



Using ArcWebSM Services for SVG Map Generation

May 2007

Copyright © 2007 ESRI
All rights reserved.
Printed in the United States of America.

The information contained in this document is the exclusive property of ESRI. This work is protected under United States copyright law and other international copyright treaties and conventions. No part of this work may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying and recording, or by any information storage or retrieval system, except as expressly permitted in writing by ESRI. All requests should be sent to Attention: Contracts and Legal Services Manager, ESRI, 380 New York Street, Redlands, CA 92373-8100 USA.

The information contained in this document is subject to change without notice.

U.S. GOVERNMENT RESTRICTED/LIMITED RIGHTS

Any software, documentation, and/or data delivered hereunder is subject to the terms of the License Agreement. In no event shall the U.S. Government acquire greater than RESTRICTED/LIMITED RIGHTS. At a minimum, use, duplication, or disclosure by the U.S. Government is subject to restrictions as set forth in FAR §52.227-14 Alternates I, II, and III (JUN 1987); FAR §52.227-19 (JUN 1987) and/or FAR §12.211/12.212 (Commercial Technical Data/Computer Software); and DFARS §252.227-7015 (NOV 1995) (Technical Data) and/or DFARS §227.7202 (Computer Software), as applicable. Contractor/Manufacturer is ESRI, 380 New York Street, Redlands, CA 92373-8100 USA.

@esri.com, 3D Analyst, ACORN, ADF, AML, ArcAtlas, ArcCAD, ArcCatalog, ArcCOGO, ArcData, ArcDoc, ArcEdit, ArcEditor, ArcEurope, ArcExplorer, ArcExpress, ArcGIS, ArcGlobe, ArcGrid, ArcIMS, ARC/INFO, ArcInfo, ArcInfo Librarian, ArcInfo—Professional GIS, ArcInfo—The World's GIS, ArcLessons, ArcLocation, ArcLogistics, ArcMap, ArcNetwork, *ArcNews*, ArcObjects, ArcOpen, ArcPad, ArcPlot, ArcPress, ArcQuest, ArcReader, ArcScan, ArcScene, ArcSchool, ArcScripts, ArcSDE, ArcSdl, ArcSketch, ArcStorm, ArcSurvey, ArcTIN, ArcToolbox, ArcTools, ArcUSA, *ArcUser*, ArcView, ArcVoyager, *ArcWatch*, ArcWeb, ArcWorld, ArcXML, Atlas GIS, AtlasWare, Avenue, Business Analyst Online, BusinessMAP, Community, CommunityInfo, Data Automation Kit, Database Integrator, DBI Kit, EDN, ESRI, ESRI BIS, ESRI—Team GIS, ESRI—*The GIS Company*, ESRI—The GIS People, ESRI—The GIS Software Leader, FormEdit, GeoCollector, Geographic Design System, Geography Matters, Geography Network, GIS by ESRI, GIS Day, GIS for Everyone, GISData Server, JTX, MapBeans, MapCafé, MapData, MapObjects, Maplex, MapStudio, ModelBuilder, MOLE, NetEngine, PC ARC/INFO, PC ARCPLOT, PC ARCSHELL, PC DATA CONVERSION, PC STARTER KIT, PC TABLES, PC ACEDIT, PC NETWORK, PC OVERLAY, PLTS, Rent-a-Tech, RouteMAP, SDE, Site-Reporter, SML, Sourcebook-America, Spatial Database Engine, StreetEditor, StreetMap, Tapestry, the ARC/INFO logo, the ArcAtlas logo, the ArcCAD logo, the ArcCAD WorkBench logo, the ArcCOGO logo, the ArcData logo, the ArcData Online logo, the ArcEdit logo, the ArcEurope logo, the ArcExplorer logo, the ArcExpress logo, the ArcGIS logo, the ArcGIS Explorer logo, the ArcGrid logo, the ArcIMS logo, the ArcInfo logo, the ArcLogistics Route logo, the ArcNetwork logo, the ArcPad logo, the ArcPlot logo, the ArcPress for ArcView logo, the ArcPress logo, the ArcScan logo, the ArcScene logo, the ArcSDE CAD Client logo, the ArcSDE logo, the ArcStorm logo, the ArcTIN logo, the ArcTools logo, the ArcUSA logo, the ArcView 3D Analyst logo, the ArcView Data Publisher logo, the ArcView GIS logo, the ArcView Image Analysis logo, the ArcView Internet Map Server logo, the ArcView logo, the ArcView Network Analyst logo, the ArcView Spatial Analyst logo, the ArcView StreetMap 2000 logo, the ArcView StreetMap logo, the ArcView Tracking Analyst logo, the ArcWorld logo, the Atlas GIS logo, the Avenue logo, the BusinessMAP logo, the Community logo, the Data Automation Kit logo, the Digital Chart of the World logo, the ESRI Data logo, the ESRI globe logo, the ESRI Press logo, the Geography Network logo, the MapCafé logo, the MapObjects Internet Map Server logo, the MapObjects logo, the MOLE logo, the NetEngine logo, the PC ARC/INFO logo, the Production Line Tool Set logo, the RouteMAP IMS logo, the RouteMAP logo, the SDE logo, The Geographic Advantage, The World's Leading Desktop GIS, *Water Writes*, www.esri.com, www.esribis.com, www.geographynetwork.com, www.gis.com, www.gisday.com, and Your Personal Geographic Information System are trademarks, registered trademarks, or service marks of ESRI in the United States, the European Community, or certain other jurisdictions.

Other companies and products mentioned herein may be trademarks or registered trademarks of their respective trademark owners.

Introduction	1
ArcWeb Services for SVG Mapping	1
Security	1
Creating an Account	1
Token Authorization	2
Retrieving an Unprojected Map	3
Projected Maps	6
The Current Map Reference	9
Other Projections and Known Coordinate Systems	10
Adding a Graticule	12
Adding Geometry on the Server Side	13
Legend, Scale Bar, and North Arrow	15
SVG Structure for GetMap Action	17
SVG Viewer Sample Application	19
Adding Map Mashups to the SVG Viewer Application	20
Conclusion	24

Introduction

ESRI ArcWeb Services provide geospatial Web services to create maps, geocode, reverse geocode, upload map features, find places, find addresses, get routes, find nearest object, generate reports, and so forth. Generally, these services are pay-as-you-go, but ArcWeb Services also provide a free Public Services program that can be used in personal, noncommercial, and nongovernmental applications.

The ArcWeb Services Representational State Transfer (REST) API can be used to access services to generate scalable vector graphics (SVG) maps, and these maps can be displayed within a browser, provided it has either an SVG plug-in or native support.

ArcWeb Services for SVG Mapping

ArcWeb Services provide extremely fast vector map services via the clusters and machinery used in the ArcWeb Services data centers. There are several reasons these vector services are so fast. An equivalent dynamically generated image map is larger in content size than its vector counterpart, which indicates that the vector map utilizes less bandwidth resources. The vector map uses less CPU resources on the servers than its associated dynamically generated image map, which implies that the map is delivered to the client more quickly and the vector servers have a larger map throughput.

