Understanding ArcWeb℠ Services:
A Developer's Overview to SOAP Implementation

An ESRI® Technical Paper • October 2002

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

Understanding ArcWeb Services: A Developer's Overview to SOAP Implementation

An ESRI Technical Paper

Contents **Page**

Understanding ArcWeb Services: A Developer's Overview to SOAP Implementation

Introduction

A Web service is a software component that can be accessed over the World Wide Web for use in other applications. ESRI's ArcWeb℠ Services are a type of Web service that provides commercially hosted spatial data and geographic information system (GIS) functionality via the Internet to ArcGIS™ and custom Web applications. ArcWeb Services offer a way for developers to include GIS content and capabilities in their applications without having to host the data or develop the necessary tools themselves. The result is significant savings of development time, expense, and computer resources. ArcWeb Services are available as prepackaged collections of related services or as individual services. They are developed using Web protocols so that different computers can easily communicate with each other.

This document provides an overview of ArcWeb Services implemented with Simple Object Access Protocol (SOAP).

ArcWeb Services Overview

ArcWeb Services are Web services that use data and related functionality to perform basic geoprocessing tasks such as address matching, map image display, and routing. As application developers, you can use ArcWeb Services to perform real-time processing on the computers where ArcWeb Services are located and return the results to your local application—all over the Internet. You do not have to maintain GIS application tools or the associated geographic data on your local system to use them in your custom application.

An ArcWeb Service enables developers to integrate mapping and location functionality into their applications without having to host the service or its data locally.

The ArcWeb Services discussed in this paper are developed with SOAP. SOAP standardizes the way a Web service communicates with a client and allows programs

written in different languages and on different platforms to be compatible with each other. SOAP works with standard Web protocols including XML, hypertext transfer protocol (HTTP), and transfer control protocol/Internet protocol (TCP/IP) as well as emerging Web service protocols such as Web Service Description Language (WSDL) files.

Note: Subsequent references to ArcWeb Services imply those implemented with SOAP.

Visit ESRI to Learn About ArcWeb Services

To learn about ESRI's offering of ArcWeb Services, visit http://www.esri.com/software/arcwebservices/index.html .

ArcWeb Services Can Communicate With Any Web-Enabled Application

ArcWeb Services can communicate with any local application that understands XML and is connected to the Web. ArcWeb Services, like many Web services, use SOAP to transfer information back and forth to clients (local applications). SOAP brings together two industry-standard languages for communicating over the Internet: HTTP and XML. ArcWeb Services use SOAP to communicate so they are compatible with some of the latest Web services toolkits. Web services toolkits, such as Microsoft .NET and The Mind Electric GLUE, simplify the implementation of ArcWeb Services because the communication protocol is handled automatically. ArcWeb Services can also communicate directly through SOAP for clients not using Web services toolkits.

ESRI Hosts ArcWeb Services

ArcWeb Services are hosted on the ESRI Internet Mapping System, which is powered by a variety of leading-edge technologies for Internet mapping and has a very high capacity for mapping and location services. It is supported by ESRI staff and consists of two complete and separate configurations maintained at two geographically separate locations in the United States to provide full system redundancy.

Examples of ArcWeb Services

Simple ArcWeb Service: Place Finder Sample Web Service

The Place Finder Sample Web Service is an example of a simple service returning a single response. It takes a single string as its input—"Paris" or "Paris, France," for example—and outputs basic location information.

How the Place Finder Sample Web Service Works

The Place Finder Sample Web Service takes a string with the name of a place and an optional extent parameter. The return is a Location object that contains an array of location parameters with the following content:

- ***x,y* coordinates**—x,y coordinates in decimal degrees of the location.
- **description1**—A complete name for the location (e.g., Redlands, California, United States).
- **description2**—A short name for the location (e.g., Redlands).

- **Score**—A number from 1–20 indicating the importance of the location. Used for ranking.
- **Type**—A letter indicating the type of place, for example, A=countries.
- **locationExtent**—The bounding extent of location.

