Reference Guide

WebSmart GeoCoder

MELISSA DATA®

WebSmart Geocoder

Reference Guide

Copyright

Trademarks

MELISSA DATA CORPORATION
22382 Avenida Empresa
Rancho Santa Margarita, CA 92688-2112
Phone: 1-800-MELISSA (1-800-635-4772)
Fax: 949-589-5211
E-mail: info@MelissaData.com
Web site: www.MelissaData.com
For the latest version of this Reference Guide, visit
http://www.MelissaData.com/tech/websmart.htm.

Document Code: WSGRFG
Revision Number: 120306.053
Last Update: March 6, 2012

Dear Programmer,

I would like to take this opportunity to introduce you to Melissa Data Corp. Founded in 1985, Melissa Data provides data quality solutions, with emphasis on address and phone verification, postal encoding, and data enhancements.

We are a leading provider of cost-effective solutions for achieving the highest level of data quality for lifetime value. A powerful line of software, databases, components, and services afford our customers the flexibility to cleanse and update contact information using almost any language, platform, and media for point-of-entry or batch processing.

This online manual will guide you through the properties and methods of our easy-to-use programming tools. Your feedback is important to me, so please don't hesitate to email your comments or suggestions to ray@MelissaData.com.

I look forward to hearing from you.

Best Wishes,

Raymond F. Melissa
President

Table of Contents

1 Welcome to WebSmart Services

The WebSmart Services are a collection of services that can be accessed by any application, allowing you to incorporate Melissa Data's technology into your programs without worrying about continually downloading and installing updates.

Melissa Data currently offers the following services:

- **Address Verifier** — Verify and standardize one or more mailing address. This service also appends ZIP + 4® and Carrier Route information.
- **Email Verifier** — Verify, correct and update, domain names from one or more email addresses.
- **GeoCoder** — Returns geographic, census, and demographic data for almost any location in the United States. Uses multisource data to return latitude and longitude down to rooftop accuracy of over 95% of all physical addresses in the United States.
- **IP Locator** — Returns name and geographic information for the owner of a public IP address.
- **Delivery Indicator** — Indicates whether an address represents a business or residential address.
- **Name Parser** — Parses and genderizes personal names and also generates salutations for correspondence.

- **Street Search** — Searches a ZIP Code™ from street address ranges matching a specific pattern and, optionally, a street number.
- **ZIP Search** — Matches city names with ZIP/Postal codes, ZIP/Postal codes with city names and searches for city names matching a pattern with a given state.
- **Phone Verifier** — Verifies and parses phone numbers, as well as identifying phone numbers as residential, business, VOIP or wireless.
- **Property** — Returns basic or detailed information about the size, ownership, and structures on a given parcel of land.

Both GeoCoder and Delivery Indicator work from an “address key” returned by the Address Verifier service, therefore, an address must first be submitted to the Address Verifier before you can use either of the other two services.

There are three ways to access the WebSmart Services:

- **SOAP** — The SOAP interface allows you to add the Web Service to an application as if it were a component object or DLL. You can then access the Web Service elements and execute commands as if they were properties and methods.
- **XML** — The Web Service can also submit a request as an XML document. It will then return the processed records as another XML document that can be parsed using whatever XML tools you utilize in your development environment.
- **REST** — This interface allows you to submit a single address record as part of a URL string and returns the processed record as an XML document identical to the one returned by the XML interface.

Using the REST service may require that you encode certain characters using the proper URL entities before adding them to a URL. Characters like spaces, slashes, ampersands and others must be replaced by special codes, which usually consist of a percent sign followed by a two-digit hexadecimal number.

The following table shows the replacements for the most common characters.

Character	URL Encoded
Space	%20 or +
*	%2A
#	%23
&	%26
%	%25

Character	URL Encoded
$	%28
+	%2B
,	%2C
/	%2F
:	%3A
;	%3B
<	%3C
=	%3D
>	%3E
?	%3F
@	%40
[	%5B
]	%5D
~	%7E

Many modern programming languages have a URL encode and URL decoding function that automates these character replacements.

