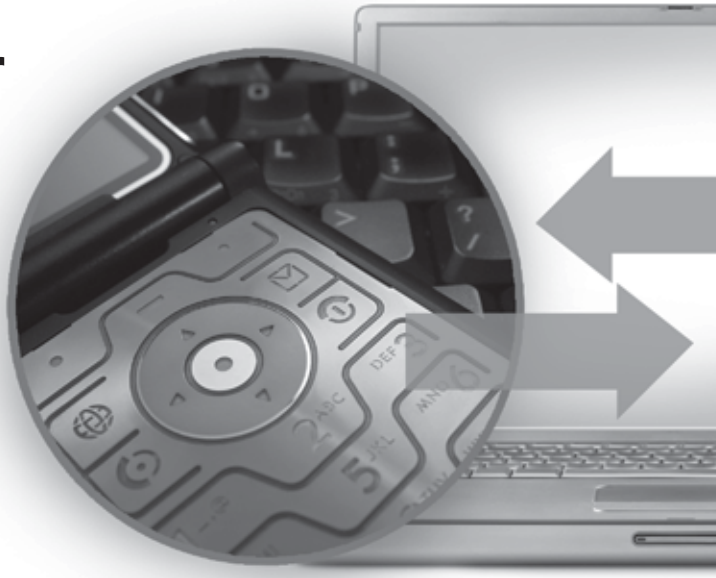
Reference Guide

WebSmart Phone Verifier



MELISSA DATA®

WebSmart Phone Verifier

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: WSPRFG
Revision Number: 131022.087
Last Update: October 22, 2013

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 Phone Verifier

WebSmart Phone Verifier allows Web sites and custom applications to verify phone numbers down to 7 and 10 digits, update area codes, and append data about the phone number.

Use Phone Verifier to:

- Verify U.S. or Canadian phone numbers down to 7 or 10 digits.
- Update the area code if it changed in the last year.
- Append data on the telephone line, distinguishing between landline, wireless numbers, or Voice Over IP (VOIP).
- Append data on the telephone owner, distinguishing between residential, business, or home office numbers.
- Parse the phone number into its components.

Adding Phone Verifier 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://phonecheck.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://phonecheck.melissadata.net/v2/XML/Service.svc/
   DoPhoneCheck
```

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://phonecheck.melissadata.net/v2/REST/Service.svc/
   doPhoneCheck
```

3 Phone Verifier Request

At the very minimum, a request to the WebSmart Phone Verifier consists of the user's Customer ID and a phone number.

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 ReqPhoneCheck As New dqwsPhoneCheck.RequestArray
Dim ResPhoneCheck As New dqwsPhoneCheck.ResponseArray
```

Step 2 – Assign the General Request Values

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

```
ReqPhoneCheck.CustomerID = strCustID
ReqPhoneCheck.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 ReqPhoneCheck.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.

```
ReqPhoneCheck.Record(intRecord) = New
   dqwsPhoneCheck.RequestArrayRecord
ReqPhoneCheck.Record(intRecord).Phone = "9495895200"
```

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 doPhoneCheck method. This sends the data to the web service and retrieves the ResponseArray object.

```
PhoneCheckClient = New dqwsPhoneCheck.Service
ResPhoneCheck =
   PhoneCheckClient.doPhoneCheck(ReqPhoneCheck)
```

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://phonecheck.melissadata.net/v2/XML/Service.svc/
   doPhoneCheck
```

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>>
      <Phone>9495895200</Phone>
   </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://phonecheck.melissadata.net/v2/REST/Service.svc/
   doPhoneCheck?id=12345678&t=RestTest&phone=9495895200
```

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 phone numbers 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>
```

Phone

This element is a required ten-character string value containing the phone number to be verified.

Remarks

The phone number should be ten characters long and include only the digits of the phone number; no parentheses, dashes or spaces.

Syntax

SOAP

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

XML

```
<RequestArray>
    <Record>
        <Phone>String</Phone>
    </Record>
</RequestArray>
```

REST

