Data Quality Web Service
Reference Guide

MELISSADATA™

Copyright

Trademarks

Document number: DQS RG 090423
Last Revision Date: April 23, 2009

MELISSA DATA CORPORATION
22382 Avenida Empresa
Rancho Santa Margarita, CA 92688

Phone: 1-800-MELISSA (1-800-635-4772)
Fax: 949-589-5211

E-mail: info@MelissaData.com
Web site: www.MelissaData.com

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

Chapter 1

Introduction

Melissa Data's Data Quality Web Service (DQWS) allows you to use the power of the internet to instantly verify, correct, update, and enhance your address data.

You can use it to:

- Verify and correct each address and phone number your customers submit with online purchases, information requests, and survey responses.
- Create applications that automatically update each record in your database.
- Personalize customer records by separating first and last names and identifying gender.
- Target markets more effectively by appending geographic data to each record.

Our Data Quality Web Service provides these services and more — saving you time and money while providing trouble-free, real-time data verification and data enhancement. You no longer need to worry about installing and maintaining software updates. Data Quality Web Service does all the work for you —freeing you to concentrate on your business and your customers!

Request Packages

The Data Quality Web Service uses several specific "packages" to process your data and return the results. Selecting a package is simple and you can select as many or as few packages as you need with a single request, with the exception that a few packages cannot be selected at the same time. These are noted in the section below.

Below is a list of the packages available with DQWS including information on the type of data they process and what they return.

Address DPV: U.S. addresses only. This package will verify that the street address, suite number, city, state and ZIP Code™ constitute a valid USPS® delivery address and return the DPV footnotes and indicate whether they address is owned by a CMRA (Commercial Mail Receiving Agency). See page 9 for more information about this package.

Address GeoCode: This package will provide basic geographic information about a submitted ZIP Code, returns the latitude and longitude of the ZIP + 4® centroid, plus county and Census Block/Tract information. This package will also return the latitude and longitude of Canadian addresses, but not the other properties. See page 13 for more information about this package.

Address Geographic Location: This package returns general geographic information about a submitted address, city, state and ZIP Code, including time zone, MSA and PMSA codes. See page 16 for more information about this package.

Address Mailing: This package standardizes and corrects a submitted address, city, state and ZIP Code, also returning detailed mailing information about the address, including Carrier Route, Congressional District and POSTNet barcode information. See page 19 for more information about this package.

Address Parse: This package takes a submitted street address and breaks it down into its component parts, including the street number, directional, street name, suffix, and suite name and number. See page 23 for more information about this package.

Cities in State: This packages matches a partial city name and state and returns a list of city names in a state that match the same pattern. In other words, “Los A*” and “CA” would return “Los Angeles” and “Los Altos”, among others. This package and ZIPs In City cannot be selected together. See page 27 for more information about this package.

LACS^{Link}: The LACS^{Link} system allows you to convert business and residential rural route addresses to street-style addresses for easy locating by emergency systems. With this package, LACS input addresses will automatically be converted to its street-style equivalent.

Name Parse: This package breaks a person’s name into its component parts and supplies gender information. It can also indicate if the submitted data contains vulgar words. See page 34 for more information about this package.

Residential Delivery Indicator: This packages returns a one-letter code indicating whether the submitted address is a residence or a business, if this information is available. See page 38 for more information about this page.

Street Data All Records: This package takes a street name pattern and returns a list of valid address ranges that fall on the same street within the same five-digit ZIP Code. This package cannot be selected in conjunction with Street Data In Range Records Only and Street Data On Address Errors Only. See page 41 for more information on this package.

Street Data In Range Records Only: This package returns an array of all street records in a city where the street name matches a submitted pattern and address range. This package cannot be selected in conjunction with Street Data All Records or Street Data On Address Errors Only. See page 47 for more information on this package.

Street Data On Address Errors Only: This package can only be selected in conjunction with Address Mailing above. If the submitted address is not a valid address for any reason, the package will return data identical to Street Data All Records instead. This pack-

age cannot be selected in conjunction with Street Data All Records or Street Data In Range Records Only. See page 53 for more information on this package.

Telephone Number: This package returns geographic information based on a submitted telephone number, including latitude and longitude of the area code/prefix, time zone, MSA and PMSA codes. It will also break down a submitted telephone number into area code, prefix, suffix and extension and supply a new area code if the number has been affected by a recent area code split. See page 59 for more information on this package.

ZIP Code Information: This package return geographic information based on a submitted ZIP Code including latitude and longitude of the ZIP™ centroid, time zone, MSA and PMSA codes, USPS last line information and facility information. See page 64 for more information about this package.

ZIPs In City: This package returns a list of all valid ZIP Codes for a submitted city and state combination. This package and Cities In State cannot be selected together. See page 68 for more information about this package.

Substitution of Special Characters

The Data Quality Web Service is XML-based and certain text characters have special meanings. As a result, these characters cannot be passed through the service directly. Instead, the web service must subsitute standard HTML character codes for these characters.

The following table lists the characters and the codes used to replace them.

Chatacter	HTML Code
&	&
‘ (apostrophe)	'
<	<
>	>
“ (double quote)	"

Submitting Data to DQWS

Data is submitted to the DQWS and returned using a SOAP (Simple Object Access Protocol) interface, therefore users of Visual Studio .NET will be able to add the web service to their project easily, directly through the development environment. Many other programming languages have SOAP interface development kits available.

In either case, the procedure for submitting a request to the DQWS and processing the returned data follows the same basic structure. Our examples show a request built and processed using Visual Basic .NET but the same principles will apply to other programming languages.

Get a Customer ID from Melissa Data

Before you can begin using the DQWS, you must first call Melissa Data Customer Service at 1-800-MELISSA (635-4772) and acquire a customer ID number. This number must be submitted along with each request to the web service.

Add DQWS to Your Project

If you are using 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:

Secure: `https://ws.melissadata.com/dqws/address.asmx`
Non-Secure: `http://ws.melissadata.com/dqws/address.asmx`

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

Create an Instance of the DQWS objects

DQWS needs three objects in order to process your request and return the results: the mdWebService object, the Request object and the Response Object.

```
'create the web service object
Dim Action as New mdWebService.mdWebServices
'create the request object
Dim Request as New mdWebService.RequestRecord
'create the response object
Dim Response as New mdWebService.ResponseRecord
```

Populate the Request Object's General Properteries

The Request object has several properties that apply to every request, regardless of the requirements of the package or packages requested. Some are required and some are optional.

CustomerID: String value

```
Request.CustomerID = "1234567890"
```

This property must included the exact customer ID you were given by Melissa Data Customer Service in order for your request to be processed.

TransmissionReference: String value

```
Request.TransmissionReference = "20Aug2004A"
```

This field is optional and can include any string value you want. It will be returned with the response object to aid you in correctly matching the response with the associated request.

Action: String value

```
Request.Action = "Request"
```

This field is used to identify object being submitted as a request.

Timeout: Integer value

```
Request.Timeout = 10
```

Timeout sets the number of seconds that the webservice will attempt to process your request before returning an error.

Version: String value

```
Request.Version = "1.0"
```

Version identifies the version of the web service in use.

Identify the Packages to be Requested

The Request object has an associated boolean value for each package. Each package is set to false by default. Setting a packages' boolean value to true results in that package being returned as part of the Response object. The properties for each package are:

Package	Property
Address Delivery Point Validation	pkgAddressDeliveryPointValidation
Address GeoCode	pkgAddressGeoCode
Address Geographic Location	pkgAddressGeographicLocation
Address Mailing	pkgAddressMailing
Address Parse	pkgAddressParsed
Cities In State	pkgZipCodeCitiesInState
LACS^{Link}	pgkLacslink
Name Parse	pkgNameParsed
Residential Delivery Indicator	pkgResidentialDeliveryIndicator
Street Data All Records	pkgStreetDataAllRecords
Street Data In Range Records Only	pkgStreetDataInRangeRecordsOnly
Street Data On Address Errors Only	pkgStreetDataOnAddressErrorsOnly
Telephone Number	pkgTelephoneNumber
ZIP Code Information	pkgZipCodeInformation
ZIPs in City	pkgZipCodeZipInCity

Populate the Input Properties of the Request Object

These properties contain the information to be sent to the DQWS for processing. Not all of these properties are required for every package. See the chapters on each package for information about which properties are required and which are optional.

Send The Request

The `doSingleRecord` method of the Web Service object returns a Response object with the results of your request. Using the example object variables from page 4:

```
Response = Action.doSingleRecord(Request)
```

Handling The Response Object

The Response object will contain the results of your request to the web service. The exact structure of the response will depend on the package requested, but the Response Object has five distinct interfaces, one of which is used by each Request package.

Address: Used by Address Delivery Point Validation, Address Geocode, Address Geographic Area, Address Mailing, Address Parse and Street Data On Address Errors Only (when the submitted address is valid).

Name: Used by Name Parse.

StreetData: Used by Street Data All Records, Street Data In Range Records Only and Street Data On Address Errors Only (when the submitted address is not valid).

Telephone: Used by Telephone Number.

ZipCodeData: Used by Cities in State, ZIP Code Information and ZIPs in City.

See the relevent section on each package for exact details on handling the data returned.

Error Messages

Some packages return specific status and error messages that alert you to problems with processing your request. See “DQWS Messages” on page 71 for information on messages returned by individual packages

The web service also returns error messages of its own to the following properties of the Response object.

Error Code

```
ErrorCode = Response.ServiceResult.ErrorCode
```

Error Description

```
ErrorDesc = Response.ServiceResult.ErrorDesc
```

These are the possible values for errors returned to these properties:

Error Code	Error Description
401	Unauthorized Access - The customer ID was invalid or not recognized. Check your CustomerID string to verify that it is correct.
700	Method Contradiction - Two packages were selected that cannot be run simultaneously.
900	General Error - Service could not complete the request, usually due to missing input data. The error desciption will contain the name of the missing property.

Batch Processing

Records can be submitted to DQWS in one of two ways: one at a time or in batches of up to 100 records. Sending a large number of records with one request will save network transmission time compared to submitting each record separately.

The following packages can be used with a batch request: **Address DPV**, **Address GeoCode**, **Address Geographic Location**, **Address Mailing**, **Address Parse**, **Name Parse**, **Residential Delivery Indicator** and **Telephone Number**.

The process for submitting multiple records is similar to submitting a single record, but there are a few changes and additional steps.

Submitting a Batch Request

When submitting a batch request, you must create instances of two additional objects in addition to the Response and Request objects described above.

```
'Create instance of request array.
Dim RequestMultiple As New dqws.RequestArray
'Create instance of response array
Dim ResponseMultiple As New dqws.ResponseArray
```

The RequestArray object contains the **Record** property, an array of Request objects, plus the general properties of the Request object, such as **CustomerID** and **TransmissionReference**. You populate these general properties for the RequestArray object only, instead of each individual Request object.

There are also a couple of properties that are unique to the RequestArray object.

The first is **TotalRecords**, which must contain the exact number of Request object records contained in the RequestArray object. The number must be less than or equal to 100. If you attempt to submit more than 100 records, DQWS will return an error and not process any records.

```
RequestMultiple.TotalRecords = 100
```

In this instance, the array index for RequestMultiple.Record would then go from 0 to 99.

The second unique RequestArray property is **pkgMultipleRecords**. This property must be set to True.

```
RequestMultiple.pkgMultipleRecords = True
```

Each Request object is populated exactly as you would when submitting a single record as described above. After assigning values to the properties, you would then add the Request object to the Record array.

```
RequestMultiple.Record(0) = Request
```

You can then re-initialize the Request object to ready it for the next record.

```
Request = New dqws.RequestRecord
```

Now you can populate the object with data from the next record and add it to the array.

When you have finished populating the RequestArray object, it's time to submit it to the web service.

```
ResponseMultiple = Action.doMultipleRecords(RequestMultiple)
```

Processing the Response from a Batch Request

The ResponseArray object is similar to the RequestArray object, in that it contains a property consisting of an array of Response object, one for each request objected submitted to the web service. Simply extract each record from the array, like this:

```
Response = ResponseMultiple.Record(0)
```

You can then process the returned data just as you would a single response. See the chapter about each package for information about the information returned by that package.

Debug Mode

DQWS has a Debug mode that can be used for troubleshooting purposes. It allows you to verify the correct property names to use when processing the return object.

When debug mode is enabled, DQWS will return all possible fields to the Response object. Only those that apply to the requested packages will be populated with data.

Enabling Debug Mode

To send a request to DQWS with debug mode enabled, set the Debug property of the Request object to True.

```
Request.Debug = True
```

For requests sent in Batch mode, use the Debug property of the RequestArray object.

Chapter 2

Address DPV™

The Address DPV package enables you to verify that an actual address exists, right down to the apartment or suite number. In addition, DPV™ can also identify an address location as a Commercial Mail Receiving Agency (CMRA). Addresses flagged as CMRA could indicate possible fraud.

This package would be useful if you were doing a mass mailing and did not want to waste postage due to undeliverable addresses. This would save you both the cost of postage and materials.

Take the following scenario involving 3 addresses located on the same block:

Example 1: A single family residence located at:

123 Main St
Anytown, CA 92688

This would return the following DPV Footnote Codes: AA, BB, signifying that the address was coded to the ZIP + 4 file and the DPV file.

Example 2: An empty lot located next door at:

125 Main St
Anytown, CA 92688

Would return the following DPV Footnote Codes: AA, signifying that the address was coded to the ZIP + 4 file only. (While this address is for an empty lot, the address still matches to the ZIP + 4 file due to the use of address ranges. For example, 101 - 199 Main St would certainly include 125 Main St.)

Example 3: A mailbox in a CMRA located on the same block at:

137 Main St. #2112
Anytown, CA 92688

Would return the following DPV Footnote Codes: AA, BB, RR, signifying that the address was coded to the ZIP + 4 file, the DPV file, and was flagged as a CMRA.

The Address Delivery Point Validation Package may be used alone or in conjunction with any other package offered by the DQWS.