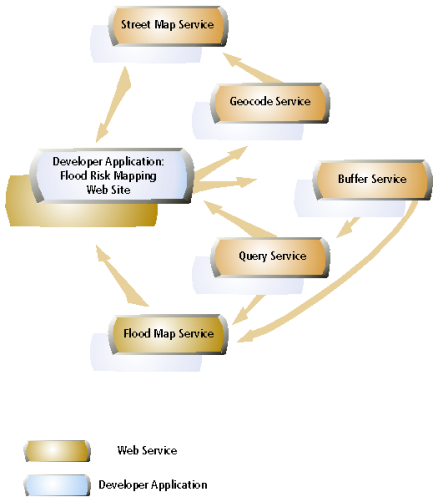
Complete Solution: Flood Risk Mapping

Developers can easily combine several ArcWeb Services to create one complete solution. The Flood Risk Mapping application combines five ArcWeb Services to help users determine the flood risk for a specific location.

How the Flood Risk Mapping Application Works

A user enters a U.S. street address through a basic Web form and receives a map and a report of the potential flood hazard for the address and surrounding area. The following ArcWeb Services are used to determine if the address is located in a flood hazard area and to generate the final map and report.

- **Geocode Service**—Determines the latitude/longitude coordinates of an address.
- **Street Map Service**—Creates a street map centered on the coordinates. User views a map of the area and confirms the point location on the map.
- **Buffer Service**—Creates a 300-foot buffer around a point (based on use guidelines published by the flood data provider).
- **Query Service**—Determines the type of the flood zone. Generates a report that identifies whether the location is in or near a flood hazard zone and, if it is, identifies the zone type.
- **Flood Map Service**—Creates a map of the area that shows flood zones and color-codes the point location based on the flood zone status.



The Flood Risk Mapping Web site uses four ArcWeb Services. Developers can create their own Web site that uses all or some of these ArcWeb Services.

ArcWeb Service Protocols

ArcWeb Services use the latest Web service standards to communicate with local applications. Like Web services, ArcWeb Services are remote objects that can be invoked by another application whenever that application needs it. ArcWeb Services use a certain set of methods or function calls that can be invoked remotely from a client machine to get the desired results. This seamless interchange requires that ArcWeb Services communicate in a way that is understood by any remote application.

This integration is accomplished through two new Internet protocols, SOAP and WSDL. To make this integration easy, ArcWeb Services are compatible with Web service toolkits. ArcWeb Services are published to a Universal Description, Discovery, and Integration (UDDI) registry so developers can quickly discover them.

SOAP

SOAP brings together XML and HTTP by setting protocols for how Web services and their clients communicate with each other. Within a SOAP framework, a client sends a request in XML over HTTP to the Web service, which, in return, sends a response in XML. It does not matter what language is used to create the request or response as long as it is wrapped in XML. This flexibility allows developers to integrate components built in different programming languages.

The Place Finder Sample Web Service provides a good example of how SOAP works. Place Finder Sample Service includes the following method:

```
findPlace (placeName: String):LocationInfo
```

To use this ArcWeb Service, the client invokes the "findPlace" method, passes in a place name, and receives a location for the place.

If a similar "find-a-place" application was stored locally, a client could use a library and invoke one of its objects. The difference with the Place Finder Sample Web Service is that the object is remote and across the Internet. Therefore, something must happen behind the scenes to transfer the findPlace request to the remote object, and the remote object must somehow return the result.

The protocol used behind the scenes is SOAP. SOAP defines a clear mapping between parameters and function calls. A findPlace request looks like this:

```
  <?xml version='1.0' encoding='UTF-8'?>
  <soap:Envelope xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
xmlns:xsd='http://www.w3.org/2001/XMLSchema'
xmlns:soap='http://schemas.xmlsoap.org/soap/envelope/'
xmlns:soapenc='http://schemas.xmlsoap.org/soap/encoding/'
soap:encodingStyle='http://schemas.xmlsoap.org/soap/encoding/'>
    <soap:Body>
      <ns0:findPlace xmlns:ns0='http://arcweb.esri.com/PlaceFinderSample'>
        <placeName xsi:type='xsd:string'>Redlands</placeName>
      </ns0:findPlace>
    </soap:Body>
  </soap:Envelope>
```

Disregarding the wrapping tags, you find that findPlace is a tag and inside it are the parameters of the function, also tags. The Place Finder Sample Web Service listens for this kind of request and returns a response that contains the location information.

