Using the Master Address Repository (MAR) Web Services

Office of the Chief Technology Officer

August 2017

Introduction

The MAR Web Services are a series of APIs which primarily call DC Government's Master Address Repository (MAR). These operations return a wealth of information about locations in Washington, DC. These locations include addresses, blocks, intersection, place names. We offer XML based returns and JSON. These APIs are available to all users. There are 68 operations. The APIs are behind a large number of applications both inside and outside DC Government. These APIs are free to use both inside and outside of government. By using the MAR Web Services, users can take advantage of DC's authoritative data sources including the best available source of addressing and street network. The data behind the MAR Web Services is updated on a regular basis.

The MAR Web Services is currently being use by DC Public Schools, Metropolitan Police Department, DC Department of Transportation, Office of Planning and many other DC Government agencies. It is also being used by organizations outside of DC Government.

MAR Web Services can Answer:

- Is this City of Washington address valid?
- Does this address contain residences?
- Which Ward is this address located in?
- Which Ward is this hundred block in?
- Which Ward is this intersection in?
- What are the coordinates of this intersection?
- Which Police Service Area is this address in?
- What is the Latitude and Longitude of the Washington Monument?
- What is the address of the Washington Monument?
- What is the address of WASHINGTON NAVY YARD BUILDING 184?
- Are the condo units at this address?
- What is the property identifier for this condo unit?
- How many apartments are there at this address?
- What is the address of a DC Recreation Center?
- What properties identifiers are associated with this address?
- What are all the apartment numbers at this address?
- What are the names of all street in SE that start with the letter T?
- What is the nearest address in DC to these spatial coordinates?
- Give me all the addresses that start with these 4 characters?

There are many other types of questions that can be answered using the MAR Web Services.

Using the MAR Web Services in a Practical Manner

There is an operation name such as findAID and also a findAID2. The operation with the number 2 at the end (ex: findAID2) allows for returns in a JSON format (as the default). The operation without the number 2 (findAID) only returns XML.

Often software developers will need to combine different operations in order to get the information that they need. Many of the applications currently using the MAR Web Services, are calling many operations.

In the appendix section of this document there is some sample code in multiple request formats.

With any serialization format, understand that DC GIS might add fields to the API response from time to time. Your code should be able to support the addition of data/fields in our response without breaking; that is one of the benefits of the JSON interchange format.

Some records will have some of their attributes returned as 'NA' which mean Not Applicable.

Keeping Up to Date on Changes

Keeping informed on changes to the the MAR Web Services is critical. Go to our DC GIS Services page (http://octo.dc.gov/service/dc-gis-services) and click on 'Sign up for DC GIS Updates'. Having multiple people who are responsible for the application sign up is even better. The DC GIS updates include changes to DC GIS data, APIs and more.

Usage Limits / Scraping

There are no set usage limits on the MAR Web Services.

However to avoid flooding our server with requests, there are a few things to consider when developing applications with the MAR web service:

- When you need to send large batches of addresses, you should use the **findLocationBatch/2** operations – this will allow you to send addresses in chunks of 1,000. This minimizes impact on our servers and is much quicker. Yesterday's batch would have been finished in a fraction of the time.
- Hold large jobs for processing **off-hours** (11 PM to 5 AM)
- Program in a **3 second pause** between geolocate requests to let our server accept other requests

Over the past few years, we've seen a sharp increase in the proliferation of bots and scrapers. These are applications that crawl web pages and services to extract information. Many of them are harmless, and some are even beneficial (e.g., Google's bot helps our pages show up in

search results). But poorly-coded bots and scrapers can hurt the performance of the site/service they're crawling by mimicking what is known as a Denial-of-Service attack, in which a web site is so overrun by traffic that it freezes and can't respond to normal user interaction.

Most of the aggressive scraping we see is on the MAR web service.

Am I aggressively scraping a web service?

If your application is accessing a service or page thousands of times with no pause in between then your code can impact the operation of our web servers, as described above. We are especially concerned if the impact occurs during DC business hours (8:00 am to 6:00 pm EST) as our web services are used in many core business operations, both within District government and outside.

So what can I do to use DC web services responsibly?

In general, these bot and scraper best practices will mitigate impacts on our server:

- Use a 3 second delay between requests to allow the server to "breathe"
- Make sure your app requests and respects our robots.txt file
- Leave larger volume processing to off-peak times, or 11:00 pm to 6:00 am EST
- If you are using the MAR web service, and have a large number of records to geocode, you can send them in batches of 1000 using findLocationBatch (for xml) or findLocationBatch2 (for json).

What if I don't adopt these practices?

To protect our shared investment, OCTO monitors these services and takes action make sure that they remain available to all. If we see you using our services in a way that meets the above description of "aggressive" then we may block your operation from completing and/or your IP address from making further calls.

API Key

There is not an API Key for any of the operations.

Anticipate Timeouts

Timeouts happen. Timeouts can happen because of network issues, severed fiber optic cables, dropped packets and animals chewing on the power cables, or any of many uncontrollable reasons. Users you can plan for how they want to handle them.

Support

Other useful documents related to the Master Address Repository (MAR).

- MAR Frequently Asked Questions (FAQ)
- MAR Address Standards
- MAR Data Dictionary
- MAR Presentation

All these above documents can be found at the MAR Webpage:https://octo.dc.gov/node/715602

To view all the operations that are part of the MAR Web Services:

http://citizenatlas.dc.gov/newwebservices/locationverifier.asmx

For questions please contact DC GIS: dcgis@dc.gov

Appendix

Using MAR Web Service at Client Side

The MAR Web Service returns XML and JSON (function name with 2 at the end) format results. It can work with any client side technologies: JavaScript, ActionScript etc.
Here we provide an XML and a JSON example to show you how to work with the MAR Web Service at the client side. If you are using client-side tools, you will need a web Proxy to work with our service refer to ESRI's GitHub for how to set up your proxy.

