

Personator™ 



Personator

Web Service

Reference Guide

Melissa Data Corporation

Copyright

Companies, names, and data used in examples herein are fictitious unless otherwise noted. No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of Melissa Data Corporation. This document and the software it describes are furnished under a license agreement, and may be used or copied only in accordance with the terms of the license agreement.

Copyright © 2014 by Melissa Data Corporation. All rights reserved.

Information in this document is subject to change without notice. Melissa Data Corporation assumes no responsibility or liability for any errors, omissions, or inaccuracies that may appear in this document.

Trademarks

The Personator Web Service is a registered trademark of Melissa Data Corp. Windows is a registered trademark of Microsoft Corp.

The following trademarks are owned by the United States Postal Service®: DPV; LACSLink; PO Box; SuiteLink; ZIP; ZIP + 4; ZIP Code; United States Postal Service; USPS.

All other brands and products are trademarks of their respective holder(s).

Melissa Data Corporation

22382 Avenida Empresa
Rancho Santa Margarita, CA 92688-2112

Phone: 1-800-MELISSA (1-800-635-4772)

Fax: 949-589-5211

E-mail: info@MelissaData.com

Internet: www.MelissaData.com

For the most recent version of this document, visit

<http://www.melissadata.com/>

Document Code: DQTWSPERRG

Revision Number: 04112014.14

Dear Developer,

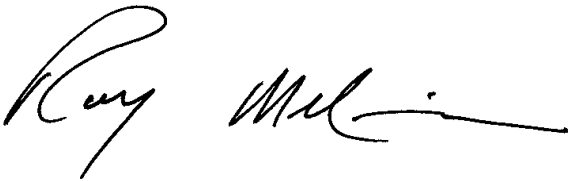
I would like to take this opportunity to thank you for your interest in Melissa Data products and introduce you to the company.

Melissa Data has been a leading provider of data quality and address management solutions since 1985. Our data quality software, Cloud services, and data integration components verify, standardize, consolidate, enhance and update U.S., Canadian, and global contact data, including addresses, phone numbers, and email addresses, for improved communications and ROI. More than 5,000 companies rely on Melissa Data to gain and maintain a single, accurate and trusted view of critical information assets.

This manual will guide you through the functions 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 me at: Ray@MelissaData.com.

I look forward to hearing from you.

Best Wishes,

A handwritten signature in black ink, appearing to read "Ray Melissa". The signature is fluid and cursive, with a long horizontal stroke at the end.

Raymond F. Melissa

President/CEO

Contents

Introduction	1
Concepts	3
Basic Procedures	3
Actions	3
Options	4
Columns	4
Results	4
Single Record vs. Batch	4
Using the Service	5
Service URLs	5
SOAP	6
Pseudocode Request	6
Response Fields	7
XML	7
Request	7
Response	8
REST	9
URL	9
Response	9
JSON	9
Request	9
Response	10
Character Replacements	11
Special Characters	11
Request Details	12

Actions	12
Check.....	12
Verify	12
Move	12
Append.....	12
Options.....	13
Check (Address)	13
Check (Phone)	14
Check (Email).....	15
Check (Name).....	15
Verify (Address, Phone, Email).....	17
Append (Address, Phone, Email).....	17
Inputs.....	18
Discrete Inputs	18
Free Form Input	20
Columns	23
Default Columns.....	23
Other Columns	24
Group Columns	24
Response Details	27
Outputs.....	27
Columns	27
Default Columns.....	27
No Group Columns.....	28
Address Details Group Columns	29
Census Group Columns.....	30
GeoCode Group Columns	31
Demographic Columns (No Group).....	32
Name Details Group Columns	35
Parsed Address Group Columns.....	36

Reference Guide

Parsed Email Group Columns	38
Parsed Phone Group Columns.....	38
Interpreting Result Codes	39
Demographics Results	41
Result Codes	45
AddressType Result Codes	45
Check (Address) Result Codes.....	45
Address Error Codes	46
Address Change Codes	47
Move (Address) Result Codes	47
GeoCoder Result Codes	48
Check (Phone) Result Codes.....	48
Check (Email) Result Codes	49
Check (Name) Result Codes	50
Verify Result Codes.....	50
Append Result Codes	51
Response Result Codes	51
Appendix.....	53
Example Requests/Responses	53
REST	53
XML.....	54

Introduction

Welcome to the Melissa Data Personator Web Service.

Personator™ Web Service is an all-in-one contact checking, verification, move update, and appending Web service. It allows you to pass in names, addresses, phone numbers, and email addresses; simultaneously parsing them, checking them for correctness, make conservative or aggressive corrections, get the latest address, and even appending data. It can also leverage all of these inputs to verify whole contact records. Since it is a Web service, Personator can be easily integrated into a wide variety of applications and you do not have to worry about finding and installing updates. Each Personator request can be configured to perform one or more of the primary actions the service provides: ***Check, Verify, Move*** and ***Append***.

The ***Check*** action allows you to pass in a name, address, phone number, and email address as one record. A record does not need to include all of those inputs, any combination of them, or even just one is sufficient to constitute a record and be checked. This action is available for US and Canada. Exception: ***Address*** field requires a *street address* and either a *city + state* or a *5 digit ZIP Code™* to be checked. ***Check*** looks at each of these subsets independently.

Whatever the input in the ***Email*** field is, it will not affect how Personator checks the ***address*** or ***phone number***. Within each field Personator parses input into its chief components. For example, email is parsed into the mailbox name, domain name, and top level domain name. Personator also makes conservative corrections, for instance correcting johndoe@gail.com to johndoe@gmail.com.

Personator is able to derive additional data from its knowledge base and correct the input data accordingly. Example: attaching the ZIP Code to the record that only has a street address, city, and state.

Personator checks the correctness of each subset of input. For example, it can determine whether the given street address exists within the given city and state or ZIP Code area.

The ***Check*** action allows you:

- to pass in a series of records and find any invalid addresses, phone numbers, or emails.
- to correct errors within the data.
- to append additional data to the records.
- or to parse out specific types of data from the input.

Within the ***Check*** action there is an optional feature called AdvancedAddressCorrection (AAC). This feature leverages the name input with the record to make more aggressive corrections and appends to an address. It can correct or add house numbers, cities, states, and ZIP Codes. AAC is available only for US addresses.