ArcWeb Services provide access to SVG map generation services in two forms: a SOAP API and a REST API. This paper focuses on the use of the REST v2006 API. The REST v2006 API works by specifying an action and a variable number of parameters as documented in the API itself. The following examples illustrate a getToken action and a getMap action.

Example getToken action:

```
https://www.arcwebservices.com/services/v2006/restmap?actn=getToken&usr=<username>&pswd=<password>
```

Example getMap action:

```
http://www.arcwebservices.com/services/v2006/restmap?actn=getMap&fmt=svg&ds=bam&sf=20000000&c=-80|40&w=600&h=600&ocs=6&g=15&gc=220000ff&gw=2&tkn=<token generated by above request>
```

The parameters can be specified in any order.

Security

ArcWeb Services are accessed via a user account and password, as the ArcWeb Services business model has the user purchase credits for service use. ArcWeb Public Services also require a user account and password, and both user name and password are necessary to retrieve a token that can be used with all service requests.

Creating an Account

To create an account, visit <http://www.arcwebservices.com/v2006/evaluate.jsp>. Sign up for an evaluation account for a free 90-day (or 5,000-credit) trial. For information on how many credits a specific service costs, visit http://www.arcwebservices.com/v2006/help/index.htm#support/basics_credits.htm. Generally, each SVG map costs about 1 credit. Credits can be purchased in blocks, and high-volume discounts are available. There are also Public Services that are free for developers working on personal, noncommercial, and nongovernmental projects. Public Services, offered through a one-year subscription, provide similar functionality to Commercial Services and a subset of Commercial Services mapping data. Available APIs

are SOAP, REST, OpenGIS Location Services (OpenLS), Java 2 Platform Micro Edition (J2ME), and ArcExplorer JavaScript API.

Token Authorization

The user name and password generate a token that is used to access map services. This token has a default timeout value of 60 minutes, with a maximum timeout of 24 hours. The expiration helps minimize the impact of a malicious client stealing and trying to reuse the authentication token to access ArcWeb Services.

The token is created with the use of the `getToken` action within the REST API:

```
https://www.arcwebservices.com/services/v2006/restmap?actn=getToken&usr=<username>&pswd=<password>
```

The following is returned:

```
<?xml version="1.0" encoding="utf-8" ?>
<TOKEN>a2NvZmZpbjoxAAEwNTU5OToyM2MyMDY0ZjllNWRjZTA3ZjhhNWYzZmlzZjcyMWEzNA%3D%3D</TOKEN>
```

The following table shows all the parameters for the `getToken` action within the REST v2006 API:

Table 1
REST v2006 API Get Token Parameters

Authentication (actn=getToken)		
Parameter	Description	Valid Values
usr (required)	ArcWeb Services account user name.	String containing valid user name. Case sensitive.
pswd (required)	ArcWeb Services account password.	String containing valid password. Case sensitive.
tout (optional)	Expiration time of token, in minutes.	Minutes to expiration. Default value is 60. Maximum value is 1440.
ip (optional)	IP address of client application viewing ArcWeb Services maps. Only use if the client machine getting the token is different from the machine requesting the maps. Use the IP utility to view the IP address seen by ArcWeb Services.	IP address in format xxx.xxx.xx.xxx such as 198.102.62.126.

Use a proxy servlet that will hide the authentication process from the end user. The client makes the REST request without using the token (`tkn`) parameter, and the proxy servlet generates the token and appends it to the parameters received from the client. The proxy servlet makes a REST API call to ArcWeb Services, collects the response, and passes the response to the client.

ArcWeb Services offer an alternative method of authentication by the use of a user ID that is generated from the user name and password information, which does not expire. The above token authentication method is preferred. For the remainder of the document, assume that the token is added via the proxy servlet, so you

can examine how to use ArcWeb Services from the client. The token is not included in any subsequent requests. In this document, proxy servlet will be referred to as:

```
http://restproxy.com/restmap?
```

instead of

```
http://www.arcwebservices.com/services/v2006/restmap?
```

Retrieving an Unprojected Map

The getMap action specifies to the REST API that a map is being requested. You can request an unprojected map (shown in figure 1) with the following:

```
http://restproxy.com/restmap?actn=getMap&fmt=svg&ds=bam&sf=100000000&c=0|30
```

The parameters specify that the map is in SVG format, with the best available map data source and a map scale of 1:100,000,000 centered at 0 degrees longitude and 30 degrees latitude (shown in JPG format for this document). Table 2 shows many of the parameter options for the getMap request.

Figure 1



Table 2
REST API Map Request Parameters

Map (actn=getMap)		
Parameter	Description	Valid Values
tkn (required if not using usrid)	Authentication token returned from a getToken request	Returned string from getToken request. Average length of string is 72-80 characters. Recommended authentication method.
usrid (required if not using tkn)	Authentication token returned from a getUserID request	Returned string from getUserID request. Average length of string is 50-70 characters. To enable and disable user IDs, visit the ArcWeb UserID account page . It is recommended you use UserIDs only during phases of evaluation, development, and testing, when they offer more flexibility and convenience. It is also recommended that you change your account password, which invalidates any existing UserIDs, and switch to using tokens before making an application public.
c (required)	Map center point in x y coordinates	The x-coordinate and the y-coordinate of the center point in geographic coordinates (long/lat), separated by a pipeline (), for example, -117.1817 34.0556.
sf (required)	Geographical scale of the map	The distance ratio between two geographic locations and those two points represented on the map. The distance measured on the map can be any units, and that multiplied by the scale factor will give the actual distance in the measured units. For example, if the distance between two points on the screen is 3 cm, for sf=100000, the geographic distance between the two points is 300,000 cm (3 multiplied by 100,000).
ds (required)	Data source	Valid values are: bam (best available map) ArcWeb:AND.Roads.World ArcWeb:TA.Streets.EU ArcWeb:TA.Streets.US ArcWeb:TA.Streets.NA If ds=bam, based on the extent of the map, the best available data source is selected from the valid values above. See Map Image data sources for details about these data sources.
w (optional)	Map image width in pixels	Default value is 400. Maximum width is 8192.
h (optional)	Map image height in pixels	Default value is 400. Maximum height is 8192.
a (optional)	Angle to rotate the map	Default value is 0. Values less than 360 rotate the map through that many degrees. Negative values are rotated in the opposite direction.

fmt (optional)	Map image format	Valid values are svg, swf, jpg, or png. Default value is swf.
stl (optional)	Style sheet used in the map	Style sheets are specific to a data source (ds). The default style for a data source is nt (Neutral). Other options are ce (Classic European), cs (Cool Steel), gs (Gray Scale), or tg (Tangerine). You can set a style for the following data sources: ArcWeb:AND.Roads.World uses nt, gs, or tg; ArcWeb:TA.Streets.EU uses nt, gs, tg, or ce; ArcWeb:TA.Streets.US uses nt, cs, gs, tg, or ce; and ArcWeb:TA.Streets.NA uses nt, gs, or tg. Use the Map Image Live Sample to preview style sheets in a map.

The center is specified in longitude and latitude. Width and height are specified in pixels. The rotation is specified in degrees. The map format can be SVG, SWF, JPG, or PNG. The style can be Neutral, Classic European, Cool Steel, Gray Scale, or Tangerine, and the data source can be ArcWeb:AND.Roads.World, ArcWeb:TA.Streets.EU, ArcWeb:TA.Streets.US, or ArcWeb:TA.Streets.NA.

