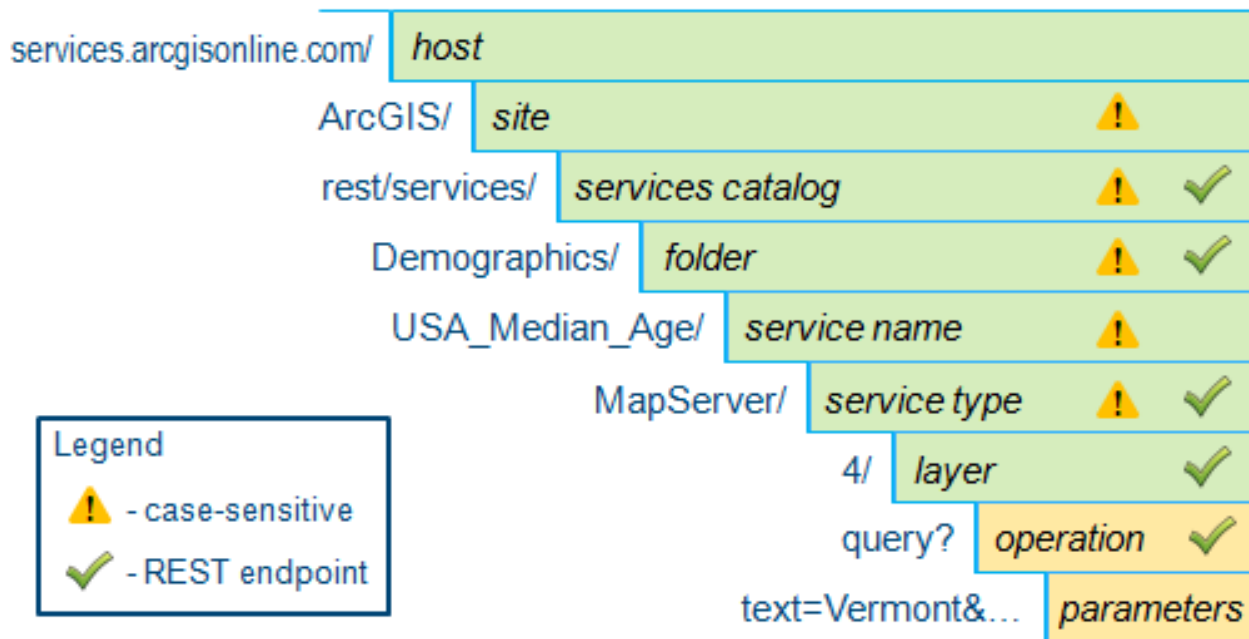
The **operation** indicates what action or task the targeted resource is being asked to perform. Different resources may support different operations as indicated on the Services Directory page for the resource. The operation is an *interactive* REST endpoint, as you will see later in this course.



The **parameters** list shows the various inputs being sent with the request. The parameters are organized into argument-value pairs, with the first pair listed after a question mark (?) and subsequent pairs separated by ampersand (&) characters. See the next slide for a detailed look at one of the argument-value pairs.



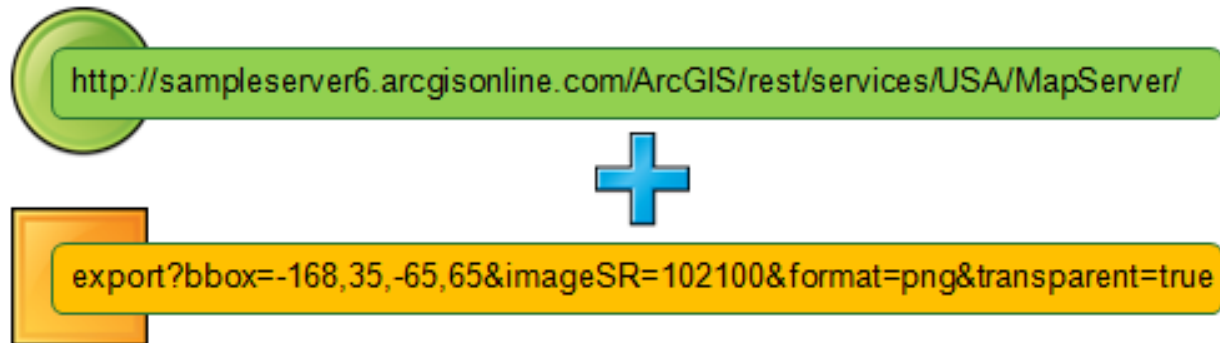
The key details from the preceding graphic are summarized in the following graphic. Remember, point your browser to any valid REST endpoint and the REST API will return the Services Directory page for that endpoint.