The *Verify* action compares different groups of data to the centric group defined by the user. It verifies the record as a whole, letting you know whether each group coincides with the centric piece of data in the Melissa Data Knowledge Base. You can define fields like address, phone number, or email as the centric data against which the other groups of data are compared. Auto-detection of the centric data is also available. The *Verify* action returns only results codes, telling you which sections of data passed verification against the centric data and which sections did not. With *Verify*, you can enter records, select the centric data as the field you are most confident in, and determine the accuracy of your input information. The *Verify* action is available only for US addresses.

The *Move* action allows you to update your US contact records with data returned by the Personator Web Service. The service allows for retrieving the most current address for a person or business. Thus if an old address is entered for a particular individual, Personator will return the latest address for that person, giving you the freshest and most up-to-date contact information.

The *Append* action allows you to enrich your US contact records with data returned by the Personator Web Service. The service will return elements based on the selected point of centrality which can either be the address, email, or phone. Through the *Append* action, you can fill in missing information in your contacts, correct them, and ensure that each of the data elements coincide, thus giving an accurate representation of each contact record.

Concepts

Basic Procedures

Using Personator starts with creating a *request*. This request must include your customer ID, which serves as a key for accessing the service, and any action(s) you want Personator to execute. Optionally, you can include which options you want to use and what columns(fields) you want returned.

The main points in preparing a request for Personator are:

- Customer ID
- Actions (*Check, Verify, Move, or Append*)
- Options
- Result Fields

You then need to cycle through all the records you want to add to it. For each record, you place all the different values into the appropriate fields and then add the record to the request structure. Once the request is finished, you send it to the service and get back the response. The response structure is very similar to the request; it contains a list of records equivalent to the one sent in the request. Each record in the response contains the output for one record from the request.

Actions

The Actions field is what determines what action the service will perform on the input data. This is a required field.

The *Check* action determines whether the data within a submitted record is valid, e.g. whether or not a given postal code contains the given city. It can also make limited corrections and appends to the data. *Check* looks at each data point separately, the inputs you put in for email don't affect what *Check* does with an address. *Check* returns results codes that describe which inputs were invalid, valid, or corrected. It also returns the input data after it has been corrected and added by the Web service. The *Check* action is available for US and Canadian addresses.

The *Verify* action allows you to select a centric data point and then determines whether the other data points are associated with it. For example, if you perform an address centric verify, it will tell you whether the name, phone, and email on that record coincide with that address in our database. *Verify* only returns the results column with results codes describing what it found. The *Verify* action is available only for US addresses.

The *Move* action allows you to get the latest move information for an individual or business. It requires that you have at the very least, a person's last name and an address or a business/company name and

an address. The returned address information will contain the updated address if a move was detected. Move also returns result codes which help you identify which addresses have a move. The *Move* action is available for US addresses.

The *Append* action will return elements based on the selected point of centrality which can either be the address, email or phone. For example, an address centric *Append* will return the name, company, phone and email associated with the given address. *Append* also returns result codes which help you identify which elements were appended. The *Append* action is available for US addresses.

Options

The Options field allows you to configure a number of options that change the way the service behaves. For instance, the UsePreferredCity option defaults to 'off.' This means that by default the service does not transform the input city name to the USPS preferred city name in the output. However, by adding 'UsePreferredCity:on' to the Options field, the service will change all city names into their preferred incarnations.

Columns

Personator allows the user to select what data the service will output. The Columns input field allows you to select either individual columns or groups which will then be returned in the output. These selected columns are returned in addition to the default columns which are always returned. Columns are only relevant when performing a *Check*, *Move*, or *Append* action, *Verify* only ever returns the results column.

Results

Every record in the response has a column called Results. This column contains a series of results codes, which are short codes that convey a great deal of information from the service. Generally, the codes tell you whether the inputs are valid, invalid, or have been changed by the service in some way. For instance, an AS01 code in the results indicates a valid, deliverable address in that record.

Single Record vs. Batch

Single record and batch requests are both made to the same endpoint. In fact, there is really no difference between single record and batch processing in Personator; single record requests are essentially batches of one. Personator can handle batches of up to 100 and it is generally recommended that you use 100 records per request as the service performs much faster the more records you use in each request.

Using the Service

A request to the Personator Web Service must consist of the Customer ID, action(s) to use, and at least one record.

The Web Service supports the following protocols all using HTTP/HTTPS:

- POST

An HTTP POST is issued with the "ContentType" header specifying the format of the request and the "Accept" header specifying the format of the response. Possible values are "application/xml" or "application/json".

- SOAP

Uses standard SOAP protocol to easily construct your request and parse the response.

SOAP requires the full request structure: all tags are required.

Please note: SOAP is slower and bloated. We strongly encourage you to use any of the other protocols instead of SOAP!

- REST

Uses HTTP GET to accept one input record and returns the response in XML format. If a JSON response is desired, append "&format=JSON" to the request string. This is useful for browser level or quick single queries.

Service URLs

The following URLs are where the Personator Web Service is hosted by Melissa Data:

Non-Secure:

SOAP <http://personator.melissadata.net/v3/SOAP/ContactVerify>

XML, REST, Etc. <http://personator.melissadata.net/v3/WEB/ContactVerify/doContactVerify>

Secure:

SOAP <https://personator.melissadata.net/v3/SOAP/ContactVerify>

XML, REST, Etc. <https://personator.melissadata.net/v3/WEB/ContactVerify/doContactVerify>

SOAP

Pseudocode Request

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

Step 1: Create the Request and Response Objects

```
Dim Request As New Personator.Request  
Dim Response As New Personator.Response
```

Step 2: Assign the General Request Values

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

```
Request.CustomerID = strCustID  
Request.TransmissionReference = strTranRef  
Request.Actions = strActions  
Request.Options = strOptions  
Request.Columns = strColumns
```

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

Step 3: Dimension the Record Array

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

```
ReDim Request.Records(99)
```

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

Step 4: Build the Record Array

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

```
Request.Records(intRecord) = New Personator.RequestRecord  
Request.Records(intRecord).AddressLine1 = "22382 Avenida Empresa"  
Request.Records(intRecord).PostalCode = "92688"  
Request.Records(intRecord).RecordID = 1
```

The lines above show only a few elements that can be sent to the Web service. See the next chapter for a description of all of the elements available to include with a Request record.