Required Request Properties

```
Request.CustomerID = String
Request.Address = String
Request.City = String
Request State = String
Request.Zip5 = String
```

Response Properties Returned

```
String = Response.Address.DPV.AddressStatus
String = Response.Address.DPV.CMRA
String = Response.Address.DPV.Footnotes
```

Request Object Syntax

The following code example shows a request for the Address DPV package with the required properties populated with valid data.

```
Dim Action As New mdWebService.mdWebServices
'request
Dim Request As New mdWebService.RequestRecord
'response
Dim Response As New mdWebService.ResponseRecord

'select package by setting boolean property to true
Request.pkgAddressDeliveryPointValidation = True
Request.action = "Request"
Request.version = "1.0"
Request.timeout = 10

'customerID required for every request
'supplied by Melissa Data Customer Service
Request.CustomerID = "1234567890"

'use TransmissionReference to assign your own identifier to a
'request
Request.transmissionReference = "ABCD1234"

'populate the input properties with the data to be processed
Request.FullName = _Name
Request.City = _CityName
Request.State = _State
Request.Zip5 = _Zip5
Request.Address = _Address
Request.Address2 = _Address2
Request.Country = _Country

'submit the request to the web service
Response = Action.doSingleRecord(Request)
```

Processing the Response Object

The Response object will contain the data returned by this package.

Address DPV returns the following properties listed below. A full description of each property can be found in the **Output Properties** chapter starting on page 85.

Sample Code

The following structure illustrates one way to extract the data returned:

```
With Response.Address.DPV
      _Status = .AddressStatus
      _CMRA =.CMRA
      _DPVFootnotes = .Footnotes
End With
```

Chapter 3

Address GeoCode

The Address GeoCode package enables you to discover more information about the data in your database. This is done through the use of the County FIPS, Census Tract, and Census Block data returned from the Address GeoCode package. Armed with these three pieces of information you can access the U.S. Census data to discover a veritable cornucopia of information pertaining to your records. Here is just a small example of what you can discover:

- How many homes are owner occupied vs. renter occupied in a given block group.
- How many children and what ages live with in a given block group.
- Average home values & household income within a given block group.

The Census data is free, for more information go to www.census.gov

Latitude and Longitude are available for both U.S. and Canadian addresses, but the other response properties in this package are returned with U.S. addresses only. Address GeoCode can be used alone or in conjunction with other packages.

Required Request Properties

```
Request.Zip5 = String
Request.Country = String
```

Response Properties Returned

```
String = Response.Address.GeoCode.Latitude
String = Response.Address.GeoCode.Longitude
String = Response.Address.Census.Block
String = Response.Address.Census.Tract
String = Response.Address.County.Fips
String = Response.Address.County.Name
```

Request Object Syntax

The following code example shows a request for the Address GeoCode package with the required properties populated with valid data.

```
Dim Action As New mdWebService.mdWebServices
'request
Dim Request As New mdWebService.RequestRecord
'response
Dim Response As New mdWebService.ResponseRecord

'select package by setting boolean property to true
Request.pkgAddressGeoCode = True
Request.action = "Request"
Request.version = "1.0"
Request.timeout = 10

'customerID required for every request
'supplied by Melissa Data Customer Service
Request.CustomerID = "1234567890"

'use TransmissionReference to assign your own identifier to a
'request
Request.transmissionReference = "ABCD1234"

'populate the input properties with the data to be processed
Request.Zip5 = _Zip5
Request.Plus4 = _Plus4

'populate the input properties with the data to be processed
Response = Action.doSingleRecord(Request)
```

Processing the Response Object

The Response object will contain the data returned by this package.

Address GeoCode returns the following properties listed below. A full description of each property can be found in the **Output Properties** chapter starting on page 85.

Sample Code

The following structure illustrates one way to extract the data returned:

```
With Response.Address
      _Latitude = .GeoCode.Latitude
      _Longitude = .GeoCode.Longitude
      _Block = .Census.Block
      _Tract = .Census.Tract
      _Fips = .County.Fips
      _County = .County.Name
      _PlaceCode = GeoCode.PlaceCode
      _PlaceName = GeoCode.PlaceName
End With
```

Chapter 4

Address Geographic Area

The Address Geographic package enables you to discover information specific to the geographic area in which the address resides. Address Geographic Area package returns the following information:

- Time Zone - Name of the Time Zone
- Time Zone Code - Numerical code used to designate the Time Zone
- CBSA - Core Based Statistical Area
- MSA - Metropolitan Statistical Area
- PMSA - Primary Metropolitan Statistical Area
- Congressional District Number

This information is pivotal to marketing in the following ways:

- Time Zone/Time Zone Code - When conducting an outbound telephone campaign you want to make sure you are not calling too early or too late in the day.
- CBSA, MSA and PMSA - Useful for determining which markets you would like to market to when utilizing Radio & Television advertising. CBSA data is based on the most recent census data but MSA and PMSA are maintained for legacy applications.

This package is for use with U.S. addresses only and can be used alone or in conjunction with other packages.

Required Request Properties

```
Request.CustomerID = String
Request.Address = String
Request.City = String
Request State = String
Request.Zip5 = String
```

Response Properties Returned

```
String = Response.Address.TimeZone.Name
String = Response.Address.TimeZone.Code
String = Response.Address.Msa.Code
String = Response.Address.Pmsa.Code
String = Response.Address.CongressionalDistrict.Code
String = Response.Address.CBSA.Code
String = Response.Address.CBSA.Title
String = Response.Address.CBSA.Level
String = Response.Address.CBSA.CBSADivision.Code
String = Response.Address.CBSA.CBSADivision.Title
String = Response.Address.CBSA.CBSADivision.Level
```

Request Object Syntax

The following code example shows a request for the Address Geographic Area package with the required properties populated with valid data.

```
Dim Action As New mdWebService.mdWebServices
'request
Dim Request As New mdWebService.RequestRecord
'response
Dim Response As New mdWebService.ResponseRecord

'select package by setting boolean property to true
Request.pkgAddressGeographicArea = True
Request.action = "Request"
Request.version = "1.0"
Request.timeout = 10

'customerID required for every request
'supplied by Melissa Data Customer Service
Request.CustomerID = "1234567890"

'use TransmissionReference to assign your own identifier to a
'request
Request.transmissionReference = "ABCD1234"

'populate the input properties with the data to be processed
Request.FullName = _Name
Request.City = _CityName
Request.State = _State
Request.Zip5 = _Zip5
```

```
Request.Address = _Address
Request.Address2 = _Address2

'submit the request to the web service
Response = Action.doSingleRecord(Request)
```

Processing the Response Object

The Response object will contain the data returned by this package.

Address Geographic Area returns the following properties listed below. A full description of each property can be found in the **Output Properties** chapter starting on page 85.

Sample Code

The following structure illustrates one way to extract the data returned:

```
With Response.Address
      _ConDist = .CongressionalDistrict.Code
      _TimeZone = .TimeZone.Name
      _TZCode = .TimeZone.Code
      _MSA = .Msa.Code
      _PMSA = .Pmsa.Code
      _CBSACode = .CBSA.Code
      _CBSATitle = .CBSA.Title
      _CBSALevel = .CBSA.Level
      _CBSADivision = .CBSA.CBSADivision.Code
      _CBSADivisionTitle = .CBSA.CBSADivision.Title
      _CBSADivisionLevel = .CBSA.CBSADivision.Level
End With
```

Chapter 5

Address Mailing

The Address Mailing package help prepare your address data for mailing. Correcting misspellings, adding missing information, standardizing, and validating your address data will help cut costs in returned postage and expensive marketing materials. All records are validated against either the U.S.P.S. or Canada Post national data files which are updated monthly. The Address Mailing package also helps cut down on data entry errors by reducing the amount of keystrokes needed to verify an address. Simply input an address and ZIP/Postal Code and the Address Mailing package will return the city/municipality and state/province plus much more.

Input:

Field	Value
Address	22382 Empresa
ZIP/Postal Code	92688

Output:

Field	Value
Address	22382 Avenida Empresa
City	Rancho Santa Margarita
City Abbreviation	Rcho Sta Marg
State	CA
ZIP/Postal Code	92688
Plus4	2112
Carrier Route	C056
Delivery Point:	82
Delivery Point Check Digit	1
Country	US

And much more….

This package is for use with U.S. & Canadian addresses only and can be used alone or in conjunction with other packages.

Required Request Properties

```
Request.CustomerID = String
Request.Address = String
Request.City = String
Request State = String
Request.Zip5 = String
```

Response Properties Returned

```
String = Response.Address.Street
String = Response.Address.Street2
String = Response.Address.Company
String = Response.Address.CarrierRoute
String = Response.Address.City.Abbreviation
String = Response.Address.City.Name
String = Response.Address.Country.Abbreviation
String = Response.Address.Country.Name
String = Response.Address.County.Fips
String = Response.Address.County.Name
String = Response.Address.DeliveryPointCheckDigit
String = Response.Address.DeliveryPointCode
String = Response.Address.Lacs
String = Response.Address.PrivateMailBox
String = Response.Address.State.Abbreviation
String = Response.Address.State.Name
String = Response.Address.Suite
String = Response.Address.Type.Code
String = Response.Address.Type.String
String = Response.Address.Urbanization.Code
String = Response.Address.Urbanization.Name
String = Response.Address.Zip.Plus4
```

```
String = Response.Address.Zip.Type
String = Response.Address.Zip.Zip5
```

Request Object Syntax

The following code example shows a request for the Address Mailing package with the required properties populated with valid data.

```
Dim Action As New mdWebService.mdWebServices
'request
Dim Request As New mdWebService.RequestRecord
'response
Dim Response As New mdWebService.ResponseRecord

'select package by setting boolean property to true
Request.pkgAddressMailing = True
Request.action = "Request"
Request.version = "1.0"
Request.timeout = 10

'customerID required for every request
'supplied by Melissa Data Customer Service
Request.CustomerID = "1234567890"

'use TransmissionReference to assign your own identifier to a
'request
Request.transmissionReference = "ABCD1234"

'populate the input properties with the data to be processed
Request.FullName = _Name
Request.City = _CityName
Request.State = _State
Request.Zip5 = _Zip5
Request.Address = _Address
Request.Address2 = _Address2
Request.Country = _Country

'submit the request to the web service
Response = Action.doSingleRecord(Request)
```

Processing the Response Object

The Response object will contain the data returned by this package.

Address Mailing returns the following properties listed below. A full description of each property can be found in the **Output Properties** chapter starting on page 85.

Sample Code

The following structure illustrates one way to extract the data returned:

```
With Response.Address
      _Street = .Street
      _Street2 = .Street2
      _Company = .Company
      _CarrierRoute = .CarrierRoute
      _City = .City.Name
      _CityAbb = .City.Abbreviation
      _Country = .Country.Abbreviation
      _County = .County.Name
      _CountyFips = .County.Fips
      _DelPtCode = .DeliveryPointCode
      _DelPtCheck = .DeliveryPointCheckDigit
      _Lacs = .Lacs
      _PMB = .PrivateMailBox
      _State = .State.Abbreviation
      _Suite = .Suite
      _AddType = .Type.String
      _AddTypeCode = .Type.Code
      _UrbCode = .Urbanization.Code
      _Zip5= .Zip.Zip5
      _Plus4 = .Zip.Plus4
      _ZipType = .Zip.Type
End With
```

Chapter 6

Address Parse

The Address Parse package returns the street address portion of a submitted address, broken down into its component parts.

Use this pack to enhance the sorting and grouping of your mailing list by separating out the street number, pre-directional, street name, street suffix, post-directional, suite name, and suite number.

Parsed address information is very useful when you wish to do the following:

- Target addresses only on Main Street
- Target addresses only on North Main Street
- Target addresses only on the 1000 block of North Main Street

The Address Parse package will first validate the address against the USPS or Canada Post data sets and then return the components of the correctly standardized address.

Example: **Input Address:** When the following address is submitted to the web service

647 Camino De Los Mares Ste 108 PMB 4500
San Clemente, CA 92673

Returned Address Components:

Field	Value
Address Range	647
Street Name	Camino De Los Mares
Suite Name	Suite
Suite Number	108
Private Mailbox	PMB 252

This package is for use with U.S. & Canadian addresses only and can be used alone or in conjunction with other packages.

Required Request Properties

```
Request.CustomerID = String
Request.Address = String
```

Response Properties Returned

```
String = Response.Address.Parsed.AddressRange
String = Response.Address.Parsed.Suffix
String = Response.Address.Parsed.Direction.Post
String = Response.Address.Parsed.Direction.Pre
String = Response.Address.Parsed.Garbage
String = Response.Address.Parsed.PrivateMailBox.Name
String = Response.Address.Parsed.PrivateMailBox.Range
String = Response.Address.Parsed.StreetName
String = Response.Address.Parsed.Suite.Name
String = Response.Address.Parsed.Suite.Range
```

Request Object Syntax

The following code example shows a request for the Address Parse package with the required properties populated with valid data.

```
Dim Action As New mdWebService.mdWebServices
'request
Dim Request As New mdWebService.RequestRecord
'response
Dim Response As New mdWebService.ResponseRecord

'select package by setting boolean property to true
Request.pkgAddressParsed = True
Request.action = "Request"
Request.version = "1.0"
Request.timeout = 10

'customerID required for every request
'supplied by Melissa Data Customer Service
```

```
Request.CustomerID = "1234567890"
'use TransmissionReference to assign your own identifier to a
'request
Request.transmissionReference = "ABCD1234"

'populate the input properties with the data to be processed
Request.Address = _Address

'submit the request to the web service
Response = Action.doSingleRecord(Request)
```

Processing the Response Object

The Response object will contain the data returned by this package.

Address Parse returns the following properties listed below. A full description of each property can be found in the **Output Properties** chapter starting on page 85.

Sample Code

The following structure illustrates one way to extract the data returned:

```
With Response.Address
      _Range = .Parsed.AddressRange
      _Suffix = .Parsed.Suffix
      _DirPost = .Parsed.Direction.Post
      _DirPre = .Parsed.Direction.Pre
      _Garbage = .Parsed.Garbage
      _PMBName = .Parsed.PrivateMailBox.Name
      _PMBNumber = .Parsed.PrivateMailBox.Range
      _StreetName = .Parsed.StreetName
      _SuiteName = .Parsed.Suite.Suite.Name
      _Suite = .Parsed.Suite.Range.Range
End With
```

Chapter 7

Cities In State

The Cities in State package returns an array of every city in a state where the city name matches a submitted pattern. The submitted pattern may contain the two standard wildcards:

* = matches up with any combination of allowable character

? = matches up with any single allowable character

The Cities in State package returns the following fields:

Count - The total number of records returned in the array. This tag is only returned once for each array while the remainder of the fields will be present in each record within the array.

City Name - The official city name as recognized by the USPS

City Abbreviation - The official city abbreviation as recognized by the USPS. If the Official city name is 13 characters or less then this field will contain the complete spelling of the official city name.

State - The official state abbreviation as recognized by the USPS

Example: **Input:**

Field	Value
City	Rancho Sa*
State	CA

Output: Count: 2

Field	Value
Record(0)	
City	Rancho Santa Fe
City Abbreviation	Rcho Santa Fe
State	CA
Record(1)	
City	Rancho Santa Margarita
City Abbreviation	Rcho Sta Marg
State	CA

The Cities in State package applies to U.S. cities only and may be used alone or in conjunction with any other package offered by the DQWS.