Here is another example with street-level scale:

```
http://restproxy.com/restmap?actn=getMap&fmt=svg&ds=bam&sf=10000&c=-75|40
```

Figure 2

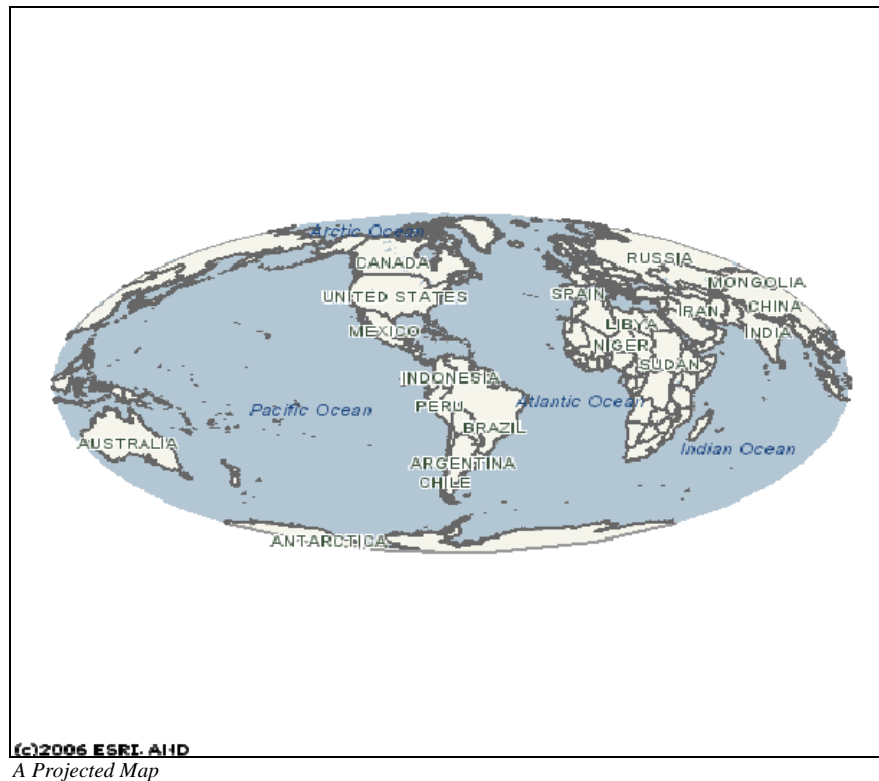


Projected Maps

For unprojected maps, geometry will be distorted in latitudes farther from the equator. For example, streets in New York City will not be perpendicular to one another and land masses will not appear in their actual geometric shape. The REST API offers projections to solve this problem with little effect on performance, as ArcWeb Services can project data in parallel for the client. The following requests a projected SVG map; results are shown in figure 3.

```
http://restproxy.com/restmap?actn=getMap&fmt=svg&ds=bam&sf=300000000&c=-70|0&ocs=0&w=500&h=500
```

Figure 3



Here an output coordinate system parameter (ocs) is used, and the width and height of the map are specified. In addition to the parameters already mentioned, the following are parameters related to the use of projections:




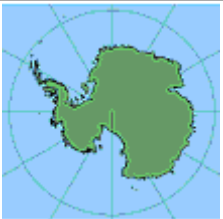
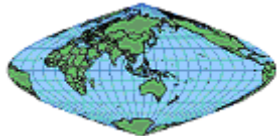

Table 3
REST API Projection-Related Parameters


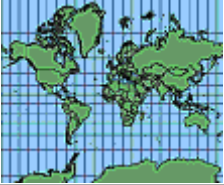

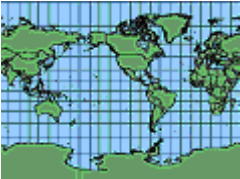
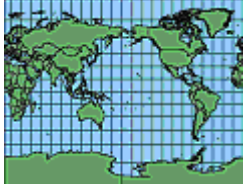
Projection		
Parameter	Description	Valid Values
ocs	The output coordinate system.	Valid values are 0-11. See OCS types for details. If you set ocs to 0 (Automatic), one of the 11 OCS types will be selected automatically based on the current extent of the map. You can also use EPSG ID or projection string. See Projected coordinates by ID or Projected coordinates listed by name (string) for valid values. Default value is 4326 (EPSG ID for GCS_WGS_1984), which is an unprojected map.
dc (required if cmr is used)	The difference between the current center and the new center point in dx dy in pixels.	Two integers representing number of pixels separated by a pipeline (). For example, 50 30 pans the map 50 pixels to the east and 30 pixels to the north. To pan the map in pixels, you must use this parameter with the cmr parameter.
cmr (required if dc is used)	Current map reference. This parameter contains all georeferencing information about the map in the following format: longitude latitude px py scale angle projection longitude: in decimal degrees; negative is West latitude: in decimal degrees; negative is South px: offset from center in pixels; negative is left py: offset from center in pixels; negative is up scale: gis scale, unitless, 1:N where scale is N angle: rotation in decimal degrees; negative is ccw (counter clockwise) projection : EPSG ID, projection number, projection string, or OCS type	For SVG format, the cmr value is contained in the first desc tag following the SVG tag of the projection request . For SWF format, the cmr value can be retrieved by the setCurrentMapReference(String) method call. The cmr value also is returned in the http header for all formats. You must use this parameter with the dc parameter.

The REST v2006 API uses a concept called projection styles. Typically in GIS applications, users select a projection for use in their map and move around that projection, until they "fall off the edge." That is to say, the center of projection stays constant as they move around the map, and since the projected map is flat, one can "fall off the edge." The REST API bypasses this problem completely by having the center of projection change with the center of the map, a concept referred to as projection styles. Projection styles work with output coordinate system types 0-11. Additionally, the SVG Map Viewer sample code supplies the developer with JavaScript code that can project points on the client side for these same projection styles.

These coordinate systems (0-11) are sphere based as opposed to ellipsoid based. The following table shows output coordinate systems 0-11:

Table 4
Output Coordinate Systems (0-11)

Type	Preview	Name	Uses and applications
0	ALL	Automatic	Automatically selects an appropriate projection from the below OCS types 1-11 based on current extent of the map.
1		ALBERS EQUAL AREA CONIC	Used for small regions or countries but not for continents.
2		CASSINI-SOLDNER	Normally used for large-scale maps of areas predominantly north-south in extent.
3		TRANSVERSE MERCATOR	State Plane Coordinate System, used for predominantly north-south state zones. USGS 7½-minute quad sheets. Most new USGS maps after 1957 use this projection.
4		LAMBERT AZIMUTHAL EQUAL AREA	Population density (area), political boundaries (area), oceanic mapping for energy, minerals, geology, and tectonics (direction). This projection can handle large areas; thus, it is used for displaying entire continents and polar regions. Equatorial aspect Africa, Southeast Asia, Australia, the Caribbean, and Central America. Oblique aspect North America, Europe, and Asia.
5		SINUSOIDAL	Used for world maps illustrating area characteristics, especially if interrupted. Used for continental maps of South America, Africa.
6		BONNE	Used during the 19th and early 20th century for atlas maps of Asia, Australia, Europe, and North America.