Repeat for each record being submitted with the current Request.

Step 5: Submit the Request Array

The final step is to create the Service Client Object and then submit the Request object doContactVerify method. This sends the data to the Web service and retrieves the Response object.

```
PersonatorClient = New Personator.Service
Response = PersonatorClient.doContactVerify(Request)
PersonatorClient.Dispose()
```

Response Fields

Get the output data from the response.

```
String outTotalRecords = Response.TotalRecords
String outTransReference = Response.TransmissionReference
String outTransResults = Response.TransmissionResults
String outVersion = Response.Version
String outAddressKey = Response.Records(intRecord).AddressKey
String outAddressLine1 = Response.Records(intRecord).AddressLine1
String outCity = Response.Records(intRecord).City
String outPostalCode = Response.Records(intRecord).PostalCode
String outState = Response.Records(intRecord).State
String outResults = Response.Records(intRecord).Results
String outRecordID = Response.Records(intRecord).RecordID
```

XML

Request

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

```
http://personator.melissadata.net/v3/WEB/ContactVerify/doContactVerify
```

Rather than an array of Record objects, an XML request can contain up to 100 <RequestRecord> elements under the <Records> element.

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

```

<Request>
<TransmissionReference>Sample</TransmissionReference>
<CustomerID>123456789</CustomerID>
<Actions>Check</Actions>
<Options/>
<Columns/>
<Records>
  <RequestRecord>
    <RecordID>1</RecordID>
    <CompanyName/>
    <FullName/>
    <AddressLine1>22382 Avenida Empresa</AddressLine1>
    <AddressLine2/>
    <Suite/>
    <City>Rancho Santa Margarita</City>
    <State>CA</State>
    <PostalCode>92688</PostalCode>
    <Country/>
    <PhoneNumber/>
    <EmailAddress/>
    <FreeForm/>
  </RequestRecord>
</Records>
</Request>

```

Response

```

<Response xmlns:i="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://schemas.datacontract.org/2004/07/WcfServiceMD.
mdContactVerify">
  <Records>
    <ResponseRecord>
      <AddressExtras></AddressExtras>
      <AddressKey>92688211282</AddressKey>
      <AddressLine1>22382 Avenida Empresa</AddressLine1>
      <AddressLine2></AddressLine2>
      <City>Rancho Santa Margarita</City>
      <CompanyName></CompanyName>
      <EmailAddress></EmailAddress>
      <NameFull></NameFull>
      <PhoneNumber></PhoneNumber>
      <PostalCode>92688-2112</PostalCode>
    </ResponseRecord>
  </Records>
</Response>

```

```

    <RecordExtras></RecordExtras>
    <RecordID>1</RecordID>
    <Reserved></Reserved>
    <Results>AS01</Results>
    <State>CA</State>
  </ResponseRecord>
</Records>
<TotalRecords>1</TotalRecords>
<TransmissionReference>Sample</TransmissionReference>
<TransmissionResults></TransmissionResults>
<Version>3.0.50</Version>
</Response>

```

REST

URL

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

```

https://personator.melissadata.net/v3/WEB/ContactVerify/
doContactVerify?t=Sample&id=123456789&act=Check&cols=&opt=&full=
&comp=&a1=22382%20avenida%20empresa&a2=&city=rancho%20santa%20
margarita&a2=&state=CA&postal=92688&ctry=&email=&phone=&ff=

```

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

Response

Same as the XML Response. See the XML “Response” on page 8.

JSON

Request

```

{
  "TransmissionReference": "Sample",
  "CustomerID": "123456789",
  "Actions": "Check",
  "Options": ,

```



```

"Columns":,
"Records":[{"
  "RecordID":"1",
  "CompanyName":,
  "FullName":,
  "AddressLine1":"22382 Avenida Empresa",
  "AddressLine2":,
  "Suite":,
  "City":"Rancho Santa Margarita",
  "State":"CA",
  "PostalCode":"92688",
  "Country":,
  "PhoneNumber":,
  "EmailAddress":,
  "FreeForm":,
}]
}

```

Response

```

{
  "Records":[{"
    "AddressExtras":,
    "AddressKey":"92688211282",
    "AddressLine1":"22382 Avenida Empresa",
    "AddressLine2":,
    "City":"Rancho Santa Margarita",
    "CompanyName":,
    "EmailAddress":,
    "NameFull":,
    "PhoneNumber":,
    "PostalCode":92688-2112,
    "RecordExtras":,
    "RecordID":"1",
    "Reserved":,
    "Results":"AS01",
    "State":"CA",
  }],
  "TotalRecords":"1",
  "TransmissionReference":"Sample",
  "TransmissionResults": "",
  "Version":"3.0.50"
}

```

}

Character Replacements

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

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

Character	URL Encoded	Character	URL Encoded	Character	URL Encoded
Space	%20 or +	/	%2F	[%5B
*	%2A	:	%3A]	%5D
#	%23	;	%3B	~	%7E
&	%26	<	%3C		
%	%25	=	%3D		
\$	%28	>	%3E		
+	%2B	?	%3F		
,	%2C	@	%40		

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

Special Characters

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

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

Request Details

A Request consists of the use of a protocol to make a call to the Web Service, detailing desired elements. The minimum required elements in a Request are your Customer ID, action(s) to use, and at least one record. If using SOAP, the full request structure (all tags) is required.

The Personator Web Service supports multiple protocols to access the Web Service, including REST, XML, JSON, and SOAP.

Each Request in the Personator Web Service has three main elements: Actions; Options; and Columns.

Actions

Actions are delimited with a “,” or “;”.

The Personator Web Service currently supports four possible actions:

Check

The *Check* action will validate the individual input data pieces for validity and correct them if possible. If the data is correctable, additional information will often be appended as well. US and Canada only.

Verify

The *Verify* action will return to you the relationships between your different input data pieces. It can show you if your name, address, email, and phone number are correlated (belonging to the same person) or not. US only.

Move

The *Move* action will return the latest address for an individual or business if a previous address was entered. Move requires either a Last Name and Address, or a Business/Company Name and Address as inputs. US only.

Move also returns results codes that help identify which addresses were updated with a move.

Append

The *Append* action will return elements based on the selected point of centrality which can either be the address, email or phone. For example, an address centric Append will return the name, company, phone and email associated with the given address. US only.