When built into a RESTful URL, these pieces ask a specific GIS service to perform a specific operation.

Communicating with GIS services

To ask a GIS service to perform a task (for example, producing a map), you need to build and send a request. A request indicates a resource and an operation, as the following example shows.



A request is built by combining a resource (green) and an operation (orange).

How do you know that "export" is an operation supported by the targeted resource? What parameters does that operation accept? Is there an easy way to build and send requests like this one?

The answer to all these questions starts with the REST API's Services Directory. The Supported Operations section of a resource's page shows which operations that resource allows.

Note: The Services Directory will be disabled for ArcGIS Online services that require a subscription. To learn more, visit the hosting server's home page URL; for example, the ArcGIS Online Geocoding Service at <http://geocode.arcgis.com>

Visit the Services Directory page for the USA (MapServer) service on sampleserver6.arcgisonline.com. What operations are supported by this resource?

Answer

- Export Map
- Identify
- Find
- Return Updates
- Generate KML

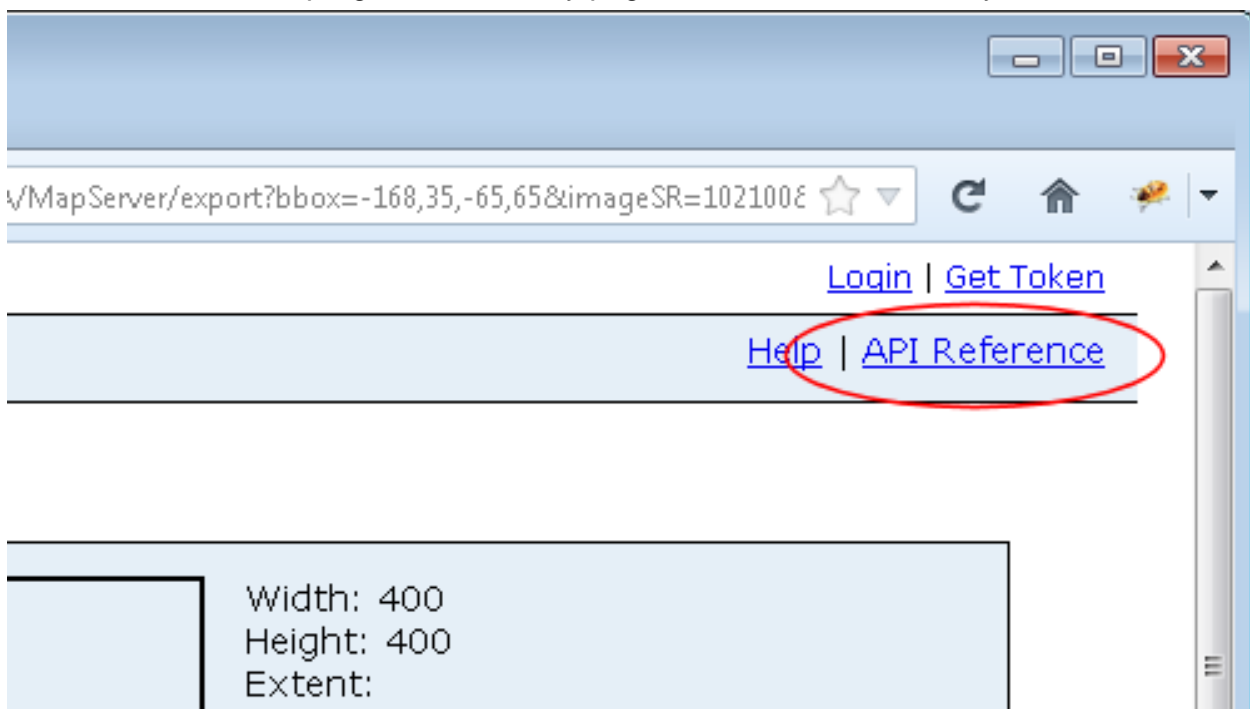
Each supported operation is a hyperlink that takes you to the REST endpoint for that operation. An operation's REST endpoint includes an HTML form that lists the operation's parameters. You can interact with the form to build requests based on the operation.

