Data Quality Web Service
Reference Guide

MELISSADATA™

Copyright

Trademarks

Document number: DQX RG 090423

Last Update: **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

Contents

Chapter 1: Introduction 1
- Overview of Data Quality Web Service 2
- How Do I Use Data Quality Web Service? 2
- Submitting XML Records to Data Quality Web Service 2
- Submitting Records for Processing 3

Chapter 2: Processes 4
- Processes 5
- Address Verify Process 5
- GeoCode Process 6
- CorrectAreaCode 7
- LookupAreaCode Process 8
- NameParse Process 9

Chapter 3: Input and Output Tags 10
- <Address> 11
- <Address2> 12
- <AddressDatabaseDate> 13
- <AddressErrorCode> 14
- <AddressErrorString> 15
- <AddressStatusCode> 16

<AddressTypeCode> 17
<AddressTypeString> 18
<AreaCodeOfZip> 19
<CarrierRoute> 20
<CBSACode> 21
<CBSADivisionCode> 22
<CBSADivisionLevel> 23
<CBSADivisionTitle> 24
<CBSALevel> 25
<CBSATitle> 26
<CensusBlock> 27
<CensusTract> 28
<City> 29
<CityAbbreviation> 30
<Company> 31
<CongressionalDistrict> 32
<CountyFips> 33
<CountyName> 34
<CustomerID> 35
<DeliveryPointCheckDigit> 36
<DeliveryPointCode> 37
<DisableAddressSwapping> 38
<ErrorString> 39
<FirstName> 40
<FullName> 41
<Gender> 42
<GeoCodeDatabaseDate> 43
<GeoCodeErrorCode> 44
<GeocodeStatusCode> 45
<Lacs> 46
<LastLine> 47
<LastName> 48
<Latitude> 49
<Longitude> 50
<MiddleName> 51
<Msa> 52
<MsaDesc> 53
<NameDatabaseDate> 54
<NamePrefix> 55
<NameStatusCode> 56
<NameSuffix> 57
<ParsedAddressRange> 58
<ParsedGarbage> 59
<ParsedPostDirection> 60
<ParsedPreDirection> 61
<ParsedStreetName> 62
<ParsedSuffix> 63
<ParsedSuiteName> 64
<ParsedSuiteRange> 65
<Phone> 66
<PhoneAreaCode> 67
<PhoneCity> 68

<PhoneCountry> 69
<PhoneCountyFips> 70
<PhoneCountyName> 71
<PhoneDatabaseDate> 72
<PhoneDistance> 73
<PhoneErrorCode> 74
<PhoneExtension> 75
<PhoneLatitude> 76
<PhoneLongitude> 77
<PhoneMsa> 78
<PhoneNewAreaCode> 79
<PhonePmsa> 80
<PhonePrefix> 81
<PhoneState> 82
<PhoneStatusCode> 83
<PhoneSuffix> 84
<PhoneTimeZone> 85
<PhoneTimeZoneCode> 86
<PlaceCode> 87
<PlaceName> 88
<Plus4> 89
<Pmsa> 90
<PmsaDesc> 91
<PrivateMailBox> 92
<RBDI> 93
<Record> 94
<RecordSet> 95
<State> 96
<Suite> 97
<TimeZone> 98
<TimeZoneCode> 99
<Urbanization> 100
<Zip> 101
Chapter 4: DPV™ – Verifying Address Accuracy 102
<CMRA> 103
<DPVFootnotes> 104
<DPVStatusCode> 105
<DPVSuiteStatusCode> 106
License Agreement 107

Chapter 1

Introduction

IN THIS CHAPTER...

"Overview of Data Quality Web Service" on page 2

"How Do I Use Data Quality Web Service?" on page 2

"Submitting XML Records to Data Quality Web Service" on page 2

"Submitting Records for Processing" on page 3

Overview of Data Quality Web Service

Data Quality Web Service uses the power of the internet to instantly verify, correct, update, and enhance data you submit online.

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 scripts 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 other database services. You no longer need to worry about installing and maintaining software applications or keeping your data updated to meet USPS® requirements. Data Quality Web Service does all the work for you — freeing you to concentrate on your business and your customers!

How Do I Use Data Quality Web Service?

To begin, you must create an account through one of Melissa Data's sales representatives. You will then receive an e-mail containing your unique CustomerID, which is required for each record you submit to Data Quality Web Service.

Next, simply provide the records you want us to process, and we'll take care of the rest! All records must be submitted in XML format.

Based on the "input" and "output" tags you include in your records, Data Quality Web Service can automatically perform as many as five separate processes on the data which your records contain. These processes include AddressVerify, GeoCode, CorrectAreaCode, LookupAreaCode, and NameParse. Descriptions of each process, including the required and optional input and output tags for each process, are included in this reference guide. You do not have to include all the optional tags a process offers —only those you want Data Quality Web Service to populate/update. In addition, any tags unique to your business (record number, and so on) will not be altered or deleted by Data Quality Web Service.

Once you submit a record, Data Quality Web Service will instantly verify, update, correct, and enhance the data it contains.

Submitting XML Records to Data Quality Web Service

Each XML submission must be sent with the following tags:

```
<?xml version="1.0"?><RecordSet><CustomerID></CustomerID><Record></Record></RecordSet>
```

The required and optional tags for each requested process should be included between the `<Record></Record>` tags in your submitted records. Following is a sample XML submission using the AddressVerify and NameParse processes:

```
<?xml version="1.0"?>
<RecordSet>
<CustomerID>0123456789</CustomerID>
<Record>
<FullName>John Smith</FullName>
<FirstName/>
<LastName/>
<Gender/>
<Address>22382 Ave Emprisa</Address>
```

```
<City/>
<State/>
<Zip>92688</Zip>
<Plus4/>
</Record>
<ErrorString/>
</RecordSet>
```

Now here is the same record after Data Quality Web Service returns it!

```
<?xml version="1.0"?>
<RecordSet>
<CustomerID>01234567890</CustomerID>
<Record>
<FullName>John Smith</FullName>
<FirstName>John</FirstName>
<LastName>Smith</LastName>
<Gender>M</Gender>
<Address>22382 Avenida Empresa</Address>
<City>Rancho Santa Margarita</City>
<State>CA</State>
<Zip>92688</Zip>
<Plus4>2112</Plus4>
</Record>
<ErrorString/>
</RecordSet>
```

The previous examples are formatted to fit the page.

SUBMITTING RECORDS FOR PROCESSING

Submit your records in XML format to:

```
http://xml.melissadata.com/xml.asp -- Non-Secure
https://xml.melissadata.com/xml.asp -- Secure
```

Backup servers:

```
http://xml3.melissadata.com/xml.asp - Non secured
https://xml3.melissadata.com/xml.asp - Secured
```

Chapter 2

Processes

IN THIS CHAPTER...

“Processes” on page 5

“Address Verify Process” on page 5

“GeoCode Process” on page 6

“CorrectAreaCode” on page 7

“LookupAreaCode Process” on page 8

“NameParse Process” on page 9

PROCESSES

Required Input Tags

The following tags must be included with all submissions to Data Quality Web Service:

```
<RecordSet>
<CustomerID>
<Record>
```

Optional Input Tags

The following tag is optional and can be used with all submissions to Data Quality Web Services:

```
<ErrorString>
```

This tag will be added if necessary, even if you do not provide it. Therefore, look for this tag on the return.

ADDRESS VERIFY PROCESS

This process compares submitted addresses to the USPS National Data File. Based on this comparison, Data Quality Web Service (SM) verifies and corrects the address data contained in your “input” tags. It can also enhances your records by adding/updating data to any “output” tags you include in your submission.

Required Input Tags

The following tags must be included (and populated) for this process to succeed:

```
<Address>
<City>
<State>
<Zip>
<Lastline>
```

City and State tags are optional if the ZIP tag is provided. Likewise, the ZIP tag is optional if the City and State tags are provided. City, State, and ZIP tags are optional if the Lastline (City, State, ZIP as one tag) is provided.

