Reference Guide

WebSmart Email Verifier

MELISSA DATA®

WebSmart Email 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: WSERFG
Revision Number: 110621.104
Last Update: June 21, 2011

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.
- **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 fields 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
$	%28

Character	URL Encoded
+	%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 Email Verifier

The WebSmart Email Verifier verifies that a submitted email address belongs to a valid domain. It can correct common misspellings of domains and update domain names if they have changed due to corporate mergers or other situations. The service also parses the email into mailbox name (the part before the "@"), the domain name and the top-level domain name (".com," ".org" and so on).

There are three ways to access the web service.

Adding WebSmart Email Verifier Web Service 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://email.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://email.melissadata.net/v2/XML/Service.svc/
   doEmailCheck
```

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://email.melissadata.net/v2/REST/Service.svc/
   doEmailCheck
```

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

3 Email Verifier Request

At the very minimum, a request to the WebSmart Email Verifier consists of the user's Customer ID and at least one email address.

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 ReqEmail As New dqwsEmail.RequestArray
Dim ResEmail As New dqwsEmail.ResponseArray
```

Step 2 – Assign the General Field Values

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

```
ReqEmail.CustomerID = strCustID
ReqEmail.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 ReqEmail.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 fields for each record in the RequestArray.

```
ReqEmail.Record(intRecord) = New
   dqwsEmail.RequestArrayRecord
ReqEmail.Record(intRecord).Email =
   "ray@mailerssoftware.com"
ReqEmail.Record(intRecord).RecordID = 1
```

The lines above show only the fields 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 fields 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 doEmail method. This sends the data to the web service and retrieves the ResponseArray object.

```
EmailClient = New dqwsEmail.Service
ResEmail = EmailClient.doEmailCheck(ReqEmail)
EmailClient.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://email.melissadata.net/v2/XML/Service.svc/
   doEmailCheck
```

Rather than an array of Record object, 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>>
      <Email>ray@mailerssoftware.com</Email>
   </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://email.melissadata.net/v2/REST/Service.svc/
   doEmailCheck?id=12345678&t=RestTest&email=
   ray%40mailerssoftware.com
```

Remember that the "@" must be replaced by a URL entity before submitting the REST request to the web service.

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

Request Fields

The following section lists the fields 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 the WebSmart Services.

Remarks

You need a customer ID to access any Melissa Data web service. If this field is not populated, the web service will return an error. To receive a customer ID, call your Melissa Data sales 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 Fields

For the SOAP and XML 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 field is a string value containing a unique identifier for the current record.

Remarks

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

When using the SOAP interface, if this field 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>
```

Email

This field must contain a well-formed email address.

Remarks

A well-formed email address contains a mailbox and a domain name separated by a “@” character.

Syntax

SOAP

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

XML

```
<RequestArray>
    <Record>
        <Email>String</Email>
    </Record>
</RequestArray>
```

REST

```
email={Email}
```

4 Email Verifier Response

The SOAP interface for the Email Verifier 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 email address.

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 email address.

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

TransmissionReference

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

Remarks

If you passed any value to the TransmissionReference field 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 of 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 field with the Results field returned with each record, described on page 21. This field 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 Email Verifier Service.

Syntax

SOAP

```
string = Response.Version
```

XML

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

Record Fields

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 fields returned by each record in the Response Array.

Record ID

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

Remarks

Use this field 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 field returns a string value containing status and error codes for the current record. Multiple codes are separated by commas.

Remarks

This field 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 Email request, see the general Result field on page 17.

The Result field 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 field will contain the value “ES01” at the very minimum and may include more of the “ES” codes. If the address could not be verified, the codes beginning with “EE” will indicate the reason or reasons why verification failed.

The possible values are:

Code	Status/Error	Description
ES01	Valid Email Domain	The domain name of the submitted email address was confirmed as valid by either the DatabaseLookup or MXLookup
ES02	Invalid Email Domain	The domain name of the submitted email address was either not located by MXLookup or was located on the list of invalid domains.
ES03	Unverified Email Domain	The domain name of the submitted email address was not confirmed as valid by either DatabaseLookup or MXLookup, but was not found on the list of invalid domain names.
ES04	Mobile Email Address	The domain name of the submitted email was identified as a mobile email address, classified as not deliverable by FCC regulations.
ES10	Syntax Was Changed	The syntax of the submitted email address was changed.