Bounding Box:	<input type="text" value="-168.7341344174033,40.6502468484659,-76.45273506959677,125.9102353763306"/>
Bounding Box Spatial Reference:	<input type="text"/>
Layers:	<input type="text"/>
Layer Definitions:	<input type="text"/>
Image Size:	<input type="text"/>
Image Spatial Reference:	<input type="text"/>
Image Format:	<input type="text" value="PNG"/>

The first few fields in the HTML form for the Export Map operation, showing default values. The form also includes eight more fields not shown here.

Completing the HTML form is easier than manually typing long request URLs, but what does each parameter mean, and what kind of input value does it need?

These questions can be answered using the REST API Reference, which can be accessed via the link near the top-right corner of any page in the Services Directory.



The API Reference link is found on each page in the Services Directory.

The REST API Reference will automatically open to the most appropriate documentation for the REST endpoint from which it was opened. For example, if you are viewing the Export Map operation endpoint and you click the API Reference link, the REST API Reference will show documentation for the Export Map operation.

The REST API Reference provides details on each parameter, including syntax and examples.

Request Parameters

Parameter	Details
<code>?</code>	Description: The response format. The default response format is <code>html</code> . If the format is <code>image</code> , the image bytes are directly streamed to the client. Values: <code>html</code> <code>json</code> <code>image</code> <code>kmlz</code>
<code>bbox</code>	Description: (Required) The extent (bounding box) of the exported image. Unless the <code>bboxSR</code> parameter has been specified, the <code>bbox</code> is assumed to be in the spatial reference of the map. Syntax: <code><minx>, <miny>, <maxx>, <maxy></code> Example: <code>bbox=-104,35.4,-94.32,81</code> The <code>bbox</code> coordinates should always use a period as the decimal separator even in countries where traditionally a comma is used.
<code>size</code>	Description: The size width *height of the exported image in pixels. If the <code>size</code> is not specified, an image with a default size of 400 * 400 will be exported. Syntax: <code><width>, <height></code> Example: <code>size=400,500</code>
<code>dpi</code>	Description: The device resolution of the exported image (dots per inch). If the <code>dpi</code> is not specified, an image with a default DPI of 96 will be exported. Example: <code>dpi=300</code>

A portion of the REST API Reference for the Export Map operation, showing the start of the Request Parameters documentation.

You can use the information in the REST API Reference to understand what each parameter does, whether the parameter is required, and the correct syntax to use in the HTML form.

Note: The examples given in the REST API Reference show the argument-value pairs as they would appear in a request URL. The HTML form requires only the *value* portion. Omit the argument and equal sign (=) when completing the HTML form.

Example: `size=600,550`

An example from the REST API Reference. When completing the HTML form, enter only the portion to the right of the equal sign (=).

The Services Directory's HTML forms make it easy to build operation requests. The next step is to send the request to the server.

Methods for sending requests

At the bottom of the Export Map HTML form are two options for sending the request to the server:



A map can be exported using two different request methods.

As indicated by these buttons, the REST API supports two HTTP methods for sending requests: GET and POST.