Optional Input Tags

The following tags are optional and are not required for this process to succeed:

```
<Company>
<Address2>
<Suite>
<Urbanization>
```

Output Tags

The following tags are not required for this process to be successful (unless they are also listed as a required input tag for the process), but can be updated/populated by Data Quality Web Service (SM) if included in your submitted records. These are the only tags populated by the AddressVerify process.

```
<ParsedAddressRange>
<ParsedPreDirection>
<ParsedStreetName>
<ParsedSuffix>
<ParsedPostDirection>
<ParsedSuiteName>
<ParsedSuiteRange>
<ParsedGarbage>
<Company>
<Address>
<Address2>
<Suite>
<City>
<CityAbbreviation>
<State>
<Zip>
<Plus4>
<AreaCodeOfZip>
<CarrierRoute>
<DeliveryPointCode>
<DeliveryPointCheckDigit>
<CountyFips>
<CountyName>
<TimeZone>
<TimeZoneCode>
<AddressErrorCode>
<AddressErrorString>
<Msa>
<Pmsa>
<AddressTypeCode>
<AddressTypeString>
<Urbanization>
<CongressionalDistrict>
<PrivateMailBox>
<Lacs>
<AddressStatusCode>
<AddressDatabaseDate>
```

GeoCode Process

This process can add geographic and census data to your records based on the ZIP™ and Plus4 codes submitted.

Required Input Tags

The following tag must be included (and populated) for this process to succeed:

```
<Zip>
```

Optional Input Tags

The following tag is optional and is not required for this process to succeed:

```
<Plus4>
```

Output Tags

The following tags are not required for this process to be successful (unless they are also listed as a required input tag for the process), but can be updated/populated by Data Quality Web Service (SM) if included in your submitted records. These are the only tags populated by the GeoCode process.

```
<Latitude>
<Longitude>
<CBSACode>
<CBSADivisionCode>
<CBSADivisionLevel>
<CBSADivisionTitle>
<CBSALevel>
<CBSATitle>
<CensusTract>
<CensusBlock>
<GeocodeStatusCode>
<GeocodeErrorCode>
<GeocodeDatabaseDate>
<PlaceCode>
<PlaceName>
```

Remarks

The GeoCode process populates output tags based on the following criteria:

1. The full ZIP and Plus 4 combination.
2. If no information is found for the Plus4 code, then the centroid of the ZIP + 2 (The ZIP Code and the first two digits of the Plus4 code) will be used.
3. If the ZIP+2 cannot be found, then the centroid for the 5-digit ZIP Code will be used.

The GeoCodeStatus output tag indicates the criteria level used.

CorrectAreaCode

This process retrieves the correct area code for the phone number and ZIP Code™ in the submitted record.

Required Input Tags

The following tags must be present and populated in the submitted record for this process to succeed:

```
<Phone>
<Zip>
```

Output Tags

The following tags are not required for this process to be successful (unless they are also listed as a required input tag for the process), but can be updated/populated by Data Quality Web Service (SM) if included in your submitted records. These are the only tags populated by the CorrectAreaCode process.

```
<PhoneAreaCode>
<PhonePrefix>
<PhoneSuffix>
<PhoneExtension>
<PhoneNewAreaCode>
<PhoneErrorCode>
<PhoneStatusCode>
<PhoneDatabaseDate>
```

LOOKUPAREACODE PROCESS

This process verifies and/or updates the telephone area code included in your records and can add information based on the combination of the verified/updated area code and the telephone number prefix contained in the submitted record.

Required Input Tags

The following tag must be included (and populated) for this process to succeed:

```
<Phone>
```

Optional Input Tags

The following tag is optional and is not required for this process to succeed:

```
<Zip>
```

If submitted, the ZIP Code is used to populate the optional <PhoneDistance> output tag with the distance in miles between the submitted area code/prefix and ZIP Code.

This input tag is required to correct area codes.

Output Tags

The following tags are not required for this process to be successful (unless they are also listed as a required input tag for the process), but can be updated/populated by Data Quality Web Service (SM) if included in your submitted records. These are the only tags populated by the LookupAreaCode process.

```
<PhoneAreaCode>
<PhonePrefix>
<PhoneSuffix>
<PhoneExtension>
<PhoneNewAreaCode>      If no new area code is found, then this tag will contain the same
                        area code as the <PhoneAreaCode> tag
<PhoneCity>
<PhoneState>
<PhoneDistance>
<PhoneCountyFips>
<PhoneCountyName>
<PhoneLatitude>
```

```
<PhoneLongitude>
<PhoneTimeZone>
<PhoneTimeZoneCode>
<PhoneMsa>
<PhonePmsa>
<PhoneCountry>
<PhoneStatusCode>
<PhoneErrorCode>
<PhoneDatabaseDate>
```

NAMEPARSE PROCESS

This process parses the names submitted in your records and can even identify the gender of the submitted name.

Required Input Tags

The following tag must be included (and populated) for this process to succeed:

```
<FullName>
```

Output Tags

The following tags are not required for this process to be successful (unless they are also listed as a required input tag for the process), but can be updated/populated by Data Quality Web Service (SM) if included in your submitted records. These are the only tags populated by the NameParse process.

```
<FirstName>
<MiddleName>
<LastName>
<NamePrefix>
<NameSuffix>
<Gender>
<NameStatusCode>
<NameDatabaseDate>
```

Chapter 3

Input and Output Tags

This chapter contains all of the input and output tags, in alphabetical order, that can be used with Data Quality Web Service.

<Address>

The delivery address provided in a submitted record.

Example:

```
<Address>22382 Avenida Empressa</Address>
```

(Data Type = String, Length = 75)

Remarks:

1. The AddressVerify process corrects and standardizes addresses in the <Address> tag. Address corrections may include fixing misspelled street names or inserting missing suffixes (for example, “ST” or “AVE”) and directionals (for example, “N”, “S”, “E”, “W”). Address standardization involves the conversion of suffixes and directionals to preferred USPS abbreviations (for example, “Street” becomes “ST”).
2. If the address contained in the <Address> tag is not verifiable and an <Address2> tag is included which contains a separate and verifiable address, the address verified in <Address2> will be moved automatically to the <Address> tag. The unverifiable address from the <Address> tag will be moved to the <Address2> tag.
3. If a suite is attached to the end of an address in the <Address> tag, it will be moved automatically to the <Suite> tag, if the <Suite> tag is present.

<Address2>

The second address line in a submitted record.

Example:

```
<Address>100 Main Street</Address>
<Address2>PO Box 1234</Address2>
```

(Data Type = String, Length = 75)

Remarks:

If the <Address2> tag contains a complete and separate address from the <Address> tag, Data Quality Web Service will attempt to verify the address in the <Address> tag. If it cannot verify the address in the <Address> tag, it will attempt to verify the address in the <Address2> tag. If successful, it will move the verified address from the <Address2> tag to the <Address> tag, and move the unverified address from <Address> to <Address2>.

If a standard suite designator (#, Apt, Apartment, Suite, Ste., etc.) is detected in <Address2>, it will be appended to the end of the <Address> tag, unless a <Suite> tag is present, in which case it will be moved to the <Suite> tag.

<AddressDatabaseDate>

The date value representing the date of the USPS National Data File used by the AddressVerify process to verify your addresses.

Example:

```
<AddressDatabaseDate>8/15/00</AddressDatabaseDate>
```

(Data Type = String, Length = 10)

Remarks:

Files are updated monthly.

<AddressErrorCode>

A single letter used to describe any problems encountered by AddressVerify when attempting to verify the address in the <Address> tag.