Append also returns result codes which help you identify which elements were appended.

Options

Options allow you to specify how the service should behave. They are passed in the format of <OptionName>:<Setting>. Multiple options are delimited with a “;”.

The default option setting when the option is not passed in the by user is listed first in italics.

Check (Address)

US and Canada only. *Check* (Address) will only be processed if the following is true:

- AddressLine1 is not empty.
- Both City and State are not empty or PostalCode is not empty.

Options

UsePreferredCity:(off, on)

Default value is off.

For every city in the United States, there is an official name that is preferred by the U.S. Postal Service. There may be one or more unofficial or “vanity” names in use. Normally, Personator allows you to verify addresses using known vanity names. If the usePreferredCity is set to on, Personator will substitute the preferred city name for all vanity names when it verifies an address.

Diacritics:(auto, on, off)

Default value is auto.

Determines whether or not French language characters are returned. If set to auto, those characters are only returned if they are in the input.

AdvancedAddressCorrection:(off, on)

Default value is off.

US only. Uses the name input to perform more advanced address corrections. This can correct or append house numbers, street names, cities, states, and ZIP codes.

LongAddressFormat:(off, on, auto)

Default value is off.

This function controls how Personator handles the abbreviations of suffixes and directionals when standardizing a street address. Setting this option to **on** will spell out any suffix and directional abbreviations (Ave to Avenue). Setting this option to **auto** will preserve the suffix and directional as

they were entered. In auto mode, if a spelled out “avenue” is entered, we will keep the suffix spelled out and vice versa.

AddressLine1 and AddressLine2 Request

AddressLine1 and AddressLine2 may each contain a full address.

- If both addresses are verified as valid, both are returned as inputted.
- If AddressLine1 has an invalid address and AddressLine2 has a valid address, the return order is switched. This returns the valid address previously in AddressLine2 into AddressLine1. Conversely, the invalid address in AddressLine1 will be returned in AddressLine2.

Alternatively, AddressLine2 may contain a suite number for AddressLine1. The suite information will be cleared from AddressLine2 and placed under its own field in the output.

For all of the possible return values for **Check** (Address) see “Check (Address) Results Codes” on page 33.

Geocode

If an address is verifiable, you can choose to geocode it. You will have the geocoding information appended (depending on your license). Geocoding is available for US and Canada.

To use Geocode, you must have the geocode columns on: GrpCensus or GrpGeocode.

Geodetic System

The GeoCoder Object uses WGS 84 standard, an Earth-centered, Earth-fixed terrestrial reference system and geodetic datum.

Geocode level

Your license will determine what geocode level you are allowed. If your license only has address checking enabled, you can geocode to 5 digits. If your license only has GeoCode enabled, you can geocode to 9 digits. If your license has GeoPoints enabled, you can geocode to 11 digits.

Geocode requires the results returned from the Web service.

Check (Phone)

US and Canada only. **Check** (Phone) will only be processed if the PhoneNumber input has a value.

Options

There are no options for **Check** (Phone).

Check (Email)

US and Canada only. *Check* (Email) will only be processed if the EmailAddress input has a value.

Options

CorrectSyntax:(on, off)

Default value is on.

If set to on, corrects the syntax of the Email field.

UpdateDomain:(on, off)

Default value is on.

If set to on, determines whether the domain name is out of date and updates it.

DatabaseLookup:(on, off)

Default value is on.

If set to on, verification of domain names will be attempted using a database of valid domains.

StandardizeCasing:(on, off)

Default value is on.

If set to on, changes all letters in the Email field to lower case before any checking occurs.

Check (Name)

US and Canada only. *Check* (FullName) will only be processed if the FullName or FirstName input has a value.

Options

If any name object option has multiple available codes and more than one are set, the first code is used.

CorrectFirstName:(on, off)

Default value is on.

If set to on, allows common spelling corrections for the FirstName field.

StandardizeCompany:(on, off)

Default value is on.

If set to on, the CompanyName field will be returned with standard abbreviation, capitalization, and punctuation rules applied.

NameHint:

(**Varying, DefinitelyFull, VeryLikelyFull, ProbablyFull, ProbablyInverse, VeryLikelyInverse, DefinitelyInverse, MixedFirstName, MixedLastName**)

Default value is Varying.

- **DefinitelyFull:** Name will always be treated as normal name order, regardless of formatting or punctuation.
- **VeryLikelyFull:** Name will be treated as normal name order unless inverse order is clearly indicated by formatting or punctuation.
- **ProbablyFull:** If necessary, statistical logic will be employed to determine name order, with a bias toward normal name order.
- **Varying:** If necessary, statistical logic will be employed to determine name order, with no bias toward either name order.
- **ProbablyInverse:** If necessary, statistical logic will be employed to determine name order, with a bias toward inverse name order.
- **VeryLikelyInverse:** Name will be treated as inverse name order unless normal order is clearly indicated by formatting or punctuation.
- **DefinitelyInverse:** Name will always be treated as inverse name order, regardless of formatting or punctuation.
- **MixedFirstName:** Name field is expected to only contain prefixes, first, and middle names.
- **MixedLastName:** Name field is expected to only contain last names and suffixes.

GenderPopulation:(Mixed, Male, Female)

Default value is Mixed. Sets the gender balance of the source data, either predominantly male, female, or mixed (evenly split).

GenderAggression:(Neutral, Conservative, Aggressive)

Default value is Neutral. Sets how aggressive genderization is for neutral first names.

MiddleNameLogic:(ParseLogic, HyphenatedLast, MiddleName)

Default value is ParseLogic. Determines the handling of middle names.

- **ParseLogic:** Middle names that are typically last names are considered to be part of a hyphenated last name.
- **HyphenatedLast:** The middle word is assumed to be part of the last name.
- **For Example:** "Matthew Edward Jones" is treated as "Matthew Edward-Jones."
- **MiddleName:** The middle word is assumed to be a middle name.
For example: "Matthew Svensson Jones." "Svensson" would be considered a middle name instead of part of the last name.

SalutationFormat:(Formal, Informal, FirstLast)

Default value is Formal. Sets the salutation format for the response:

- Formal: Dear Mr. Smith
- Informal: Dear John
- First/Last: Dear John Smith

Verify (Address, Phone, Email)

US only. The type of search performed is dependent upon your defined centric piece of information.