GET is the default HTTP method for sending requests. With the GET method, the operation and parameters are passed as part of the URL, making it easy to see what's being sent to the server.



GET passes the operation and parameters as part of the request URL.

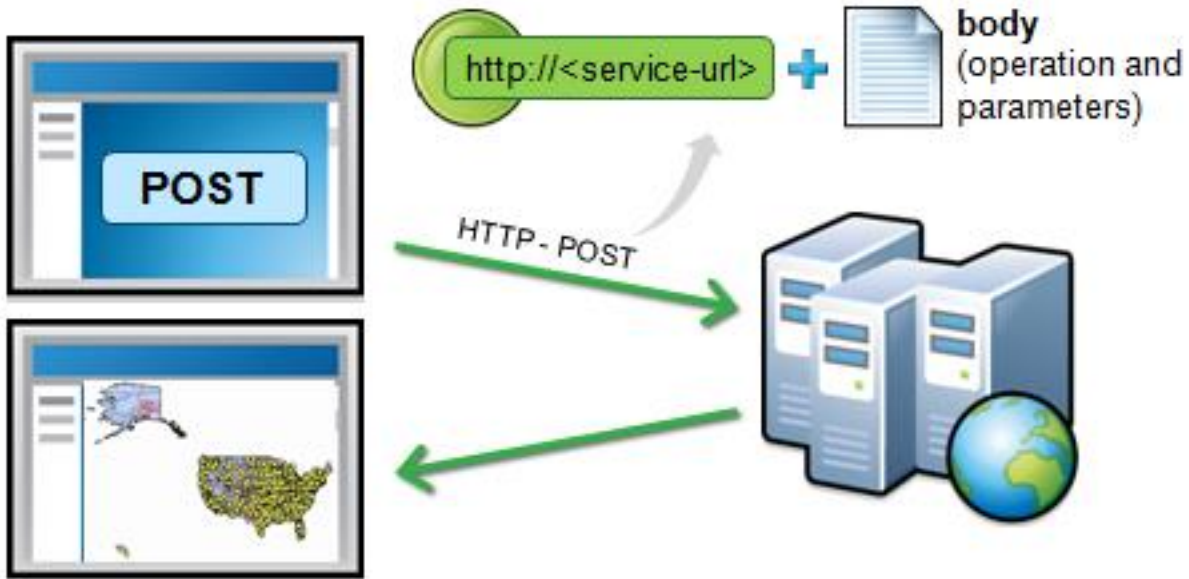
Using GET allows you to bookmark specific resources and operations (with specific parameters) in your browser. For example, you can bookmark a particular map query that you want to perform on a regular basis.

But there are two common situations in which the GET method won't meet your needs:

- **Long requests**—The length of a URL is limited to as few as 1,024 characters by some browsers. Request URLs often exceed that length when they include geometry information (coordinates), spatial reference text descriptions, or even sufficiently complex attribute query criteria.
- **Altering the server**—In some situations, you may want to modify the data stored on the server. For example, you may want to add a new feature to an editable data layer. For safety reasons, requests that effect a change to the state of the server cannot be

sent using GET (for example, think about the dangers of bookmarking a request to delete all features).