Example:

```
<AddressErrorCode>M</AddressErrorCode>
```

(Data Type = String, Length = 1)

Remarks:

Address codes and their related error types are as follows:

Code	Error	Description
Empty	No Error	Address is correct.
M	Multiple Matches	Matches More than one record in the USPS National Data File matches the address in the <Address> tag. Including more information, such as city or urbanization names, can help reduce the number of multiple match errors.
N	No Street Data for ZIP	The ZIP Code exists, but no street begins with the same letter in that ZIP Code.
R	Address out of Range	The number was found but the street number was not between the low and high range in the USPS National Data File.
T	Component Mismatch	Either the directionals or the suffix field do not match the USPS National Data File, and there is more than one choice for correcting the address. For example, if the <Address> tag contains “100 Main St” and the USPS National Data File contains only “100 E Main St” and “100 Main Ave”, this error code is returned because we do not know whether to add the directional “E” to the address or change the suffix from “St” to “Ave”.
U	Unknown Street	An exact street name could not be found attempts to match the street name phonetically resulted in no matches or matches to more than one street name.
X	Non-Delivarable Address	The physical location exists but there are no homes on this street.
Z	ZIP Code Error	The ZIP Code provided does not exist, and no ZIP Code could be matched to the city and state provided.

<AddressErrorString>

A textual description of a problem found when attempting to verify/correct an address.

Example:

```
<AddressErrorString>Unknown Street</AddressErrorString>
```

(Data Type = String, Length = 24)

Remarks:

See **"<AddressErrorCode>" on page 14** for additional details on address errors.

Address Error Code	Address Error String Returned
M	Multiple Matched
N	No Data Available for City
R	Range Error
T	Component Error
U	Unknown Street
X	Undeliverable Address
Z	Invalid ZIP Code
<Empty>	No Error

<AddressStatusCode>

The level of coding performed on the submitted address.

Example:

```
<AddressStatusCode>9</AddressStatusCode>
```

(Data Type = String, Length = 1)

Remarks:

Address status codes and their related levels are as follows:

Code	Level
X	Address was not coded.
S	The address was standardized but not coded.
5	Multiple matches were found for the address, but all possible matches are located in the same ZIP Code. The <Zip> data returned will be correct but no <CarrierRoute> or <Plus4> output will be available.
6	Fully-coded Canadian address
7	Multiple matches were found for the address, but all possible matches are located in the same ZIP Code and carrier route. The <Zip> and <CarrierRoute) data returned will be correct but no <Plus4> output will be available.
9	The address was fully coded.
V	Street number validated to DPV level.

If an "S" or "X" is returned, check the <AddressErrorCode> tag, if you included it in your submission, for the reason why the address could not be coded.

<AddressTypeCode>

The type of address coded in the submitted record.

Example:

```
<AddressTypeCode>S</AddressTypeCode>
```

(Data Type = String, Length = 1)

Remarks:

The type of address indicated by each code letter is as follows::

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

If an address cannot be verified, this tag will not be populated.

<AddressTypeString>

The textual description of the type of address in the submitted record. See **"<AddressTypeCode>" on page 17**.

Example:

```
<AddressTypeString>Rural Route</AddressTypeString>
```

(Data Type = String, Length = 30)

Remarks:

Address types returned for this tag are as follows:

Address Type Code	Address Type String
F	Firm or company
G	General delivery
H	High-rise or business complex
P	PO Boxs
R	Rural Route
S	Street or residential

<AreaCodeOfZip>

The area code associated with the ZIP Code in the record's <Zip> tag.

Example:

```
<AreaCodeOfZip>214</AreaCodeOfZip>
```

(Data Type = String, Length = 3)

Remarks:

Since some ZIP Codes may encompass more than one area code, the data returned for this tag may not be as accurate as the data returned for <AreaCode> and <NewAreaCode> when utilizing phone correction process.

<CarrierRoute>

The carrier route associated with the address verified in the submitted record.

Example:

```
<CarrierRoute>C056</CarrierRoute>
```

(Data Type = String, Length = 4)

Remarks:

The first character in a carrier route is always a letter, and the last three characters are always numbers. The letter indicates the type of delivery associated with the address:

B = PO Box

C = City Delivery

G = General Delivery

H = Highway Contract

R = Rural Route

<CBSACode>

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

Example:

```
<CBSACode>31100</CBSACode>
```

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.

<CBSADivisionCode>

This tag 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.

Example:

```
<CBSADivisionCode>31084</CBSADivisionCode>
```

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.

<CBSADivisionLevel>

This tag 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.

Example:

```
<CBSADivisionLevel>Metropolitan Statistical Area</CBSADivisionLevel>
```

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.

<CBSADivisionTitle>

This tag 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.

Example:

```
<CBSADivisionTitle>Los Angeles-Long Beach-Glendale, CA</CBSADivisionTitle>
```

Remarks

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

<CBSALevel>

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

Example:

```
<CBSALevel>Metropolitan Statistical Area</CBSALevel>
```

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.”

<CBSATITLE>

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

Example:

```
<CBSATitle>Los Angeles-Long Beach-Santa Ana, CA</CBSATitle>
```

Remarks

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

<CensusBlock>

The Census Block Group Number for the ZIP + 4® code verified in the submitted record.

Example:

```
<CensusBlock>1</CensusBlock>
```

(Data Type = String, Length = 1)

Remarks:

Block groups are the next level above census blocks in the geographic hierarchy. A BG is a combination of census blocks that is a subdivision of a census tract or block numbering area (BNA). (A county or its statistically equivalent entity contains either census tracts or BNAs; it can not contain both.) A BG consists of all census blocks whose numbers begin with the same digit in a given census tract or BNA; for example, BG 3 includes all census blocks numbered in the 300s. The BG is the smallest geographic entity for which the decennial census tabulates and publishes sample data.

The block group returns a one-character string containing the block group number.

If the GeoCode process is unsuccessful, this tag remains empty.

<CensusTract>

The Census Tract for the ZIP + 4 Code of the submitted record.

Example:

```
<CensusTract>121002</CensusTract>
```

(Data Type = String, Length = 6)

Remarks:

Census Tracts are small, relatively permanent statistical subdivisions of a county. Census Tracts are delineated for all metropolitan areas MSA's and other densely populated counties by local census statistical areas committees, following Census Bureau guidelines. Census Tracts returned by the GeoCode process are six digits in length.

<City>

The city provided in the submitted record, or the official city name for the ZIP Code provided in the submitted record.

Example:

```
<City>Dallas</City>
```

(Data Type = String, Length = 28)

Remarks:

1. The <City> tag is optional if a populated <Zip> tag is submitted.
2. If a city name is not provided in a submitted record, the AddressVerify process will add the official city name for the ZIP Code submitted in the record.
3. If a city name is provided, the AddressVerify process will only change the provided city name if it is incorrect or an unapproved mailing name. In these cases, the official city name for the ZIP Code will replace the existing city name.
4. If the supplied city and state do not match the ZIP Code, AddressVerify will give preference to the city name instead of the ZIP Code. This logic is based on the assumption that a ZIP Code with one typo will result in more incorrect address matches that a city name with a few typos.

<CITYABBREVIATION>

This is the official 13-letter city name abbreviation for the city name provided in the <City> tag.

Example:

```
<City>Rancho Santa Margarita</City>
<CityAbbreviation>Rcho Sta Mar</CityAbbreviation>
```

(Data Type = String, Length = 13)

Remarks:

If after being verified by the AddressVerify process, the contents of the <City> tag exceed 13 characters, AddressVerify will populate this tag with the official abbreviation for the city name.

If the city name in the <City> tag is 13 letters or shorter, the <CityAbbreviation> tag will contain the entire city name. Therefore, you would always get the city name from this tag.

<COMPANY>

The company name entered in the submitted record.

Example:

```
<Company>Melissa Data</Company>
```

(Data Type = String, Length = 50)

Remarks:

1. If the <Company> tag contains the name of a company that has been assigned a specific Plus4 code by the USPS, AddressVerify will add that Plus4 code to the <Plus4> tag, if a <Plus4> tag is provided.
2. If a company name is not supplied for a company that has been assigned a specific Plus4 code, the more generic “street level default” Plus4 code will be added to the <Plus4> tag of the record instead, if a <Plus4> tag is provided.
3. Contents of the <Company> tag are not altered or changed by the AddressVerify process.

<CongressionalDistrict>

The congressional district number associated with the address in the submitted <Address> tag.