Options

CentricHint:(Auto, Address, Phone, Email)

Default value is Auto. When set to Auto, it first uses Address if available, followed by Phone if no Address is available, and lastly Email if neither Address nor Phone are available. Use this to tell the service which piece of information to use as the primary pivot when verifying information.

Append (Address, Phone, Email)

US only. The appended information is dependent upon your defined centric piece of information.

Options

CentricHint: (Auto, Address, Phone, Email)

Default value is Auto. When set to Auto, it first uses Address if available, followed by Phone if no Address is available, and lastly Email if neither Address nor Phone are available. Use this to tell the service which piece of information to use as the primary point of reference when appending data.

Append:(Blank, CheckError, Always)

Setting the *Append* option to Blank will cause the service to return information only when the input address, phone, email, name or company is blank.

Setting the *Append* option to CheckError will cause the service to return information when there are errors to either the address, phone, email, name or company. What an error entails are defined as follows:

Address Error: If the input address was not found in the database, was not at the least partially verified, or could not be corrected. (does not contain AS01, AS02, or AS03).

Phone Error: If the input phone number was not matched to either the 10 digit or 7 digit level at the least. (does not contain PS01 or PS02).

Email Error: If the input email address was not found in the database, or if the email is unconfirmed. (does not contain ES01 or ES03).

Name Error: If the input name did not parse successfully (does not contain NS01)

Company Error: If the input company was blank.

Setting the *Append* option to Always will cause the service to return information all the time, regardless of whether the input address, phone, email, name or company is blank or incorrect.

For a complete list of results codes, see “Verify Results Codes” on page 38.

For a complete list of status codes, see “Response Results Codes” on page 39.

For request examples, see “Example Requests/Responses” on page 41.

Inputs

Personator allows for inputting data using either discrete/individual columns, or with single string input using Free Form.

Discrete Inputs

Definition

Personator allows for inputting each of the following discrete domains:

Input	Definition
TransmissionReference	Serves as a unique identifier for this set of records. This allows you to match a response to a request.
CustomerID	Customer ID is your license string. This must be valid for you to access the Web service.
Actions	Specify which actions to do.
Columns	Specify which columns to be returned.
Options	Specify which options to use for the selected action.
FirstName	First name.
LastName	Last name.
FullName	Full name.
CompanyName	Company name.
AddressLine1	Street address. If including a suite, you can add it to the end of AddressLine1 field or enter it into the AddressLine2 field.

Input	Definition
AddressLine2	Street address. Can be a continuation of AddressLine1 (ex: suite) or another address.
City	City.
State	State. Accepts either two-character abbreviation or full state name.
PostalCode	Postal Code or ZIP Code.
Country	Country.
LastLine	City + State + ZIP.
Email	Email address.
Phone	Telephone number.
FreeForm	Single Line Input containing Address, Name, Phone, Email, and Company. Can be fielded or unfielded.

Code

Input	Code
TransmissionReference	XML/SOAP: TransmissionReference REST: &t
CustomerID	XML/SOAP: CustomerID REST: &id
Actions	XML/SOAP: Actions REST: &act
Columns	XML/SOAP: Columns REST: &cols
Options	XML/SOAP: Options REST: &opt
FirstName	XML/SOAP: FirstName REST: &first
LastName	XML/SOAP: LastName REST: &last
FullName	XML/SOAP: FullName REST: &full
CompanyName	XML/SOAP: CompanyName REST: &comp
AddressLine1	XML/SOAP: AddressLine1 REST: &a1
AddressLine2	XML/SOAP: AddressLine2 REST: &a2

Input	Code
City	XML/SOAP: City REST: &city
State	XML/SOAP: State REST: &state
PostalCode	XML/SOAP: PostalCode REST: &postal
Country	XML/SOAP: Country REST: &ctry
LastLine	XML/SOAP: LastLine REST: &lastline
Email	XML/SOAP: Email REST: &email
Phone	XML/SOAP: Phone REST: &phone
FreeForm	XML/SOAP: Freeform REST: &ff
Format	REST: &format=JSON

Format only needs to be set if you want a JSON response.

Free Form Input

FreeForm's powerful entity recognition and identification algorithms allow extraction of contact information from fielded or unfielded textual data in a single string. This functionality will identify, parse and reorganize input data into usable data types, assuring that even the most inconsistent data entry will be properly Checked, Verified, Appended and Moved through the Personator Web Service. It can parse a single line of input text containing many datatypes - whether or not it contains a delimiter.

FreeForm is able to identify and parse the following Contact Domains:

- Address
- Suite
- City
- State
- PostalCode
- Email Address
- Phone Number
- Company Name
- Full Name
- Country

Implementation:

It is important to remember that FreeForm should be used by itself and exclusive of other RequestRecord Inputs. The FreeForm Input will be disregarded if any of the other <RequestRecord> Fields (eg. AddressLine1, PhoneNumber, EmailAddress, etc.) are populated.

Here is an example of a proper request for utilizing FreeForm in XML:

```
<Request>
  <Actions>Check</Actions>
  <Columns/>
  <CustomerID>Customer ID Here</CustomerID>
  <Options/>
  <Records>
    <RequestRecord>
      <RecordID>1</RecordID>
      <FreeForm>22382 Avenida Empresa, Rancho Santa Margarita,
        CA, 92688</FreeForm>
    </RequestRecord>
  </Records>
</Request>
```

Here is an example of an incorrect request using FreeForm in XML:

```
<Request>
  <Actions>String</Actions>
  <Columns/>
  <CustomerID>String</CustomerID>
  <Options>String</Options>
  <Records>
    <RequestRecord>
      <RecordID>1</RecordID>
      <AddressLine1>22382 Avenida Empresa</AddressLine1>
      <City>Rancho Santa Margarita</City>
      <State>CA</State>
      <PostalCode>92688</PostalCode>
      <PhoneNumber>8008006245</PhoneNumber>
      <FreeForm>22382 Avenida Empresa, Rancho Santa Margarita,
        CA, 92688</FreeForm>
    </RequestRecord>
  </Records>
</Request>
```

Columns

The Personator Web Service does not have a static output structure. The output is dynamic and will only return a set of default fields, plus any additional fields that are selected. To select additional fields, we specify either the column name(s) or the group name(s) on the <Columns> Input, delimited by a comma (“,”).