Required Request Properties

```
Request.CustomerID = String
Request.City = String
Request State = String
```

Response Properties Returned

```
String = Response.ZipCodeData.count
String = Response.ZipCodeData.Record(idx).City.Name
String = Response.ZipCodeData.Record(idx).City.Abbreviation
String = Response.ZipCodeData.Record(idx).State.Abbreviation
```

Request Object Syntax

The following code example shows a request for the Cities In State package with the required properties populated with valid data.

```
Dim Action As New mdWebService.mdWebServices
'request
Dim Request As New mdWebService.RequestRecord
'response
Dim Response As New mdWebService.ResponseRecord

'select package by setting boolean property to true
Request.ZipCodeCitiesInState = True
Request.action = "Request"
Request.version = "1.0"
Request.timeout = 10

'customerID required for every request
'supplied by Melissa Data Customer Service
Request.CustomerID = "1234567890"

'use TransmissionReference to assign your own identifier to a
'request
Request.transmissionReference = "ABCD1234"

'populate the input properties with the data to be processed
Request.City = _CityName
Request.State = _State

'submit the request to the web service
Response = Action.doSingleRecord(Request)
```

Processing the Response Object

The Response object will contain the data returned by this package.

Cities In State returns the following properties listed below. A full description of each property can be found in the **Output Properties** chapter starting on page 85.

Sample Code

The following structure illustrates one way to extract the data returned:

```
With Response.ZipCodeData
      'cycle through the records returned by the response
      For I = 0 to .count - 1
            With .Record(I)
                  _City(I) = .City.Name
                  _CityAbb(I) = .City.Abbreviation
                  _State(I) = .State.Abbreviation
            End With
      Next
End With
```

Chapter 8

LACS^{Link}

LACS^{Link} is a process where some rural route addresses are modified to city-style addresses to allow emergency services (for example, ambulance, police, fire, and so on) to find these addresses more efficiently.

The LACS^{Link} service matches the old address with the updated address and corrects it as part of the address checking process.

This package will automatically updates all updated addresses submitted to the web service when used with the Address Mailing package. The Address Mailing package must be requested in order to use the LACS^{Link} package.

Required Request Properties

Any request calling the LACS^{Link} package must also call the Address Mailing package, therefore it requires the same request properties as the Address Mailng package. See page 20 for more information.

Response Properties Returned

```
String = Response.Address.Lacslink.LacsStatusCode
String = Response.Address.LacsLink.LacsReturnCode
```

Request Object Syntax

The following code example shows a request for the LACS[Link] package with the required properties populated with valid data.

```
Dim Action As New mdWebService.mdWebServices
'request
Dim Request As New mdWebService.RequestRecord
'response
Dim Response As New mdWebService.ResponseRecord

'select package by setting boolean property to true
Request.pkgAddressMailing = True
Request.pkgLacsLink = True
Request.action = "Request"
Request.version = "1.0"
Request.timeout = 10

'customerID required for every request
'supplied by Melissa Data Customer Service
Request.CustomerID = "1234567890"

'use TransmissionReference to assign your own identifier to a
'request
Request.transmissionReference = "ABCD1234"

'populate the input properties with the data to be processed
Request.FullName = _Name
Request.City = _CityName
Request.State = _State
Request.Zip5 = _Zip5
Request.Address = _Address
Request.Address2 = _Address2
Request.Country = _Country

'submit the request to the web service
Response = Action.doSingleRecord(Request)
```

Processing the Response Object

The Response object will contain the data returned by this package.

LACS^{Link} returns the following properties listed below. A full description of each property can be found in the **Output Properties** chapter starting on page 85.

LACS Return Code (U.S. Only) .....see page 100
LACS Status Code (U.S. Only).....see page 101

Sample Code

The following structure illustrates one way to extract the data returned:

```
With Response.Address.LacsLink
      _Status = .LacsStatusCode
      _ReturnCode = .LascReturnCode
End With
```

Chapter 9

Name Parse

The Name Parse package allows you to:

- Parse a persons name into its individual elements. This allows you to sort by last name even if your customer enter their full name as a single string value.
- In most cases, assign gender to greatly increase the accuracy of gender specific marketing
- Prevent garbage names from entering your database by flagging vulgar words.

The Name Parse package returns the following fields:

Prefix
First Name
Middle Name
Last/Surname
Suffix
Gender

Example: **Input:** mr. john wayne brown phd

Output:

Field	Value
Prefix	Mr.
First Name	John
Middle Name	Wayne
Last Name	Brown
Suffix	PhD
Gender	M

The Name Parse package may be used alone or in conjunction with any other package offered by the DQWS.

Required Request Properties

```
Request.CustomerID = String
Request.FullName = String
```

Response Properties Returned

```
String = Response.Name.First
String = Response.Name.Full
String = Response.Name.Gender
String = Response.Name.Last
String = Response.Name.Middle
String = Response.Name.Prefix
String = Response.Name.Suffix
```

Request Object Syntax

The following code example shows a request for the Name Parse package with the required properties populated with valid data.

```
Dim Action As New mdWebService.mdWebServices
'request
Dim Request As New mdWebService.RequestRecord
'response
Dim Response As New mdWebService.ResponseRecord

'select package by setting boolean property to true
Request.pkgNameParsed = True
Request.action = "Request"
Request.version = "1.0"
Request.timeout = 10

'customerID required for every request
'supplied by Melissa Data Customer Service
Request.CustomerID = "1234567890"

'use TransmissionReference to assign your own identifier to a
'request
Request.transmissionReference = "ABCD1234"

'populate the input properties with the data to be processed
Request.FullName = _FullName

'submit the request to the web service
Response = Action.doSingleRecord(Request)
```

Processing the Response Object

The Response object will contain the data returned by this package.

Name Parse returns the following properties listed below. A full description of each property can be found in the **Output Properties** chapter starting on page 85.

Sample Code

The following structure illustrates one way to extract the data returned:

```
With Response.Name
      _First = .First
      _Gender = .Gender
      _Last = .Last
      _Middle = .Middle
      _Prefix = .Prefix
      _Suffix = .Suffix
End With
```

Chapter 10

Residential Business Delivery Indicator

The Residential Business Delivery Indicator package returns a one letter code indicating whether a submitted address is a residential or business address. This information is useful when calculating shipping charges since carriers such as UPS and FedEx include a surcharge for residential deliveries.

Input:

Field	Value
Address:	22382 Avenida Empresa
City:	Rancho Santa Margarita
State:	CA
ZIP Code:	92688

Output:

Field	Value
Delivery Indicator Code	B

The Residential Business Delivery Indicator is for use with U.S. addresses only and can be used alone or in conjunction with other packages.

Required Request Properties

```
Request.CustomerID = String
Request.Address = String
Request.City = String
Request State = String
Request.Zip5 = String
```

Response Properties Returned

```
String = Response.Address.DeliveryIndicator.Code
```

Request Object Syntax

The following code example shows a request for the Residential Business Delivery Indicator package with the required properties populated with valid data.

```
Dim Action As New mdWebService.mdWebServices
'request
Dim Request As New mdWebService.RequestRecord
'response
Dim Response As New mdWebService.ResponseRecord

'select package by setting boolean property to true
Request.pkgDeliveryIndicator = True
Request.action = "Request"
Request.version = "1.0"
Request.timeout = 10

'customerID required for every request
'supplied by Melissa Data Customer Service
Request.CustomerID = "1234567890"

'use TransmissionReference to assign your own identifier to a
'request
Request.transmissionReference = "ABCD1234"

'populate the input properties with the data to be processed
Request.City = _CityName
Request.State = _State
Request.Zip5 = _Zip5
Request.Address = _Address
Request.Address2 = _Address2
Request.Country = _Country

'submit the request to the web service
Response = Action.doSingleRecord(Request)
```

Processing the Response Object

Residential Business Delivery Indicator returns the following properties listed below. A full description of each property can be found in the **Output Properties** chapter starting on page 85.

Delivery Indicator .....see page 95

Sample Code

The following structure illustrates one way to extract the data returned:

```
With Response.Address
      _RBD = .DeliveryIndicator.Code
End With
```

Chapter 11

Street Data All Records

The Street Data All Records package returns an array of all street records in a city where the street name matches a submitted pattern. The submitted pattern may contain the two standard wildcards:

* = matches up with any combination of allowable character

? = matches up with any single allowable character

The Street Data All Records package returns the following fields:

Note: Only one record shown for example purposes.

Example

Input: The following data is submitted to the web service.

Field	Value
Street	223 Empresa
ZIP Code	92688

Output: The following record is returned.

Field	Value
Count	31
Record #1	
Id	1
Company	Empty
Urbanization Code	Empty
AddressType	S

Field	Value
BaseAlternateIndicator	B
CarrierRoute	C056
CongressionalDistrict	42
County Fips	06059
Lacs	Empty
LastLine Number	Z22621
Primary Range Low	0000022300
Primary Range High	0000022398
Primary Range OddEven	E
Pre Direction	Empty
Street Name	EMPRESA
Post Direction	Empty
Street Suffix	AVDA
Suite Name	Empty
Suite Range Low	Empty
Suite Range High	Empty
Suite Range OddEven	Empty
Zip5	92688
Plus4.Low	2112
Plus4.High	2112

The Street Data All Records package applies to U.S. only and may be used alone or in conjunction with any other package offered by the DQWS except the Street Data in Range Records Only and Street Data on Address Errors Only packages.

Required Request Properties

```
Request.CustomerID = String
Request.Address = String
Request.City = String
Request State = String
Request.Zip5 = String
```

Response Properties Returned

```
String = Response.StreetData.count
String = Response.StreetData.Record(idx).AddressType
String = Response.StreetData.Record(idx).BaseAlternateIndicator
String = Response.StreetData.Record(idx).CarrierRoute
String = Response.StreetData.Record(idx).Company
String = Response.StreetData.Record(idx).CongressionalDistrict.Code
String = Response.StreetData.Record(idx).County.Fips
String = Response.StreetData.Record(idx).Direction.Post
String = Response.StreetData.Record(idx).Direction.Pre
String = Response.StreetData.Record(idx).Lacs
String = Response.StreetData.Record(idx).LastLineNumber
String = Response.StreetData.Record(idx).PreferredLastLineNumber
String = Response.StreetData.Record(idx).PrimaryRange.High
String = Response.StreetData.Record(idx).PrimaryRange.Low
String = Response.StreetData.Record(idx).PrimaryRange.OddEven
String = Response.StreetData.Record(idx).Street.Name
String = Response.StreetData.Record(idx).Street.Suffix
String = Response.StreetData.Record(idx).Street.Suite.Range.High
String = Response.StreetData.Record(idx).Street.Suite.Range.Low
String = Response.StreetData.Record(idx).Street.Suite.Range.OddEven
String = Response.StreetData.Record(idx).Street.Suite.SuiteName
String = Response.StreetData.Record(idx).Type.Code
String = Response.StreetData.Record(idx).Type.String
String = Response.StreetData.Record(idx).Urbanization.Code
String = Response.StreetData.Record(idx).Zip.Zip5
String = Response.StreetData.Record(idx).Zip.Plus4.High
String = Response.StreetData.Record(idx).Zip.Plus4.Low
String = Response.StreetData.Record(idx).Zip.Type
```

Request Object Syntax

The following code example shows a request for the Street Data All Records package with the required properties populated with valid data.

```
Dim Action As New mdWebService.mdWebServices
'request
Dim Request As New mdWebService.RequestRecord
'response
Dim Response As New mdWebService.ResponseRecord

'select package by setting boolean property to true
Request.pkgStreetDataAllRecords = True
Request.action = "Request"
Request.version = "1.0"
Request.timeout = 10

'customerID required for every request
'supplied by Melissa Data Customer Service
Request.CustomerID = "1234567890"

'use TransmissionReference to assign your own identifier to a
'request
Request.transmissionReference = "ABCD1234"

'populate the input properties with the data to be processed
Request Address = _Address
Request.City = _CityName
Request.State = _State
Request.Zip5 = _Zip5
Request.Zip5 = _Zip5
Request.Plus4 = _Plus4

'submit the request to the web service
Response = Action.doSingleRecord(Request)
```

Remarks

This package cannot be selected as part of the same Request as the Street Data In Range Records Only package. Setting both properties of the Request Object to “True” will cause the service to return a error condition.

Processing the Response Object

The Response object will contain the data returned by this package.

Street Data All Records returns the following properties listed below. A full description of each property can be found in the **Output Properties** chapter starting on page 85.

Sample Code

The following structure illustrates one way to extract the data returned:

```
With Response.StreetData
      'cycle through the records returned by the response
      For I = 0 to .count - 1
            With Record(I)
                  _AddressType(I)   = .AddressType
                  _BaseAltInd(I)    = .BaseAlternateIndicator
                  _CarrierRt(I)     = .CarrierRoute
                  _ConDistrict(I)   = .CongressionalDistrict.Code
                  _CountyFips(I)    = .County.Fips
                  _PostDir(I)       = .Direction.Post
                  _PreDir(I)        = .Direction.Pre
                  _Lacs(I)          = .Lacs
                  _LastLine(I)      = .LastLineNumber
                  _PrefLastLine(I)  = .PreferredLastLineNumber
                  _RangeHigh(I)     = .PrimaryRange.High
                  _RangeLow(I)      = .PrimaryRange.Low
                  _RangeOddEven(I)  = .PrimaryRange.OddEven
                  _StreetName(I)    = .Street.Name
                  _StreetSuffix(I)  = .Street.Suffix
                  _SuiteHigh(I)     = .Street.Suite.Range.High
                  _SuiteLow(I)      = .Street.Suite.Range.Low
                  _SuiteOddEven(I)  = .Street.Suite.Range.OddEven
                  _SuiteName(I)     = .Street.Suite.SuiteName
                  _AddressType(I)   = .Type.Code
                  _Urb(I)           = .Urbanization.Code
                  _Zip5(I)          = .Zip.Zip5
                  _Plus4High(I)     = .Zip.Plus4.High
                  _Plus4Low(I)      = .Zip.Plus4.Low
                  _ZipType(I)       = .Zip.Type
            End With
      Next
End With
```

Chapter 12

Street Data In Range Records Only

The Street Data In Range Records Only package returns an array of all street records in a city where the street name matches a submitted pattern and address range. The submitted pattern may contain the two standard wildcards:

* = matches up with any combination of allowable character

? = matches up with any single allowable character

The Street Data In Range Records Only package returns the following fields:

Note: Only one record shown for example purposes.