Example:

```
<CongressionalDistrict>48</CongressionalDistrict>
```

(Data Type = String, Length = 2)

Remarks:

This two-digit number is accurate to the ZIP + 4 level. For states with only one congressional representative, the value “01” is returned.

<CountyFips>

The 5-digit county FIPS code associated with the address submitted in the <Address> tag.

Example:

```
<CountyFips>06059</CountyFips>
```

(Data Type = String, Length = 5)

Remarks:

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

The county FIPS code returned in this tag is accurate to the ZIP + 4 level.

<CountyName>

The county name associated with the address submitted in the <Address> tag.

Example:

```
<CountyName>Orange</CountyName>
```

(Data Type = String, Length = 26)

Remarks:

The county name is associated with the County FIPS (see **“<CountyFips>” on page 33**) and is accurate to the ZIP + 4 level.

<CustomerID>

This tag must contain the 10 digit alpha-numeric ID for the Data Quality Web Service (SM) customer submitting the record(s).

Example:

```
<CustomerID>Flipper151</CustomerID>
```

(Data Type = String, Length = 10)

Remarks:

The <CustomerID> tag must contain the correct customer ID in order for the submitted record(s) to be processed.

<DeliveryPointCheckDigit>

The single-digit number representing the delivery point check digit associated with the address verified by the AddressVerify process.

Example:

```
<DeliveryPointCheckDigit>1</DeliveryPointCheckDigit>
```

(Data Type = String, Length = 1)

Remarks:

The delivery point check digit makes up the 12th position of a 12-digit POSTNET barcode. In the12-digit POSTNET barcode, the ZIP Code is used for positions 1 through 5, the Plus4 code for positions 6 through 9, the delivery point code for positions 10 and 11, and this check digit for position 12.

See **“<DeliveryPointCode>” on page 37** for more information.

<DeliveryPointCode>

The two-digit delivery point code associated with the address verified by the AddressVerify process.

Example:

```
<DeliveryPointCode>82</DeliveryPointCode>
```

(Data Type = String, Length = 2)

Remarks:

This 2-digit string makes up the 10th and 11th positions of a 12-digit POSTNET barcode. See **"<DeliveryPointCheckDigit>" on page 36**.

<DisableAddressSwapping>

If address data in the <Address> tag is uncodable, DQWS checks the <Address2> tag. If address data in the <Address2> tag is codable, DQWS “swaps” the data, moving the codable address from the <Address2> tag to the <Address> tag, and the uncodable address information from the <Address> tag to the <Address2> tag. The swapping lets you know which address lines were coded.

DQWS performs address swapping by default. To disable address swapping, pass this tag with a value of “1”.

Example:

```
<DisableAddressSwapping>1</DisableAddressSwapping>
```

Remarks:

If address swapping is disabled, the <ErrorCode> tag will contain either a “1” or “2” to indicate which address was coded.

<ErrorString>

This tag contains a textual description of any errors that occur during record processing.

Example:

```
<ErrorString>The CustomerID tag could not be found or the number contained in the tag was invalid</ErrorString>
```

(Data Type = String, Length = 255

Remarks:

This tag is populated by Data Quality Web Service when an error is discovered.

While it is impossible to document every possible error string that could be returned in this tag, the following is what you could expect:

1. Any exceptions that are returned from the XML DOM parser. In the case of an improperly formed XML document.
2. Any SQL exceptions that are generated.
3. Invalid customer number or an inactive customer number.
4. A general error message along the lines of "Transmission Error" for any exceptions that are thrown by our objects that should not be.

<FIRSTNAME>

The name submitted in the <FirstName> tag or added by the NameParse process when parsing the contents of the <FullName> tag.

Example:

```
<FirstName>Jonathan</FirstName>
```

(Data Type = String, Length = 12)

Remarks:

The NameParse process will populate this tag with the first name found in the <FullName> tag.

<FullName>

This tag contains the full name entered into a <FullName> tag in a submitted record, including the name prefix, first name, middle name, and name suffix.

Example:

```
<FullName>Dr. John Leon Nydam III</FullName>
```

(Data Type = String, Length = 50)

Remarks:

The NameParse process can parse the contents of this tag into its component parts.

Example:

```
<NamePrefix> = "Dr"
<FirstName> = "John"
<MiddleName> = "Leon"
<LastName> = "Nydam"
<NameSuffix> = III
```

<GENDER>

This tag indicates whether the gender of the first name in the <FirstName> tag is male, female, neutral, or unknown.

Example:

```
<Gender>M</Gender>
```

(Data Type = String, Length = 1)

Remarks:

The NameParse process will return one of the following values for this tag::

Code	Gender
F	Female
M	Male
N	Neutral (Name could be Male or Female)
U	Unknown — the name does not exist in our database

<GeoCodeDatabaseDate>

The date of the data used by the GeoCode process.

Example:

```
<GeoCodeDatabaseDate>8/15/00</GeoCodeDatabaseDate>
```

(Data Type = String, Length = 10)

Remarks:

None.

<GeoCodeErrorCode>

The error code returned by the GeoCode process when it fails to match a ZIP + 4 code submitted in a record.

Example:

```
<GeoCodeErrorCode>Z</GeoCodeErrorCode>
```

(Data Type = String, Length = 1)

Remarks:

The GeoCode process returns one of the following error codes when it cannot match a ZIP + 4 code submitted in a record::

Code	Description
Z	Bad ZIP Code – Invalid ZIP Code submitted

<GeocodeStatusCode>

The GeocodeStatusCode indicates how precisely the ZIP + 4 code in the submitted record matches the ZIP + 4 codes in the GeoCode database. This determines the accuracy of the geographic and census-related data returned by the GeoCode process.

Example:

```
<GeocodeStatusCode>9</GeocodeStatusCode>
```

(Data Type = String, Length = 1)

Remarks:

The GeoCode process returns one of the following values to this tag::

Code	Description
9	Data returned is accurate to the ZIP + 4 centroid.
7	Data returned is accurate to the ZIP+2 centroid.
5	Date returned is accurate to the 5-digit ZIP Code centroid.
X	No data was returned.

<Lacs>

This indicator notifies you if the submitted address has undergone a LACS conversion

Example:

```
<Lacs>L</Lacs>
```

(Data Type = String, Length = 1)

Remarks:

There are two possible values for this tag::

Value	Description
L	The address in this record has undergone a LACS conversion.
<empty>	The address in this record has not undergone a LACS conversion.

The LACS tag is used when rural route addresses are converted to city-style addresses to allow emergency services (ambulance, police, fire, etc.) to find the addresses more efficiently.

After a conversion, the old address is retained in the USPS National Data File for a period of one year. Afterwards, no Plus4 code will be returned by AddressVerify for this address.

The new addresses are not contained in the USPS National Data File. To change the old address to the newly converted address, you will need to send addresses marked “L” by AddressVerify to a company that does LACS processing. Melissa Data will be glad to assist you with this process.

<LACSLinkReturnCode>

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.

Example:

```
<LACSLinkReturnCode>A</LACSLinkReturnCode>
```

(Data Type = String, Length = 1)

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.

Code	Description
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.

<LACSLinkIndicator>

This property returns a “Y” if the submitted address was corrected to a new address found in the LACS^{Link} data. A “N” is returned if the address was not updated by LACS^{Link}.

Example:

```
<LACSLinkIndicator>Y</LACSLinkIndicator>
```

(Data Type = String, Length = 1)

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.

If the submitted address was found in the LACS^{Link} database and changed to the new address, this one-character property will return a "Y," otherwise it will return an "N." An “S” is returned if the submitted address was corrected to a new address found in the LACS^{Link} data but contained a suite that could not be matched.

<LastLine>

Submits the city, state and ZIP Code as a single string.

Example:

```
<LastLine>Rancho Santa Margarita, CA 92688</LastLine>
```

(Data Type = String, Length = 40)

Remarks:

You can use this tag as an alternative method to setting the City, State and Zip properties individually. For example, set LastLine to "Rancho Santa Margarita, CA 92688-2212" rather than setting the City tag to "Rancho Santa Margarita" and the State tag to "CA."