Special Characters

Because the WebSmart Services are XML-based, certain characters cannot be passed as data. They would be interpreted as part of the XML structure and would cause errors. The following codes must be substituted for these characters.

Character	URL Encoded
&	& (ampersand)
“	" (left/right quotes should be replaced with straight quotes)
‘	' (apostrophe)
<	< (less-than)
>	> (greater-than)

2 An Introduction to the WebSmart GeoCoder

The WebSmart GeoCoder returns geographic, demographic and census data for a given location, defined by the address key returned by the Address Verifier service (or a postal code for Canadian addresses).

The GeoCoder Service:

- Returns the latitude and longitude (U.S. and Canada).
- Returns the time zone (U.S. and Canada).
- Returns FIPS County information Place Code, Census Block and Census Tract and Core Based Statistical Area (U.S. only).

An option of GeoCoder, GeoPoint uses multisource data to return latitude and longitude coordinates down to the rooftop of over 95% of all physical addresses in the United States.

There are two levels of accuracy within GeoPoint:

- Rooftop - The most accurate latitude and longitude available with coverage of approximately 75 million individual addresses.
- Interpolated Rooftop - These coordinates are computed using mathematical algorithms and street shape maps. While these coordinates are educated estimates, they should be quite accurate for the majority of the United States. There may be cases where the location of interpolated points is not very near the actual residence, mostly in rural areas with long streets and non-standard house numbering. However, these coordinates are still much more accurate than ZIP + 4 coordinates.

Adding GeoCoder to a Project

If you are using the SOAP service with Visual Studio .NET, you need to add a web reference to the service to your project. Click on the Project menu and select Add Web Reference... Enter the following URL on the Add Web Reference dialog box:

```
https://GeoCoder.melissadata.net/v2/SOAP/Service.svc
```

If you are not using Visual Studio .NET, see the documentation for your SOAP interface for the procedure for adding the service to your project.

Submitting an XML Request

After building your XML string from your data, an XML request to the web service is submitted using an HTTP POST operation to the following URL:

```
https://GeoCoder.melissadata.net/v2/XML/Service.svc/
   doGeoCode
```

Building a REST Request

Query strings are sent to the web service as part of the URL using an HTTP Get operation appended to following URL:

```
https://GeoCoder.melissadata.net/v2/REST/Service.svc/
   doGeoCode
```

3 GeoCoder Request

At the very minimum, a request to the WebSmart GeoCoder consists of the user's Customer ID and at least one AddressKey.

SOAP Request

The following Visual Basic Code shows a simple order of operations for building and submitting a RequestArray object, submitting it to the Web Service and retrieving a response object.

Step 1 – Create the Request and Response Objects

```
Dim ReqGeoCode As New dqwsGeoCode.RequestArray
Dim ResGeoCode As New dqwsGeoCode.ResponseArray
```

Step 2 – Assign the General Request Values

There are three properties of the Request Array object that apply to the request as a whole. CustomerID is required.

```
ReqGeoCode.CustomerID = strCustID
ReqGeoCode.TransmissionReference = strTranRef
```

The Transmission Reference is a unique string value that identifies this request array.

Step 3 – Dimension the Record Array

The maximum number of records per request is 100, therefore the largest dimension will be 99.

```
ReDim ReqGeoCode.Record(99)
```

For maximum efficiency, you should dimension the array using the exact number of records being submitted minus one.

Step 4 – Build the Record Array

The exact method for building the array will depend on the exact database software in use, but you will need to loop through every record to be submitted and assign the required values to the corresponding elements for each record in the RequestArray.

```
ReqGeoCode.Record(intRecord) = New
   dqwsGeoCode.RequestArrayRecord
ReqGeoCode.Record(intRecord).AddressKey = "92688211282"
ReqGeoCode.Record(intRecord).RecordID = 1
```

The lines above show only the elements that are absolutely required to submit a record to the web service. See the rest of this chapter for a description of all of the elements available to include with a request record.

Repeat for each record being submitted with the current RequestArray.