*Note: Doing a Verify Action by itself will only return the <Results> Column

Default Columns

The following default columns are always returned as part of the output.

Column Name
AddressExtras (Default)
AddressKey (Default)
AddressLine1 (Default)
AddressLine2 (Default)
City (Default)
CompanyName(Default)
EmailAddress (Default)
NameFull (Default)
PhoneNumber (Default)
PostalCode (Default)
State (Default)

Other Columns

The following columns will be included as part of the output if they are requested in the <Columns> Input.

Column Name
Plus4
PrivateMailBox
Suite

For Example:

```
<Columns>Plus4,PrivateMailBox,Suite</Columns>
```

Group Columns

The following columns will be included as part of the output if either the GroupName or the Individual Column Names are requested in the <Columns> Input.

Column	Group
AddressTypeCode	GrpAddressDetails
CarrierRoute	GrpAddressDetails
CityAbbreviation	GrpAddressDetails
CountryCode	GrpAddressDetails
CountryName	GrpAddressDetails
DeliveryIndicator	GrpAddressDetails
DeliveryPointCheckDigit	GrpAddressDetails
DeliveryPointCode	GrpAddressDetails
StateName	GrpAddressDetails
UrbanizationName	GrpAddressDetails
UTC	GrpAddressDetails
CBSACode	GrpCensus
CBSADivisionCode	GrpCensus
CBSADivisionLevel	GrpCensus
CBSADivisionTitle	GrpCensus
CBSALevel	GrpCensus
CBSATitle	GrpCensus
CensusBlock	GrpCensus

Column	Group
CensusTract	GrpCensus
CongressionalDistrict	GrpCensus
CountyFIPS	GrpCensus
CountyName	GrpCensus
PlaceCode	GrpCensus
PlaceName	GrpCensus
Latitude	GrpGeocode
Longitude	GrpGeocode
DateOfBirth	
HouseholdIncome	
LengthOfResidence	
OwnRent	
Occupation	
PresenceOfChildren	
MaritalStatus	
DateOfDeath	
DemographicsGender	
Gender	GrpNameDetails
Gender2	GrpNameDetails
NameFirst	GrpNameDetails
NameFirst2	GrpNameDetails
NameLast	GrpNameDetails
NameLast2	GrpNameDetails
NameMiddle	GrpNameDetails
NameMiddle2	GrpNameDetails
NamePrefix	GrpNameDetails
NamePrefix2	GrpNameDetails
NameSuffix	GrpNameDetails
NameSuffix2	GrpNameDetails
Salutation	GrpNameDetails
AddressDeliveryInstallation	GrpParsedAddress
AddressHouseNumber	GrpParsedAddress
AddressLockBox	GrpParsedAddress

Column	Group
AddressPostDirection	GrpParsedAddress
AddressPreDirection	GrpParsedAddress
AddressPrivateMailboxName	GrpParsedAddress
AddressPrivateMailboxRange	GrpParsedAddress
AddressRouteService	GrpParsedAddress
AddressStreetName	GrpParsedAddress
AddressStreetSuffix	GrpParsedAddress
AddressSuiteName	GrpParsedAddress
AddressSuiteNumber	GrpParsedAddress
DomainName	GrpParsedEmail
MailboxName	GrpParsedEmail
TopLevelDomain	GrpParsedEmail
AreaCode	GrpParsedPhone
NewAreaCode	GrpParsedPhone
PhoneExtension	GrpParsedPhone
PhonePrefix	GrpParsedPhone
PhoneSuffix	GrpParsedPhone

For Example:

This will return all the columns associated with GrpAddressDetails

```
<Columns>GrpAddressDetails</Columns>
```

This will return only the specified columns

```
<Columns>CBSACode,CarrierRoute</Columns>
```

Response Details

A response is the result of a request. This consists of returned results codes and parsed, corrected, and/or appended request elements, depending on the options selected.

Depending on the protocol used to make the request, the response will be in a certain protocol. Personator Web Service supports three possible response protocols: SOAP; XML; and JSON.

Outputs

Columns

Columns are delimited with a “,”.

The Personator Web Service returns specific columns for input data based on your needs. At a minimum, default columns are always returned. Default columns for specific actions are designated in the “Default Action” column. Beyond the default columns, you can request the presence of additional columns individually by specifying their column name, or the group that contains that column.

Default Columns

AddressExtras

Any extra information that does not fit in the AddressLine fields.

AddressKey

Returns a unique identifier for an address. This key can be used with other current and future Melissa Data services.

AddressLine1

Returns the address entered in the AddressLine field. If two addresses were entered and only one is valid, the valid address is returned instead. This includes the suite and private mailbox.

AddressLine2

If two addresses are passed into the AddressLine field, the second address is returned here. If only one of two addresses is valid, the valid address will be returned in AddressLine1.

City

Returns the city entered in the City field.

CompanyName

Returns the company name.

EmailAddress

Returns the email address entered in the Email field.

NameFull

Returns the full name for the record.

PhoneNumber

Returns the standardized phone number for the record.

PostalCode

Returns the 9-digit postal code for U.S. addresses and 6-digit postal code for Canadian addresses.

State

Returns the state for the record.

No Group Columns

These columns are not default and have no group. They normally will not be needed unless for specific legacy requirements.

Plus4

Returns the 4-digit plus4 for the input address. If this column is requested, the PostalCode field will only contain the 5-digit ZIP for U.S. addresses.

PrivateMailBox

Returns the private mail box number for the address in the AddressLine field, if any. Private mailboxes are private mail boxes in commercial mail receiving agencies, like a UPS Store. If requested, the Private mailbox will be populated in this field instead of the Address field.

Suite

Returns the suite for the address in the AddressLine field, if any. If requested, the suite will be populated in this field instead of the Address field.

Address Details Group Columns

These columns are not default and belong to the GrpAddressDetails group.

AddressTypeCode

Returns a code for the address type in the AddressLine field. Please see the appendix for the list of possible codes.

CarrierRoute

Returns a 4-character code defining the carrier route for this record.

CityAbbreviation

Returns an abbreviation for the city entered in the City field, if any.

CountryCode

Returns the country code for the country in the Country field.

CountryName

Returns the country name for the record.