This tag is useful if you have address data that is composed of lines of plain text and the City, State and ZIP Codes do not have their own separate fields.

<LastName>

The last name as submitted in the <LastName> tag or parsed from the <FullName> tag by the NameParse process.

Example:

```
<LastName>Smith</LastName>
```

(Data Type = String, Length = 32)

Remarks:

See **“NameParse Process” on page 9**“ and **“<FullName>” on page 41** for a description of name parsing.

<LATITUDE>

The latitude of the ZIP Code or ZIP + 4 centroid for the submitted record.

Example:

```
<Latitude>85.123456</Latitude>
```

(Data Type = String, Length = 9)

Remarks:

Latitude is the geographic coordinate of a point measured in degrees north or south of the equator.

The contents of this tag are returned by the GeoCode process based on the <ZIP> or <ZIP><Plus4> codes submitted in the record. The level of accuracy for the value returned is displayed in the <GeocodeStatusCode> tag.

Since all U.S. ZIP Codes are north of the equator, this value will always be a positive number.

<LONGITUDE>

The longitude of the ZIP Code or ZIP+9 centroid for the submitted record.

Example:

```
<Longitude>-117.356789</Longitude>
```

(Data Type = String, Length = 11)

Remarks:

Longitude is the geographic coordinate of a point measured in degrees east or west of the Greenwich meridian.

The contents of this tag are returned by the GeoCode process based on the <ZIP> or <ZIP> <Plus4> codes submitted in the record. The level of accuracy for the value returned is displayed in the <GeocodeStatusCode> tag.

A negative longitude value indicates a westerly longitude. (All points west of the Greenwich meridian, including the United States, have a negative longitude.)

<MIDDLENAME>

The middle name as submitted in the <MiddleName> tag or parsed from the <FullName> tag by the NameParse process.

Example:

```
<MiddleName>Lee</MiddleName>
```

(Data Type = String, Length = 12)

Remarks:

See **“NameParse Process” on page 9**“ and **“<FullName>” on page 41** for a description of name parsing.

<Msa>

The MSA number associated with the address verified by the AddressVerify process.

Example:

```
<Msa>5495</Msa>
```

(Data Type = String, Length = 4)

Remarks:

The federal Office of Management and Budget defines a Metropolitan Statistical Area (MSA) as a geographic entity containing a core area, plus adjacent communities. There must be one city with 50,000 or more people, or the presence of an Urbanized Area and a total population of at least 100,000 people (75,000 people in New England).

<MsaDesc>

The name of the region corresponding to the code returned by the <Msa> tag.

Example:

```
<MsaDesc>LOS ANGELES-RIVERSIDE-ORANGE COUNTY, CA </MsaDesc>
```

Remarks:

See **“<Msa>” on page 52** for more information about MSAs.

<NameDatabaseDate>

The date value representing the date of the database used by the NameParse process.

Example:

```
<NameDatabaseDate>8/15/00</NameDatabaseDate>
```

(Data Type = String, Length = 10)

Remarks:

None.

<NamePrefix>

The name prefix as submitted in the <NamePrefix> tag or parsed from the <FullName> tag by the NameParse process.

Example:

```
<NamePrefix>Dr.</NamePrefix>
```

(Data Type = String, Length = 10)

Remarks:

See **“NameParse Process” on page 9** and **“<FullName>” on page 41** for a description of name parsing.

<NameStatusCode>

The status code representing the result of the NameParse process.

Example:

```
<NameStatusCode>V</NameStatusCode>
```

(Data Type = String, Length = 1)

Remarks:

Descriptions of codes returned in this tag are as follows::

Code	Description
<Empty>	Name successfully parsed.
V	Vulgar word found in full or first name.
X	Unable to parse name

<NameSuffix>

The name suffix as submitted in the <NameSuffix> tag or parsed from the <FullName> tag by the NameParse process.

Example:

```
<NameSuffix>III</NameSuffix>
```

(Data Type = String, Length = 10)

Remarks

See **“NameParse Process” on page 9** and **“<FullName>” on page 41** for a description of name parsing.

<PARSEDADDRESSRANGE>

The address number contained in the <Address> tag.

Example:

```
<Address>147 N Main Street SE, Apt 25, 2nd Floor</Address>

<ParsedAddressRange>147</ParsedAddressRange>
```

(Data Type = String, Length = 10)

Remarks:

None.

<ParsedGarbage>

This tag contains any leftover text after an address has been successfully verified (and parsed).

Example:

```
<Address>147 N Main Street SE, Apt 25, 2nd Floor</Address>

<ParsedGarbage>2nd Floor</ParsedGarbage>
```

(Data Type = String, Length = 50)

Remarks:

This tag contains all the characters and/or tokens that do not appear to be part of the successfully verified address. Common items added here include community center names, names of people, additional delivery instructions, etc.

<PARSEDPOSTDIRECTION>

Any geographical directional that follows the street name in a successfully verified address.

Example:

```
<Address>147 N Main Street SE, Apt 25, 2nd Floor</Address>

<ParsedPostDirection>SE</ParsedPostDirection>
```

(Data Type = String, Length = 2)

Remarks:

The post direction included in the delivery address will be abbreviated when parsed to this tag. For example, “Southeast” will be abbreviated to “SE”.

<PARSEDPREDIRECTION>

Any geographical directional that precedes the street name in a successfully verified address.

Example:

```
<Address>147 SE Main Street, Apt 25, 2nd Floor</Address>

<ParsedPreDirection>SE</ParsedPreDirection>
```

(Data Type = String, Length = 2)

Remarks:

The pre-direction included in the delivery address will be abbreviated when parsed to this tag. For example, “Southeast” will be abbreviated to “SE.”

<ParsedStreetName>

The name of the street contained in the verified address.

Example:

```
<Address>147 N Main Street SE, Apt 25, 2nd Floor</Address>

<ParsedStreetName>Main</ParsedStreetName>
```

(Data Type = String, Length = 28)

Remarks:

For street names that beginning with any of the following Spanish words, the first word in the street name will be moved to the <ParsedSuffix> tag and the next word in the street name will be parsed to this tag. This occurs because Spanish street names have the suffix at the front of the name.

The Spanish words that get moved to the <ParsedSuffix> tag are: “Avenida”, ”Calle”, ”Camino”, ”Paseo”, and “Via.”

Example:

Street Name = “Avenida Empresa” is parsed to:

```
<ParsedStreetName>Empresa</ParsedStreetName>
<ParsedSuffix>Avda<ParsedSuffix>
```

<ParsedSuffix>

The suffix portion of a successfully verified address.

Example:

```
<Address>147 N Main Street SE, Apt 25, 2nd Floor</Address>
<ParsedSuffix>St</ParsedSuffix>
```

(Data Type = String, Length = 2)

Remarks:

The AddressVerify process puts the abbreviated suffix of the successfully verified address in this tag. For example, the word “Street” is abbreviated to “St” when parsed to this tag.

Spanish abbreviations:

Avenida = Avda

Camino = Cmno

Calle = Clle

Paseo = PSO

Via = Viia

<ParsedSuiteName>

The secondary unit name from a successfully verified address.

Example:

```
<Address>147 N Main Street SE, Apt 25, 2nd Floor</Address>

<ParsedSuiteName>Apt</ParsedSuiteName>
```

(Data Type = String, Length = 8)

Remarks:

If included in a submitted address that is successfully verified, AddressVerify will parse any of the following values to this tag:

“#”, “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”.

<PARSEDSUITERANGE>

The range of any secondary unit from a successfully verified address.

Example:

```
<Address>147 N Main Street SE, Apt 25, 2nd Floor</Address>

<ParsedSuiteRange>25</ParsedSuiteRange>
```

(Data Type = String, Length = 10)

Remarks:

None.

<PHONE>

The telephone area code, prefix (first 3 digits after the area code), suffix (last 4 digits of the phone number), and extension (if any) submitted.