Example

Input: The following data is submitted to the web service.

Field	Value
Street	223 Empresa
ZIP Code	92688

Output: The following record is returned.

Field	Value
Count	1
Record #1	
Id	1
Company	Empty
Urbanization Code	Empty
AddressType	S

Field	Value
BaseAlternateIndicator	B
CarrierRoute	C056
CongressionalDistrict	42
County Fips	06059
Lacs	Empty
LastLine Number	Z22621
Primary Range Low	0000022300
Primary Range High	0000022398
Primary Range OddEven	E
Pre Direction	Empty
Street Name	EMPRESA
Post Direction	Empty
Street Suffix	AVDA
Suite Name	Empty
Suite Range Low	Empty
Suite Range High	Empty
Suite Range OddEven	Empty
Zip5	92688
Plus4.Low	2112
Plus4.High	2112

The Street Data In Range Records Only package applies to U.S. only and may be used alone or in conjunction with any other package offered by the DQWS except the Street Data All Records and Street Data on Address Errors Only packages.

Required Request Properties

```
Request.CustomerID = String
Request.Address = String
Request.City = String
Request State = String
Request.Zip5 = String
```

Response Properties Returned

```
String = Response.StreetData.count
String = Response.StreetData.Record(idx).AddressType
String = Response.StreetData.Record(idx).BaseAlternateIndicator
String = Response.StreetData.Record(idx).CarrierRoute
String = Response.StreetData.Record(idx).Company
String = Response.StreetData.Record(idx).CongressionalDistrict.Code
String = Response.StreetData.Record(idx).County.Fips
String = Response.StreetData.Record(idx).Direction.Post
String = Response.StreetData.Record(idx).Direction.Pre
String = Response.StreetData.Record(idx).Lacs
String = Response.StreetData.Record(idx).LastLineNumber
String = Response.StreetData.Record(idx).PreferredLastLineNumber
String = Response.StreetData.Record(idx).PrimaryRange.High
String = Response.StreetData.Record(idx).PrimaryRange.Low
String = Response.StreetData.Record(idx).PrimaryRange.OddEven
String = Response.StreetData.Record(idx).Street.Name
String = Response.StreetData.Record(idx).Street.Suffix
String = Response.StreetData.Record(idx).Street.Suite.Range.High
String = Response.StreetData.Record(idx).Street.Suite.Range.Low
String = Response.StreetData.Record(idx).Street.Suite.Range.OddEven
String = Response.StreetData.Record(idx).Street.Suite.SuiteName
String = Response.StreetData.Record(idx).Type.Code
String = Response.StreetData.Record(idx).Type.String
String = Response.StreetData.Record(idx).Urbanization.Code
String = Response.StreetData.Record(idx).Zip.Zip5
String = Response.StreetData.Record(idx).Zip.Plus4.High
String = Response.StreetData.Record(idx).Zip.Plus4.Low
String = Response.StreetData.Record(idx).Zip.Type
```

Request Object Syntax

The following code example shows a request for the Street Data In Range Records Only package with the required properties populated with valid data.

```
Dim Action As New mdWebService.mdWebServices
'request
Dim Request As New mdWebService.RequestRecord
'response
Dim Response As New mdWebService.ResponseRecord

'select package by setting boolean property to true
Request.pkgStreetDataInRangeRecordsOnly = True
Request.action = "Request"
Request.version = "1.0"
Request.timeout = 10

'customerID required for every request
'supplied by Melissa Data Customer Service
Request.CustomerID = "1234567890"

'use TransmissionReference to assign your own identifier to a
'request
Request.transmissionReference = "ABCD1234"

'populate the input properties with the data to be processed
Request Address = _Address
Request.City = _CityName
Request.State = _State
Request.Zip5 = _Zip5
Request.Zip5 = _Zip5
Request.Plus4 = _Plus4

'submit the request to the web service
Response = Action.doSingleRecord(Request)
```

Remarks

This package cannot be selected as part of the same Request as the Street Data All Records Only package. Setting both properties of the Request Object to “True” will cause the service to return a error condition.

Processing the Response Object

The Response object will contain the data returned by this package.

Street Data In Range Records Only returns the following properties listed below. A full description of each property can be found in the **Output Properties** chapter starting on page 85.

Sample Code

The following structure illustrates one way to extract the data returned:

```
With Response.StreetData
      'cycle through the records returned by the response
      For I = 0 to .count - 1
            With Record(I)
                  _AddressType(I)   = .AddressType
                  _BaseAltInd(I)    = .BaseAlternateIndicator
                  _CarrierRt(I)     = .CarrierRoute
                  _ConDistrict(I)   = .CongressionalDistrict.Code
                  _CountyFips(I)    = .County.Fips
                  _PostDir(I)       = .Direction.Post
                  _PreDir(I)        = .Direction.Pre
                  _Lacs(I)          = .Lacs
                  _LastLine(I)      = .LastLineNumber
                  _PrefLastLine(I)  = .PreferredLastLineNumber
                  _RangeHigh(I)     = .PrimaryRange.High
                  _RangeLow(I)      = .PrimaryRange.Low
                  _RangeOddEven(I)  = .PrimaryRange.OddEven
                  _StreetName(I)    = .Street.Name
                  _StreetSuffix(I)  = .Street.Suffix
                  _SuiteHigh(I)     = .Street.Suite.Range.High
                  _SuiteLow(I)      = .Street.Suite.Range.Low
                  _SuiteOddEven(I)  = .Street.Suite.Range.OddEven
                  _SuiteName(I)     = .Street.Suite.SuiteName
                  _AddressType(I)   = .Type.Code
                  _Urb(I)           = .Urbanization.Code
                  _Zip5(I)          = .Zip.Zip5
                  _Plus4High(I)     = .Zip.Plus4.High
                  _Plus4Low(I)      = .Zip.Plus4.Low
                  _ZipType(I)       = .Zip.Type
            End With
      Next
End With
```

Chapter 13

Street Data On Address Errors Only

The Street Data On Address Errors Only package works in conjunction with the Address Mailing package. If the Address Mailing package is unable to code an address and the Street Data On Address Errors Only package is also selected then a list of possible streets records will be returned. The Street Data On Address Errors Only package functions exactly like the Street Data All Records package in the fact that it returns an array of all street records in a city where the street name matches.

The Street Data On Address Errors Only package returns the following fields:

Note: Only one record shown for example purposes.

Example

Input: The following data is submitted to the web service.

Field	Value
Street	223 Empresa
ZIP Code	92688

Output: The following record is returned.

Field	Value
Count	1
Record #1	
Id	1
Company	Empty
Urbanization Code	Empty
AddressType	S

Field	Value
BaseAlternateIndicator	B
CarrierRoute	C056
CongressionalDistrict	42
County Fips	06059
Lacs	Empty
LastLine Number	Z22621
Primary Range Low	0000022300
Primary Range High	0000022398
Primary Range OddEven	E
Pre Direction	Empty
Street Name	EMPRESA
Post Direction	Empty
Street Suffix	AVDA
Suite Name	Empty
Suite Range Low	Empty
Suite Range High	Empty
Suite Range OddEven	Empty
Zip5	92688
Plus4.Low	2112
Plus4.High	2112

The Street Data On Address Errors Only package applies to U.S. only and may only be used when the Address Mailing package is selected and may not be used in conjunction with the Street Data in Range Records Only and Street Data All Records packages.

Required Request Properties

```
Request.CustomerID = String
Request.Address = String
Request.City = String
Request State = String
Request.Zip5 = String
```

Response Properties Returned

```
String = Response.StreetData.count
String = Response.StreetData.Record(idx).AddressType
String = Response.StreetData.Record(idx).BaseAlternateIndicator
String = Response.StreetData.Record(idx).CarrierRoute
String = Response.StreetData.Record(idx).Company
String = Response.StreetData.Record(idx).CongressionalDistrict.Code
String = Response.StreetData.Record(idx).County.Fips
String = Response.StreetData.Record(idx).Direction.Post
String = Response.StreetData.Record(idx).Direction.Pre
String = Response.StreetData.Record(idx).Lacs
String = Response.StreetData.Record(idx).LastLineNumber
String = Response.StreetData.Record(idx).PreferredLastLineNumber
String = Response.StreetData.Record(idx).PrimaryRange.High
String = Response.StreetData.Record(idx).PrimaryRange.Low
String = Response.StreetData.Record(idx).PrimaryRange.OddEven
String = Response.StreetData.Record(idx).Street.Name
String = Response.StreetData.Record(idx).Street.Suffix
String = Response.StreetData.Record(idx).Street.Suite.Range.High
String = Response.StreetData.Record(idx).Street.Suite.Range.Low
String = Response.StreetData.Record(idx).Street.Suite.Range.OddEven
String = Response.StreetData.Record(idx).Street.Suite.SuiteName
String = Response.StreetData.Record(idx).Type.Code
String = Response.StreetData.Record(idx).Type.String
String = Response.StreetData.Record(idx).Urbanization.Code
String = Response.StreetData.Record(idx).Zip.Zip5
String = Response.StreetData.Record(idx).Zip.Plus4.High
String = Response.StreetData.Record(idx).Zip.Plus4.Low
String = Response.StreetData.Record(idx).Zip.Type
```

Request Object Syntax

The following code example shows a request for the Street Data On Address Errors Only package with the required properties populated with valid data.

```
Dim Action As New mdWebService.mdWebServices
'request
Dim Request As New mdWebService.RequestRecord
'response
Dim Response As New mdWebService.ResponseRecord

'select package by setting boolean property to true
Request.pkgStreetDataOnAddressErrorsOnly = True

'Address Mailing must be requested also
Request.pkgAddressMailing = True
Request.action = "Request"
Request.version = "1.0"

'customerID required for every request
'supplied by Melissa Data Customer Service
Request.timeout = 10
Request.CustomerID = "1234567890"

'use TransmissionReference to assign your own identifier to a
'request
Request.transmissionReference = "ABCD1234"

'populate the input properties with the data to be processed
Request Address = _Address
Request.City = _CityName
Request.State = _State
Request.Zip5 = _Zip5
Request.Zip5 = _Zip5
Request.Plus4 = _Plus4

'submit the request to the web service
Response = Action.doSingleRecord(Request)
```

Processing the Response Object

The Response object will contain the data returned by this package.

Street Data On Address Errors Only returns the following properties listed below. A full description of each property can be found in the **Output Properties** chapter starting on page 85.

Sample Code

The following structure illustrates one way to extract the data returned:

```
With Response.StreetData
      'cycle through the records returned by the response
      For I = 0 to .count - 1
            With Record(I)
                  _AddressType(I)   = .AddressType
                  _BaseAltInd(I)    = .BaseAlternateIndicator
                  _CarrierRt(I)     = .CarrierRoute
                  _ConDistrict(I)   = .CongressionalDistrict.Code
                  _CountyFips(I)    = .County.Fips
                  _PostDir(I)       = .Direction.Post
                  _PreDir(I)        = .Direction.Pre
                  _Lacs(I)          = .Lacs
                  _LastLine(I)      = .LastLineNumber
                  _PrefLastLine(I)  = .PreferredLastLineNumber
                  _RangeHigh(I)     = .PrimaryRange.High
                  _RangeLow(I)      = .PrimaryRange.Low
                  _RangeOddEven(I)  = .PrimaryRange.OddEven
                  _StreetName(I)    = .Street.Name
                  _StreetSuffix(I)  = .Street.Suffix
                  _SuiteHigh(I)     = .Street.Suite.Range.High
                  _SuiteLow(I)      = .Street.Suite.Range.Low
                  _SuiteOddEven(I)  = .Street.Suite.Range.OddEven
                  _SuiteName(I)     = .Street.Suite.SuiteName
                  _AddressType(I)   = .Type.Code
                  _Urb(I)           = .Urbanization.Code
                  _Zip5(I)          = .Zip.Zip5
                  _Plus4High(I)     = .Zip.Plus4.High
                  _Plus4Low(I)      = .Zip.Plus4.Low
                  _ZipType(I)       = .Zip.Type
            End With
      Next
End With
```

Chapter 14

Telephone Number

The Telephone Number package returns geographic data associated with a submitted phone number. All geographic information returned is derived from the location of the wire center for the Area Code/Prefix and not the exact physical location of the phone number.

This package also returns the submitted phone number parsed into its component parts (area code, prefix, suffix and extension).

If the associated ZIP Code is included with the request and the phone number has been subject to a recent area code split, the package will also return both the new and old area codes, enabling you to keep your telephone database current in the face of frequent area code splits. If a new area code is found, the geographic information is based on the center of the new area code.

If no ZIP Code is submitted or no split is detected, this package will return the current area code and the geographic information associated with that.

The information returned is useful in determining the appropriate time to conduct telephone correspondence as well as determining the geographic area of your inbound calls which is very useful in determining future marketing efforts.

This package is also useful for easily sorting your database by area code and prefix as well as catching duplicate records that differ only by extension or the format used to enter the phone number.

Example 1

Input: If you submit the following phone number and zip code. No area code split is detected:

Field	Value
Phone Number	(949) 589-5200 x123
Zip Code	92688

Output: The web service returns the following data:

Field	Value
Latitude	33.636000
Longitude	-117.600000
TimeZone Name	Pacific Time
TimeZone Code	08
City Name	Trabuco Canyon
State Abbreviation	CA
County Name	Orange
County FIPS	06059
Country Abbreviation	US
Country Name	United States of America
MSA Code	4472
PMSA Code	5945
Area Code	949
New Area Code	949
Prefix	589
Suffix	5200
Extension	123

Example 2

Input: If you submit the following phone number and zip code. An area code split is detected:

Field	Value
Phone Number	(714) 589-5200 x123
Zip Code	92688

Output: The web service returns the following data. Unaffected fields omitted for brevity.

Field	Value
Area Code	714
New Area Code	949

Example 3

Input: If you submit the following phone number but no zip code. The service does not attempt to detect an area code:

Field	Value
Phone Number	(949) 589-5200 x123

Output: The web service returns the following data. Unaffected fields omitted for brevity.

Field	Value
Area Code	949
New Area Code	949

The Telephone package applies to U.S. phone numbers only and may be used alone or in conjunction with any other package offered by the DQWS.