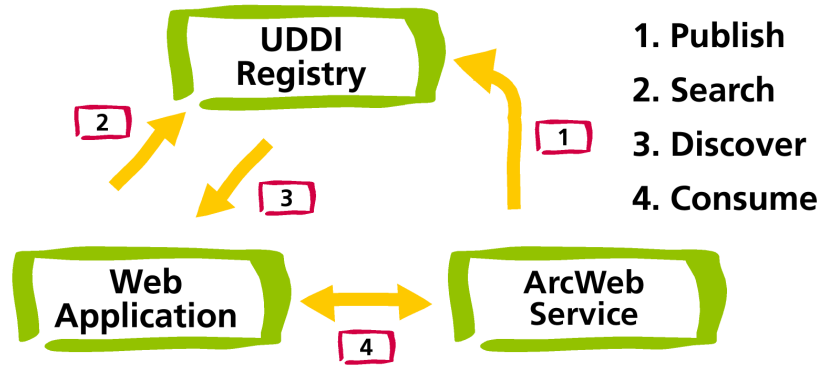
Without SOAP, a client would have to create an XML request, send it in a POST message to a URL over HTTP, then parse the request using an XML parser.

WSDL

ArcWeb Services use another XML syntax (and W3C standard) to communicate with local applications. A WSDL document describes an ArcWeb Service so the client knows what the service does. WSDL defines such things as which methods are available, what their parameters are, and the parameters' types. WSDL documents are most useful if used with a toolkit. Together, WSDL and SOAP allow any application connected to the Web to communicate with an ArcWeb Service.

UDDI

ArcWeb Services are published on the UDDI registry, a universal database of Web services. Developers can search any UDDI site to discover services on the registry, making UDDI a powerful resource for publishers and consumers alike. Discovery happens either through a Web interface (e.g., https://www-3.ibm.com/services/uddi/protect/registry.html) or SOAP calls.



ArcWeb Services are published to the UDDI registry so Web developers can easily discover and consume them.

More Information on SOAP, WSDL, and UDDI

The following Web sites provide overviews of SOAP, WSDL, and UDDI.

- http://www-106.ibm.com/developerworks/webservices/
- http://www.uddi.org/
- http://dcb.sun.com/practices/webservices/
- http://msdn.microsoft.com/webservices

Web Services Toolkits

Web services toolkits integrate disparate systems so they can work together. They are often called frameworks because they are so comprehensive in what they accomplish. A toolkit automatically reads the WSDL document and creates the client code for the service. Some toolkits create a context menu that displays all the possible methods and arguments. Without a toolkit, the developer needs to more fully understand the SOAP exchange between the server and client. The developer needs to construct SOAP requests and make an HTTP post to the server URL. With a toolkit, all you have to do is point the toolkit to the WSDL. The toolkit converts all parameters and requests into a SOAP message, which it sends to the service. It also takes care of converting the returned SOAP message into a usable object.

ESRI tests ArcWeb Services using two popular toolkits. We will test others as they become stable and are readily used by the developer community.

The toolkits tested by ESRI are

- **Microsoft .NET**
- **The Mind Electric GLUE**

Web services toolkit technology is new. For this reason, you may encounter an error when trying to use a toolkit with a particular Web service. Each ArcWeb Service lists compatible toolkits and describes any known issues you may encounter.

Example of Using an ArcWeb Service With a Web Services Toolkit

A Web services toolkit can decrease the amount of work developers have to do to integrate services into their applications. Each toolkit is slightly different in how it works, but the general methodologies and end results are the same. The example below uses the GLUE toolkit (which uses Java) to call the Place Finder Web Service.