XML Example

The general XMLHttpRequest was used to accomplish this task.

```
/* Common values for the ReadyState of the XMLHttpRequest object */
var READYSTATE_UNINITIALIZED = 0;
var READYSTATE_LOADING = 1;
var READYSTATE_LOADED = 2;
var READYSTATE_INTERACTIVE = 3;
var READYSTATE_COMPLETE = 4;
/* Common values for HTTP status codes */
```

```
var HTTPSTATUS_OK = 200;

function CreateXmlHttpRequestObject() {
	var xmlObj;
	if (window.ActiveXObject) {
	try {
		xmlObj = new ActiveXObject("Microsoft.XMLHTTP");
	} catch (e) {
		xmlObj = new ActiveXObject("Msxml2.XMLHTTP");
	}
	}
	else
	xmlObj = new XMLHttpRequest();

	return xmlObj;
}
```

```
function verifyAddress(inAddress) {
	var xmlHttpObj = CreateXmlHttpRequestObject();
	if (xmlHttpObj) {
	// We want this request asynchronous
	xmlHttpObj.open("POST",
"https://citizenatlas.dc.gov/newwebservices/locationverifier.asmx?op=findLocation", true);

	xmlHttpObj.onreadystatechange = function () {
		if (xmlHttpObj.readyState == READYSTATE_COMPLETE) {
		if (xmlHttpObj.status == HTTPSTATUS_OK) {
			var doc = xmlHttpObj.responseXML;
			var nodesType = doc.getElementsByTagName("sourceOperation");
			var queryType = nodesType[0].childNodes[0].nodeValue;
			var nodesX = doc.getElementsByTagName("LATITUDE");
			var nodesY = doc.getElementsByTagName("LONGITUDE");
			var nodesName;
			switch (queryType) {
			case "DC Place":
				nodesName = doc.getElementsByTagName("ALIASNAME");
				break;
			case "DC Intersection":
				nodesName = doc.getElementsByTagName("FULLINTERSECTION");
				break;
			case "DC Address":  //case "DC AID":
				nodesName = doc.getElementsByTagName("FULLADDRESS");
				break;
			case "DC Block":
```

```
                        case "DC Block Address":
                                nodesName = doc.getElementsByTagName("FULLBLOCK");
                                break;
                        default: break;

                        }
                        var confidences = doc.getElementsByTagName("ConfidenceLevel");

                        if (!nodesName) {
                        //MAR web service system is experiencing difficult at this moment
                          return;
                        }
                        else if (nodesName.length == 0) {
                        //No any match found
                        return;
                        }
                        else if (nodesName.length == 1 && confidences[0].childNodes[0].nodeValue
== "100") {
                        //one perfect match found
                        var addr = nodesName[0].childNodes[0].nodeValue;
                        var x = nodesX[0].childNodes[0].nodeValue;
                        var y = nodesY[0].childNodes[0].nodeValue;
                        return;
                        } else {
                        var addr, x, y;
                        //found one non-perfect match or found multiple matches
                        for (var i = 0; i < nodesName.length; i++) {
                                addr = nodesName[i].childNodes[0].nodeValue;
                                x = nodesX[i].childNodes[0].nodeValue;
                                y = nodesY[i].childNodes[0].nodeValue;
                        }
                        }
                }
                }
        }
        xmlHttpObj.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
        xmlHttpObj.setRequestHeader("SOAPAction", "str");
        xmlHttpObj.send("str=" + inAddress);
        }
}
```

JSON Example

The function name ends with “2” has the capability to handle XML and JSON returns by specify the “f” parameter.

Following example we adopt the JQuery and AJAX to handle JSON returns.

```
function verifyAddress(inAddress) {
	var marURL =
"https://citizenatlas.dc.gov/newwebservices/locationverifier.asmx?op=findLocation2";

	$.ajax({
	type: "POST",
	contentType: "application/json; charset=utf-8",
	url: marURL,
	data: '{f:"json", str:"' + inAddress + '"}',
	dataType: "json",
	async: true,
	success: function (data, textStatus) {
		parseMARReturn(data);
	},
	error: function (data, status, error) {
		alert("error");
	}
	});
}

function parseMARReturn(response) {
	var nodesType = response.sourceOperation;
	var nodesLat = [], nodesLng = [], nodesX = [], nodesY = [], nodesName = [], confidences
= [];
	//parse returns
	for (var i = 0, ic = response.returnDataset.Table1.length; i < ic; i++) {
	var candidate = response.returnDataset.Table1[i];
	switch (nodesType) {
		case "DC Place":
		  nodesName.push(candidate.ALIASNAME);
		nodesX.push(candidate.XCOORD);
		nodesY.push(candidate.YCOORD);
		  nodesLat.push(candidate.LATITUDE);
		  nodesLng.push(candidate.LONGITUDE);
		  confidences.push(candidate.ConfidenceLevel);
		break;
		case "DC Intersection":
		  nodesName.push(candidate.FULLINTERSECTION);
		nodesX.push(candidate.REFX);
		nodesY.push(candidate.REFY);
		  nodesLat.push(candidate.LATITUDE);
		  nodesLng.push(candidate.LONGITUDE);
		  confidences.push(candidate.ConfidenceLevel);
```

```
                break;
                case "DC Address":
                  nodesName.push(candidate.FULLADDRESS);
                nodesX.push(candidate.XCOORD);
                nodesY.push(candidate.YCOORD);
                  nodesLat.push(candidate.LATITUDE);
                  nodesLng.push(candidate.LONGITUDE);
                  confidences.push(candidate.ConfidenceLevel);
                break;
                case "DC Block":
                case "DC Block Address":
                  nodesName.push(candidate.FULLBLOCK);
                  nodesX.push(candidate.CENTROIDX);
                  nodesY.push(candidate.CENTROIDY);
                  nodesLat.push(candidate.LATITUDE);
                nodesLng.push(candidate.LONGITUDE);
                  confidences.push(candidate.ConfidenceLevel);
                break;
                case "DC AID":
                  nodesName.push(candidate.FULLADDRESS);
                nodesX.push(candidate.XCOORD);
                nodesY.push(candidate.YCOORD);
                  nodesLat.push(candidate.LATITUDE);
                  nodesLng.push(candidate.LONGITUDE);
                  confidences.push(candidate.ConfidenceLevel);
                break;
                default: break;
        }
        }

        //process parsed results
        if (nodesName.length == 0) {
        alert("No matches found");
        }
        else if (nodesName.length == 1 && confidences[0] === "100") {
        //found one perfect match
        alert(nodesName[0] + "_" + nodesLat[0] + "_" + nodesLng[0] + "_" + nodesX[0] + "_" +
nodesY[0] + "_" + confidences[0]);
        }
        else {
        //found one non-perfect match or found multiple matches
        alert("found one non-perfect match or multiple matches");
        }
}
```