Example:

```
<Phone>949-589-5200 ext 100</Phone>
```

(Data Type = String, Length = 25)

Remarks:

1. If the area code has been split for the submitted area code/prefix combination, the new area code will be added to the <PhoneNewAreaCode> tag, if present.
2. If a correction was made to the area code based on the ZIP Code submitted with the record, the corrected area code will be added to the <PhoneNewAreaCode> tag, if present.

<PHONEAREACODE>

The telephone area code submitted with the record or parsed by the LookupAreaCode or CorrectAreaCode processes.

Example:

```
<PhoneAreaCode>714</PhoneAreaCode>
```

(Data Type = String, Length = 3)

Remarks:

1. If this tag is empty and the LookUpAreaCode or CorrectAreaCode processes detect less than 10 digits in the <Phone> tag, they will assume that the area code is missing from the submitted telephone number and this tag will remain empty.
2. If the area code has been split for the submitted area code/prefix combination, the new area code will be added to the <PhoneNewAreaCode> tag, if present.
3. If a correction was made to the area code based on the ZIP Code submitted with the record, the corrected area code will be added to the <PhoneNewAreaCode> tag, if present.
4. The <PhoneNewAreaCode> tag contains the same area code if no changes were detected.

<PhoneCity>

The city associated with the telephone number submitted in the <Phone> tag.

Example:

```
<PhoneCity>Rancho Santa Margarita</PhoneCity>
```

(Data Type = String, Length = 28)

Remarks:

None.

<PHONECOUNTRY>

The official two-character country code associated with the telephone number in the <Phone> tag.

Example:

```
<PhoneCountry>US</PhoneCountry>
```

(Data Type = String, Length = 2)

Remarks:

This tag only applies to the United States and Canada.

US = United States

CA = Canada

<PhoneCountyFips>

The County FIPS number (see **"<CountyFips>" on page 33**) associated with the telephone number submitted in the <Phone> tag.

Example:

```
<PhoneCountyFips>06037</PhoneCountyFips>
```

(Data Type = String, Length = 5)

Remarks:

None.

<PHONECOUNTYNAME>

The county name associated with the telephone number submitted in the <Phone> tag.

Example:

```
<PhoneCountyName>Orange</PhoneCountyName>
```

(Data Type = String, Length = 25)

Remarks:

None.

<PhoneDatabaseDate>

The date of the phone database used by the CorrectAreaCode and LookupAreaCode processes.

Example:

```
<PhoneDatabaseDate>8/15/00</PhoneDatabaseDate>
```

(Data Type = String, Length = 10)

Remarks:

None.

<PHONEDISTANCE>

The distance between the centroid of the submitted area code/prefix number in the <Phone> tag and the centroid of the submitted ZIP Code in the <Zip> tag.

Example:

```
<PhoneDistance>2</PhoneDistance>
```

(Data Type = String, Length = 4)

Remarks:

Distance is entered in miles. If the ZIP Code is not submitted, no distance is returned.

<PhoneErrorCode>

The error code representing the result of the LookupAreaCode or CorrectAreaCode processes.

Example:

```
<PhoneErrorCode>A</PhoneErrorCode>
```

(Data Type = String, Length = 1)

Remarks:

Descriptions for error codes appearing in this tag are as follows::

Code	Description
A	Bad Area Code — The area code submitted does not exist in the database or contains characters.
B	Blank — The <Phone> tag is empty.
E	Bad Phone Number — Too many or too few digits in the <Phone> tag.
M	Multiple Match — Could not choose between two or more area codes because a bad or missing area code was submitted and the distance between the area codes was too close to choose one over the other.
P	Bad Prefix — The prefix submitted in the <Phone> tag does not exist in the database.
Z	Missing or Bad ZIP Code — The ZIP Code submitted in the <Zip> tag does not exist or no ZIP Code was entered.

<PhoneExtension>

The extension submitted in the <PhoneExtension> tag or parsed as part of the LookupAreaCode or CorrectAreaCode processes.

Example:

```
<PhoneExtension>100</PhoneExtension>
```

(Data Type = String, Length = 10)

Remarks:

None.

<PhoneLatitude>

The degree of latitude (see **“<Latitude>” on page 49**) associated with the telephone number submitted in the <Phone> tag.

Example:

```
<PhoneLatitude>85.1234</PhoneLatitude>
```

(Data Type = String, Length = 7)

Remarks:

Latitude is the geographic coordinate of the NPA/NXX wire center measured in degrees north or south of the equator (NPA/NXX = area code/prefix). The Latitude property is a seven-character (maximum) string value set by a call to the Lookup Method.

Example:

“85.1234”

<PHONELONGITUDE>

The longitude (see **“<Longitude>” on page 50**) associated with the telephone number submitted in the <Phone> tag.

Example:

```
<PhoneLongitude>-100.1234</PhoneLongitude>
```

(Data Type = String, Length = 8)

Remarks:

Longitude is the geographic coordinate of the NPA/NXX wire center measured in degrees east or west of the Greenwich Meridian(NPA/NXX = area code/prefix). The Longitude property is a nine-character (maximum) string value set by a call to the Lookup Method. It is accurate to four decimal places, and the negative sign is used to indicate a longitude in the United States.

Example:

“-100.1234”

<PhoneMsa>

The MSA number (see **“<Msa>” on page 52**) associated with the telephone number submitted in the <Phone> tag.

Example:

```
<PhoneMsa>5495</PhoneMsa>
```

(Data Type = String, Length = 4)

Remarks:

None.

<PhoneNewAreaCode>

The new or corrected area code for the area code/prefix combination submitted in the <Phone> tag.

Example:

```
<PhoneNewAreaCode>949</PhoneNewAreaCode>
```

(Data Type = String, Length = 3)

Remarks:

1. If this tag is empty and the LookUpAreaCode or CorrectAreaCode processes detect less than 10 digits in the <Phone> tag, they will assume that the area code is missing from the submitted telephone number and this tag will remain empty.
2. If the area code has been split for the submitted area code/prefix combination, the new area code will be added to the <PhoneNewAreaCode> tag, if present.
3. If a correction was made to the area code based on the ZIP Code submitted with the record, the corrected area code will be added to the <PhoneNewAreaCode> tag, if present.

<PhonePmsa>

The Pmsa number (see **"<Pmsa>" on page 90**) associated with the telephone number submitted in the <Phone> tag.

Example:

```
<PhonePmsa>5495</PhonePmsa>
```

(Data Type = String, Length = 4)

Remarks:

None.

<PHONEPREFIX>

The first 3 digits of the telephone number, not counting the area code. This tag is populated by parsing the <Phone> tag during the CorrectAreaCode or LookupAreaCode processes.

Example:

```
<PhonePrefix>589</PhonePrefix>
```

(Data Type = String, Length = 3)

Remarks:

None.

<PHONESTATE>

The state (see **"<State>" on page 96**) associated with the telephone number submitted in the <Phone> tag.

Example:

```
<PhoneState>CA</PhoneState>
```

(Data Type = String, Length = 2)

Remarks:

None.

<PHONESTATUSCODE>

The results of the LookupAreaCode or CorrectAreaCode process.

Example:

```
<PhoneStatusCode>C</PhoneStatusCode
```

(Data Type = String, Length = 1)

Remarks:

Descriptions of the codes found in this tag are as follows:

Code	Description
C	Corrected the area code by changing it according to the ZIP Code in the <Zip> tag of the submitted record.
U	Updated area code. The new area code is located in <NewAreaCode>. This change was identified using area code/prefix data.
X	Bad phone number.

<PhoneSuffix>

The last four digits of the telephone number in the <Phone> tag, as parsed by the CorrectAreaCode or LookupAreaCode processes.

Example:

```
<PhoneSuffix>5200</PhoneSuffix>
```

(Data Type = String, Length = 4)

Remarks:

None.

<PHONETIMEZONE>

The Time Zone (see **“<TimeZone>” on page 98**) associated with the telephone number submitted in the <Phone> tag.