7		STEREOGRAPHIC	The oblique aspect has been used to map circular regions on the moon, Mars, and Mercury.
8		MERCATOR	Standard sea navigation charts (direction). Other directional uses: air travel, wind direction, ocean currents.
9		MOLLWEIDE	Suitable for thematic or distribution mapping of the entire world.
10		MILLER CYLINDRICAL	Used for general-purpose world maps.
11		GALL'S STEREOGRAPHIC	Used for world maps in British atlases. Used only for world maps.

The Current Map Reference

Every map returned by the REST API has a current map reference (cmr) that is embedded in the reply. For SVG, this spatial reference information is contained in the <desc> tag located immediately after the <svg> tag. For example:

```
<svg width="400px " height="400px" viewBox="0 0 400 400" xmlns="http://www.w3.org/2000/svg"
version="1.0" xmlns:xlink="http://www.w3.org/1999/xlink">
<desc>
-117.0|34.0|0.0|0.0|50000.0|0.0|3
</desc>
```

In this example, the current map reference (cmr) value is:

```
-117.0|34.0|0.0|0.0|50000.0|0.0|3
```

Here -117 and 34 represent the longitude and latitude of the center of the map. The next two values represent the offset of the actual map from the center of the screen map. (This is generally used if the map of the world is zoomed all the way out.) The next value (50000) is the map scale—1:50,000. The rotation in degrees is specified in the next to last value, and the last value represents the coordinate system, which is 3 (Transverse Mercator) in the above example.

What good is this current map reference? You can use this as a parameter in your next getMap REST request and specify an offset in pixels (dc) that indicates the new center location. To change the center of the map (panning), use the dc parameter to specify the pixel offset from the last center. If the user specifies a new zoom window (on the screen), use dc parameter to specify the pixel offset from the existing center to the center of the zoom window, and use the sf parameter to specify the new scale, which is directly related to the zoom window's height (or width) to the map's height (or width) in pixels. This method (center and scale) allows a general solution to panning and zooming on a map that has been rotated and projected without having client-side support for projections.

Note that ArcWeb Services supply JavaScript code for the projections (0-11) in the sample SVG Map Viewer application.

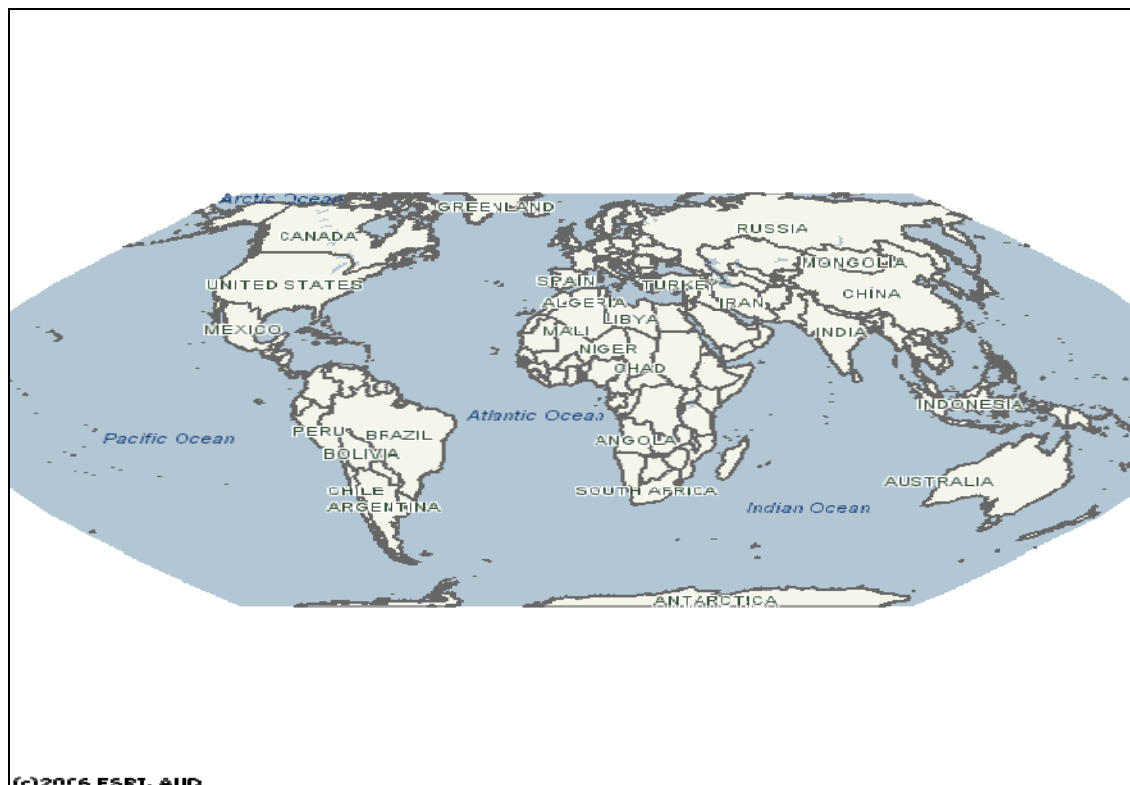
Other Projections and Known Coordinate Systems

In addition to supporting projection styles, the REST v2006 API supports known coordinate systems and ellipsoidal projections. However, the projection center of these coordinate systems stays constant, which may or may not be amenable to the client application.

As noted above in table 3, the ocs parameter can use the projection styles (0-11), EPSG ID, or projection string format. EPSG IDs are numbers in the range 2000-104000. These refer to either a known projected coordinate system or a known geographic coordinate system. The following requests a map using ocs=54010 (Eckert VI Projection), resulting in the map shown in figure 4.

```
http://restproxy.com/restmap?actn=getMap&fmt=png&ds=bam&sf=200000000&c=0|0&w=600&h=600&ocs=54010
```

Figure 4



©2006 ESRI, All Rights Reserved
Eckert VI Projection

Likewise the projection string also refers to either a projected or geographic coordinate system; however, these can be tailor-made. An example of the ESRI projection string format is:

```
PROJCS["World_Robinson",GEOGCS["GCS_WGS_1984",DATUM["D_WGS_1984",SPHEROID["WGS_1984",6378137.0,298.257223563]],PRIMEM["Greenwich",0.0],UNIT["Degree",0.0174532925199433]],PROJECTION["Robinson"],PARAMETER["False_Easting",0.0],PARAMETER["False_Northing",0.0],PARAMETER["Central_Meridian",0.0],UNIT["Meter",1.0]]
```

Thus the following request results in the map shown in figure 5:

```
http://www.arcwebservices.com/services/v2006/restmap?actn=getMap&tkn=a2NvZmZpbjoxOTExMjg4OTpmNDE0MjVkdVMTgzNTQyMmY5YzcyZDkwNjBkYzAzZg%3D%3D&fmt=svg&ds=bam&sf=3000000&c=0&w=500&h=500&ocs=PROJCS["World_Robinson",GEOGCS["GCS_WGS_1984",DATUM["D_WGS_1984",SPHEROID["WGS_1984",6378137.0,298.257223563]],PRIMEM["Greenwich",0.0],UNIT["Degree",0.0174532925199433]],PROJECTION["Robinson"],PARAMETER["False_Easting",0.0],PARAMETER["False_Northing",0.0],PARAMETER["Central_Meridian",0.0],UNIT["Meter",1.0]]
```

Figure 5



Robinson WKT Specified Projection

Adding a Graticule

Add a graticule to the map by using the following graticule parameters in addition to the other parameters within the getMap action.