Step 5 – Submit the Request Array

The final step is to create the Service Client Object and then submit the RequestArray object doGeoCode method. This sends the data to the web service and retrieves the ResponseArray object.

```
GeoCodeClient = New dqwsGeoCode.Service
ResGeoCode = GeoCodeClient.doGeoCode(ReqGeoCode)
GeoCodeClient.Dispose()
```

XML Request

The raw XML request is built using whatever XML tools are available via your development tools and submitted to the following URL using an HTTP POST request.

```
https://GeoCoder.melissadata.net/v2/XML/Service.svc/
   doGeoCode
```

Rather than an array of Record objects, an XML request contains a <Record> element for each address record, up to 100.

The following XML Code contains the same request as the SOAP example above.

```
<RequestArray>
   <TransmissionReference>Web Service Test 2008/12/31
   </TransmissionReference>
   <CustomerID>123456789</CustomerID>
   <Record>
      <RecordID>1</RecordID>>
      <AddressKey>92688211282</AddressKey>
   </Record>
   <Record>
   ...
   </Record>
</RequestArray>
```

REST Request

A REST request can submit a single address record via an HTTP GET. The following example uses the same address as the SOAP and XML samples.

```
https://GeoCoder.melissadata.net/v2/REST/Service.svc/
   doGeoCode?id=12345678&t=RestTest&key=92688211282
```

The record ID element does not exist for the REST interface, since you can only submit a single record per request.

Request Elements

The following section lists the elements that set the basic options for each and identify the user to the Web Service.

Customer ID

This is a required string value containing the identifier number issued to the customer when signing up for Melissa Data WebSmart Services.

Remarks

You need a customer ID to access any Melissa Data Web Service. If this element is not populated, the web service will return an error. To receive a customer ID, call your Melissa Data sale representative at 1-800-MELISSA.

Syntax

SOAP

```
Request.CustomerID = string
```

XML

```
<RequestArray>
    <CustomerID>String</CustomerID>
</RequestArray>
```

REST

```
id={CustomerID}
```

Transmission Reference

This is an optional string value that may be passed with each Request Array to serve as a unique identifier for this set of records.

Remarks

This value is returned as sent by the Response Array, allowing you to match the Response to the Request.

Syntax

SOAP

```
Request.TransmissionReference = string
```

XML

```
<RequestArray>
    <TransmissionReference>String</TransmissionReference>
</RequestArray>
```

REST

```
t={transMissionReference}
```

Record Elements

For the SOAP and XML web services, the Request Array will contain an element or property called Record. In SOAP this property is an array of object variables of the type Record. XML will have as many Record elements as there are addresses being submitted to the web service.

The REST interface only allows a single record per request.

Record ID

This element is a string value containing a unique identifier for the current record.

Remarks

Use this element to match the record with the record returned with the Response Array.

When using the SOAP interface, if this element is not populated, the web service will automatically insert a sequential number for each record.

There is no equivalent for Record ID for the REST interface.

Syntax

SOAP

```
Request.Record().RecordID = string
```

XML

```
<RequestArray>
    <Record>
        <RecordID>String</RecordID>
    </Record>
</RequestArray>
```

AddressKey

This element is a required eleven-character string value containing a unique number representing a specific location or a six-character Canadian postal code.

Remarks

This value can beobtained by submitting an address record to the Address Verifier service.

For Canadian GeoCoding, submit the Postal Code instead of an address key..

Syntax

SOAP

```
Request.Record().AddressKey = string
```

XML

```
<RequestArray>
    <Record>
        <AddressKey>String</AddressKey>
    </Record>
</RequestArray>
```

REST

```
key={AddressKey}
```

4 GeoCoder Response

The SOAP interface for the GeoCoder service returns a ResponseArray Object. The primary component of this object is an array of Record objects, one for each record submitted with the RequestArray, containing the verified and standardized address data.

The XML interface returns an XML document containing a number of <Record> elements, one for each record submitted with the Request, containing the verified and standardized address data.

The REST interface returns an XML document with a single <Record> element.

TransmissionReference

Returns a string value containing the contents of the TransmissionReference element from the original Request.

Remarks

If you passed any value to the TransmissionReference element when building your request, it is returned here. You can use this property to match the response to the request.