Example:

```
<PhoneTimeZone>Pacific Time</PhoneCountyFips>
```

(Data Type = String, Length = 20)

Remarks:

1. The Time Zone does not account for daylight savings time.
2. If a telephone number is not present or is “bad” (see **“<PhoneStatusCode>” on page 83**), this tag will be empty.
3. These are the Time Zones that can be returned:

   “Atlantic Time”

   “Eastern Time”

   “Central Time”

   “Mountain Time”

   “Pacific Time”

   “Alaska Time”

   “Hawaii Time”

   “Samoa Time”

   “Marshall Island Time”

   “Guam Time”

   “Palau Time”

<PHONETIMEZONECODE>

The 1 or 2 digit code representing the Time Zone (see **"<TimeZoneCode>" on page 99**) associated with the telephone number submitted in the <Phone> tag.

Example:

```
<PhoneTimeZoneCode>8</PhoneTimeZoneCode>
```

(Data Type = String, Length = 2)

Remarks:

1. The Time Zone does not account for daylight savings time.
2. If a telephone number is not present or is "bad" (See **"<PhoneStatusCode>" on page 83**), this tag will be empty.
3. These are the Time Zone codes that can be returned:

Code	Time Zone
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

<PLACECODE>

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

Example

```
<PlaceCode>0643000</PlaceCode>
```

(Data Type = String, Length = 7)

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.

<PLACENAME>

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

Example

```
<PlaceName>Los Flores</PlaceName>
```

(Data Type = String, Length = 60)

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."

<Plus4>

The 4-digit ZIP Code add-on associated with the address verified in the submitted record.

Example:

```
<Plus4>2112</Plus4>
```

(Data Type = String, Length = 4)

Remarks:

If the address is successfully verified, this tag will contain the 4-digit ZIP Code add-on. If the address is not corrected, this tag will be blank.

<PMSA>

The PMSA number associated with the verified address in a submitted record.

Example:

```
<Pmsa>5495</Pmsa>
```

(Data Type = String, Length = 4)

Remarks:

In cases where an address belongs to multiple MSAs, the Primary Metropolitan Statistical Area (PMSA) is the market the address most closely identifies with.

<PmsaDesc>

The name of the region corresponding to the code returned by the <Pmsa> tag.

Example:

```
<PmsaDesc>ORANGE COUNTY</PmsaDesc>
```

Remarks:

See **“<Pmsa>” on page 90** for more information about PMSAs.

<PrivateMailBox>

The private mail box number associated with a CMRA (Commercial Mail Receiving Agency).

Example:

```
<PrivateMailBox>5200</PrivateMailBox>
```

(Data Type = String, Length = 10)

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 its customers’ private mailboxes.

<RBDI>

Returns a one-character code indicating whether the submitted address is a residence, a business or the status is unknown.

Example:

Parcel delivery to residential addresses is more expensive than to businesses. You can use this property to filter out one type of address or the other.

If the RBDI Add-on is installed, this property will return one of the following codes.

Code	Definition
R	Residential Address
B	Business Address
U	Uknown

<RECORD>

This tag delineates the beginning and end of each individual record you submit.

Example:

```
<Record>
<FirstName>John</FirstName>
<LastName>Doe</LastName>
<Address>22382 Avenida Empresa</Address>
<City>Rancho Santa Margarita</City>
<State>CA</State>
<Zip>92688</Zip>
<Plus4>2112</Plus4>
</Record>
```

Remarks:

None.

<RecordSet>

This tag signifies the beginning and end of a set of record submissions.

Example:

```
<RecordSet>
<Record>
<FullName>John Doe</FullName>
<FirstName/>
<LastName/>
<Gender/>
<Phone>714-555-1234</Phone>
<NewAreaCode/>
</Record>
<Record>
<FullName>Jane Smith</FullName>
<FirstName/>
<LastName/>
<Gender/>
<Phone>213-555-5678</Phone>
<NewAreaCode/>
</Record>
</RecordSet>
```

Remarks:

None.

<STATE>

The state associated with the address submitted in a record.

Example:

```
<State>CA</State>
```

(Data Type = String, Length = 2)

Remarks:

If the state is incorrect or not supplied, the correct two-letter state abbreviation for the address will be added to this tag during the AddressVerify process.

<SUITE>

The Suite name and number associated with the address submitted in a record.

Example:

```
<Suite>Ste 100</Suite>
```

(Data Type = String, Length = 30)

Remarks:

If a suite name and number is found at the end of the <Address> or <Address2> tag, it will be moved to the <Suite> tag during the AddressVerify process.

<TIMEZONE>

The Time Zone associated with the address verified in the submitted record.

Example:

```
<TimeZone>Pacific Time</TimeZone>
```

(Data Type = String, Length = 2o)

Remarks:

1. The Time Zone does not account for daylight savings time.
2. If an address cannot be verifed, this tag will be empty.
3. Following are the Time Zones that can be returned:

   “Military”

   “Atlantic Time”

   “Eastern Time”

   “Central Time”

   “Mountain Time”

   “Pacific Time”

   “Alaska Time”

   “Hawaii Time”

   “Samoa Time”

   “Marshall Island Time”

   “Guam Time”

   “Palau Time”

<TimeZoneCode>

The 1 or 2 digit code representing the Time Zone associated with the address verified in the submitted record.

Example:

```
<TimeZoneCode>8</TimeZoneCode>
```

(Data Type = String, Length = 2)

Remarks:

1. The Time Zone does not account for daylight savings time.
2. If an address cannot be verified, this tag will be empty.
3. These are the Time Zone codes that can be returned:

Code	Time Zone
0	Military Time (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

<URBANIZATION>

The urbanization tag applies to Puerto Rican address only.

Example:

```
<Urbanization>URB Camino Reposeo</Urbanization>
```

(Data Type = String, Length = 28)

Remarks:

The <Urbanization> tag is used to break “ties” when a ZIP Code is linked to multiple instances of the same address. The tag indicates which “neighborhood” the address is located in so that it can more accurately match the correct address.

<ZIP>

The ZIP Code (5-digit or ZIP + 4) associated with the address in the submitted record.

Example:

```
<Zip>92688</Zip>
```

(Data Type = String, Length = 10)

Remarks:

1. This is an optional tag if the city and state are provided in a record.
2. If you do not provide a city and state, the <Zip> tag is required in order to successfully verify the <Address> tag.
3. If the ZIP Code is incorrect or not provided, the AddressVerify process will add the correct 5-digit ZIP Code to this tag as long as the correct city and state are included in the record.
4. The <Phone> tag uses the <Zip> tag.

Chapter 4

DPV™ – Verifying Address Accuracy

The DPV feature 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).

DPV is a subscription service available for an additional fee, and is only available for U.S. addresses. For more information, please contact your sales representative. DPV related tags will not be populated beyond the extent of a regular subscription if you are not subscribed to the service.

DPV is accessible through the same URL that you are currently using. All that is required is to include the new output tags described in this section.

If you are not sure you have a subscription to DPV or would like to upgrade your current subscription to include DPV, please contact your customer service representative at 1-800-MELISSA (635-4772).

<CMRA>

This tag indicates whether or not an address is actually a private mailbox at a Commercial Mail Receiving Agency (CMRA) such as Mailboxes, Etc.

Example:

```
<CMRA>Y</CMRA>
(Data Type = String, Length 1)
```

Remarks:

This tag is a one-character value:

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

<DPVFOOTNOTES>

When an address is checked against the DPV data files, the applicable standard USPS footnote codes are returned here.

Example:

```
<DPVFootnotes>AABBRR</DPVFootnotes>
(Data Type = String, Length; Up To 15)
```

Remarks:

The following standard footnotes are available. More than one code may be returned.

Code	Description
AA	Input Address Matched to the ZIP + 4 file.
A1	Input Address Not Matched to the ZIP + 4 file.
BB	Input Address Matched to DPV (all components)
CC	Input Address Primary Number Matched to DPV but Secondary Number not Matched (present but invalid).
N1	Input Address Primary Number Matched to DPV but Highrise Address Missing Secondary Number.
M1	Input Address Primary Number Missing.
M3	Input Address Primary Number Invalid.
P1	Input Address Missing PO, RR, or HC Box number.
P3	Input Address PO, RR or HC number invalid.
RR	Input Address Matched to CMRA
R1	Input Address Matched to CMRA but Secondary Number 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.