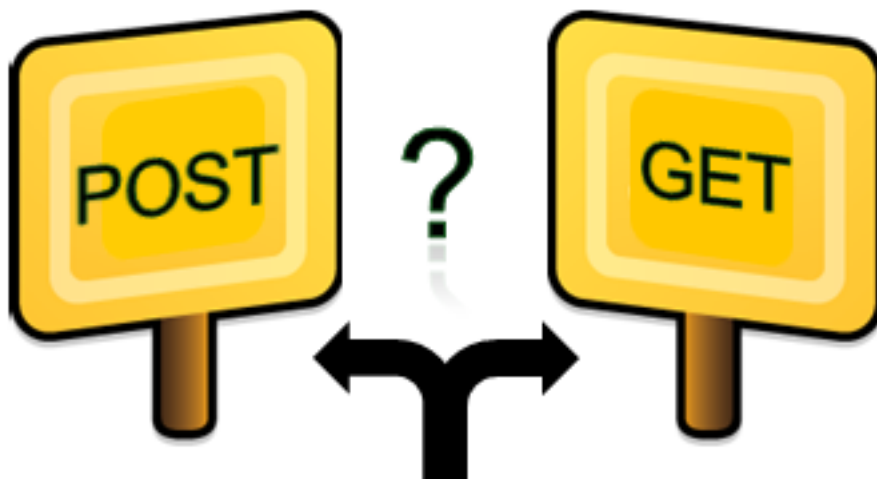
In situations in which GET isn't suitable, use the POST method. POST sends the operation and parameters in the body of the request, separate from the URL.



POST passes the operation and parameters in the body of the request.

The length of the POST body can safely exceed 1,024 characters. Requests sent by POST can't be bookmarked because bookmarks record only the URL, not the body, of the request.

Which method should you choose?



The choice between the GET and the POST methods depends on several factors.

With the REST API, use the GET method whenever possible. Requests sent via GET are easier to troubleshoot and can be copy-pasted or bookmarked.

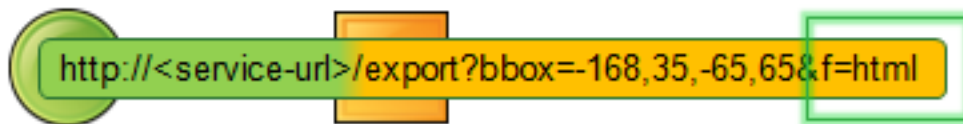
More information

- If the GET method is used with an operation that only allows POST (for example, the FeatureServer's Apply Edits operation), a 405 (Method not allowed) error will be returned by the server.
- In some cases, GET can be used to send a long request that is stored as a file on a public server. For more information, see *Understand options for sending long JSON objects in a request* (item 6) in the API Reference topic [Getting Started](#)
- When you are working with custom Server Object Extensions (SOEs), certain operations may only support POST. For more information, see the API Reference topic [Map Service Extension](#)

Formatting the responses

When you send a request to a GIS service, what do you want to receive in response? The answer often depends on what you want to do with the response. For example, when exporting a map, you often want to display the exported map image, but sometimes you might only want to know the scale of the exported image so you can display the map scale information.

Server responses can be returned in a variety of formats. You can choose the format that best meets the needs of your particular situation. You specify the response format by using the `f=format` parameter when building the request, as shown.



Use of the format (`f=`) parameter in a request URL.

In the previous example, the `f=html` parameter tells the server to return the response in HTML format (the format of the Services Directory).

Note: HTML is the default format. If a RESTful URL's operation doesn't specify `f=format`, the server will assume `f=html` and will return the response as a Services Directory page.

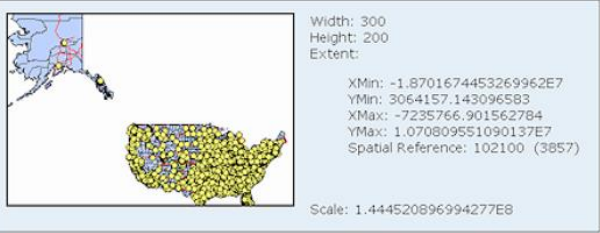
The following graphic describes situations in which you might use the format.

Click an item below.

- f=html**
- f=json
- f=pjson
- f=amf
- f=nmf
- f=image
- f=kmz
- f=lyr

ArcGIS REST Services Directory
[Home](#) > [services](#) > [USA \(MapServer\)](#) > [export](#)

Export Map (USA)



Bounding Box: -168.35;-65.85
Bounding Box Spatial Reference:

Displays the Services Directory page (HTML) for the targeted endpoint. This is the default format. It is used to discover, explore, and test GIS services.

Click an item below.