Table 5
Graticule Parameters in REST v2006 API

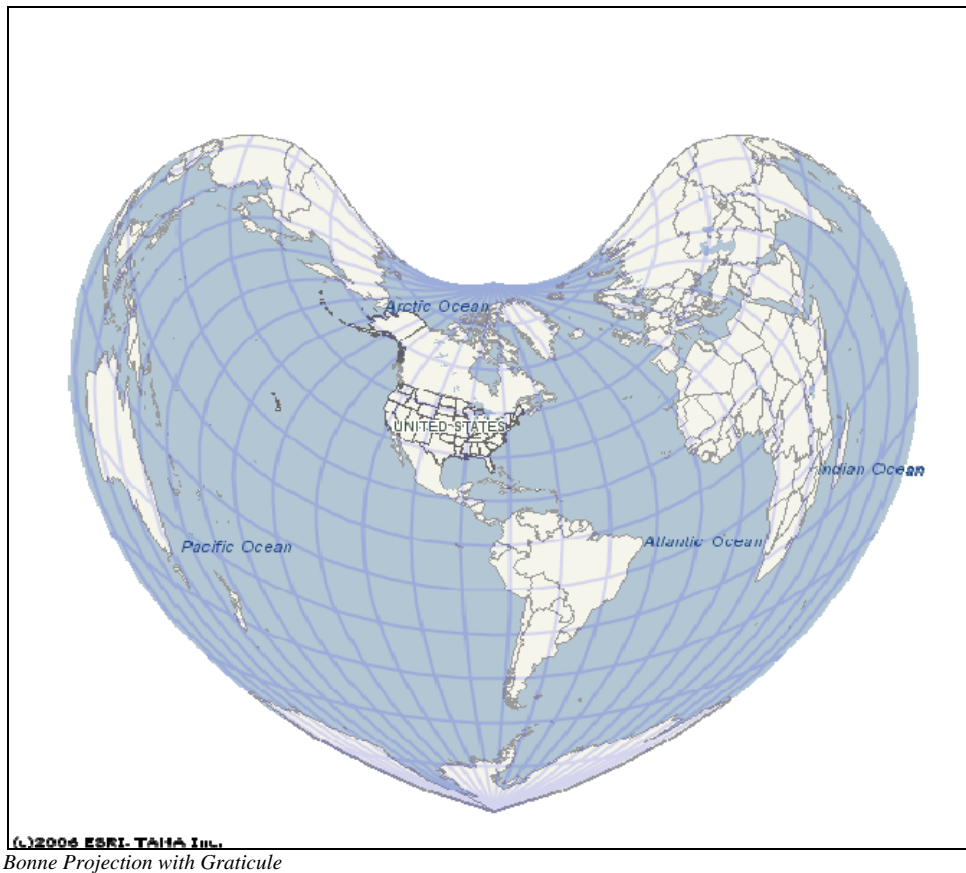
Graticule		
Note: If you do not set g, the graticule will not display on the map.		
Parameters	Description	Valid Values
g	The distance in degrees between two adjacent latitudes or two adjacent longitudes on the map.	Values can range from 1 to 30 and are numeric. Default value is 5.
gc (optional)	Color of graticule.	For example, gc=aaRRGGBB, insert a hexadecimal number in the place of aaRRGGBB. The aa is optional and represents transparency. Default value is 770000ff (blue).
gw (optional)	Width in pixels of the graticule.	Values can range from 1 to 10 and be numeric. Default value is 1.

The following (Bonne projection with graticule) uses g=15, gc=22000ff, and gw=2:

```
http://restproxy.com/restmap?actn=getMap&fmt=svg&ds=bam&sf=200000000&c=-80|40&w=600&h=600&ocs=6&g=15&gc=220000ff&gw=2
```

Note that the above example uses the aaRRGGBB hex values as color. In this example, the alpha or transparency is set to hex 22 on a scale from 0 to ff, and the graticule lines are two pixels wide (gw) and drawn every 15 degrees (g). This results in the map shown in figure 6.

Figure 6



Adding Geometry on the Server Side

The REST v2006 API allows you to add server-side geometry such as points, lines, and polygons on top of the map. The geometry added is assumed to be geographic data, that is, in latitude/longitude decimal degrees. Additionally, the data is automatically warped to whatever coordinate system is specified as the output coordinate system. This added geometry is called acetate geometry: acetate points, acetate lines, and acetate polygons.

**Table 6
Acetate Point Parameters**

Acetate Point		
Note: You must use the Acetate Point parameters p and pl.		
Parameter	Description	Valid Values
p	Point location in x y coordinates.	To display a point, set p=-117.1817 34.0556, where -117.1817 is the x coordinate and 34.0556 is the y coordinate of the center point, separated by a pipeline (). Use with the pl and/or pi parameters.

pl	Label for the point. Font format is Verdana, normal, 10 pt. Label location is set by the point parameter.	Sets a label value for each point or icon on your map. Use with the p parameter.
pi	Icon to use with the point. Icon location is set by the point parameter.	See REST icons for a list of valid values. For example, to display a red pushpin icon, set pi=pushpin_red.gif. Note that for svg and swf images to display, add pi to your URL so that it is not the last parameter listed. Use with the p parameter.

**Table 7
Acetate Line Parameters**

Acetate Line		
Note: You must use all or none of the Acetate Line parameters.		
Parameter	Description	Valid Values
ln	The x1 y1 x2 y2...xn yn coordinates of a polyline.	A series of x and y coordinates of a polyline that are separated by a pipeline (). To display a polyline, set ln=-117 34 -95 45 -85 40, where -117 34 is start point of the polyline, -95 45 is the x,y coordinate of the connecting segment, and -85 45 is the end point of the polyline. ln parameters can be used multiple times in a request.
lnw	Width in integers.	Values can range from 1 to 10. Default value is 1.
lnc	Line color.	Line color in html color format (aaRRGGBB). Can be used multiple times in a request. Default is "FF000000" (opaque and black). The lnc parameter will be ignored if the lnw parameter is not specified.

**Table 8
Acetate Polygon Parameters**

Acetate Polygon		
Note: You must use both or neither of the Acetate Polygon parameters.		
Parameter	Description	Valid Values
plg	The x1 y1 x2 y2...xn yn coordinates of a polygon.	A series of x- and y-coordinates of vertices in a polygon, separated by a pipeline (). To display a polygon, set plg=-117 34 -120 45 -100 50, where -117 34 is one vertex, -120 45 is the second vertex, and -100 50 is the third vertex.
plc	Polygon color.	In html color format (aaRRGGBB). Can be used multiple times in a request. Default is "FF000000" (opaque and black). The plc parameter will be ignored if the plg parameter is not specified.