```
phone={PhoneNumber}
```

4 Phone Verifer Response

The SOAP interface for the Phone Verifer 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 parsed and verified phone data.

The XML interface returns an XML document containing a number of <Record> elements, one for each record submitted with the Request, containing the parsed and verified phone 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	Short Description	Long Description
SE01	Web Service Internal Error	The web service experienced an internal error.
GE01	Empty Request Structure	The SOAP, JSON, or XML request structure is empty.
GE02	Empty Request Record Structure	The SOAP, JSON, or XML request record structure is empty.
GE03	Records Per Request Exceeded	The counted records sent more than the number of records allowed per request.
GE04	Empty CustomerID	The CustomerID is empty.
GE05	Invalid CustomerID	The CustomerID is invalid.
GE06	Disabled CustomerID	The CustomerID is disabled.
GE07	Invalid Request	The SOAP, JSON, or 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 Phone Verifer 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 since it only allows single requests.

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, this element 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 Phone Verifer 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 “PS01” or “PS02” at the very minimum and may include more of the “PS” codes. If the address could not be verified, the codes beginning with “PE” will indicate the reason or reasons why verification failed.

The possible values are:

Code	Short Description	Long Description
PS01	10-Digit Match	The first 10-digits of the phone number have been verified as valid.
PS02	7-Digit Match	The first 7-digits of the phone number has been verified as valid.
PS06	Updated Area Code	The area code was changed due to an area code split. The updated code is located within NewAreaCode.
PS07	Cellular Line	The exchange type of the phone number indicates the number is a cellular number.
PS08	Land Line	The exchange type of the phone number indicates the number is a land line number.
PS09	VOIP Line	The exchange type of the phone number indicates the number is a VOIP number.
PS10	Residential Number	The phone number belongs to a residence.
PS11	Business Number	The phone number belongs to a business.
PS12	SOHO Number	The phone number belongs to a small office or home office.

Code	Short Description	Long Description
PE01	Bad Area Code	The area code does not exist in our database or contains non-numbers.
PE02	Blank Phone Number	The phone number is blank.
PE03	Bad Phone Number	The phone number has too many or too few digits.
PE04	Multiple Match	Two or more possible area codes are available as a fix and their distance is too close to choose one over the other.
PE05	Bad Prefix	The phone prefix does not exist in our database.

Syntax

SOAP

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

XML

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

Area Code

For each record in the Response Array, this element returns a string value containing the original area code from the submitted phone number.

Remarks

If there are not enough digits for the area code, this element will be empty. If the area code/prefix combination has been split, the new area code will be returned by the NewAreaCode element.

Syntax

SOAP

```
string = Response.Record().Phone.AreaCode
```

XML

```
<ResponseArray>
    <Record>
        <Phone>
            <AreaCode>String</AreaCode>
        </Phone>
    </Record>
</ResponseArray>
```

New Area Code

For each record in the Response Array, this element returns a string value containing the new area code for the submitted phone number, if the submitted area code has recently undergone a split.

Remarks

The New Area Code element will return an updated area code based on the submitted phone number.

An updated area code is a new area code that is based on the input of an area code/prefix combination that has been split. If no new area code was found, then this element will be empty..

Syntax

SOAP

```
string = Response.Record().Phone.NewAreaCode
```

XML

```
<ResponseArray>
    <Record>
        <Phone>
            <NewAreaCode>String</NewAreaCode>
        </Phone>
    </Record>
</ResponseArray>
```

Prefix

For each record in the Response Array, this element returns a string value containing the prefix (first three digits after the area code) of the submitted phone number.

Remarks

The prefix or exchange is the three-digit portion of the phone number that immediately follows the area code.

Syntax

SOAP

```
string = Response.Record().Phone.Prefix
```

XML

```
<ResponseArray>
    <Record>
        <Phone>
            <Prefix>String</Prefix>
        </Phone>
    </Record>
</ResponseArray>
```

Suffix

For each record in the Response Array, this element returns a string value containing the suffix (last four digits) of a submitted phone number.

Remarks

The suffix is the last four digits of the phone number

Syntax

SOAP

```
string = Response.Record().Phone.Suffix
```

XML

```
<ResponseArray>
    <Record>
        <Phone>
            <Suffix>String</Suffix>
        </Phone>
    </Record>
</ResponseArray>
```

Extension

For each record in the Response Array, this element returns a string value containing the extension information, if any, of a submitted phone number.

Remarks

If the submitted phone number did not contain an extension, this element will be blank.

Syntax

SOAP