- f=html
- f=json
- f=pjson
- f=amf
- f=nmf
- f=image
- f=kmz
- f=lyr

```
{ "href": "http://sampleserver6.arcgisonline.com/arcgis/rest/directories/arcgisoutput/USA_MapServer/_ags_map1dd5286c172448559d81bdb4cc66bf57.png", "width": 400, "height": 400, "extent": { "xmin": -168, "ymin": -1.5, "xmax": -65, "ymax": 101.5, "spatialReference": { "wkid": 4326, "latestWkid": 4326 } }, "scale": 108217785.19129333 }
```

Returns a consistently formatted JSON object that other APIs can parse to extract desired information. Commonly used when working with the ArcGIS Web APIs.

Click an item below.

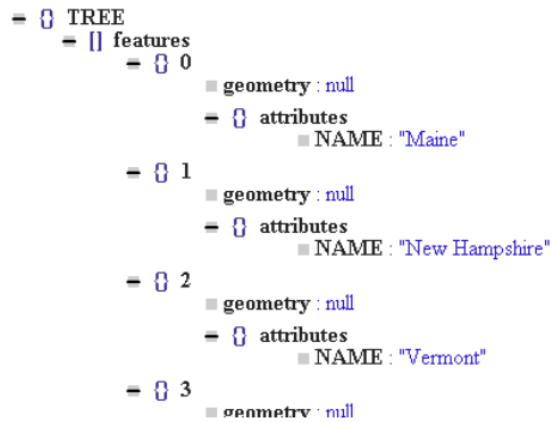
- f=html
- f=json
- f=pjson
- f=amf
- f=nmf
- f=image
- f=kmz
- f=lyr

```
{ "href": "http://sampleserver6.arcgisonline.com/arcgis/r", "width": 300, "height": 200, "extent": { "xmin": -18701674.453269962, "ymin": 3064157.143096583, "xmax": -7235766.9015627839, "ymax": 10708095.510901369, "spatialReference": { "wkid": 102100, "latestWkid": 3857 } }, "scale": 144452089.69942769 }
```

Returns a "prettified" JSON object that has been formatted with spaces and line breaks for improved readability. Should be used only for troubleshooting because it is less compact (and therefore requires more bandwidth) than f=json.

Click an item below.

- f=html
- f=json
- f=pjson
- f=amf**
- f=nmf
- f=image
- f=kmz
- f=lyr



Returns an Adobe ActionScript object that has been serialized into binary Action Message Format. Used primarily by Adobe Flex and Adobe AIR clients, this format is more efficient than JSON for communicating large query results. Screenshot produced using the online Amf Viewer utility at <http://amfview.org>.

Click an item below.

- f=html
- f=json
- f=pjson
- f=amf
- f=nmf**
- f=image
- f=kmz
- f=lyr

```

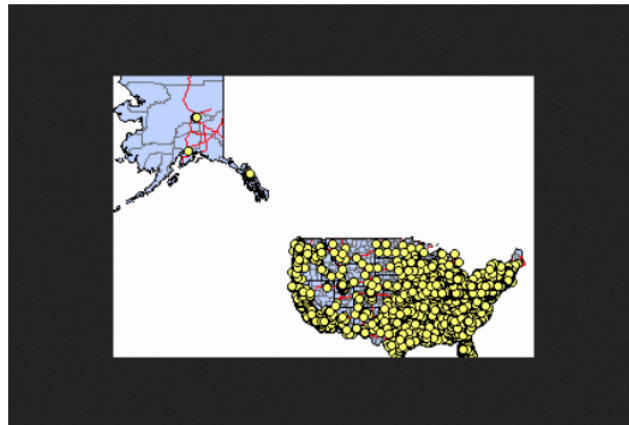
<?xml version="1.0" encoding="utf-8"?>
<esri:ArcGISExplorerDocument xmlns:esri='http://services.arcgisonline.com/ArcGIS/
<E2Layers xsi:type="esri:ArrayOfE2Layer">
  <E2Layer xsi:type="esri:E2ServerLayer">
    <DisplayName>World_Topo_Map</DisplayName>
    <Visible>true</Visible>
    <LODTileFetch>true</LODTileFetch>
    <HideFromContents>false</HideFromContents>
    <Transparency>0</Transparency>
    <ServerType>2</ServerType>
    <GlobeLayerType>0</GlobeLayerType>
    <Url>http://services.arcgisonline.com/ArcGIS/
    <ServiceName>World_Topo_Map</ServiceName>
    <SubServiceName>World_Topo_Map</SubServiceName>
    <E2GlobeLayerProperties xsi:type="esri:E2Se
    <MinimumVisibleDistance>0</MinimumVisibleD
    <MaximumVisibleDistance>0</MaximumVisibleD