The following example creates a red line of line width 4 pixels from 20,0 to -70,70 with longitude, latitude. The resulting map is shown in figure 7.

```
http://restproxy.com/restmap?actn=getMap&fmt=svg&ds=bam&sf=200000000&c=-20|30&w=600&h=600&ocs=7&ln=20|0|-70|70&inc=ff0000&lnw=4
```

Figure 7



Acetate Line Example

Legend, Scale Bar, and North Arrow

Legend, Scale Bar, and North Arrow can also be specified as parameters for the getMap action. These are shown in tables 9–11.

**Table 9
Legend Parameters**

Legend		
Note: If you do not set values, the legend will not display on the map.		
Parameter	Description	Valid Values
l	The upper left corner of the legend in pixels, from the top left corner of the map.	The x,y coordinate of the legend, separated by a pipeline (). For example, to display the legend 10 pixels from the left side of the map and 5 pixels from the top of the map, set l=10 5, where 10 is the x coordinate and 5 is the y coordinate. Use positive integers.

**Table 10
North Arrow Parameters**

North Arrow		
Note: You must use all or none of the North Arrow parameters.		
Parameter	Description	Valid Values
na	The center point of the north arrow in pixels, whose origin (0,0) is from the lower left corner of the map.	The x,y coordinate of the center point, separated by a pipeline (). A value of (0,0) will display a partially visible north arrow because the origin of the north arrow object is placed in the lower left corner of the map. To display a fully visible north arrow, use the width (naw) and height (nah) when setting the center point. For example, a value of 20 30 with a width (naw) = 20 and height (nah) = 30, places the north arrow 10 pixels from the y axis and 15 from the x axis.
naw	Width in pixels of the north arrow.	To ensure that the north arrow is fully visible, use height (nah) and center point (na) when setting the width.
nah	Height in pixels of the north arrow.	To ensure that the north arrow is fully visible, use the width (naw) and center point (na) when setting the height.
nat	Type of north arrow.	Valid values are: bk (black), bu (blue), or gr (green). Default value is bk.

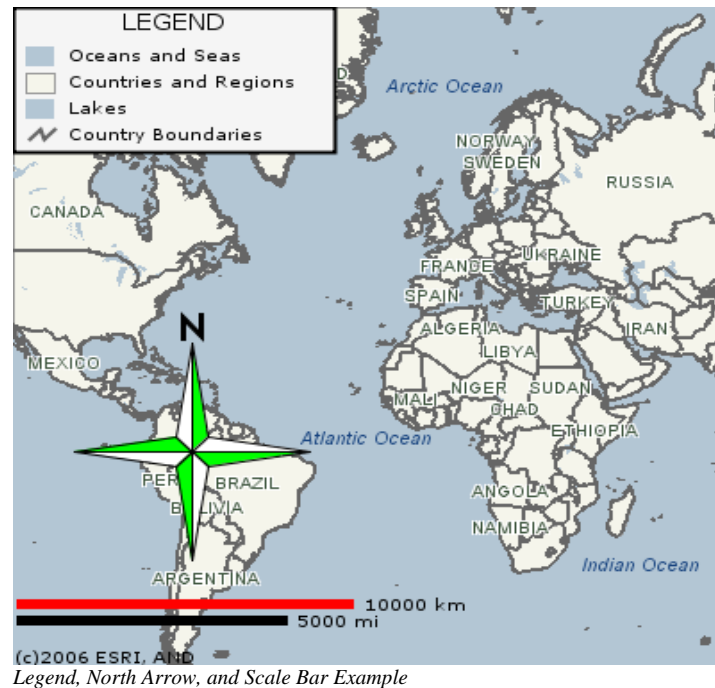
**Table 11
Scale Bar Parameters**

Scale Bar		
Note: You must use all or none of the Scale Bar parameters.		
Parameter	Description	Valid Values
sbx	The lower left corner of the scale bar in pixels, from the bottom left corner of the map.	The x,y coordinate of the scale bar, separated by a pipeline (). For example, to display the scale bar 10 pixels from the left side of the map and 10 pixels from the bottom of the map, set sbx=10 10, where 10 is the x coordinate and 10 is the y coordinate.
sbw	Width in pixels of the scale bar.	Use positive integer.
sbt	Type of scale bar.	Valid values are 1 (transparent) or 2 (color). Default value is 1.

The request below creates a legend, north arrow, and scale bar and results in the map in figure 8.

<http://restproxy.com/restmap?actn=getMap&fmt=svg&ds=bam&sf=200000000&c=-20|30&ocs=8&l=0|0&na=100|130&naw=150&nah=150&nat=gr&sbx=0|20&sbw=200&sbt=2>

Figure 8



SVG Structure for GetMap Action

The SVG is returned from the getMap request in a specific structure. It contains a current map reference, as mentioned before, and the geometry is structured so that multiple styles can be applied to it, such as in street casings. The following document was altered so that you can examine the structure more closely; large parts of the document were replaced with ellipses (. . .).

```

<svg width="400px " height="400px" viewBox="0 0 400 400" xmlns="http://www.w3.org/2000/svg"
version="1.0" xmlns:xlink="http://www.w3.org/1999/xlink">
<desc>
0.0|30.0|0.0|0.0|1.0E8|0.0|4326
</desc>
<title>ESRI Map Server</title>
<defs>
  <g id='L0-0'>
<path d='M0 400v-400h400l0 400 z'/>
  </g>
  <g id='L10-0'>
    <path d='M400 1411-1 -1 -2 -113 -3 z'/>
  </g>
  ...
</defs>

<g id='Oceans and Seas'>
  <use xlink:href='#L0-0' fill-rule='evenodd' fill='rgb(179,198,212)'
stroke='none'/>
</g>
<g id='Lakes'>
  <use xlink:href='#L10-0' fill-rule='evenodd' fill='rgb(179,198,212)'
stroke='none'/>
</g>
...
<g id='TEXT-Oceans and Seas'>
</g>
<g id='TEXT-Lakes'>
</g>
<text x='0' y='399' style='font-size:9px;text-
anchor:start;fill:rgb(0,0,0)'>(c)2006 ESRI, AND</text>
</svg>

```

This is an SVG document that is 400 pixels in width and 400 pixels in height. The view box is also 0 to 400 in both directions, and all locations, widths, and heights are in pixels. Some text classes (styles) are defined within the CDATA section, which can be applied to text in later sections.

All geometry (except acetate geometry) is defined within the <defs> tag. You may also notice that all geometry with common styling is combined within one geometric path. The geometry is combined for performance, as the SVG plug-ins and native SVG engines seem to work much faster with one large geometric object than with many small geometries. All geometry is in pixel coordinates for performance reasons also. Thus you should assume that with any change of scale, a new request is necessary as all map services are scale dependent because of the massive amount of data backing them. The clusters generate the map in a sub-100 millisecond time frame, and the majority of time is spent on the client waiting for the plug-in (or native engine) to render the data.

Styles are applied after the </defs> tag, and each layer is circumscribed here by a <g> element. Any sublayers are likewise styled inside the layer <g> and are themselves delimited by a <g> tag. The id attribute (within the <g>) is very important and identifies whether this layer is a text component of the layer (prefixed by TEXT-) or a vector component (no TEXT- prefix). The id of the vector layer is the actual entry in the legend. One could parse this content and generate a legend or a UI component that turns the layer on and off.