Required Request Properties

```
Request.CustomerID = String
Request.Telephone = String
Request.Zip5 = String
```

Response Properties Returned

```
String = Response.Telephone.City.Name
String = Response.Telephone.Country.Abbreviation
String = Response.Telephone.Country.Name
String = Response.Telephone.County.Fips
String = Response.Telephone.County.Name
String = Response.Telephone.Extension
String = Response.Telephone.GeoCode.Latitude
String = Response.Telephone.GeoCode.Longitude
String = Response.Telephone.Msa.Code
String = Response.Telephone.NewAreaCode
String = Response.Telephone.PhoneAreaCode
String = Response.Telephone.Pmsa.Code
String = Response.Telephone.State.Abbreviation
String = Response.Telephone.State.Name
String = Response.Telephone.TimeZone.Code
String = Response.Telephone.TimeZone.Name
```

Request Object Syntax

The following code example shows a request for the Telephone package with the required properties populated with valid data.

```
Dim Action As New mdWebService.mdWebServices
'request
Dim Request As New mdWebService.RequestRecord
'response
Dim Response As New mdWebService.ResponseRecord

'select package by setting boolean property to true
Request.pkgTelephoneNumber = True
Request.action = "Request"
Request.version = "1.0"
Request.timeout = 10

'customerID required for every request
'supplied by Melissa Data Customer Service
Request.CustomerID = "1234567890"

'use TransmissionReference to assign your own identifier to a
'request
Request.transmissionReference = "ABCD1234"

'populate the input properties with the data to be processed
Request.Telephone = _Telephone

'submit the request to the web service
Response = Action.doSingleRecord(Request)
```

Processing the Response Object

The Response object will contain the data returned by this package.

Telephone returns the following properties listed below. A full description of each property can be found in the **Output Properties** chapter starting on page 85.

Sample Code

The following structure illustrates one way to extract the data returned:

```
With Response.Telephone
      _City = .City.Name
      _CityAbb = .City.Abbreviation
      _Country = .Country.Abbreviation
      _CountryName = .Country.Name
      _County = .County.Name
      _CountyFips = .County.Fips
      _Ext = .Extension
      _Latitude = .Geocode.Latitude
      _Longitude = .Geocode.Longitude
      _Msa = .Msa.Code
      _Pmsa = .Pmsa.Code
      _Prefix = .Prefix
      _State = .State.Abbreviation
      _StateName = .State.Name
      _TimeZone = .TimeZone.Name
      _TimeZoneCode = .TimeZone.Code
      _State = .State.Abbreviation
End With
```

Chapter 15

ZIP Code Information

The ZIP Code Information package returns an array of all records pertaining to the input ZIP Code. Since a ZIP Code can have more then one city the possibility exists that more then one record will be returned. For this reason an array is returned.

Note: Only one record is shown for example purposes.

Example

Input: If the following data is submitted:

Field	Value
ZIP Code	92688

Output: The web service returns the following information:

Field	Value
ZipCodeData.count:	2
Record #: 1	
ZIP5	92688
Automation	D
City Name	RANCHO SANTA MARGARITA
City Abbreviation	Rcho Sta Marg
County Name	Orange
County FIPS	06059
Facility Code	B
GeoCode Latitude	33.6390
GeoCode Longitude	-117.6030

Field	Value
LastLineIndicator	L
LastLineNumber	Z22621
PreferredLastLineNumber	Z22621
State Abbreviation	CA
TimeZone Name	Pacific Time
TimeZone Code	08
AreaCode	949
ZipType	Standard
MSA Code	4472
PMSA Code	5945

The ZIP Code Information package is for use with U.S. addresses only and can be used alone or in conjunction with other packages.

Required Request Properties

```
Request.CustomerID = String
Request.Zip5 = String
```

Response Properties Returned

```
String = Response.ZipCodeData.count
String = Response.ZipCodeData.Record(idx).AreaCode
String = Response.ZipCodeData.Record(idx).Automation
String = Response.ZipCodeData.Record(idx).City.Name
String = Response.ZipCodeData.Record(idx).City.Abbreviation
String = Response.ZipCodeData.Record(idx).County.Fips
String = Response.ZipCodeData.Record(idx).FacilityCode
String = Response.ZipCodeData.Record(idx).GeoCode.Latitude
String = Response.ZipCodeData.Record(idx).GeoCode.Longitude
String = Response.ZipCodeData.Record(idx).LastLineIndicator
String = Response.ZipCodeData.Record(idx).LastLineNumber
String = Response.ZipCodeData.Record(idx).Msa.Code
String = Response.ZipCodeData.Record(idx).Pmsa.Code
String = Response.ZipCodeData.Record(idx).PreferredLastLineNumber
String = Response.ZipCodeData.Record(idx).State.Abbreviation
String = Response.ZipCodeData.Record(idx).TimeZone.Name
String = Response.ZipCodeData.Record(idx).TimeZone.Code
String = Response.ZipCodeData.Record(idx).Zip.Zip5
String = Response.ZipCodeData.Record(idx).Zip.Type
```

Request Object Syntax

The following code example shows a request for the ZIP Code Information package with the required properties populated with valid data.

```
Dim tStr1 As String
Dim Action As New mdWebService.mdWebServices
'request
Dim Request As New mdWebService.RequestRecord
'response
Dim Response As New mdWebService.ResponseRecord

'select package by setting boolean property to true
Request.pkgZipCodeInformation = True
Request.action = "Request"
Request.version = "1.0"
Request.timeout = 10

'customerID required for every request
'supplied by Melissa Data Customer Service
Request.CustomerID = "1234567890"

'use TransmissionReference to assign your own identifier to a
'request
Request.transmissionReference = "ABCD1234"

'populate the input properties with the data to be processed
Request.Zip5 = _Zip5

'submit the request to the web service
Response = Action.doSingleRecord(Request)
```

Processing the Response Object

The Response object will contain the data returned by this package.

ZIP Code Information returns the following properties listed below. A full description of each property can be found in the **Output Properties** chapter starting on page 85.

Sample Code

The following structure illustrates one way to extract the data returned:

```
With Response.ZipCodeData
      'cycle through the records returned by the response
      For I = 0 to .count - 1
            With .Record(I)
                  _AreaCode(I) = .AreaCode
                  _Auto(I) = .Automation
                  _City(I) = .City.Name
                  _CityAbb(I) = .City.Abbreviation
                  _County(I) = .County.Name
                  _CountyFips(I) = .County.Fips
                  _Facility(I) = .FacilityCode
                  _LastLineInd(I) = .LastLineIndicator
                  _LastLine(I) = .LastLineNumber
                  _Latitude(I) = .Geocode.Latitude
                  _Longitude(I) = .Geocode.Longitude
                  _Msa(I) = .Msa.Code
                  _Pmsa(I) = .Pmsa.Code
                  _State(I) = .State.Abbreviation
                  _TimeZone(I) = .TimeZone.Name
                  _TimeZoneCode(I) = .TimeZone.Code
                  _Zip5(I) = .Zip5
            End With
      Next
End With
```

Chapter 16

ZIPs In City

The ZIPs In City package returns an array of all ZIP Codes within the input city/state combination. Since a City can have more then one ZIP Code the possibility exists that more then one record will be returned. For this reason an array is returned.

Note: Only one record is shown for example purposes.

Example

Input: The following data is submitted to the web service:

Field	Value
City:	Rancho Santa Margarita
State:	CA

Output: The following record is returned:

Field	Value
ZipCodeData.count:	1
Record #: 1	
Zip5	92688
City Name	RANCHO SANTA MARGARITA
City Abbreviation	Rcho Sta Marg
ZipType	

The ZIPs In City package is for use with U.S. addresses only and can be used alone or in conjunction with other packages.

Required Request Properties

```
Request.CustomerID = String
Request.City = String
Request State = String
Request.Zip5 = String
```

Response Properties Returned

```
String = Response.ZipCodeData.count
String = Response.ZipCodeData.Record(idx).City.Name
String = Response.ZipCodeData.Record(idx).City.Abbreviation
String = Response.ZipCodeData.Record(idx).Zip.Zip5
```

Request Object Syntax

The following code example shows a request for the ZIPs In City package with the required properties populated with valid data.

```
Dim Action As New mdWebService.mdWebServices
'request
Dim Request As New mdWebService.RequestRecord
'response
Dim Response As New mdWebService.ResponseRecord

'select package by setting boolean property to true
Request.ZipCodeZipInCity = True
Request.action = "Request"
Request.version = "1.0"
Request.timeout = 10

'customerID required for every request
'supplied by Melissa Data Customer Service
Request.CustomerID = "1234567890"

'use TransmissionReference to assign your own identifier to a
'request
Request.transmissionReference = "ABCD1234"

'populate the input properties with the data to be processed
Request.City = _CityName
Request.State = _State

'submit the request to the web service
Response = Action.doSingleRecord(Request)
```

Processing the Response Object

The Response object will contain the data returned by this package.

ZIPs In City returns the following properties listed below. A full description of each property can be found in the **Output Properties** chapter starting on page 85.

Sample Code

The following structure illustrates one way to extract the data returned:

```
With Response.ZipCodeData
      'cycle through the records returned by the response
      For I = 0 to .count - 1
            With .Record(I)
                  _City(I) = .City.Name
                  _CityAbb(I) = .City.Abbreviation
                  _Zip5(I) = .Zip.Zip5
            End With
      Next
End With
```

Chapter 17

DQWS Messages

In addition to the data that it returns, the Data Quality Web Service will also return several properties to the Response Object that indicate how successful a call to the service was. If a call to the web service was not successful, then these properties will indicate why this condition exists.

These messages fall into three categories.

Service Faults

Service Faults generally occur when the web service detects that a required piece of input data, such as a ZIP Code, is missing from a request, and the service cannot proceed with processing. Faults do not indicate if the input was invalid, merely that it was not present.

Status Codes

Status Codes indicate how to what degree the web service was able to process your request. If the Status Code indicates that the service was unable to fully process the request, you would then check the Error Code to determine why this condition occured.

Error Codes

Error Codes occur when the service was unable to fully process the data submitted. This is different from a Fault in that Faults occur when the input data is incomplete and Error Codes are returned when the data submitted to the service was complete but incorrect or unverifiable.

Service Faults

Service Faults are returned when DQWS was unable to proceed with processing the input data because a required element was missing from the Request object.

Processing the Fault Messages

Fault Messages are returned via whichever interface of the Response Object is returned by the requested package. For example, faults caused by the Address Mailing package will be returned via the Address interface. See “Handling The Response Object” on page 6.

Each interface of the Reponse object has a “Faults” collection. The count property will indicated the number of faults. The Faults collection is zero-based, therefore the last fault will have an index of count minus one. In other words, if the package returned three fault conditions, the first fault will be “0” and the third would be “2”.

Each fault has the four properties listed below:

Property	Description.
Code	Returns the name of the fault.
Description	Returns a more detailed description of the fault.
Source	Returns the name of the object or property that caused the fault.
Detail	Contains the name of the HTML file at Melissa Data’s support site which contains help for correcting the fault.

Examples

The following code sample shows how you would read the fault information from a package using the Address interface.

```
'Test if the Faults collection has any objects
If Response.Address.Faults.Count > 0 Then
      'Cycle through the Faults collection
      For I = 0 to Response.Address.Faults.Count - 1
            'Extract the properties of each Fault item
            With Resonse.Address.Faults.Fault(0)
                  strFaultCode(I) = .Code
                  strFaultDesc(I) = .Desc
                  strFaultSource(I) = .Source
                  strFaultDetail(I) = .Detail
            End With
      Next
End If
```

If the package requested used the StreetData interface instead, the code would begin like this:

```
If Response.StreetData.Faults.Count > 0 Then
```

For other packages, simply substitute the name of the interface for Address and StreetData above

The following is a list of possible Faults. Not all faults are returned by all packages.

Fault Code	Explanation
EmptyStreetAddress	The Address property of the Request object was not populated
EmptyCity	The City property of the Request object was not populated.
EmptyFullName	The FullName property of the Request object was not populated.
EmptyState	The State property of the Request object was not populated.
EmptyTelephoneNumber	The Telephone property of the Request object was not populated
EmptyZip	The Zip5 property of the Request object was not populated.
MethodContradiction	Two packages that cannot be run simultaneously were included in the request. The combinations that will cause this fault are: • Street Data All Records and Street Data In Ranges Records Only • Cities In State and ZIPs in City
TotalRecords > 100	A batch or multiple-record request contained more than 100 records. Processing was not carried out.

See each chapter for specific Request package information regarding which properties are required by that package. If a property is not required, the associated fault will not be returned.

Status Codes

A Request package will always return a Status code unless that package has no Status codes associated with it. The Status Code indicates how successful the web service was at processing the submitted data, as well as other information.

If the Status code indicates that processing was incomplete or unsuccessful. The Error code (see below) can be examined to determine why this condition occured.

Processing Status Codes

Each interface of the Response object has its own set of Status codes. Not all Status codes pertain to every package that uses that interface.

Address

The following code sample shows one method of reading a Status code returned via the Address interface.

```
Dim strStatusCode as String, strStatusDesc as String

strStatusCode = Response.Address.Result.StatusCode
strStatusDesc = Response.Address.Result.StatusDesc
```

The following shows the Status codes that are possible when using Address Delivery Point Validation, Address Geographic Area, Address Mailing, Address Parse and Street Data On Address Errors Only (when the submitted address is valid).

Code	Level
S (U.S.)	The address was standardized but not coded. Standardization means that some conversion was done on the address (for example, changing Post Office Box to PO Box or abbreviating street suffixes).
V	Street number validated to DPV level.
X	Address was not coded.
6	A Canadian address was fully coded.
7 (U.S.)	There were multiple matches for the address but they were all in the same ZIP Code and carrier route. The returned ZIP Code and carrier route will be correct but you will not get any +4 information.
9 (U.S.)	The address was fully coded.

The following shows the Status codes that are possible when using the Address Geocode package:

Code	Level
9	Information was coded to the ZIP + 4 centroid.
7	Information was coded to the ZIP+2 centroid.
5	Information was coded to the 5-digit ZIP Code centroid.
X	Information was not coded.

Name

The following code sample shows one method of reading a Status code returned via the Name interface.

```
Dim strStatusCode as String, strStatusDesc as String

strStatusCode = Response.Name.Result.StatusCode
strStatusDesc = Response.Name.Result.StatusDesc
```

The following shows the Status codes that are possible when using Name Parse:

Code	Level
Empty	The name was successfully parsed.
V	The name contained a vulgar word.
X	Unable to parse name.

StreetData

Packages that use the the StreetData interface does not return Status codes. If the Request was successfully processed, the count property of the StreetData interface will be greater than zero.