Using MAR Web Service at Server Side

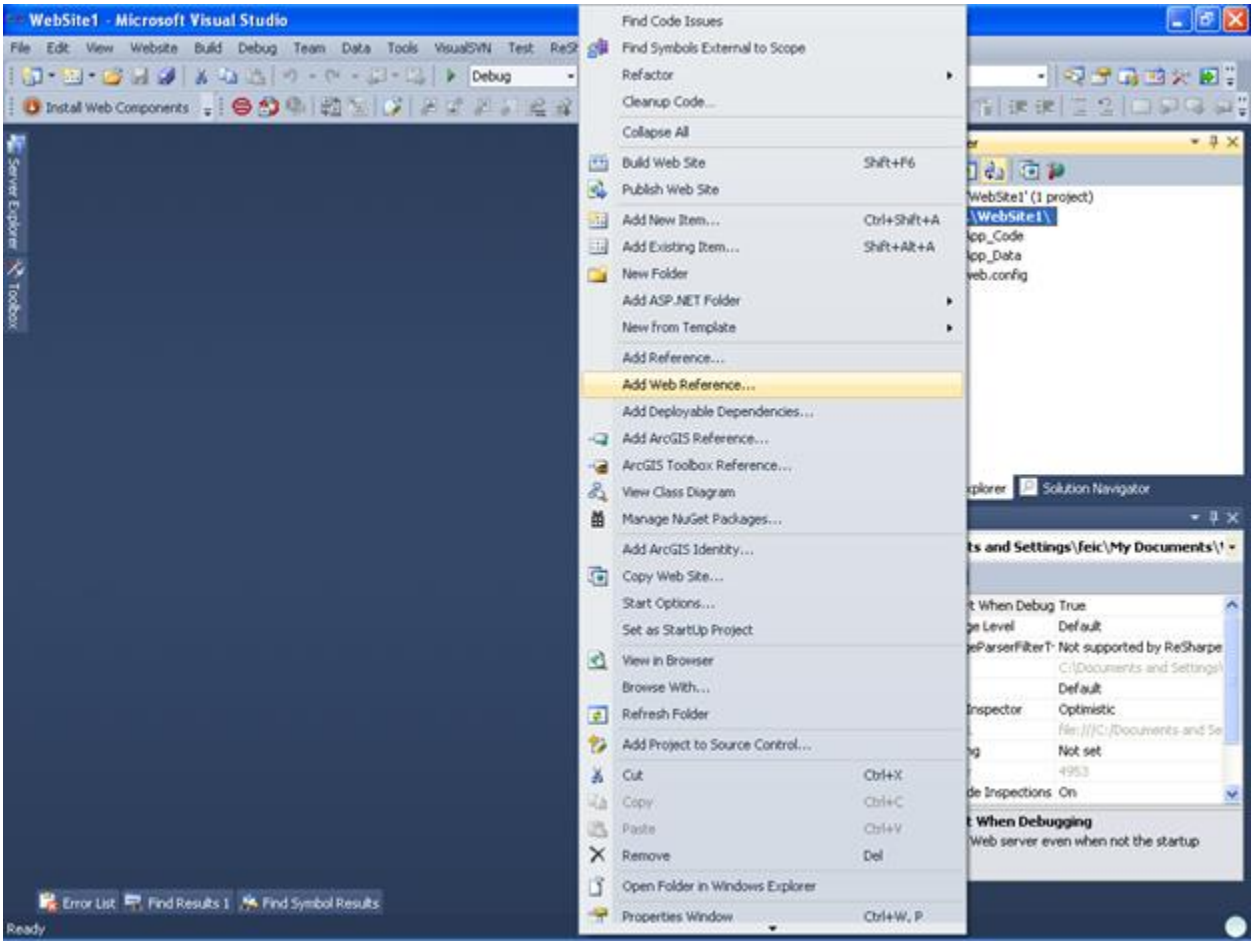
MAR Web Service was developed on .NET ASMX web service technologies. It can seamlessly integrate with any .NET project. The return result from MAR Web Service is

System.Data.DataTable. Developers can easily navigate this object to extract any info of interest to them.
MAR Web Service also provides the SOAP interface. ther server side platforms, like Java, can work with MAR via the SOAP protocol.

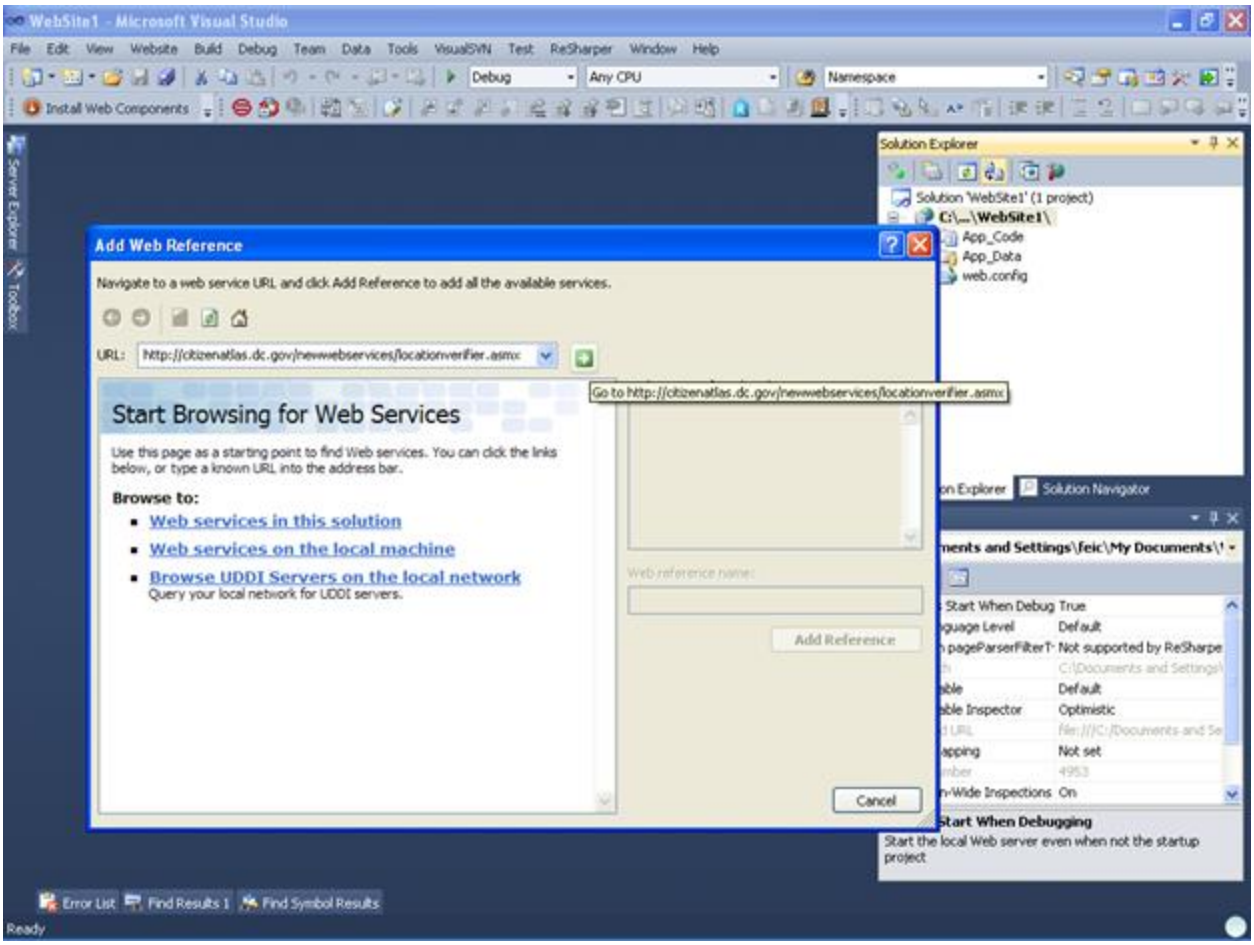
To use MAR Web Service in your .NET project, please simply follow these steps:
1). Add the web reference of the MAR Web Service by pointing to the SOAP WSDL;
2). Call any functions provided by MAR Web Service;
3). Handle the return results.