Code	Status/Error	Description
ES11	Top Level Domain Changed.	The top level domain of the submitted email address was changed.
ES12	Domain Changed (Spelling)	The domain of the submitted email address was corrected for spelling.
ES13	Domain Changed (Update)	The domain of the submitted email address was updated due to a domain name change.
EE01	Syntax Error	There is a syntax error in the submitted email address.
EE02	Top Level Domain Not Found	The top level domain of the submitted email address was not found.
EE03	Mail Server Not Found	The mail server (domain) of the submitted email address was not found.
EE04	Invalid Mailbox Name	An invalid mailbox name was detected (IE: noreply). To configure invalid mailbox names, review mdEmailConfig.ini.
DE	Email Address was empty.	Submitted email address field was an empty string or null value.

Syntax

SOAP

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

XML

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

EmailAddress

For each record in the Response Array, this field returns the updated, corrected email address.

Remarks

If the submitted address could not be verified, this will return the contents of the Email field submitted with the Request Array.

Syntax

SOAP

```
string = Response.Record().Email.EmailAddress
```

XML

```
<ResponseArray>
    <Record>
        <Email>
            <EmailAddress>String</EmailAddress>
        </Email>
    </Record>
</ResponseArray>
```

Domain Name

For each record in the Response Array, this field returns the domain name portion of the email address passed to the Request Array record, excluding the Top Level Domain, including any changes or corrections that have been made by the WebSmart Email Verifier.

Remarks

This field returns a string value containing the domain name portion of the corrected email address. For example, it returns all characters that come after the "@" character, not including the Top Level Domain, such as ".com." If the final address is "jsmith@melissadata.com," this property just returns "melissadata."

To get the full domain name, combine the results of this field with a "." character and the contents of the TopLevelDomain Name field.

Syntax

SOAP

```
string = Response.Record().Email.DomainName
```

XML

```
<ResponseArray>
    <Record>
        <Email>
            <DomainName>String</DomainName>
        </Email>
    </Record>
</ResponseArray>
```

Mailbox Name

For each record in the Response Array, this field returns the mailbox or user name portion of the email address passed to the VerifyEmail address, including any changes or corrections that have been made by the Web Service.

Remarks

This field returns a string value containing the mailbox name or user name portion of the email address (all characters that precede the "@" character). If the final address is "jsmith@melissadata.com," this property returns "jsmith."

Syntax

SOAP

```
string = Response.Record().Email.MailboxName
```

XML

```
<ResponseArray>
   <Record>
      <Email>
         <MailboxName>String</MailboxName>
      </Email>
   </Record>
</ResponseArray>
```

Top Level Domain

For each record in the Response Array, the Name field returns the Top Level Domain name portion of the email address passed to the Web Service address, including any changes or corrections that have been made by the Web Service. The Description field returns the long-form description of the Top Level Domain name portion of the email address.

Remark

The Name field returns a string value containing the Top Level Domain name portion of the corrected email address, such as “com”. If the final address is “jsmith@melissadata.com,” this field returns “com.”

Syntax

SOAP

```
string = Response.Record().Email.TopLevelDomain.Name
string = Response.Record().Email.TopLevelDomain.Description
```

XML

```
<ResponseArray>
   <Record>
      <Email>
         <TopLevelDomain>
            <Name>String</Name>
            <Description>String</Description>
         </TopLevelDomain>
      </Email>
   </Record>
</ResponseArray>
```

Web Response XML Format

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

```
<?xml version="1.0" encoding="UTF-8"?>
<ResponseArray xmlns="urn:mdWebServiceEmail"
   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>
         <Email>
            <EmailAddress>string</EmailAddress>
            <DomainName>string</DomainName>
            <MailboxName>string</MailboxName>
            <TopLevelDomain>
               <Name>string</Name>
               <Description>string</Description>
            </TopLevelDomain>
      </Email>
   </Record>
</ResponseArray>
```