DeliveryIndicator

Returns an indicator of whether an address is a business address or residential address.

Code	Definition
B	Business
R	Residential
U	Unknown

DeliveryPointCheckDigit

Returns a string value containing the 1-digit delivery point check digit.

DeliveryPointCode

Returns a string value containing the 2-digit delivery point code.

StateName

Returns the full name of the state entered in the State field.

UrbanizationName

Returns the urbanization name for the address entered in the AddressLine field. Usually only used if the address is in Puerto Rico.

UTC

Returns the time zone of the requested record.

All Melissa Data products express time zones in UTC (Coordinated Universal Time).

Census Group Columns

These columns are not default and belong to the GrpCensus group.

CBSACode

Census Bureau's Core Based Statistical Area (CBSA).

Returns the 5-digit code for the CBSA associated with the requested record.

CBSADivisionCode

Returns the code for a division associated with the requested record, if any.

CBSADivisionLevel

Returns whether the CBSA division, if any, is metropolitan or micropolitan.

CBSADivisionTitle

Returns the title for the CBSA division, if any.

CBSALevel

Returns whether the CBSA is metropolitan or micropolitan.

CBSATitle

Returns the title for the CBSA.

CensusBlock

Returns a 4-digit string containing the census block number associated with the requested record.

Census blocks are the smallest geographic area for which the Bureau of the Census collects and tabulates decennial census data.

CensusTract

Returns a 4-to 6-digit string containing the census tract number associated with the requested record.

Census tracts are small subdivisions of a county.

CongressionalDistrict

Returns the 2-digit congressional district that the requested record belongs to.

CountyFIPS

Returns the FIPS code for the county in the County field.

FIPS code is a 5-digit code. The first two digits are a state code and the last three indicate the county within the state.

CountyName

Returns the county name.

PlaceCode,

PlaceName

When ZIP codes overlap, the City field will always return the city that covers most of the ZIP area. If the address is located outside of that city but within the ZIP Code, PlaceCode/PlaceName will refer to that area

GeoCode Group Columns

These columns are not default and belong to the GrpGeocode group.

Latitude

Returns the geocoded latitude for the address entered in the AddressLine field.

Longitude

Returns the geocoded longitude for the address entered in the AddressLine field.

Demographic Columns (No Group)

If any demographics are enabled, <DemographicsResults> will return a comma delimited string containing all the results of the demographics combined.

DateOfBirth

Returns the date of birth in the format YYYYMM. Accuracy is only to the month.

HouseholdIncome

Returns the range of the household's income.

Value
Unknown
\$0-\$15,000
\$15,001-\$20,000
\$20,001-\$30,000
\$30,001-\$40,000
\$40,001-\$50,000
\$50,001-\$60,000
\$60,001-\$70,000
\$70,001-\$80,000
\$80,001-\$90,000
\$90,001-\$100,000
\$100,001-\$125,000
\$125,001-\$150,000
\$150,001+

LengthOfResidence

Returns the range of the individual's length of residency in their current address.

Value

Less than 1 year

1-2 years

2-3 years

3-4 years

4-5 years

5-6 years

6-7 years

7-8 years

8-9 years

9-10 years

10-11 years

11-12 years

12-13 years

13-14 years

14-15 years

15+ years

Unknown

PresenceOfChildren

Returns the presence of children in the household.

Value

Unknown

No Children Present

Children Present

MaritalStatus

Returns the individual's marital status.

Value

Unknown

Definitely Single

Possibly Single

Possibly Married

Value

Definitely Married

DateOfDeath

Returns the full date of death in the format YYYYMMDD.

DemographicsGender

Returns gender based on demographics data.

Value

Unknown

Male

Female

Neutral

OwnRent

Returns the individual's status as owner or renter of the property.

Value

Unknown

Definite Renter

Probable Renter

Probable Owner

Definite Owner

Occupation

Returns the category for the individual's occupation.

Value

Physician/Dentist

Healthcare

Lawyer/Judge

Professional/Technical

Management

Teacher/Educator

Value

Sales/Marketing

Clerical/Service Worker

Tradesmen/Laborer

Farmer

Student

Homemaker

Retired

Federal Employee

Unknown

Military

Military Retired

Other

Business Owner

Religious

Self Employed

Financial

Name Details Group Columns

These columns are not default and belong to the GrpNameDetails group.

Gender

Returns a gender for the name in the FullName field.

Gender2

Only used if 2 names are in the FullName field. Returns a gender for the second name in the FullName field.

NameFirst

Returns the first name in the FullName field.

NameFirst2

Only used if 2 names are in the FullName field. Returns the second name in the FullName field.

NameLast

Returns the last name in the FullName field.

NameLast2

Only used if 2 names are in the FullName field. Returns a last name for the second name in the FullName field.

NameMiddle

Returns a middle name for the name in the FullName field.

NameMiddle2

Only used if 2 names are in the FullName field. Returns a middle name for the second name in the FullName field.

NamePrefix

Returns a prefix for the name in the FullName field.

NamePrefix2

Only used if 2 names are in the FullName field. Returns a prefix for the second name in the FullName field.

NameSuffix

Returns a suffix for the name in the FullName field.

NameSuffix2

Only used if 2 names are in the FullName field. Returns a suffix for the second name in the FullName field.

Salutation

Returns a salutation for the name in the FullName field.

Parsed Address Group Columns

These columns are not default and belong to the GrpParsedAddress group.

AddressDeliveryInstallation (Canada Only)

Returns the parsed delivery installation for the address entered in the AddressLine field.

AddressHouseNumber

Returns the parsed house number for the address entered in the AddressLine field.

AddressLockBox (Canada Only)

Returns the parsed lock box number for the address entered in the AddressLine field.

AddressPostDirection

Returns the parsed post-direction for the address entered in the AddressLine field.

AddressPreDirection

Returns the parsed pre-direction for the address entered in the AddressLine field.

AddressPrivateMailboxName

Returns the parsed private mailbox name for the address entered in the AddressLine field.

AddressPrivateMailboxRange

Returns the parsed private mailbox range for the address entered in the AddressLine field.

AddressRouteService (Canada Only)

Returns the parsed route service number for the address entered in the AddressLine field.

AddressStreetName

Returns the parsed street name for the address entered in the AddressLine field.