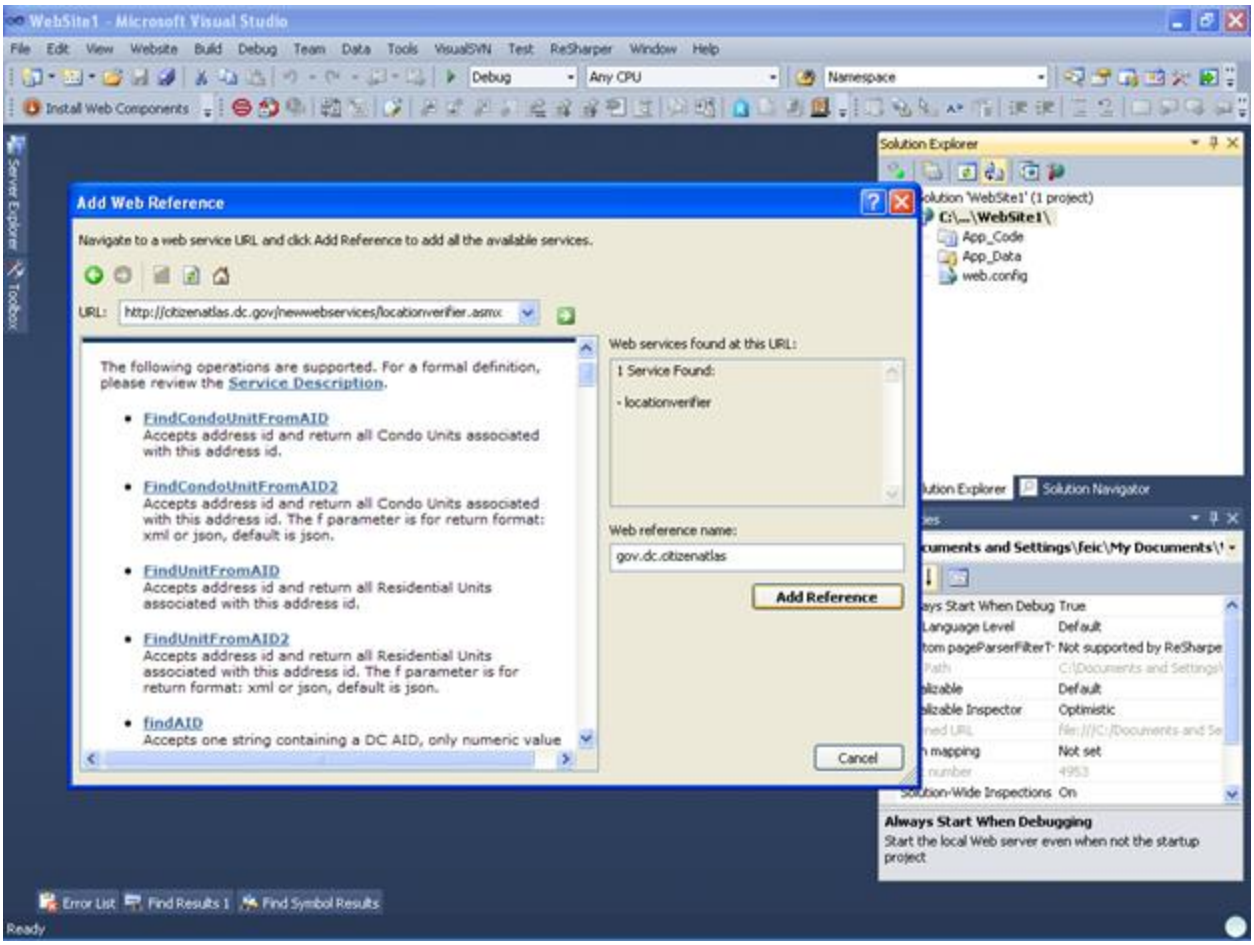
An example Visual Studio 2010 Project is provided as a reference or starting point.

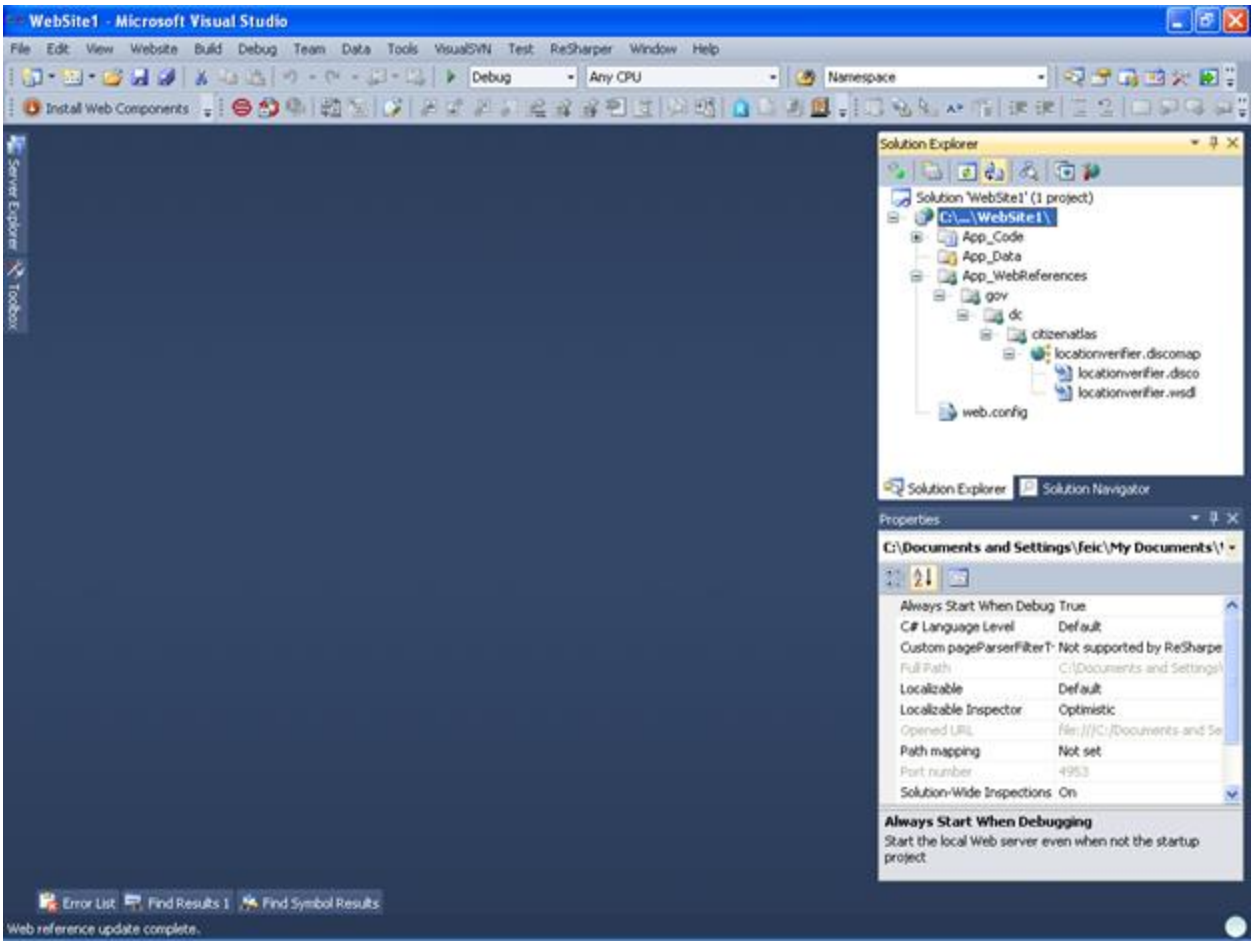
Step 1:
Adding MAR web service as Web Reference to your project:



Input the MAR web service URL:
https://citizenatlas.dc.gov/newwebservices/locationverifier.asmx


WebSite1 - Microsoft Visual Studio
File Edit View Website Build Debug Team Data Tools VisualSVN Test ReSharper Window Help
Debug
Any CPU
Namespace
Install Web Components
Solution Explorer
Solution 'WebSite1' (1 project)
C:\...\WebSite1\
App_Code
App_Data
web.config
Add Web Reference
Navigate to a web service URL and click Add Reference to add all the available services.
URL:
http://citizenatlas.dc.gov/newwebservices/locationverifier.asmx
Go to http://citizenatlas.dc.gov/newwebservices/locationverifier.asmx
Start Browsing for Web Services
Use this page as a starting point to find Web services. You can click the links below, or type a known URL into the address bar.
Browse to:
Web services in this solution
Web services on the local machine
Browse UDDI Servers on the local network
Query your local network for UDDI servers.
Web reference name:
Add Reference
Cancel
Solution Navigator
Start When Debug True
Language Level Default
Not supported by ReSharper
Default
Optimistic
Not set
Start When Debugging
Start the local Web server even when not the startup project
Error List
Find Results 1
Find Symbol Results
Ready


WebSite1 - Microsoft Visual Studio
File Edit View Website Build Debug Team Data Tools VisualSVN Test ReSharper Window Help
Debug
Any CPU
Namespace
Install Web Components
Add Web Reference
Navigate to a web service URL and click Add Reference to add all the available services.
URL:
http://citizenatlas.dc.gov/newwebservices/locationverifier.asmx
The following operations are supported. For a formal definition, please review the Service Description.
FindCondoUnitFromAID
Accepts address id and return all Condo Units associated with this address id.
FindCondoUnitFromAID2
Accepts address id and return all Condo Units associated with this address id. The f parameter is for return format: xml or json, default is json.
FindUnitFromAID
Accepts address id and return all Residential Units associated with this address id.
FindUnitFromAID2
Accepts address id and return all Residential Units associated with this address id. The f parameter is for return format: xml or json, default is json.
findAID
Accepts one string containing a DC AID, only numeric value
Web services found at this URL:
1 Service Found:
- locationverifier
Web reference name:
gov.dc.citizenatlas
Add Reference
Cancel
Solution Explorer
Solution 'WebSite1' (1 project)
C:\...\WebSite1\
App_Code
App_Data
web.config
Solution Navigator
Always Start When Debugging
Start the local Web server even when not the startup project
Error List
Find Results 1
Find Symbol Results
Ready