Telephone

The following code sample shows one method of reading a Status code returned via the Telephone interface.

```
Dim strStatusCode as String, strStatusDesc as String

strStatusCode = Response.Telephone.Result.StatusCode
strStatusDesc = Response.Telephone.Result.StatusDesc
```

The following shows the Status codes that are possible when using the Telephone package:

Code	Description
C	Corrected area code that was changed according to the ZIP Code it falls into.
U	Updated area code that had split. The new area code is in the Response.Telephone.NewAreaCode property.
X	Bad telephone number.

ZipCodeData

Packages that use the ZipCodeData Interface do not return Status codes. If the Request was successfully processed, the count property of the ZipCodeData interface will be greater than zero.

Error Codes

A Request package will return a Error code if the web service was unable to fully process the submitted data, unless that package has no Error codes associated with it. The Error Code indicates the reason why the web service could not process the request

Processing Error Codes

Each interface of the Response object has its own set of Error codes. Not all Error codes pertain to every package that uses that interface.

Address

The following code sample shows one method of reading an Error code returned via the Address interface.

```
Dim strErrorCode as String, strErrorDesc as String

strErrorCode = Response.Address.Result.ErrorCode
strErrorDesc = Response.Address.Result.ErrorDesc
```

The following shows the Error codes that are possible when using Address Delivery Point Validation, Address Geographic Area, Address Mailing, Address Parse and Street Data On Address Errors Only (when the submitted address is valid):

Error Code	Error Desc	Explanation
Space (“ “)	OK	Address is correct
F	DPV Offline	DPV processing was terminated due to the detection of what is determined to be an artificially created address. No address beyond this point has been DPV validated. In accordance with the License Agreement between USPS and Melissa Data, DPV shall be used to validate legitimately obtained addresses only, and shall not be used for the purpose of artificially creating address lists. The written Agreement between Melissa Data and you, its customer shall also include this same restriction against using DPV to artificially create address lists. Continuing use of DPV requires compliance with all terms of the License Agreement. If you believe this address was identified in error, please contact Melissa Data.
M	Multiple Matches	More than one record matches the address and there is not enough information available in the submitted address to break the tie between multiple records. Passing information, such as city/municipality names or urbanization names, can help reduce the number of multiple match errors.
N	No Street Data for ZIP/ Postal Code	The ZIP/Postal Code exists but no streets begin with the same letter in that ZIP/Postal Code.
R	Address out of Range	The street was found but the street number in the submitted address was not between the low and high range of the post office database.
T	Component Mismatch	Either the directionals or the suffix field did not match the post office database, and there was more than one choice for correcting the address. For example, if the given address was "100 Main St" and the only addresses found were "100 E Main St" and "100 Main Ave", the error code "T" would be returned because we do not know whether to add the directional "E" or to change the suffix to "Ave."
U	Unknown Street	An exact street name match could not be found and phonetically matching the street name resulted in either no matches or matches to more than one street name.
W	Early Warning System	This address has been identified in the Early Warning System (EWS) data file, and should be included in the next national database update.
X	Non-Deliverable Address	The physical location exists but there are no homes on this street. One reason might be railroad tracks or rivers running alongside this street, as they would prevent construction of homes in this location.
Z	ZIP/Postal Code Error	The ZIP/Postal Code does not exist and could not be determined by the city/municipality and state/province.

The following shows the Error codes that are possible when using the Address Geocode package:

Error Code	Error Desc	Explanation
Empty	OK	A valid ZIP + 4 Code was entered and located.
N	Record Not Found	Unable to locate the +4 without the ZIP Code.
Z	Bad ZIP Code	An invalid ZIP Code was entered.

DQWS returns a blank space as the error code when the submitted data was successfully coded and processed. This is to maintain full compatibility with the error codes returned by Melissa Data's other Data Quality tools.

Name

Packages that use the Name Interface do not return a ErrorCode.

StreetData

Packages that use the the StreetData interface does not return error codes. If the Request was successfully processed, the count property of the StreetData interface will be greater than zero.

Telephone

The following code sample shows one method of reading a Error code returned via the Telephone interface.

```
Dim strErrorCode as String, strErrorDesc as String

strErrorCode = Response.Telephone.Result.ErrorCode
strErrorDesc = Response.Telephone.Result.ErrorDesc
```

The following shows the Error codes that are possible when using the Telephone package:

Code	Description
A	Bad Area Code (area code does not exist in the database or the area code contains characters)
B	Blank (phone number is blank)
E	Bad Phone Number (too many or too few digits)
M	Multiple Match (could not choose between 2 or more area codes as a bad or missing area code was encountered and the distance between the area codes was too close to choose one over the other)
P	Bad Prefix (this prefix does not exist in the database)
Z	Bad ZIP Code (an invalid ZIP Code was entered, for example, 00001)

ZipCodeData

Packages that use the ZipCodeData Interface do not return Error codes. If the Request was successfully processed, the count property of the ZipCodeData interface will be greater than zero.

Chapter 18

Input Properties

The properties of the Request object described in this chapter are used to send data to the web service to be processed. Not all properties are used by each Request package. Some properties are optional only if others are used.

Address

This is the street address portion of the information to be processed.

Data Type String value

Max Length 32 characters

Syntax

```
Request.Address = "1234 Main Street"
```

Required For Address Delivery Point Validation, Address Geocode, Address Mailing, Address Parse, Street Data All Records, Street Data In Range Records Only, Street Data On Address Errors Only

Remarks This can contain suite information or that information can be entered as the Suite property or as the Address2 property.

Address Mailing package: If the contents of the Address property are not a valid address, the web service will attempt use the contents of Address2 instead.

Street Data All Records package: This property contains the full or partial street name to search for in the submitted five digit ZIP Code.

Address2

This property contains is the second part of the street address portion of the information to be processed.

Data Type String value

Max Length 32 characters

Syntax

```
Request.Address2 = "PO Box 1234"
```

Required For None

Optional For Address Delivery Point Validation, Address Geocode, Address Mailing, Address Parse, Street Data All Records, Street Data In Range Records Only, Street Data On Address Errors Only

Remarks This property can contain the suite information for the submitted data or an alternate street address.

Address Mailing package: If the contents of the Address property are not a valid address, the web service will attempt use the contents of Address2 instead.

City

This property identifies the city or municipality associated with the submitted address data.

Data Type String value

Max Length 28 characters

Syntax `Request.City = "Rancho Santa Margarita"`

Required For Cities In State, Zips In City

Optional For Address Delivery Point Validation, Address Geocode, Address Mailing, Address Parse, Residential and Business Delivery Indicator, Street Data All Records, Street Data In Range Records Only, Street Data On Address Errors Only

Remarks When this property is used, the city or municipality is usually only optional if you provide the correct ZIP or Postal Code. If you do not provide a ZIP or Postal Code, the City and State properties are required to successfully check an address.

If the City Property is not supplied, the address check logic will use the official city or municipality name of the ZIP or Postal Code instead.

If the City Property is supplied, the address check logic will only change the city or municipality name if it is an incorrect or unapproved mailing name. In these cases, the official city or municipality name for the ZIP or Postal Code will be returned. However, if the official city or municipality name or an approved vanity name for the ZIP or Postal Code is entered, the address check logic will return that city as entered.

If the supplied city and state do not match the ZIP or Postal Code, the address checking logic will give preference to the city name. Matches will be attempted within the supplied city instead of the ZIP or Postal Code. This logic is based on the assumption that a ZIP or Postal Code with one typo will result in more incorrect address matches than a city name with a few typos.

Cities In State Package: This property is required and can contain a partial city name and a wildcard character (“*”) to match city names in the requested state.

Company

The company identifies the business name associated with the address data being submitted.

Data Type String value

Max Length 40 characters

Syntax

```
Request.Company = "Melissa Data Corp."
```

Required For None

Optional For Address Delivery Point Validation, Address Geocode, Address Mailing, Residential and Business Delivery Indicator

Remarks Because some companies can have unique ZIP + 4 codes assigned to them, this information can help pinpoint ZIP + 4 information more precisely with some packages.

Country

This property contains the name or abbreviation for the country associated with the submitted address data.

Data Type String value

Max Length 50 characters

Syntax

```
Request.Country = "US"
```

Remarks The web service can process address data for the United States or Canada.

Debug

This property enables debug mode.

Data Type Boolean value

Syntax

```
Request.Debug = True
```

```
RequestArray.Debug = True
```

Remarks Use this property to enable debug mode in either single record or batch processing. Debug mode causes DQWS to return all fields to the response object. Only those fields returned by the requested packages will be populated, however.

Full Name

This property includes a person's name to be parsed.

Data Type String value

Max Length 100 characters

Syntax

```
Request.FullName = "John Q. Smith"
```

Required For Name Parse

Plus4

The four digit add-on portion of the full ZIP + 4 associated with the submitted data.

Data Type String value

Max Length 4 characters

Syntax

```
Request.Plus4 = "2112"
```

Optional For Address Geocode, Street Data All Records, Street Data In Range Records Only, Street Data On Address Errors Only, ZIP Code Information, Address Delivery Point Validation, Address Geographic Area, Address Mailing, Address Parse

Remarks If you are including the full nine-digit ZIP + 4 Code in your request, place the second part here or include it in the "Zip5" property.

State

This identifies the state or province associated with the submitted address data.

Data Type String value

Max Length 2 characters

Syntax

```
Request.State = "CA"
```

Required For Cities In State, Zips In City

Optional For Address Delivery Point Validation, Address Geocode, Address Mailing, Address Parse, Residential and Business Delivery Indicator, Street Data All Records, Street Data In Range Records Only, Street Data On Address Errors Only

Remarks When this property is used, the city or municipality is usually only optional if you provide the correct ZIP or Postal Code. If you do not provide a ZIP or Postal Code, the City and State properties are required to successfully check an address.

Suite

This property contains the suite information assocated the submitted address.

Data Type String value

Max Length 16 characters

Syntax

```
Request.Suite = "Apt. 101"
```

Remarks You can include the suite information here, as the Address2 property as part of the Address property.

Telephone

This property contains a telephone number to be processed.

Data Type String value

Max Length 24 characters

Syntax

```
Request.Telephone = "949-589-5200"
```

Required For Telephone Number

Remarks The area code, prefix and suffix of the phone number do not need to be separated by dashes or parentheses.

Urbanization

This property would contains the Urbanization code for a Puerto Rican Address

Data Type String value

Max Length 28 characters

Syntax

```
Request.Urbanization = "URB Camino Reposeo"
```

Optional For Address Delivery Point Validation, Address Geocode, Address Mailing, Address Parse, Street Data All Records, Street Data In Range Records Only, Street Data On Address Errors Only

Remarks This property applies to Puerto Rican addresses only It can break ties when the same street name occur more than once in the same ZIP Code.

Zip5

This property contains the ZIP Code to be processed, either the five-digit ZIP Code or the full nine-digit ZIP + 4 for US addresses and the six-character Postal Code for Canadian addresses.

Data Type String value

Max Length 10 characters

Syntax

```
Request.Zip5 = "92688"
```

or

```
Request.Zip5 = "92688-2112"
```

Required For Address Geocode, Street Data All Records, Street Data In Range Records Only, Street Data On Address Errors Only, ZIP Code Information

Optional For Address Delivery Point Validation, Address Geographic Area, Address Mailing, Address Parse, Residential and Business Delivery Indicator, Telephone Number

Remarks If including the full nine-digit ZIP + 4, the dash is optional.

Address Delivery Point Validation, Address Geographic Area, Address Mailing and Address Parse packages: The ZIP or Postal Code is only optional when you provide the correct city/municipality and state/province. If you do not provide a city/municipality and state/province, the ZIP Code will be required to successfully check an address.

Chapter 19

Output Properties

The properties described in this chapter are returned by the Response Object. Not all properties are returned by every package. See the page for each property for the packages returning that property.

Also, the stucture for some properties will vary based on the package requested. See the chapter for each package to learn how the each property is returned by that package.

Address Range

This property returns a string value with the delivery number of the submitted address.

Returned By Address Parse

Remarks “1234 N. Main Street” would return an Address Range of “1234”.

Address Status

This property returns a string value based on whether or not the submitted address could be fully coded and validated.

Returned By Address Delivery Point Validation

Remarks These are the possible values for Address Status:

Status	Explanation
AddressVerified	The address is a valid deliverable address.
SuiteOutOfRange	The suite number is outside the deliverable range for the submitted address.
SuiteMissing	The submitted requires a suite number to be a deliverable address and none was submitted with the Request.
AddressOutOfRange	The submitted street address is outside the deliverable range for the street within that ZIP Code.

Address Suffix

This property returns a string value with the suffix portion of the submitted address.

Returned By Address Parse

Remarks This property contains the official abbreviation for the street suffix, such as “St”, “Rd” or “Ave.”

“1234 N. Main Street” would return an Address Suffix of “St”.

Address Type Code (U.S. Only)

Returns a string value containing a one-letter code for the type of address that was coded (PO Box, Rural Route, and so on), using the submitted address.

Returned By

Address Mailing, Street Data All Records, Street Data In Range Records Only, Street Data On Address Errors Only

Remarks

This 1-character code indicates the type of address that was returned:

Code	Type
F	Firm or company address
G	General Delivery address
H	High-rise or business complex
P	PO Box address
R	Rural route address
S	Street or residential address

Address Type String (U.S. Only)

Returns a string value describing the type of address coded.

Returned By

Address Mailing

Remarks

This string value indicates the type of address that was returned:

Code	Type
F	Firm or company address
G	General Delivery address
H	High-rise or business complex
P	PO Box address
R	Rural route address
S	Street or residential address

Area Code (ZIP Code)

Returns a string value containing the telephone area code associated with the submitted ZIP Code.

Returned By Zip Code Information

Remarks If a ZIP Code has more than one area code assigned to it, the dominant area code will be displayed.

Automation

Returns a string value containing the carrier route rate mail indicator associated with the submitted zip code.

Returned By Zip Code Information

Remarks Automation specifies the following conditions for letter-size carrier route sorted mail:

Code	Explantion
A	Carrier route sortation rates apply for this ZIP Code and merging is permitted.
B	Carrier route sortation rates apply for this ZIP Code and merging is not permitted.
C	Carrier route sortation rates do not apply for this ZIP Code and merging is permitted.
D	Carrier route sortation rates do not apply for this ZIP Code and merging is not permitted.

Base Alternate Indicator

This property returns a string value with the base alternate indicator for the address record returned.

Returned By Street Data All Records, Street Data In Range Record Only, Street Data On Address Errors Only.

Remarks This code specifies whether or not a record is a base (preferred) or alternate record. Base records (indicated by a "B") can represent a range of addresses or an individual address, such as a firm record, while alternate records (indicated by an "A") are individual delivery points. Base records are generally preferred over alternate records. The base record for an alternate record can be found by matching up the ZIP + 4 ranges.