Syntax

SOAP

```
string = Response.TransmissionReference
```

XML

```
<ResponseArray>
    <TransmissionReference>
        String
    </TransmissionReference>
</ResponseArray>
```

Total Records

Returns a string value containing the number records returned with the current response.

Remarks

This property returns the number of records processed and returned by the response as a string value.

Syntax

SOAP

```
string = Response.TotalRecords
```

XML

```
<ResponseArray>
   <TotalRecords>String</TotalRecords>
</ResponseArray>
```

Results

Returns a string value containing the general and system error messages from the most recent request sent to the service.

Remarks

Do not confuse this element with the Results element returned with each record, described on page 20. This element returns error messages caused by the most recent request as a whole.

The possible values are:

Code	Description
SE01	Web Service internal error.
GE01	General Error — Empty XML request structure.
GE02	General Error — Empty XML request record structure.
GE03	General Error — Counted records send more than number of records allowed per request.
GE04	General Error — CustomerID is empty.
GE05	General Error — CustomerID is invalid.
GE06	General Error — CustomerID is disabled.
GE07	General Error — XML request is invalid.

Syntax

SOAP

```
string = Response.Results
```

XML

```
<ResponseArray>
   <Results>String</Results>
</ResponseArray>
```

Version

Returns a string value containing the current version number of the GeoCoder web service.

Syntax

SOAP

```
string = Response.Version
```

XML

```
<ResponseArray>
   <Version>String</Version>
</ResponseArray>
```

Record Elements

The SOAP version of the Response Array returns a property called Record which is an array of Record objects, one for each record submitted with the original Request Array.

The XML service returns one <Record> element for every record submitted with the original request.

The REST response is identical to the XML response, but will only contain a single <Record> element.

The following section describes the elements returned by each record in the Response Array.

Record ID

For each record in the Response Array, this element returns a string value containing the unique identifier for the current record if one was passed to the Request Array.

Remarks

Use this element to match the record in the Response Array with the record originally passed with the request.

Syntax

SOAP

```
string = Response.Record().RecordID
```

XML

```
<ResponseArray>
    <Record>
        <RecordID>String</RecordID>
    </Record>
</ResponseArray>
```

Results

For each record in the Response Array, returns a string value containing status and error codes for the current record. Multiple codes are separated by commas.

Remarks

This element returns the status and error messages for each record in the Response Array. For the general status and error messages generated by the most recent GeoCoder request, see the general Result element on page 16.

The Result element may return one or more four-character strings, separated by commas, depending on the result generated by the current record.

If the address in the current record was verified, this element will contain the value “GS01” at the very minimum and may include more of the “GS” codes. If the address could not be verified, the codes beginning with “GE” will indicate the reason or reasons why verification failed.

The possible values are:

Code	Description
GS01	Record was coded to the ZIP + 4 centroid.
GS02	Record was coded to the ZIP + 2 centroid.
GS03	Record was coded to the 5-digit ZIP Code centroid.
GS04	A Canadian Record was geocoded.
GS05	Record was coded to rooftop level.
GS06	Record was coded to interpolated rooftop level.
GE02	ZIP Code Error. An invalid ZIP Code was entered.
DE	AddressKey was empty or less than 11 characters were sent or a non-numeric character was encountered.

Syntax

SOAP

```
string = Response.Record().Results
```

XML

```
<ResponseArray>
   <Record>
      <Results>String</Results>
   </Record>
</ResponseArray>
```

County

For each record in the Response Array, these elements return the county name and FIPS Code for the address key submitted for the current record.

Remarks

The Federal Information Processing Standard (FIPS) is a 5-digit code defined by the U.S. Bureau of Census. The first two digits are a state code and the last three indicate the county within the state.

06037 is the County FIPS for Los Angeles, CA. “06” is the state code for California and “037” is the county code for Los Angeles.

Syntax

SOAP

```
string = Response.Record().Address.County.Fips
string = Response.Record().Address.County.Name
```

XML

```
<ResponseArray>
    <Record>
        <Address>
            <County>
                <Fips>String</Fips>
                <Name>String</Name>
            </County>
        </Address>
    </Record>
</ResponseArray>
```

Latitude

For each record in the Response Array, this element returns a string value containing the latitude for the centroid of the location described by the submitted address key or postal code.