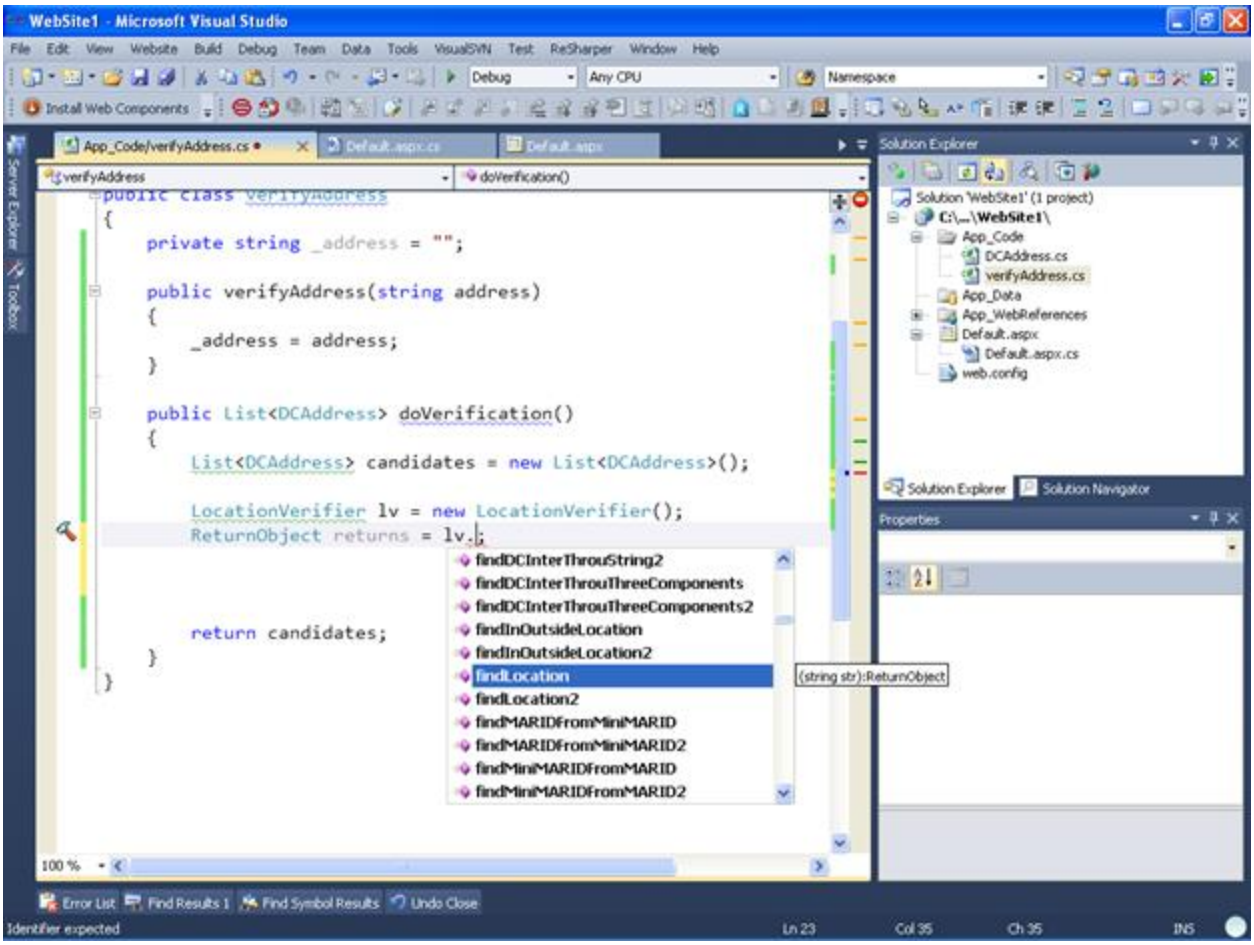


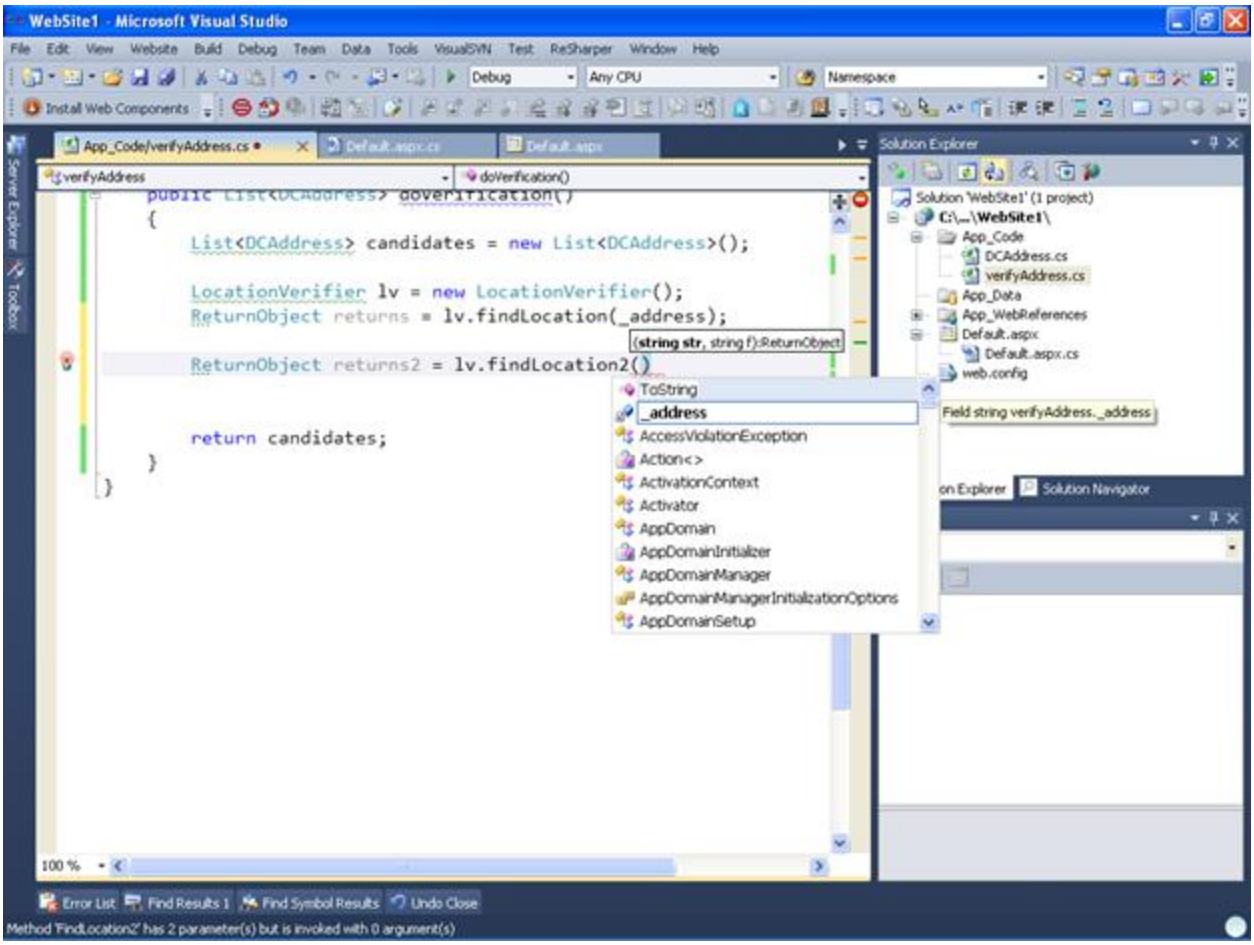
Step 2: Calling any functions provided by MAR Web Service

Importing the MAR web service into your code:

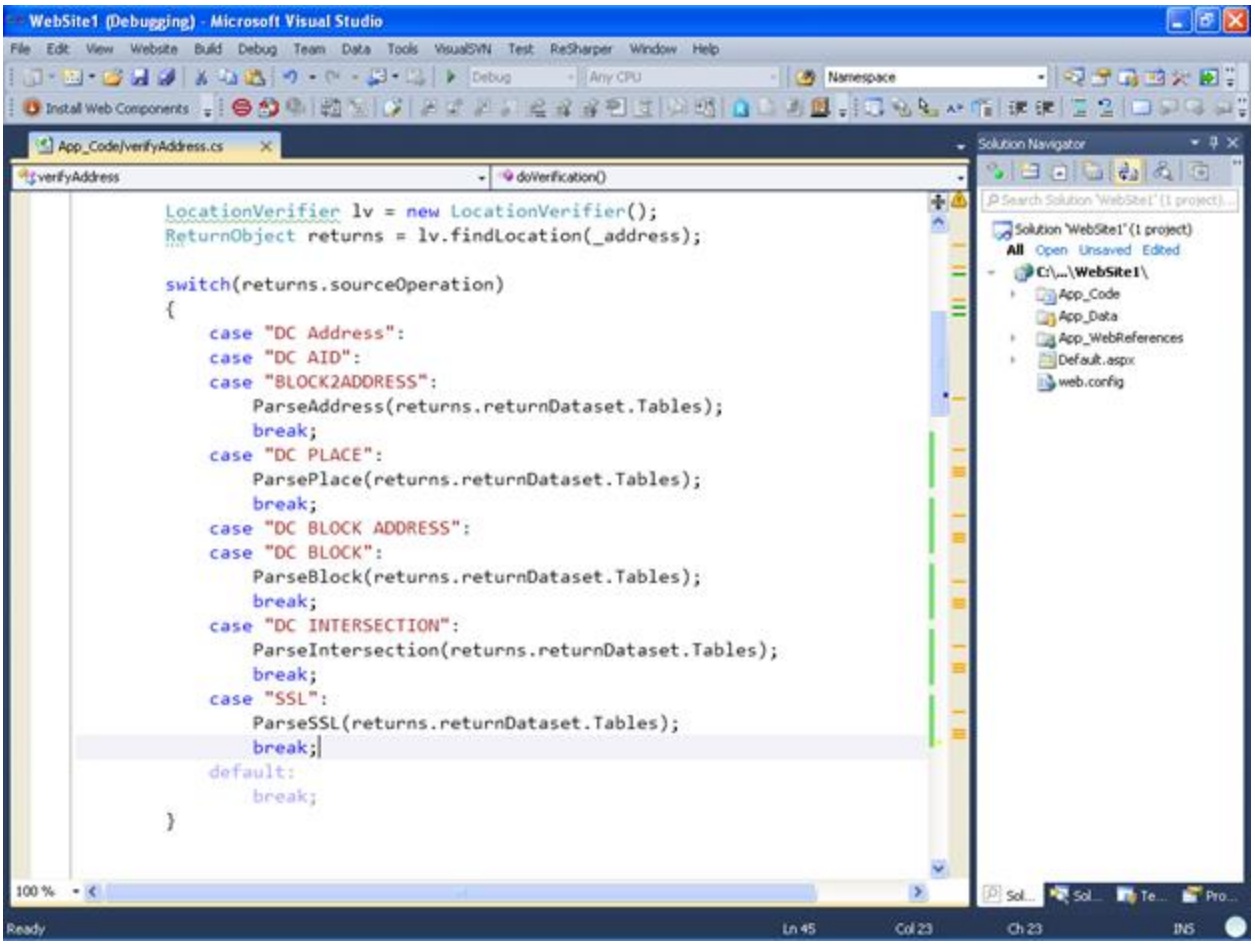
```
using gov.dc.citizenatlas;
```

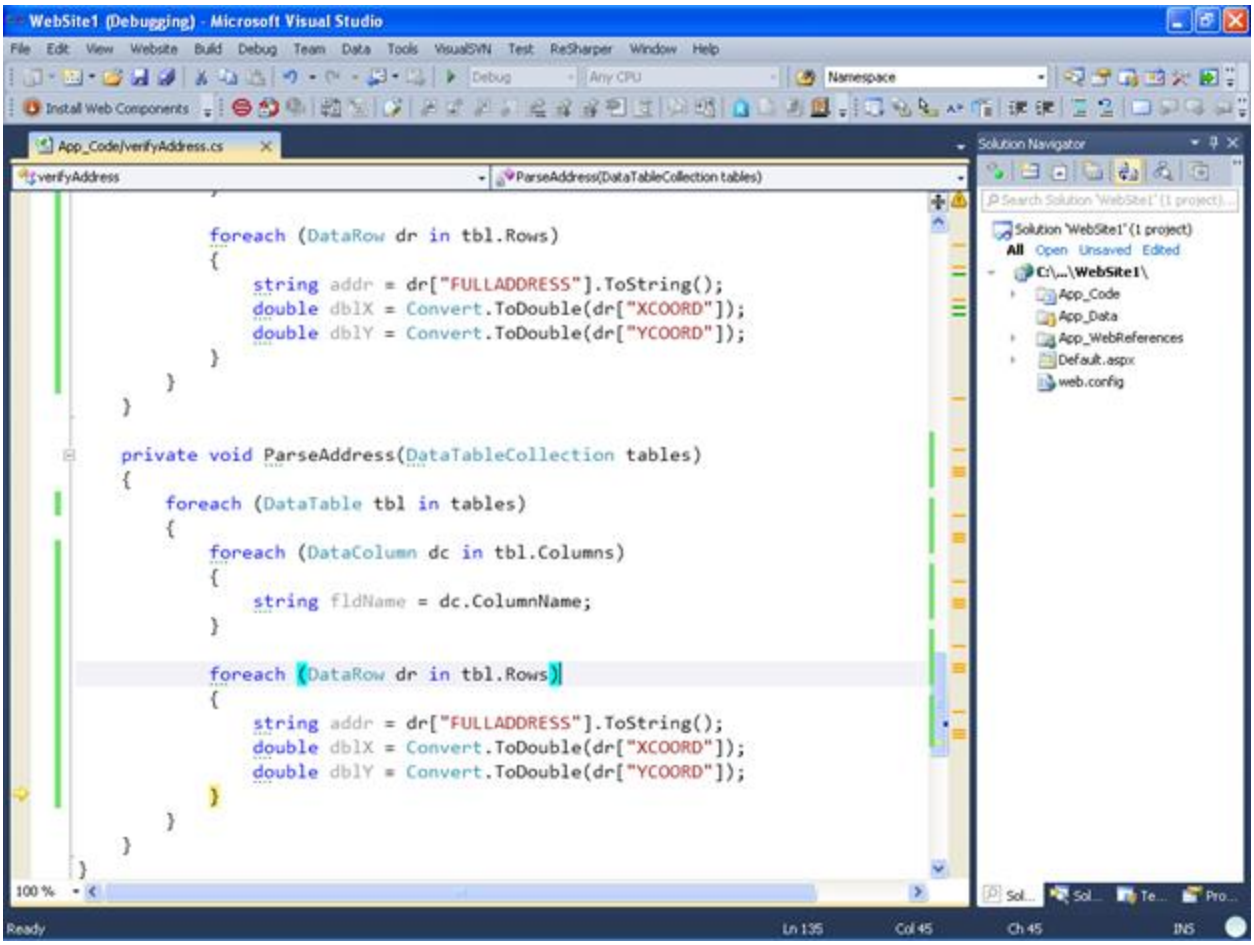
Using the MAR web service in your code:


WebSite1 - Microsoft Visual Studio
File Edit View Website Build Debug Team Data Tools VisualSVN Test ReSharper Window Help
Debug
Any CPU
Namespace
Install Web Components
App_Code/verifyAddress.cs
Default.aspx.cs
Default.aspx
verifyAddress
doVerification()
public class verifyAddress
{
private string _address = "";
public verifyAddress(string address)
{
_address = address;
}
public List<DCAddress> doVerification()
{
List<DCAddress> candidates = new List<DCAddress>();
LocationVerifier lv = new LocationVerifier();
ReturnObject returns = lv.;
return candidates;
}
}
findDCInterThrou5tring2
findDCInterThrouThreeComponents
findDCInterThrouThreeComponents2
findInOutsideLocation
findInOutsideLocation2
findLocation
findLocation2
findMARIDFromMiniMARID
findMARIDFromMiniMARID2
findMiniMARIDFromMARID
findMiniMARIDFromMARID2
(string str):ReturnObject
Solution Explorer
Solution 'WebSite1' (1 project)
C:\...\WebSite1\
App_Code
DCAddress.cs
verifyAddress.cs
App_Data
App_WebReferences
Default.aspx
Default.aspx.cs
web.config
Solution Explorer
Solution Navigator
Properties
100 %
Error List
Find Results 1
Find Symbol Results
Undo Close
Identifier expected
Ln 23
Col 35
Ch 35
INS



Step 3 Handling the return results:


WebSite1 (Debugging) - Microsoft Visual Studio
File Edit View Website Build Debug Team Data Tools VisualSVN Test ReSharper Window Help
Debug
Any CPU
Namespace
Install Web Components
App_Code/verifyAddress.cs
verifyAddress
doVerification()
LocationVerifier lv = new LocationVerifier();
ReturnObject returns = lv.findLocation(_address);
switch(returns.sourceOperation)
{
case "DC Address":
case "DC AID":
case "BLOCK2ADDRESS":
ParseAddress(returns.returnDataset.Tables);
break;
case "DC PLACE":
ParsePlace(returns.returnDataset.Tables);
break;
case "DC BLOCK ADDRESS":
case "DC BLOCK":
ParseBlock(returns.returnDataset.Tables);
break;
case "DC INTERSECTION":
ParseIntersection(returns.returnDataset.Tables);
break;
case "SSL":
ParseSSL(returns.returnDataset.Tables);
break;
default:
break;
}
100 %
Solution Navigator
Search Solution 'WebSite1' (1 project)...
Solution 'WebSite1' (1 project)
All Open Unsaved Edited
C:\...\WebSite1\
App_Code
App_Data
App_WebReferences
Default.aspx
web.config
Sol...
Sol...
Te...
Pro...
Ready
Ln 45
Col 23
Ch 23
INS