```
string = Response.Record().Phone.Extension
```

XML

```
<ResponseArray>
    <Record>
        <Phone>
            <Extension>String</Extension>
        </Phone>
    </Record>
</ResponseArray>
```

City

For each record in the Response Array, this element returns a string value containing the name of the city associated with the submitted phone number.

Remarks

Because of phone number portability, geographical information may not reflect the true location of the owner of the phone number for wireless and VOIP numbers.

Syntax

SOAP

```
string = Response.Record().Phone.City
```

XML

```
<ResponseArray>
    <Record>
        <Phone>
            <City>String</City>
        </Phone>
    </Record>
</ResponseArray>
```

State

For each record in the Response Array, this element returns a string value containing the name of the state associated with the submitted phone number.

Remarks

Because of phone number portability, geographical information may not reflect the true location of the owner of the phone number for wireless and VOIP numbers.

Syntax

SOAP

```
string = Response.Record().Phone.State
```

XML

```
<ResponseArray>
   <Record>
      <Phone>
         <State>String</State>
      </Phone>
   </Record>
</ResponseArray>
```

Country

For each record in the Response Array, these two elements return string values containing the full name and abbreviation for the country, either the United States or Canada, associated with the submitted phone number.

Remarks

The possible values for Country Code are “US” or “CA.” The possible values for Country Name are “United States of America” or “Canada.”

Because of phone number portability, geographical information may not reflect the true location of the owner of the phone number for wireless and VOIP numbers.

Syntax

SOAP

```
string = Response.Record().Phone.Country.Abbreviation
string = Response.Record().Phone.Country.Name
```

XML

```
<ResponseArray>
   <Record>
      <Phone>
         <Country>
            <Abbreviation>String</Abbreviation>
            <Name>String</Name>
         </Country>
      </Phone>
   </Record>
</ResponseArray>
```

Latitude

For each record in the Response Array, this element returns a string value containing the latitude of the NPA/NXX wire center for the submitted phone number.

Remarks

Because of phone number portability, geographical information may not reflect the true location of the owner of the phone number for wireless and VOIP numbers.

Syntax

SOAP

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

XML

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

Longitude

For each record in the Response Array, this element returns a string value containing the longitude of the NPA/NXX wire center for the submitted phone number.

Remarks

Because of phone number portability, geographical information may not reflect the true location of the owner of the phone number for wireless and VOIP numbers.

Syntax

SOAP

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

XML

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

TimeZone

For each record in the Response Array, these elements return string values containing the name and numeric code for the time zone associated with the submitted phone number.

Remarks

Following are the possible values for the time zone code and name:

Code	Name	Code	Name
4	Atlantic Time	10	Hawaii Time
5	Eastern Time	11	Samoa Time
6	Central Time	13	Marshall Island Time
7	Mountain Time	14	Guam Time
8	Pacific Time	15	Palau Time
9	Alaska Time		

Because of phone number portability, geographical information may not reflect the true location of the owner of the phone number for wireless and VOIP numbers.

Syntax

SOAP

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

XML

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

Response Object XML Format

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

```
<?xml version="1.0" encoding="utf-8"?>
<ResponseArray>
   <Version>String</Version>
   <TransmissionReference>String</TransmissionReference>
   <Results>String</Results>
   <TotalRecords>String</TotalRecords>
   <Record>
      <RecordID>String</RecordID>
      <Results>String</Results>
      <Phone>
         <AreaCode>String</AreaCode>
         <NewAreaCode>String</NewAreaCode>
         <Prefix>String</Prefix>
         <Suffix>String</Suffix>
         <Extension>String</Extension>
         <City>String</City>
         <State>String</State>
         <Country>
            <Abbreviation>String</Abbreviation>
            <Name>String</Name>
         </Country>
         <Latitude>String</Latitude>
         <Longitude>String</Longitude>
         <TimeZone>
            <Name>String</Name>
            <Code>String</Code>
         </TimeZone>
      </Phone>
   </Record>
</ResponseArray>
```