Remarks

Latitude is the geographic coordinate of a point measured in degrees north or south of the equator. The web service uses the WGS-84 standard for determining latitude.

Latitude is returned for both the U.S. and Canada.

Since all North American latitude coordinates are north of the equator, this value will always be positive.

Syntax

SOAP

```
string = Response.Record().Address.Latitude
```

XML

```
<ResponseArray>
   <Record>
      <Address>
         <Latitude>String</Latitude>
      </Address>
   </Record>
</ResponseArray>
```

Longitude

For each record in the Response Array, this element returns a string value containing the longitude for the centroid of the location described by the submitted address key or postal code.

Remarks

Longitude is the geographic coordinate of a point measured in degrees east or west of the Greenwich meridian. The web service uses the WGS-84 standard for determining longitude.

Longitude is returned for both the U.S. and Canada.

Since all North American longitude coordinates are west of the Greenwich meridian, this value will always be negative.

Syntax

SOAP

```
string = Response.Record().Address.Longitude
```

XML

```
<ResponseArray>
   <Record>
      <Address>
         <Longitude>String</Longitude>
      </Address>
   </Record>
</ResponseArray>
```

Place

For each record in the Response Array, these elements return the Census Bureau's Place Code and Place Name for the address key submitted with the current record.

Remarks

ZIP Code boundaries sometime overlap with city limits and unincorporated areas. The ZIP Code may place a location within one city even though it is physically located within a neighboring area. These properties returns the Census Bureau's official name for the area containing the location described the submitted address key.

For example, the 92688 ZIP Code is located mostly within the city of Rancho Santa Margarita. However, it also contains parts of the unincorporated area of Los Flores. For these ZIP + 4 codes, the City property of the Address Verifier service would return "Rancho Santa Margarita," but the Name property will return "Los Flores."

Syntax

SOAP

```
string = Response.Record().Address.Place.Code
string = Response.Record().Address.Place.Name
```

XML

```
<ResponseArray>
    <Record>
        <Address>
            <Place>
                <Code>String</Code>
                <Name>String</Name>
            </Place>
        </Address>
    </Record>
</ResponseArray>
```

Time Zone

For each record in the Response Array, these elements return the numeric code and descriptive name for the time zone containing the location described by the submitted address key or postal code.

Remarks

These elements can return the following values:

Code	Name	Code	Name
0	Military (APO or FPO)	9	Alaska Time
4	Atlantic Time	10	Hawaii Time
5	Eastern Time	11	Samoa Time
6	Central Time	13	Marshall Islands Time
7	Mountain Time	14	Guam Time
8	Pacific Time	15	Palau Time

Time zone information is returned for both U.S, and Canada.

Syntax

SOAP

```
string = Response.Record().Address.TimeZone.Code
string = Response.Record().Address.TimeZone.Name
```

XML

```
<ResponseArray>
   <Record>
      <Address>
         <TimeZone>
            <Code>String</Code>
            <Name>String</Name>
         </TimeZone>
      </Address>
   </Record>
</ResponseArray>
```

Core Based Statistical Area

For each record in the Response Array, these six elements return the U.S. Census Bureau's Core Based Statistical Area (CBSA) data for the location associated with the submitted address key.

Remarks

Metropolitan and micropolitan statistical areas (metro and micro areas) are geographic entities defined by the U.S. Office of Management and Budget (OMB) for use by Federal statistical agencies in collecting, tabulating, and publishing Federal statistics. The term "Core Based Statistical Area" (CBSA) is a collective term for both metro and micro areas. A metro area contains a core urban area of 50,000 or more population, and a micro area contains an urban core of at least 10,000 (but less than 50,000) population. Each metro or micro area consists of one or more counties and includes the counties containing the core urban area, as well as any adjacent counties that have a high degree of social and economic integration (as measured by commuting to work) with the urban core.

The CBSA Code is a five-digit code for the specific CBSA associated with the location described by the submitted address key. The Level states whether the particular CBSA is a metropolitan or micropolitan area. The Title returns the official U.S. Census Bureau name for the CBSA.