```

Returns an ArcGIS Explorer Map (.nmf) file containing XML that points to the requested resource. Used by ArcGIS Explorer Desktop.

Click an item below.

- f=html
- f=json
- f=pjson
- f=amf
- f=nmf
- f=image
- f=kmz
- f=lyr



Returns only an image, streamed directly from the server. Use this format to efficiently retrieve an exported map image in a single request (versus JSON, where a first request returns JSON containing the image URL, and a second request is then needed to retrieve the image). Not to be confused with the Export Map operation's `format=<image-format>` parameter.

Click an item below.

- f=html
- f=json
- f=pjson
- f=amf
- f=nmf
- f=image
- f=kmz
- f=lyr

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns:atom="http://www.w3.org/2005/Atom"
  <GroundOverlay>
    <Icon>
      <href>http://sampleserver6.arcgisonline
    </Icon>
    <LatLonBox>
      <north>84.33333333333334</north>
      <south>15.666666666666671</south>
      <east>-65.000000000000004</east>
      <west>-168.000000000000006</west>
    </LatLonBox>
  </GroundOverlay>
</kml>
```

Returns a zipped (compressed) Keyhole Markup Language (.kml) file containing formatted XML. Used primarily by Google Earth.

Click an item below.

- f=html
- f=json
- f=pjson
- f=amf
- f=nmf
- f=image
- f=kmz
- f=lyr**



Returns an ArcGIS layer (.lyr) file containing layer properties in binary format. Used by ArcGIS for Desktop, especially ArcMap.

Click an item below.

- f=pjson
- f=amf
- f=nmf
- f=image
- f=kmz
- f=lyr
- f=jsapi**
- f=help

ArcGIS JavaScript API: World_Topo_Map



Returns an HTML/JavaScript web page that displays the resource in a simple viewer built from the ArcGIS API for JavaScript. Use this format to quickly see how a resource appears and behaves in a simple web application.

Click an item below.

- f=pjson
- f=amf
- f=nmf
- f=image
- f=kmz
- f=lyr
- f=jsapi
- f=help**

Export Map

URL:	http:// <mapservice-url>/export
Required Capability:	Map
Version Introduced:	9.3

Description

The export operation is performed on a [map service resource](#). The provides information about the exported map image such as its UI

Apart from the usual response formats of html and json, users can perform an export with the format of image. When users perform an export with the format of image, the client. One must note that with this approach you don't get an actual image.

Returns the REST API Reference page for the resource or operation. This is the same effect as clicking the **API Reference** link on the resource/operation's Services Directory page. You can use `f=help` to access the REST API Reference directly, without first going to the Services Directory.

More about JSON

JavaScript Object Notation (JSON) is a plain-text format that can represent complex data, such as feature geometry and attributes, in a consistent way. For more JSON-related information and tools, see:

- Wikipedia: <http://en.wikipedia.org/wiki/JSON>
- JSON.org: <http://json.org>
- JSONLint, a JSON validation and formatting tool: <http://jsonlint.com>

You have learned how RESTful requests are constructed and sent, and how to request responses in a variety of formats. You're ready to put this knowledge to work!

[See Exercise2.pdf: Testing a service's operations](#)