The definition of the Place Finder Sample Web Service is found at the following URL:

- http://arcweb.esri.com/services/v1/PlaceFinderSample.wsdl

The WSDL document defines the service as having three methods.

- findPlace (placeName: String): LocationInfo
- findPlace (placeName: String, placeFinderOptions:PlaceFinderOptions): LocationInfo
- getVersion ():String

GLUE comes with an executable called wsdl2java. When this is run and pointed toward the above URL it generates all the classes needed to call the ArcWeb Service. In this case it creates seven classes.

Class File	Description
IPlaceFinderSample.java	The interface that contains the method findPlace and is used to make the Web service call.
PlaceFinderSampleHelper.java	A helper class created by GLUE to instantiate the right objects.
Location.java	A class that defines the location return class.
PlaceFinderSample.map	A mapping class that maps types from XML to Java.
Envelope.java	A class that defines an envelope. Used as an input in PlaceFinderOptions object to restrict the search as well as an output to define the extent of the returned location.
LocationInfo.java	A class that contains a list of Location objects.
PlaceFinderOptions.java	A class that defines different optional parameters to restrict the search .

Once these classes are created, they can be used in your program as follows:

```
…
electric.xml.io.Mappings.readMappings("PlaceFinderSample.map");
IPlaceFinderSample placeFinderSample = PlaceFinderSampleHelper.bind();
LocationInfo locInfo = placeFinderSample.findPlace("Redlands");
…
```

J-8853

The mapping file is loaded, the objects are bound to the service, and the findPlace request is made. The first two lines are set up and only need to be done once; the last can be repeated over and over.

To the program making the request, this looks like any other invocation method. This is because the underlying framework takes care of converting the call and parameters into XML, sending it to the correct service, returning a response in XML, and converting it into the appropriate objects.

Example of Using an ArcWeb Service Without a Web Services Toolkit

You do not have to use a Web services toolkit to use an ArcWeb Service. Most Web services toolkits are relatively new, and some are not fully functional. Developers may be wary of using these new products on their production sites until they feel confident they are stable and compatible with their systems.

Because SOAP is simply an XML protocol, it is fairly straightforward to use the SOAP protocol with an HTTP POST (but more complicated than using a toolkit).

Documentation on the SOAP requests and responses is provided with each ArcWeb Service so developers not using a toolkit will know how to call the service directly. Once the SOAP inputs and outputs are known, a legal request can be crafted, all the parameters can be inserted correctly, and a request can be sent to the ArcWeb Service.

Once an XML request is created, the request needs to be sent using the POST method. This is much like a form posted from a regular browser page. The difference is that a SOAP service also expects to see a special header called "SOAPAction."

Here is an example in Java of a routine that sends a SOAP request and parses the response.

```
// Create SOAP Message
String soapRequest = "<?xml version='1.0' encoding='UTF-8'?>..."
…
// Open a URL to the ArcWeb Service
java.net.URL url = new
java.net.URL("http://arcweb.esri.com/services/v1/PlaceFinderSample");
java.net.URLConnection conn = url.openConnection();
conn.setUseCaches(false);
conn.setDoInput(true);
conn.setDoOutput(true);

// Set the SOAPAction header to tell the server that this is a SOAP
request conn.setRequestProperty("SOAPAction", "");

// Post the SOAP Message
java.io.DataOutputStream dos = new java.io.DataOutputStream( new
java.io.BufferedOutputStream( conn.getOutputStream()));
dos.writeBytes( soapRequest.toString());
dos.flush();
dos.close();

// Parse the reponse using an XML parser
com.sun.xml.tree.XmlDocument xmlDocument =
com.sun.xml.tree.XmlDocument.createXmlDocument( new
java.io.BufferedInputStream(conn.getInputStream()), false);
org.w3c.dom.Element parent = xmlDocument.getDocumentElement();
System.out.println(parent.toString());
```