Carrier Route (U.S. Only)

Returns a string value containing the carrier route associated with the submitted address.

Returned By Address Mailing, Street Data All Records

Remarks The CarrierRoute Property is a 4 character string. The first character of this property is always alphabetic, and the last three characters are numeric. The alphabetic letter indicates the type of delivery associated with this address.

Letter	Type of Delivery
B	PO Box
C	City Delivery
G	General Delivery
H	Highway Contract
R	Rural Route

Example "R001" or "C027" would be typical carrier routes.

CBSA Code

This property returns a five-digit Core Based Statistical Area (CBSA) number associated with the submitted address data.

Returned By Address Georgraphic Area

Remarks **From the U.S. Census Bureau:** 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.

CBSA Division Code

This property returns a string value containing the five-digit code for the division within a Core Base Statistical Area associated with the submitted address data. Will be blank if the CBSA is not broken into divisions.

Returned By Address Geographical Area

Remarks Some large Core Base Statistical Areas are broken into two more divisions which have their own code and title. If the CBSA does not have divisions, this property will be empty.

CBSA Division Level

This property returns a string value specifying whether a division with the Core Based Statistical Area associated with the submitted address data is a metropolitan or micropolitan area. Will be blank if the CBSA is not broken into divisions.

Returned By Address Geographical Area

Remarks Every Core Based Statistical Area is either a Metropolitan or Micropolitan area. This property has two possible values: “Metropolitan Statistical Area” or “Micropolitan Statistical Area.” Currently, all CBSAs with divisions are part Metropolitan Statistical Areas.

This property will be blank if there are no divisions in a CBSA.

CBSA Division Title

This property returns a string value containing the name of the Statistical Division associated with the submitted address data. Will be blank if the CBSA is not broken into divisions.

Returned By Address Geographic Area

Remarks This string contains the text that describes the cities or counties contained within a Statistical Division.

CBSA Level

This property returns a string value specifying whether a Core Based Statistical Area associated with submitted address data is a metropolitan or micropolitan area.

Returned By Address Geographic Area

Remarks Every Core Based Statistical Area is either a Metropolitan or Micropolitan area. This property has two possible values: “Metropolitan Statistical Area” or “Micropolitan Statistical Area.”

CBSA Title

This property returns a string value containing the name of the Core Based Statistical Area associated with the submitted address data.

Returned By Address Geographic Area

Remarks This string contains the text that describes the counties or cities contained within a Core Based Statistical Area (CBSA).

Census Block

This property returns a string value with the census block group number associated with the submitted address data.

Returned By Address Geocode

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 CensusBlock property is a 4-character string value set by a call to the GeoCode method. 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.

Census Tract

This property returns a string value with the census tract number associated with the submitted address data.

Returned By Address Geocode

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.

Example This property could contain "1210" or "121002"

City Abbreviation

Returns a string value containing the 13-letter abbreviation for the city or municipality associated with the submitted data.

Returned By Address Mailing, Cities In State, Zip Code Information, Zip In City

Remarks If the City name returned is longer than 13 letters, City Abbreviation will contain the official abbreviation the post office has associated with that city or municipality name. For example, "Fort Lauderdale," will return the abbreviation "Ft Lauderdale" for City Abbreviation.

If the City name is 13 letters or shorter, City Abbreviation will contain the city or municipality name.

City Name

Returns a string value containing the name of the city or municipality associated with the submitted data.

Returned By Address Mailing, Cities In State, Telephone Number, Zip Code Information, Zip In City

Remarks **Address Mailing:** If the City name is not supplied, the web service will use the official city or municipality name of the ZIP or Postal Code instead.

If the City and State is supplied, the address check logic will only change the city or municipality name if it is an incorrect or unapproved mailing name. In these cases, the official city or municipality name for the ZIP or Postal Code will be returned. However, if the official city or municipality name or an approved vanity name for the ZIP or Postal Code is entered, the web service will return that city as entered.

If the supplied city and state do not match the ZIP or Postal Code, the web service will give preference to the city name. Matches will be attempted within the supplied city instead of the ZIP or Postal Code. This logic is based on the assumption that a ZIP or Postal Code with one typo will result in more incorrect address matches than a city name with a few typos.

CMRA

This property returns a one-character string based on whether or not the submitted address is actually a private mailbox at a Commercial Mail Receiving Agency (CMRA).

Returned By Address Delivery Point Validation

Remarks There are two possible values for CMRA:

Code	Explanation
Y	The address belongs to a CMRA.
N	The address does not belong to a CMRA.

Congressional District (U.S. Only)

The congressional district number associated with the submitted address data.

Returned By Address Geographic Area, Street Data All Records

Remarks Congressional District is a 2-digit number of the congressional district for the address given and is accurate to the ZIP + 4 level. For states with only one congressional district, this value is always "01".

Company

Returns a string value containing the company name included with the submitted address.

Returned By Address Mailing

Remarks This property returns the Company property of the Request object.

Count

For packages that can return more than one records, this property contains the number of records returned.

Returned By Cities In State, Street Data All Records, Street Data In Range Records Only, Street Data On Address Errors Only, Zip Code Information, Zip In City

Remarks Collections returned by the web service are zero based, therefore the last item will have an index of Count minus one.

Country Abbreviation

Returns a string value containing a two-character abbreviation that indicates the country associated with the submitted address.

Returned By Address Mailing, Telephone Number

Remarks The Data Quality Web Service includes data for American and Canadian addresses. The 2-character abbreviation "US" is returned for addresses in the United States and the 2-character abbreviation "CA" is returned for Canadian addresses.

Country Name

Returns a string value containing the name the country associated with the submitted address.

Returned By Address Mailing, Telephone Number

Remarks The Data Quality Web Service includes data for American and Canadian addresses. "USA" is returned for addresses in the United States and "CANADA" is returned for Canadian addresses.

County FIPS

This property returns a string value with the FIPS Code of the county associated with the submitted ZIP Code.

Returned By Address Geocode, Address Mailing, Street Data All Records, Street Data In Range Records Only, Street Data On Address Errors Only, Telephone Number, Zip Code Information.

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.

Example 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.

County Name

This property returns a string value with the name of the county associated with the submitted address data.

Returned By Address Geocode, Address Mailing, Telephone Number, Zip Code Information

Remarks The County Name has a maximum length of 25 characters.

Example “Los Angeles” or “Manhattan”

Delivery Indicator (U.S. Only)

Return a string value containing a one-letter code which indicates whether the submitted address was a residence, a business or unknown.

Returned By Residential Delivery Indicator

Remarks This 1-character code indicates the type of address that was submitted:

Code	Type
R	Residential Address
B	Business Address
U	Unknown Delivery Type

Delivery Point Code (U.S. Only)

Returns a string value containing the 2-digit delivery point code associated with the submitted address.

Returned By Address Mailing

Remarks The DeliveryPointCode Property is a 2-digit character string that makes up the 10th and 11th positions of a 12-digit POSTNet barcode.

See **Delivery Point Check Digit** below for a complete guide to producing POSTNet barcodes.

Delivery Point Check Digit (U.S. Only)

Returns a string value containing the one-digit number representing the check digit associated with the submitted address.

Returned By Address Mailing

Remarks This 1-digit string makes up the 12th position of a 12-digit POSTNet barcode.

In 12-digit POSTNet barcodes, the ZIP Code is used for positions 1 to 5, the Plus4 code for positions 6 to 9, the delivery point code for positions 10 and 11, and this check digit for position 12.

Direction Post

This property returns a string value with any directional information that follows the street name in the submitted address.

Returned By Address Parse, Street Data All Records

Remarks Direction Post will contain the abbreviated post-direction of the full address string. Directions such as "North" will be changed to "N" before they are stored in this property.

“1234 Main Street Northwest” would return a Post Direction of “NW”.

Direction Pre

This property returns a string value with any directional information that precedes the street name in the submitted address.

Returned By Address Parse, Street Data All Records

Remarks Direction Pre will contain the abbreviated pre-direction of the full address string. Directions such as "North" will be changed to "N" before they are stored in this property.

“1234 South Main Street” would return a Post Direction of “S”.

DPV Footnotes

This property returns a string value containing the applicable USPS footnote codes for the submitted address.

Returned By Address DPV

Remarks DPV Footnotes returns a six-character string containing up to three of the following standard footnotes:

Code	Description
AA	Submitted Address Matched to the ZIP + 4 file.
A1	Submitted Address Not Matched to the ZIP + 4 file.
BB	Submitted Address Matched to DPV (all components)
CC	Submitted Address Primary Number Matched to DPV but Secondary Number not Matched (present but invalid).
N1	Submitted Address Primary Number Matched to DPV but Highrise Address Missing Secondary Number.
M1	Submitted Address Primary Number Missing.
M3	Submitted Address Primary Number Invalid.
P1	Submitted Address Missing PO, RR, or HC Box number.
P3	Input Address PO, RR or HC number invalid.
RR	Submitted Address Matched to CMRA and PMB designator present.
R1	Submitted Address Matched to CMRA but PMD designator not present.
F1	Address Was Coded to a Military Address
G1	Address Was Coded to a General Delivery Address
U1	Address Was Coded to a Unique ZIP Code.

Extension

Returns a string value containing the extension portion of the submitted telephone number.

Returned By Telephone Number

Remarks Returns digits following a “x” character or digits in excess of ten from the submitted phone number.

Facility Code (U.S. Only)

Returns a string value containing the code for the type of postal facility associated with the submitted ZIP Code.

Returned By ZIP Code Information

Remarks These are the possible values for Facility Code

Code	Type
A	Airport Mail Facility
B	Branch
C	Community Post Office
D	Area Distribution Center
E	Sectional Center Facility
F	Delivery Distribution Center
G	General Mail Facility
K	Bulk Mail Facility
M	Money Order Unit
N	Community or place name
P	Post Office™
S	Station
U	Urbanization (Used in Puerto Rico)
X	Vanity name (Should not be used)

First Name

This property returns a string value with the first name from the submitted name.

Returned By Name Parse

Remarks First Name is a 12-character (maximum) string value.

Example “Mr. John W. Smith, Jr.” would return “John”.

Garbage

This property returns a string value containing any garbage characters from the submitted address.

Returned By Address Parse

Remarks

Example

Gender

This property returns a one-letter code for the gender of the submitted name.

Returned By Name Parse

Remarks Possible return values for Gender:

Code	Desciption
F	Female
M	Male
N	Neutral (Male or Female)
U	Unknown

Example “Mr. John W. Smith, Jr.” would return “M”.

LACS Code (U.S. Only)

Returns a string value containing the indicator which notifies you whether or not the submitted address has undergone a LACS conversion.

Returned By

Address Mailing, Street Data All Records, Street Data In Range Records Only, Street Data On Address Errors Only

Remarks

Locatable Address Conversion Service (LACS) is a process where some rural route addresses are modified to city-style addresses to allow emergency services (for example, ambulance, police, fire, and so on) to find these addresses more efficiently.

LACS Code is a 1-character value with two possible values. An empty space indicates that the address has not undergone a LACS conversion. A value of "L" in the LACS field indicates that the address has undergone a conversion. After a conversion, the old address is retained in the ZIP + 4 file for a period of one year. After the one year period, the old addresses will be dropped from the ZIP + 4 file and the address checking logic will not assign a +4 for this address.

In order to actually update the flagged address, you must call enable the LACS^{Link} package offered by the Web Service. See page 31 for more information.

LACS Return Code (U.S. Only)

Returns a two-character number code indicating the degree to which the submitted address was matched to the LACS^{Link} data and if the address was updated.

Returned By

LACS^{Link}

Remarks

Some rural route addresses are modified to city-style addresses to allow emergency services (for example, ambulance, police, fire, and so on) to find these addresses more efficiently.

The LACS^{Link} service matches the old address with the updated address and corrects it as part of the Address Mailing package.

This property returns one of the following codes:

Code	Description
A	**LACS Record Match** - The input record matched to a record in the master file. A new address could be furnished.
00	**No Match** - The input record *could not be* matched to a record in the master file. A new address could not be furnished.
14	**Found LACS Record: New Address Would Not Convert at Run Time** - The input record matched to a record in the master file. The new address could not be converted to a deliverable address.
92	**LACS Record: Secondary Number Dropped from Input Address** - The input record matched to a master file record, but the input address had a secondary number and the master file record did not. The record is a ZIP + 4 street level or highrise match.

LACS Status Code (U.S. Only)

This property indicates whether or not the submitted address was corrected to a new address found in the LACS^{Link} data.

Returned By LACS^{Link}

Remarks LACS^{Link} is a process where some rural route addresses are modified to city-style addresses to allow emergency services (for example, ambulance, police, fire, and so on) to find these addresses more efficiently.

The LACS^{Link} service matches the old address with the updated address and corrects it as part of the Address Check process.

Code	Description
Y	The submitted address was found in the LACS^{Link} database and changed to the new address.
N	The submitted address was not found in the LACS^{Link} database.
S	Yhe submitted address was corrected to a new address found in the LACS^{Link} data but contained a suite that could not be matched.

Last Line Indicator

Returns a string value containing the indicator that the city name returned with this record is the official USPS city name for the ZIP Code.

Returned By Zip Code Information

Remarks An "L" in this field indicates that the city name is the official U.S. Postal Service® name for the ZIP Code (Only one record per ZIP Code is coded with an "L").

Last Line Number

Returns a string value containing the last line number for the address record returned.

Returned By Street Data All Records, Street Data In Range Records Only, Street Data On Address Errors Only, Zip Code Information

Remarks The last line number is a 6-character string used for advanced address matching. This number can be associated with last line numbers returned from the Zip Code Information package to break ties based off of city names.

Last Name

This property returns a string value with the last name from the submitted name.

Returned By Name Parse

Remarks Last Name is a 20-character (maximum) string value.

Example “Mr. John W. Smith, Jr.” would return “Smith”.

Latitude

This property returns a string value with the latitude associated with the submitted data.

Returned By Address Geocode, Telephone Number, Zip Code Information

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.

Since all U.S. ZIP Code latitude coordinates are north of the equator, this value will always be positive.

Example “85.123456”

Longitude

This property returns a string value with the longitude associated with the submitted data.

Returned By Address Geocode, Telephone Number, Zip Code Information

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.

Since all U.S. ZIP Code latitude coordinates are west of the Greenwich meridian, this value will always be negative.

Example “-100.123456”

Middle Name

This property returns a string value with the middle name or initial from the submitted name.

Returned By Name Parse

Remarks Last Name is a 12-character (maximum) string value.

Example “Mr. John W. Smith, Jr.” would return “W.”.