WebSite1 (Debugging) - Microsoft Visual Studio
File Edit View Website Build Debug Team Data Tools VisualSVN Test ReSharper Window Help
Install Web Components
App_Code/verifyAddress.cs
verifyAddress
ParseAddress(DataTableCollection tables)
foreach (DataRow dr in tbl.Rows)
{
string addr = dr["FULLADDRESS"].ToString();
double dblX = Convert.ToDouble(dr["XCOORD"]);
double dblY = Convert.ToDouble(dr["YCOORD"]);
}
}
}
private void ParseAddress(DataTableCollection tables)
{
foreach (DataTable tbl in tables)
{
foreach (DataColumn dc in tbl.Columns)
{
string fldName = dc.ColumnName;
}
foreach (DataRow dr in tbl.Rows)
{
string addr = dr["FULLADDRESS"].ToString();
double dblX = Convert.ToDouble(dr["XCOORD"]);
double dblY = Convert.ToDouble(dr["YCOORD"]);
}
}
}
100 %
Solution Navigator
Solution 'WebSite1' (1 project)
All Open Unsaved Edited
C:\...\WebSite1\
App_Code
App_Data
App_WebReferences
Default.aspx
web.config
Ready
Ln 135
Col 45
Ch 45
INS

SOAP 1.1

The following is a sample SOAP 1.1 request and response. The `placeholders` shown need to be replaced with actual values.

```
POST /newwebservices/locationverifier.asmx HTTP/1.1
Host: citizenatlas.dc.gov
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://tempuri.org/newWebServices/LocationVerifier/findAliasFromAID"

<?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/">
  <soap:Body>
    <findAliasFromAID xmlns="http://tempuri.org/newWebServices/LocationVerifier">
      <AID>string</AID>
    </findAliasFromAID>
  </soap:Body>
</soap:Envelope>

HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?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/">
  <soap:Body>
    <findAliasFromAIDResponse
xmlns="http://tempuri.org/newWebServices/LocationVerifier">
      <findAliasFromAIDResult>
        <returnCodes>string</returnCodes>
        <details>string</details>
        <returnDataset>
          <xsd:schema>schema</xsd:schema>xml</returnDataset>
        <returnBlkAddrDataset>
          <xsd:schema>schema</xsd:schema>xml</returnBlkAddrDataset>
        <returnCDDataSet>
          <xsd:schema>schema</xsd:schema>xml</returnCDDataSet>
        <UNIT>string</UNIT>
        <UNITNUMBER>string</UNITNUMBER>
        <sourceOperation>string</sourceOperation>
        <processTime>string</processTime>
      </findAliasFromAIDResult>
    </findAliasFromAIDResponse>
  </soap:Body>
</soap:Envelope>
```

SOAP 1.2

The following is a sample SOAP 1.2 request and response. The placeholders shown need to be replaced with actual values.

```
POST /newwebservices/locationverifier.asmx HTTP/1.1
Host: citizenatlas.dc.gov
Content-Type: application/soap+xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap12:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap12="http://www.w3.org/2003/05/soap-envelope">
  <soap12:Body>
    <findAliasFromAID xmlns="http://tempuri.org/newWebServices/LocationVerifier">
      <AID>string</AID>
    </findAliasFromAID>
  </soap12:Body>
</soap12:Envelope>

HTTP/1.1 200 OK
Content-Type: application/soap+xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap12:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap12="http://www.w3.org/2003/05/soap-envelope">
  <soap12:Body>
    <findAliasFromAIDResponse
xmlns="http://tempuri.org/newWebServices/LocationVerifier">
      <findAliasFromAIDResult>
        <returnCodes>string</returnCodes>
        <details>string</details>
        <returnDataset>
          <xsd:schema>schema</xsd:schema>xml</returnDataset>
        <returnBlkAddrDataset>
          <xsd:schema>schema</xsd:schema>xml</returnBlkAddrDataset>
        <returnCDDataSet>
          <xsd:schema>schema</xsd:schema>xml</returnCDDataSet>
        <UNIT>string</UNIT>
        <UNITNUMBER>string</UNITNUMBER>
        <sourceOperation>string</sourceOperation>
        <processTime>string</processTime>
      </findAliasFromAIDResult>
    </findAliasFromAIDResponse>
  </soap12:Body>
</soap12:Envelope>
```

HTTP GET

The following is a sample HTTP GET request and response. The placeholders shown need to be replaced with actual values.

```
GET /newwebservices/locationverifier.asmx/findAliasFromAID?AID=string HTTP/1.1
Host: citizenatlas.dc.gov


HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<ReturnObject xmlns="http://tempuri.org/newWebServices/LocationVerifier">
  <returnCodes>string</returnCodes>
  <details>string</details>
  <returnDataset>
    <schema
xmlns="http://www.w3.org/2001/XMLSchema">schema</schema>xml</returnDataset>
  <returnBlkAddrDataset>
    <schema
xmlns="http://www.w3.org/2001/XMLSchema">schema</schema>xml</returnBlkAddrDataset>
  <returnCDDataSet>
    <schema
xmlns="http://www.w3.org/2001/XMLSchema">schema</schema>xml</returnCDDataSet>
  <UNIT>string</UNIT>
  <UNITNUMBER>string</UNITNUMBER>
  <sourceOperation>string</sourceOperation>
  <processTime>string</processTime>
</ReturnObject>
```

HTTP POST

The following is a sample HTTP POST request and response. The placeholders shown need to be replaced with actual values.

```
POST /newwebservices/locationverifier.asmx/findAliasFromAID HTTP/1.1
Host: citizenatlas.dc.gov
Content-Type: application/x-www-form-urlencoded
Content-Length: length

AID=string

HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<ReturnObject xmlns="http://tempuri.org/newWebServices/LocationVerifier">
  <returnCodes>string</returnCodes>
  <details>string</details>
  <returnDataset>
    <schema
xmlns="http://www.w3.org/2001/XMLSchema">schema</schema>xml</returnDataset>
  <returnBlkAddrDataset>
    <schema
xmlns="http://www.w3.org/2001/XMLSchema">schema</schema>xml</returnBlkAddrDataset>
  <returnCDDataSet>
    <schema
xmlns="http://www.w3.org/2001/XMLSchema">schema</schema>xml</returnCDDataSet>
  <UNIT>string</UNIT>
  <UNITNUMBER>string</UNITNUMBER>
  <sourceOperation>string</sourceOperation>
  <processTime>string</processTime>
</ReturnObject>
```