The geometry id attribute L10-0 refers to layer 10 sublayer 0. It is used within the <g id='Lakes'>, which occurs after the </defs>. For example, if this layer had sublayers of large lakes and small lakes, then there

would occur two <g> entries within <g id='Lakes'>, something like <g id='Large'> and <g id='Small'>. This is coded in the following (notice how the geometry is referenced in this example):

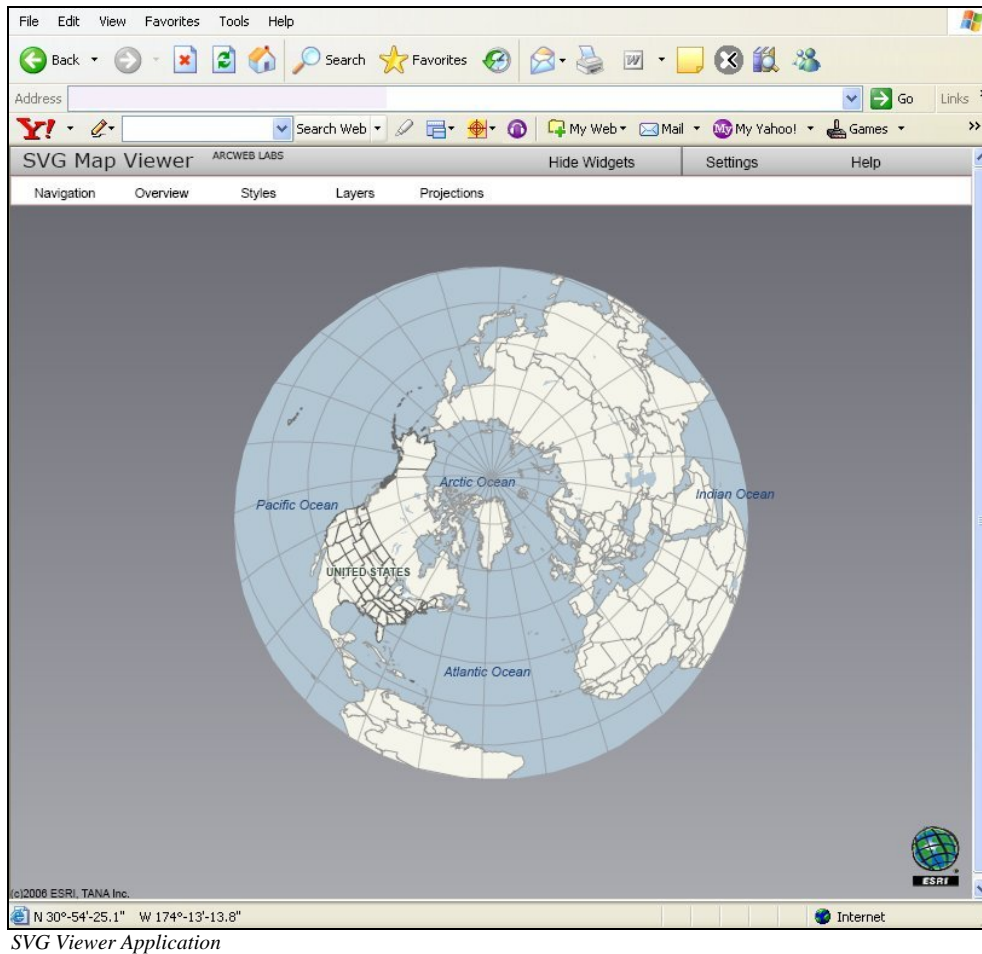
```
...
<defs>
  <g id='L10-0'>
    <path d='M400 1411-1 -1 -2 -113 -3 z' />
  </g>
  <g id='L10-1'>
    <path d='M200 1411-1 -1 -2 -113 -3 z' />
  </g>
...
</defs>
...
<g id='Lakes'>
  <g id='Small'>
    <use xlink:href='#L10-0' . . ./>
  </g>
  <g id='Large'>
    <use xlink:href='#L10-1' . . ./>
  </g>
</g>
...
```

Since the map document is grouped this way, one could write code that could modify the styling of the geometry (by layer) entirely on the client. (The SVG Map Viewer application does not do this.) The text of layers is added after all layers have been drawn, and since the SVG file is drawn in order, text is essentially added after all geometric styling (use) has been specified. Text can have sublayers also, and you can link the text to the geometry (common names) when you parse the content. This is implemented with SVG Map Viewer in the Layer Widget. Note that all layer styles are applied by the <use> element. Acetate geometry is added after the layer style definitions.

SVG Map Viewer Sample Application

The SVG Map Viewer is a free sample application available with source code and can be seen at <http://www2.arcwebservices.com/v2006/solutions/sc.jsp>. The SVG Map Viewer was built in SVG and JavaScript. It uses the REST v2006 API to generate on-demand SVG maps. Panning is provided with a single click (specifies a new map center). You can window in on an area within the map by clicking and dragging to specify a rectangle. There are five map widgets that provide navigation, overview, styles, layers, and projections, as shown in figure 9.

Figure 9



Adding Map Mashups to the SVG Map Viewer Application

All source code is provided with the SVG Map Viewer sample application. This includes projection source for the projection style coordinate systems (0-11). Map mashups are easy to create with the SVG Map Viewer application, and two simple examples are enclosed with the source to illustrate how to do this. You have two choices on how to achieve this: create an HTML file and use interdocument communication that works on a smaller target platform base, or add your code directly to the SVG application and allow it to be somewhat more portable. The example shown uses an HTML file and a JavaScript file. To add it to the SVG application, you must add it to the SVG Map Viewer application itself.

HTML for Mashup

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
<meta http-equiv="content-type" content="text/html; charset=iso-8859-1" />
  <title>SVG Map Viewer</title>
  <meta name="generator" content="BBEdit 7.1.4" />
  <script src="addbigcities.js">
```

```

</script>
<script type="text/javascript">
  <!--
      function openWindow(url,attributes) {
          window.open(url,"newWindow",attributes);
      }
      function setWindowStatus(status) {
          window.status=status;
      }
  //-->
</script>
<style type="text/css">
html{
    height:100%;
    overflow:hidden;
}
body{
    height:100%;
    margin:0;
    padding:0;
}
</style>
</head>
<!--

```

This sample loads the map, then zooms to an area so that all the points added through addbigcities.js are displayed on the map. Click the city icons to view the Web site associated with the city.