Some CBSA's are broken into parts known as divisions. In this case, the CBSA Division elements will also be populated. If not, these elements will be empty. Each division also has a Code, Level and Title.

Syntax

SOAP

```
string = Response.Record().Address.CBSA.Code
string = Response.Record().Address.CBSA.Level
string = Response.Record().Address.CBSA.Title
string = Response.Record().Address.CBSA.CBSADivisionCode
string = Response.Record().Address.CBSA.CBSADivisionLevel
string = Response.Record().Address.CBSA.CBSADivisionTitle
```

XML

```
<ResponseArray>
   <Record>
      <Address>
         <CBSA>
            <Code>String</Code>
            <Level>String</Level>
            <Title>String</Title>
            <CBSADivisionCode>String</CBSADivisionCode>
            <CBSADivisionLevel>String</CBSADivisionLevel>
            <CBSADivisionTitle>String</CBSADivisionTitle>
         </CBSA>
      </Address>
   </Record>
</ResponseArray>
```

Census Block

For each record in the Response Array, this element returns the Census Block number associated with the location described by the submitted address key.

Remarks

Census blocks, the smallest geographic area for which the Bureau of the Census collects and tabulates decennial census data, are formed by streets, roads, railroads, streams and other bodies of water, other visible physical and cultural features, and the legal boundaries shown on Census Bureau maps.

The Census Block is a four-character string value. The first digit is the Block Group and the last three characters (if any) are the Block Number.The block group returns a one-character string containing the block group number.

Syntax

SOAP

```
string = Response.Record().Address.Census.Block
```

XML

```
<ResponseArray>
   <Record>
      <Address>
         <Census>
            <Block>String</Block>
         </Census>
      </Address>
   </Record>
</ResponseArray>
```

Census Tract

For each record in the Response Array, this element returns the Census Tract number associated with the location described by the submitted address key.

Remarks

Census Tracts are small, relatively permanent statistical subdivisions of a county. Census Tracts are delineated for all metropolitan areas (MA's) and other densely populated counties by local census statistical areas committees following Census Bureau guidelines (more than 3,000 Census Tracts have been established in 221 counties outside MA's).

The CensusTract property is usually returned as a 4-digit string. However, in areas that experience substantial growth, a Census Tract may be split to keep the population level even. When this happens, a 6-digit number will be returned.

The web service requires a full nine-digit ZIP with a valid Plus 4 add-on to return the Census Tract. If a five-digit ZIP is submitted, the Census Tract will not be returned.

Syntax

SOAP

```
string = Response.Record().Address.Census.Tract
```

XML

```
<ResponseArray>
    <Record>
        <Address>
            <Census>
                <Tract>String</Tract>
            </Census>
        </Address>
    </Record>
</ResponseArray>
```

Response Object XML Format

The following shows the structure of the XML document returned by the GeoCoder Service.

```
<?xml version="1.0" encoding="UTF-8"?>
<ResponseArray xmlns="urn:mdWebServiceGeoCode"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   >
   <version>String</version>
   <TransmissionReference>String
   </TransmissionReference>
   <Results>String</Results>
   <TotalRecords>String</TotalRecords>

   <Record>
      <RecordID>String</RecordID>
      <Results>String</Results>
      <Address>
         <County>
            <Name>String</Name>
            <Fips>String</Fips>
         </County>
         <Latitude>String</Latitude>
         <Longitude>String</Longitude>
         <Place>
            <Code>String</Code>
            <Name>String</Name>
         </Place>
         <TimeZone>
            <Name>String</Name>
            <Code>String</Code>
         </TimeZone>
         <CBSA>
            <Code>String</Code>
            <Title>String</Title>
            <Level>String</Level>
            <CBSADivisionCode>String</CBSADivisionCode>
            <CBSADivisionTitle>String</CBSADivisionTitle>
            <CBSADivisionLevel>String</CBSADivisionLevel>
         </CBSA>
         <Census>
```

```
            <Block>String</Block>
            <Tract>String</Tract>
         </Census>
      </Address>
   </Record>
</ResponseArray>
```