MSA Code (U.S. Only)

This property returns a four-digit string value with the Metropolitan Statistical Area (MSA) Code in which the submitted address is located.

Returned By Address Geographic Area, Telephone Number, Zip Code Information

Remarks The Office of Management and Budget defines the Metropolitan Statistical Area (MSA).

An MSA consists of one or more counties forming a large population with adjacent communities and having a high degree of social and economic integration.

MSA is based on 1990 census data and is not being updated. It is maintained for compatibility with legacy applications. For the most up-to-date data, use the CBSA properties instead.

Example “4472”, the MSA Code for Los Angeles, Orange and Riverside counties in California.

New Area Code

Returns a string value containing the new area code of the submitted phone number, if the old area code has recently split and a valid ZIP code was submitted to the sevice.

Returned By Telephone Number

Remarks If there has been no area code split or no ZIP Code was submitted, this property will contain the same information as Phone Area Code.

Phone Area Code (Telephone)

Returns a string value containing the area code portion of the submitted phone number.

Returned By Telephone Number

Remarks If there are not enough digits in the submitted phone number for the area code, this property will be empty. If the area code/prefix combination has been split, the new area code will be in New Area Code.

Place Code

Returns the census place code associated with the submitted ZIP + 4 code.

Returned By Address Geocode

Remarks This property returns a seven-digit string value containing the census place code for the submitted ZIP + 4 code.

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. The place code matches the ZIP + 4 code with the Census Bureau's official name for that physical location.

Place Name

Returns the census place code associated with the submitted ZIP + 4 code.

Returned By Address Geocode

Remarks The PlaceName property returns a 60-digit string value containing the census place name for the submitted ZIP + 4 code.

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. This property returns the Census Bureau's official name for the ZIP + 4 code.

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 Mailing package would return "Rancho Santa Margarita," but this property will return "Los Flores."

Plus 4 (U.S. Only)

Returns a string value containing the four-digit ZIP Code add-on associated with the submitted address.

Returned By Address Mailing

Remarks When correcting addresses, this property is usually ignored. However, if the submitted ZIP code is a "unique" ZIP code (a ZIP code that is assigned to one company), the submitted Plus4 code will be retained under certain circumstances.

If the user does not set the Plus4 Property, the address checking logic will still code the address, but it may use a "default" Plus4 code for these "unique" ZIP Codes.

Plus4 Range High (U.S. Only)

Returns a string value containing the last Plus4 Code associated with the returned address range.

Returned By Street Data All Records, Street Data In Range Records Only, Street Data On Address Errors Only

Remarks This four-character string gives the last Plus4 add-on for a ZIP + 4 range.

Example If the ZIP + 4 range is 1234-1334, this field will show the "1334" value.

Plus4 Range Low

Returns a string value containing the first Plus4 Code associated with the returned address range.

Returned By Street Data All Records, Street Data In Range Records Only, Street Data On Address Errors Only

Remarks This four-character string gives the first Plus4 add-on for a ZIP + 4 range.

Example If the ZIP + 4 range is 1234-1334, this field will show the "1234" value.

PMSA Code (U.S. Only)

This property returns a four-digit string value with the Primary Metropolitan Statistical Area (MSA) Code in which the submitted address is located.

Returned By Address Geographic Area, Telephone Number, Zip Code Information

Remarks The Office of Management and Budget defines the Primary Metropolitan Statistical Area (PMSA) for regions that contain a population of more than one million.

PMSA is based on 1990 census data and is not being updated. It is maintained for compatibility with legacy applications. For the most up-to-date data, use the CBSA properties instead.

Example “5495”, the PMSA for Orange County, CA.

Preferred Last Line Number

Returns a string value containing the preferred last line number for the submitted ZIP Code.

Returned By Zip Code Information

Remarks This is the equivalent of the “LastLineNumber" record mentioned earlier. If the preferred last line number is the same as the last line number, this city name is the one the post office will recognize as the official name for that ZIP code.

Primary Range High

Returns a string value contain the last number in the address returned.

Returned By Street Data All Records, Street Data In Range Records, Street Data On Address Errors

Remarks Primary Range High is a 10 character maximum string value which gives the last number in the address range. For example, if the address range is 100 to 200 Main Street, this field will show the "200" value.

All numeric data returned is padded with zeroes on the left.

A hyphen in front of the range field indicates a significant leading zero. That means that the leading zero is part of the range and is required. For example, -7 would indicate the range is 07, and cannot be just 7. No hyphen symbol (-) indicates that the leading zeros are not a part of this range.

Primary Range Low

Returns a string value containing the first number in the address returned.

Returned By Street Data All Records, Street Data In Range Records, Street Data On Address Errors

Remarks Primary Range High is a 10 character maximum string value which gives the first number in the address range. For example, if the address range is 100 to 200 Main Street, this field will show the "100" value.

All numeric data returned is padded with zeroes on the left.

A hyphen in front of the range field indicates a significant leading zero. That means that the leading zero is part of the range and is required. For example, -7 would indicate the range is 07, and cannot be just 7. No hyphen symbol (-) indicates that the leading zeros are not a part of this range.

Primary Range Odd/Even

Returns a string value containing the Odd/Even indicator for the address range returned.

Returned By Street Data All Records, Street Data In Range Records, Street Data On Address Errors

Remarks This one-character field shows either an "O" for Odd, an "E" for Even, or a "B" for Both. An "O" indicates that the address range contains only odd numbered addresses and an "E" indicates that only even numbered addresses are present in the address range. A "B" indicates that both odd and even address numbers are present in the address range. For example, an "O" means that in the 101 to 201 address range, only the numbers 101, 103, 105, 107, 109, and so on, are valid address numbers.

Private Mailbox (U.S. Only)

Returns a string value containing the private mail box number associated with a CMRA (Commercial Mail Receiving Agency).

Returned By Address Mailing

Remarks CMRAs are private businesses that provide a mailing address and "post office" box for their customers. Mail is delivered by the Postal Service™ to the CMRA, which then distributes the mail to the customer's private mail box.

Example "PMB 252"

Prefix (Name)

This property returns a string value with the prefix from the submitted name.

Returned By Name Parse

Remarks Prefix is a 10-character (maximum) string value.

Example “Mr. John W. Smith, Jr.” would return “Mr.”.

Prefix (Telephone)

Returns a string value containing the prefix portion of the submitted phone number (first three digits after the area code).

Returned By Telephone Number

Private Mail Box Name

This property returns a string value with the name portion of a private mail box number, if any.

Returned By Address Parse

Example If the address countain the private mail box number “PMB 101”, the web service would return “PMB” to this property.

Private Mail Box Number

This property returns a string value with the name portion of a private mail box number, if any.

Returned By Address Parse

Example If the address countain the private mail box number “PMB 101”, the web service would return “101” to this property.

State Abbreviation

Returns a string value containing the state or province associated with the submitted address.

Returned By Address Mailing, Cities In State, Telephone Number, Zip Code Information

Remarks This property returns the two-letter code for the state or province.

State Name

Returns a string value containing the state or province associated with the submitted address.

Returned By Address Mailing, Telephone Number

Remarks This property returns the full name for the state or province.

Street

Returns a string value containing the standardized and corrected form of the Address property of the Request object.

Returned By Address Mailing

Remarks The Street Property will return the contents of the Address property of the Request object along with any corrections or standardizations performed by the address checking logic of the web service.

Address corrections may include fixing misspelled street names or inserting missing suffixes and directionals.

Address standardization involves the conversion of suffixes and directionals to preferred postal abbreviations (for example,"Street" to "St").

Note: If a suite name is attached to the end of an address, it will be moved to the Suite Property.

Street Name

This property returns a string value with the street name portion of the returned address data.

Returned By

Address Parse, Street Data All Records, Street Data In Range Records Only, Street Data On Address Errors Only

Remarks

Street Name is a maximum 28-character string.

For street names that begin with certain spanish words, that first word will be moved to the suffix field and the next word in the street name will be returned as the street name. This is because spanish street names have the suffix at the front of the address. If this word was not moved to the suffix field, it would appear that every street in some Puerto Rican ZIP codes began with the same letter. The spanish words that get moved to the suffix field are: "Avenida", "Calle", "Camino", "Paseo" and "Via".

Example

“1234 South Main Street” would return “Main”.

Street Suffix

Returns a string value containing the suffix of the street for the address record returned.

Returned By

Street Data All Records, Street Data In Range Records Only, Street Data On Address Errors Only.

Remarks

Street Suffix is a four-character maximum string value.

When a street names are used more than once within a city or area, street suffixes are often used to distinguish them. For example, if there is a Main Street but also a Main Avenue, this field will hold either the "St" or "Ave" suffix, depending on which one is being referred to.

Street2

Returns a string value containing the standardized and corrected form of the Address2 property of the Request object, if supplied.

Returned By

Address Mailing

Remarks

This property returns the optional Address2 property of the Request object. It can contain either a suite or a completely different address.

If the web service detects a common suite name such as #, Apt, Ste, and so on, this will be moved to the Suite Property.

If the address entered in the Address Property was not verifiable and a separate and complete verified address was submitted in the Address2 property, the address entered in the Address2 property will be returned to the Street property and the address submitted in the Address property will be moved to the Street2 Property.

Examples

If the Address Property is not verifiable, the Address2 Property will be verified and the address properties will be swapped.

Input	Address = 123 Main St Apt 10 (could not be verified) Address2 = PO Box 223 (could be verified)
Output	Street = PO Box 223 Street2 = 123 Main St Apt 10

If the Address property can be verified, the Address2 property will not be considered.

Input	Address = PO Box 223 (could be verified) Address2 = 123 Main St Apt 10 (not looked at)
Output	Street = PO Box 223 Street2 = 123 Main St Apt 10

Suffix (Name)

This property returns a string value with the suffix from the submitted name.

Returned By

Name Parse

Remarks

Suffix is a 10-character (maximum) string value.

Example

“Mr. John W. Smith, Jr.” would return “Jr.”.

Suffix (Telephone)

Returns a string value containing the suffix portion (last four digits) of the submitted phone number.

Returned By Telephone Number

Suite

Returns a string value containing the suite name and number associated with the submitted address.

Returned By Address Mailing

Remarks If the suite is found at the end of the Address property of the Request object, it will be moved to the Suite property of the Response object. This move helps maintain a clean database and allows the user to limit secondary ranges to a separate field.

Suite Name

This property returns a string value with the name of any secondary unit in the submitted address.

Return Address Parse, Street Data All Records, Street Data In Range Records Only, Street Data On Address Errors Only

Remarks Suite Name is a 4-character maximum string value.

Possible return values are:

"#", "APT", "BLDG", "BOX", "BSMT", "DEPT", "FL", "FRNT", "HNGR", "LBBY", "LOT", "LOWR", "OFC", "PH" (Penthouse), "PIER", "REAR", "RM", "SIDE", "SLIP", "SPC", "STE", "STOP", "TRLR", "UNIT", "UPPR".

Example “1234 Main Street, #A101” could return “#” or “APT”.

Suite Number

This property returns a string value with the numeric portion of any secondary unit in the submitted address.

Returned By Address Parse

Remarks Suite Number is a 6-character maximum string value.

Example “1234 Main Street, #A101” would return “A101”

Suite Range High

Returns a string value containing the last number in the suite range for the address range returned.

Returned By Street Data All Records, Street Data In Range Records Only, Street Data On Address Errors Only

Remarks The SuiteRangeHigh Property is an eight-character maximum string value, which gives the last number in the suite range.

Example For the 1001 to 2001 suite range, this field will return the "2001" value.

Suite Range Low

Returns a string value containing the first number in the suite range for the address range returned.

Returned By Street Data All Records, Street Data In Range Records Only, Street Data On Address Errors Only

Remarks Suite Range Low is an eight-character maximum string value, which gives the first number in the suite range.

Example For the 1001 to 2001 suite range, this field will return the "1001" value.

Suite Range Odd/Even

Returns a string value containing the Odd/Even indicator of the suite range for the address range returned.

Returned By

Street Data All Records, Street Data In Range Records Only, Street Data On Address Errors Only

Remarks

This 1-character field has an "O" for Odd, an "E" for Even, or a "B" for Both. An "O" indicates that the suite range contains only odd numbers, an "E" indicates that only even numbers are present in the suite range, and a "B" indicates that both odd and even numbers are included in the suite range.

Example

"O" will indicate that, in the 1001 to 2001 suite range, only suite numbers 1001, 1003, 1005, 1007, and so on, are valid.

Time Zone Name

This property returns a string value with the name of the time zone associated with the submitted address data.

Returned By

Address Geographic Area, Telephone Number, Zip Code Information

Remarks

These are the values that can be returned for Time Zone Name:

"Military", "Atlantic Time", "Eastern Time", "Central Time", "Mountain Time", "Pacific Time", "Alaska Time", "Hawaii Time", "Samoa Time", "Marshall Island Time", "Guam Time" or "Palau Time"

The TimeZone Property does not account for daylight savings time regardless of whether an area is affected by it or not.

Time Zone Code

This property returns a string value with the numeric code for the time zone in which the submitted address is located.

Returned By Address Geographic Area, Telephone Number, Zip Code Information

Remarks These are the possible values for the TimeZone Property:

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

The TimeZoneCode Property does not account for daylight savings time regardless of whether an area is affected by it or not.

Urbanization Code (U.S. Only)

Returns a string value containing the urbanization name associated with the submitted address

Returned By Address Mailing, Street Data All Records, Street Data In Range Records Only, Street Data On Address Errors Only

Remarks Applies to Puerto Rican addresses only.

Example "URB Camino Reposeo"

ZIP Code

Returns a string value containing the five-digit ZIP Code or six-character Postal Code associated with the submitted address.

Returned By

Address Mailing, Street Data All Records, Street Data In Range Records Only, Street Data On Address Errors Only, Zip In City

Remarks

If a ZIP or Postal Code was submitted with the address, this property returns the five-digit ZIP Code or six-character Postal Code. If no ZIP or Postal Code was included, this will include the value that was matched to the submitted city and state.

Zip Type (U.S. Only)

Returns a string value containing the ZIP Code type associated with the submitted address.

Returned By

Address Mailing, Street Data All Records, Street Data In Range Records Only, Street Data On Address Errors Only, Zip Code Information

Remarks

The ZipType Property is a 1-character string value which defines the ZIP Code for delivery purposes.

Code	ZIP Type
P	A ZIP Code used only for PO Boxes.
U	Unique: A ZIP Code assigned to an organization or government institution such as the IRS.
M	Military: A ZIP Code assigned to an APO/FPO.
Blank	A standard ZIP Code.