```

-->
<!-- calls the function testcase2 defined in addbigcities.js -->
<body onload="TestCase2(window.awsetup.mapControl)">
<center>
<embed src="gtmap.jsp?IE=.svg" type="image/svg+xml" width='100%'
height='100%'
pluginspage="http://www.adobe.com/svg/viewer/install/main.html" />
</center>
</body>
</html>

```

JavaScript Code for Mashup

```

/**
 * This file shows how you can add your own data on the map.
 * It uses the functions defined in map_control.js file that comes with the download.
 *
 * Add an image icon representing a city on the map at xy (decimal degrees).
 * name parameter contains the user-specified city name.
 * website parameter contains user-supplied URL of the Web site associated with city.
 */
function addCity(name, x, y, website){
    //define an icon for the city
    var myIcon = new awgeom.ImageElement(
        {point:new awgeom.Point(x,y),

```

```
        width:30, height:30, url:"star_green.gif", pixelOffset:new
        awgeom.Point(-15, -20), scaleMax:4000000});
var state="Maine";
if (name=="Berlin"){
    state="New Hampshire";
}

//set the data and url for the icon
myIcon.xdata = name+", "+state;
myIcon.website = website;

//handle events for city image icon
myIcon.handleEvent=function(evt){
    if(evt.type=="mouseover")
        //display rollover text (city name) when mouse is rolled
        // over the object that triggered this event.
        awsetup.mapControl.rollOverUI(this.xdata, evt);
    else if(evt.type=="mouseout")
        //Clear the rollover text (city name) when mouse cursor
        // goes out of the icon.
        awsetup.mapControl.rollOutUI(evt);
    else if(evt.type=="mousedown"){
        //Open a new window with the url associated with the object
        //that triggered this event.
        var attributes = "height=600,width=800, toolbar=yes,
            location=yes, directories=yes, status=yes, menubar=yes,
            scrollbars=yes, copyhistory=yes, resizable=yes";
        openWindow(this.website, attributes);
        evt.stopPropagation();
    }
}
//Associate the event to the object that will be handling it
//(or you can specify a function).
//In this case, mouseover, mouseout and mousedown events on the
//myIcon will be handled by myIcon
myIcon.setEvents({mouseover:myIcon, mouseout:myIcon,
    mousedown:myIcon});

//Add image icon to map.
awsetup.mapControl.addUserElement(myIcon);

//Add user-supplied city name text to the map.
awsetup.mapControl.addUserElement(new awgeom.TextElement(
    {point:new awgeom.Point(x,y),
    text:name, style:"stroke:green;fill:green;font-
    style:italic;font-size:12pt;text-anchor:middle",
    pixelOffset:new awgeom.Point(0, 25),scaleMax:2000000}
));
}
/**
 * setup the data
 */
function TestCase22(){
    addCity("Berlin", -71.176714, 44.476319,
```

```

    "http://www.ci.berlin.nh.us/");
addCity("Paris", -70.496479, 44.262212,
    "http://www.city-data.com/city/South-Paris-Maine.html");
addCity("Sweden", -70.65053, 44.094635,
    "http://experts.about.com/e/s/sw/Sweden,_Maine.htm");
addCity("Norway", -70.575423, 44.222529,
    "http://www.norwaymaine.com/");
addCity("Oxford", -70.509772, 44.121355,
    "http://www.oxfordcountyfair.com/");
addCity("Naples", -70.604241, 43.96423,
    "http://www.naplesgolfcourse.com/");
addCity("Denmark", -70.784598, 43.978479,
    "http://history.rays-place.com/me/denmark-me.htm");
addCity("Peru", -70.434312, 44.492236,
    "http://en.wikipedia.org/wiki/Peru,_Maine");
addCity("Mexico",-70.5136,44.5737, "http://www.mexicomaine.net");
addCity("China", -69.5864, 44.3960,
    "http://htusa.webmaine.com/maine/china");
addCity("Lebanon", -70.9084, 43.4243,
    "http://www.raynorshyn.com/meGenWeb/york/lebanon/");
addCity("Rome",-69.7504, 44.4960,
    "http://www.hometownusa.com/maine/Rome.html");
addCity("Athens", -69.6637, 44.9533,
    "http://en.wikipedia.org/wiki/Athens,_Maine");
addCity("Moscow", -69.8823, 45.13483,
    "http://userpages.prexar.com/townofmoscow/");

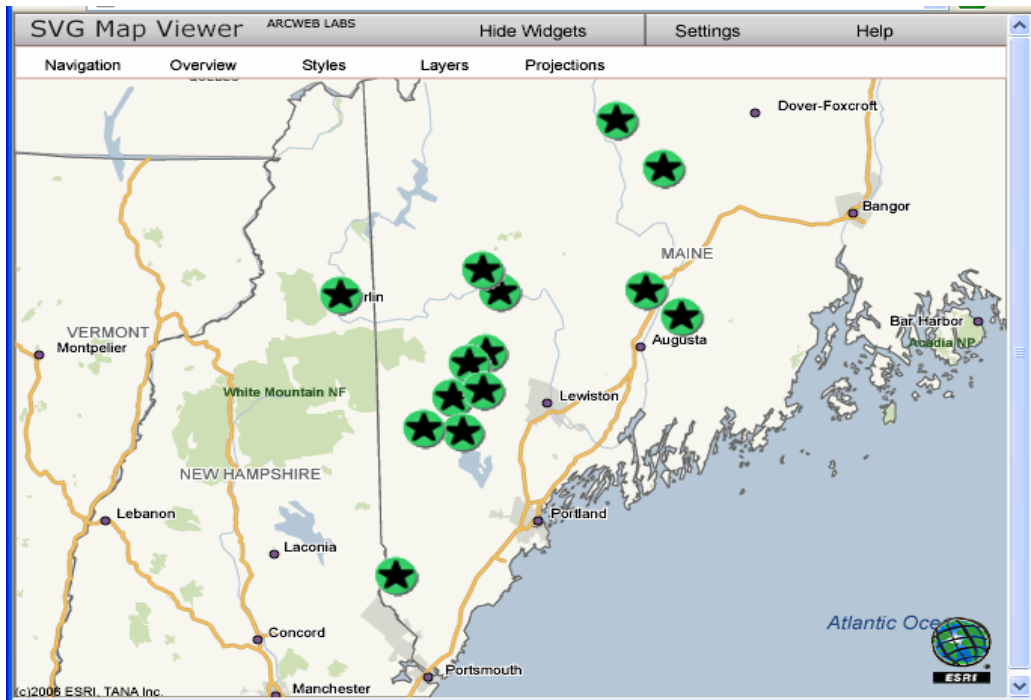
//Get the map envelope that will display all cities specified above.
var myEnvelope = awsetup.mapControl.getUserDataEnvelopeDD();
//Zoom to the envelope and expand it by a factor of 1.5.
awsetup.mapControl.zoomToEnvelopeDD(myEnvelope, 1.5);
}
function TestCase2(){
    //wait for the map to come back
    if (awsetup.mapControl.getRequestsInProgress() != 0){
        setTimeout("TestCase2();", 1000);

    }else{
        TestCase22();
    }
}
}

```

The sample application provides a JavaScript API with which you can add points, lines, and polygons directly with their SVG styling parameters. Additionally, you can add events to these objects, as illustrated in the above example. Internally the JavaScript API has two layers (above the map content): a normal drawing (user) layer and an animated layer. The example (shown in figure 10) does not use the animated layer but instead uses the user data, as the data is static.

Figure 10



Map Mashup Example

Conclusion

ESRI's ArcWeb Services provide SVG map generation services for on-demand mapping. The REST API supplies the SVG developer with many parameter options that will prove useful in mapping applications. The returned SVG offers the developer a means to interact with its display. This is shown in the SVG Map Viewer application at <http://www.arcwebservices.com/v2006/solutions/sc.jsp>.