Here is the same example using Visual Basic with the Microsoft XML Parser.

```
Dim pXml As MSXML.XMLHTTPRequest
Dim PostStr, TextStr As String

Set pXml = New MSXML.XMLHTTPRequest

' Create the SOAP Message
PostStr = "<?xml version='1.0' encoding='UTF-8'?>"
…

' Setup a POST to the ArcWeb Service URL
pXml.Open "POST", "http://arcweb.esri.com/services/v1/PlaceFinderSample",
False

' Add the SOAPAction header (can be empty or contain the URL of the
Service)
pXml.setRequestHeader "SOAPAction",
"http://arcweb.esri.com/services/v1/PlaceFinderSample"
pXml.send (PostStr)

' Read response back into XML parser
Set pDomDocument = pXml.responseXML
```

Once you know the SOAP request and response syntax, using the ArcWeb Service should be straightforward. You do not need to use a Web services toolkit to access an ArcWeb Service. The primary advantage of using a Web services toolkit is that it parses and generates all SOAP messages and thus makes development a quicker process.

Service Types

ESRI offers two types of ArcWeb Services.

- Public ArcWeb Services—Accessible to anyone who accepts the licensing agreement. Restrictions on how the service can be used may apply.
- Restricted ArcWeb Services—Restricted to users who have permission to access the service. Some services are encrypted due to the sensitive nature of the information contained in them.

Procedures for using the two types of ArcWeb Services are slightly different.

Public ArcWeb Services

No authentication steps are required. Users must agree on the intended use of the service. For example, the Place Finder Sample Service may be used freely in low-volume, noncommercial Internet applications. It cannot be resold as is or as part of a custom application.

Restricted ArcWeb Services

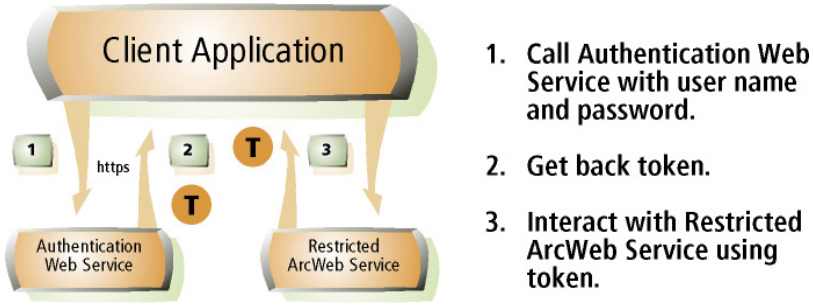
Only authorized users have access to Restricted ArcWeb Services. These users identify themselves with a user name and password. Authentication is done through the Authentication Web Service.

Restricted ArcWeb Services use a token to avoid the security implications of sending a user name and password along with each request. The first time a client accesses a Restricted ArcWeb Service, it must call a separate Authentication Web Service with a user name and password over a secure hypertext transfer protocol (HTTPS) secure sockets layer (SSL) connection. These security measures minimize the risk that an unauthorized user can access a Restricted ArcWeb Service by "stealing" valid login

information. If the client sends valid user name and password, the Authentication Web Service returns a token. The token is a binary encoded string that client application uses to call a Restricted ArcWeb Service.

Each time an ArcWeb Service is called, the token must be passed in. This token has a default time-out, after which a new token must be requested from the Authentication Web Service. The relatively short time-out helps minimize the possibility of having the token stolen. A token with a longer time-out may be requested in certain situations.

Some Restricted ArcWeb Services require additional security because of the highly sensitive nature of the data the ArcWeb Services provide. These services work over HTTPS and SSL so none of the information passed between client and server can be intercepted or decoded. Each ArcWeb Service is available over HTTP and HTTPS. Because more processing occurs with HTTPS connections, performance is not as fast as with HTTP connections. If security is critical to your application, you should access ArcWeb Services over HTTPS. Otherwise, use HTTP for optimal performance.



Client applications access a Restricted ArcWeb Service with a token from the Authentication Web Service.