<DPVStatusCode>

The DPV status code indicates the level of coding that was performed on the input address.

Example:

```
<DPVStatusCode>9</DPVStatusCode>”
(Data Type = String, Length 1)
```

Remarks:

The status codes and their related coding levels are as follows:

Code	Level
E	Expired database.
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.
7	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	The address was fully coded.

If an ‘S’ or ‘X’ code is returned, check the ErrorCode tag to find out why the address did not code.

DPVStatusCode refers to the level of coding achieved for the street number of a given address. To check the level of coding achieved for the a suite/apartment number of a given address, check the DPVSuiteStatusCode tag.

<DPVSuiteStatusCode>

The status code indicates the success or failure to validate the suite/apartment number of a submitted address.

Example:

```
<DPVSuiteStatusCode>V</DPVSuiteStatusCode>
(Data Type = String, Length 1)
```

Remarks:

The status codes and their related coding levels are as follows:

Code	Level
M	A suite number is required for the given street address but is missing from the submitted record.
R	A suite number is present on the submitted record but is either not required or is out of range for the given street address.
V	The suite field was verified.
X	The suite field was not coded.

License Agreement

1. NOTICE. MELISSA DATA CORPORATION IS WILLING TO LICENSE THE ENCLOSED SOFTWARE TO YOU, ONLY ON THE CONDITION THAT YOU ACCEPT ALL OF THE TERMS CONTAINED IN THIS LICENSE AGREEMENT. PLEASE READ THIS LICENSE AGREEMENT CAREFULLY BEFORE OPENING THE SEALED DISK PACKAGE. BY OPENING THIS PACKAGE (OR IN THE CASE OF DOWNLOADED SOFTWARE, YOU REQUEST UNLOCKING CODE FROM THE PUBLISHER) YOU AGREE TO BE BOUND BY THE TERMS OF THIS AGREEMENT. IF YOU DO NOT AGREE TO THESE TERMS WE ARE UNWILLING TO LICENSE THE SOFTWARE TO YOU, AND YOU SHOULD NOT OPEN THE DISK PACKAGE. IN SUCH CASE, PROMPTLY RETURN THE UNOPENED DISK PACKAGE AND ALL OTHER MATERIAL IN THIS PACKAGE ALONG WITH PROOF OF PAYMENT, TO THE AUTHORIZED DEALER FROM WHOM YOU OBTAINED IT FOR A FULL REFUND OF THE PRICE YOU PAID.

2. Ownership and License. This is a license agreement and NOT an agreement for sale. We continue to own the copy of the Software (including, but not limited to, object code, dynamic link libraries, and sample programs, together with the accompanying documentation contained in this package and all other copies that you are authorized by this Agreement to make collectively known as “Software”). Your rights to use the Software are specified in this Agreement, and we retain all rights not expressly granted to you in this Agreement. This Software is protected by U.S. copyright laws and international treaties. Nothing in this Agreement constitutes a waiver of our rights under U.S. Copyright law or any other federal or state law or international treaty.

3. Permitted Uses. You are granted the following rights to the Software:

(a) Right to Install and Use.

(1) Standalone Computer - Single Installation: You may install and use the Software on the hard disk drive of any single compatible computer that you own. However, you may not under any circumstances have the Software installed onto the hard drives of two or more computers at the same time, (nor may you install the Software onto the hard disk drive of one computer and then use the original CD-ROM on another computer). If you wish to use the Software on more than one computer, you must either erase the Software from the first hard drive before you install it onto a second hard drive, or else license an additional copy of the Software for each additional computer on which you want to use it.

(2) Network Use: If the single computer on which you install the Software is a network or Internet server, you may use the Software on any computer attached to the network, provided that it is only installed on the server. You may install and use this Software on a single file server regardless of the number of workstations attached to the network.

(b) Right to Copy. You may copy the Software for backup and archival purposes, provided that the original and each copy is kept in your possession, and that your installation and use of the Software does not exceed that allowed in part (a) above.

(1) Solely with the respect to the manual and Help files, you may make an unlimited number of copies (either in hard-copy or electronic form), provided that such copies shall be used only for internal purposes and are not republished or distributed beyond the licensee's premises.

(2) Copy, bundle, or redistribute the DEMO software with any commercial product (including books, CD-ROM, computer hardware, or software products). Your promotional and/or packaging materials must clearly disclose that the Software is copyrighted software of Melissa Data, that no charge is made by Melissa Data or you for it, and that it is not a fully supported commercial version.

(c) Right to Modify. You may modify the Software and/or merge it into another computer program to the extent necessary for your own use on (a single computer or server as specified above); however, any portion of the Software merged into another computer program will continue to be subject to the terms of this Agreement. You may use and modify the source code version of those Software portions that the documentation identifies as sample code ("SAMPLE CODE"), provided you do not distribute the SAMPLE CODE or any modified version of the SAMPLE CODE, in source form.

(d) Right to Transfer. You may not rent, lend, or lease this Software. However, you may transfer this license to use the Software to another party on a permanent basis by transferring this copy of the License Agreement, at least one unaltered copy of the Software, and all documentation. You must, at the same time, either transfer to the other party or destroy all your other copies of the Software or destroy all of your copies. Such transfer of possession terminates your license from us. Such other party shall be licensed under the terms of this Agreement upon its acceptance of this Agreement by its initial use of the Software. If you transfer the Software, you must remove the Software from your hard disk and you may not retain any copies of the Software for your own use.

4. Prohibited Uses. You may not, without written permission from us:

(a) Use, copy, modify, merge, or transfer copies of the Software or documentation except as provided in this Agreement;

(b) Use any backup or archival copies of the Software (or allow someone else to use such copies) for any purpose other than to replace the original copy in the event it is destroyed or becomes defective;

(c) Disassemble, decompile or reverse engineer, or in any manner decode the Software for any reason;

(d) Distribute, Sublicense, lease, or rent the Software, Data files and/or Dynamic Link Libraries of the Software.

(e) Expose the interfaces of the Software through your application (e.g. an OCX, DLL, class library, etc.).

5. Limited Warranty. We make the following limited warranties, for a period of 180 days from the date you acquired the Software from us.

(a) Media. The disks and documentation in this package will be free from defects in materials and workmanship under normal use. If the disks or documentation fail to conform to this warranty, you may, as your sole and exclusive remedy, obtain a replacement free of charge if you return the defective disk or documentation to us with a dated proof of purchase.

(b) Software. The Software in this package will materially conform to the documentation that accompanies it. If the Software fails to operate in accordance with this warranty, you may, as your sole and exclusive remedy, return all of the Software and

the documentation to the authorized dealer from whom you acquired it, along with a dated proof of purchase, specifying the problem, and we will provide you with a new version of the Software or a full refund at our election.

(c) WARRANTY DISCLAIMER. WE DO NOT WARRANT THAT THIS SOFTWARE WILL MEET YOUR REQUIREMENTS OR THAT ITS OPERATION WILL BE UNINTERRUPTED OR ERROR-FREE. WE EXCLUDE AND EXPRESSLY DISCLAIM ALL EXPRESS AND IMPLIED WARRANTIES NOT STATED HEREIN, INCLUDING THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

6. Termination. This license and your right to use this Software automatically terminate if you fail to comply with any provisions of this Agreement, destroy the copies of the Software in your possession, or voluntarily return the Software to us. Upon termination you will destroy all copies of the Software and documentation. Otherwise, the restrictions on your rights to use the Software will expire upon expiration of the copyright to the Software.

7. Miscellaneous Provisions. This Agreement will be governed by and construed in accordance with the substantive laws of California. This is the entire agreement between us relating to the contents of this package, and supersedes any prior purchase order, communications, advertising or representations concerning the contents of this package. No change or modification of this Agreement will be valid unless it is in writing, and is signed by us.