AddressStreetSuffix

Returns the parsed street suffix for the address entered in the AddressLine field.

AddressSuiteName

Returns the parsed suite name for the address entered in the AddressLine field.

AddressSuiteNumber

Returns the parsed suite number for the address entered in the AddressLine field.

Parsed Email Group Columns

These columns are not default and belong to the GrpParsedEmail group.

DomainName

Returns the parsed domain name for the email entered in the Email field.

MailboxName

Returns the parsed mailbox name for the email entered in the Email field.

TopLevelDomain

Returns the parsed top-level domain name for the email entered in the Email field.

Parsed Phone Group Columns

These columns are not default and belong to the GrpParsedPhone group.

AreaCode

Returns the parsed area code for the phone number entered in the Phone field.

NewAreaCode

Returns the parsed new area code for the phone number entered in the Phone field.

PhoneExtension

Returns the parsed extension for the phone number entered in the Phone field.

PhonePrefix

Returns the parsed prefix for the phone number entered in the Phone field.

PhoneSuffix

Returns the parsed suffix for the phone number entered in the Phone field.

Interpreting Result Codes

Personator uses a wide variety of result codes. So much so that interpreting them can seem intimidating at first. However, a glance at the result codes can quickly give you some very good information if you know what to look for.

The result codes for *Check* all basically follow this pattern: the first character tells you which data point it pertains to (A=address, N=name, p=Phone, etc.). The second character tells you what kind of code it is. An 'S' generally indicates that the input is valid for that data point. A 'C' indicates that something in that data point was changed/appended. An 'E' tells you that some part of that data point was determined to be invalid. The next two characters can be to look up the code and determine what exactly it is telling you, but generally you can look at a result record and tell from the number of XS codes vs the number of XE codes how good the data for that record is.

Verify codes are a little bit different. Each VR code represents two data points in a record that intersect in our database. VS codes can indicate that a datapoint cannot be found in our database and therefore cannot be verified, or that there was an incomplete match between two datapoints (such as an address only being matched to a last name instead of the full name.) If no codes are returned it means that none of the data input matched up with the centric element in our database.

Our advice is to use Result codes as a filter between good and bad records. Determine which results would qualify a good record, then let all the records that don't satisfy the criteria qualify as bad records. See the following simple examples:

Address:

Good: AS01, or AS02, or AS03

Bad: Everything else.

Description: All records with a valid or correctable address are good.

Address and Phone:

Good: (AS01, or AS02, or AS03) and (PS01 or PS02)

Bad: Everything else.

Description: All records with a good address and a good phone are good.

Address and Name Verify:

Good: (AS01, or AS02, or AS03) and VR01

Semi-Good: AS01, or AS02, or AS03

Bad: Everything else.

Address and Name Verify:

Description: Valid addresses that match (at least partially) the name are good. Otherwise, good addresses are still 'semi-good'. Everything else is bad.

As you can see, Result codes can be cascaded as well to define multiple categories. It is important to read and understand all the result codes to figure out which ones apply to your business needs.

Demographics Results

If any demographics fields are enabled through the <Columns> field, the <DemographicsResults> field will also be returned with a comma delimited string containing all the results for the enabled demographics fields combined.

HouseholdIncome

Returns the range of the household's income.

Code	Definition
ID00	Unknown
ID01	\$0-\$15,000
ID02	\$15,001-\$20,000
ID03	\$20,001-\$30,000
ID04	\$30,001-\$40,000
ID05	\$40,001-\$50,000
ID06	\$50,001-\$60,000
ID07	\$60,001-\$70,000
ID08	\$70,001-\$80,000
ID09	\$80,001-\$90,000
ID10	\$90,001-\$100,000
ID11	\$100,001-\$125,000
ID12	\$125,001-\$150,000
ID13	\$150,001+

LengthOfResidence

Returns the range of the individual's length of residency in their current address.

Code	Definition
LD00	Less than 1 year
LD01	1-2 years
LD02	2-3 years
LD03	3-4 years
LD04	4-5 years
LD05	5-6 years

Code	Definition
LD06	6-7 years
LD07	7-8 years
LD08	8-9 years
LD09	9-10 years
LD10	10-11 years
LD11	11-12 years
LD12	12-13 years
LD13	13-14 years
LD14	14-15 years
LD15	15+ years
LD99	Unknown

PresenceOfChildren

Returns the presence of children in the household.

Code	Definition
CD00	Unknown
CD01	No Children Present
CD02	Children Present

MaritalStatus

Returns the individual's marital status.

Code	Definition
MD00	Unknown
MD01	Definitely Single
MD02	Possibly Single
MD03	Possibly Married
MD04	Definitely Married

DateOfDeath

Returns the full date of death in the format YYYYMMDD.

Code	Definition
DD00	Not Deceased
DD01	Deceased

DemographicsGender

Returns gender based on demographics data.

Code	Definition
GD00	Unknown
GD01	Male
GD02	Female
GD03	Neutral

OwnRent

Returns the individual's status as owner or renter of the property.

Code	Definition
RD00	Unknown
RD01	Definite Renter
RD02	Probable Renter
RD03	Probable Owner
RD04	Definite Owner

Occupation

Returns the category for the individual's occupation.

Code	Definition
WD01	Physician/Dentist
WD02	Healthcare
WD03	Lawyer/Judge
WD04	Professional/Technical
WD05	Management
WD06	Teacher/Educator
WD07	Sales/Marketing
WD08	Clerical/Service Worker
WD09	Tradesmen/Laborer
WD10	Farmer
WD11	Student
WD12	Homemaker
WD13	Retired
WD14	Federal Employee
WD00	Unknown
WD15	Military
WD16	Military Retired
WD99	Other
WD17	Business Owner
WD18	Religious
WD19	Self Employed
WD20	Financial

Result Codes

All *Check*, *Verify*, *Move* and *Append* Result Codes are returned in the <Results> field.

Response Result Codes are returned in the <RequestResults> field.

AddressType Result Codes

Code	Description
G	General Delivery
M	Military Address
P	PO Box™ Address
R	Rural Route Address
S	Standard Address
U	Unique/LVR

Check (Address) Result Codes

Address Status Codes

Code	Short Description	Long Description
AS01	Address Fully Verified	The address is valid and deliverable according to official postal agencies.
AS02	Street Only Match	The street address was verified but the suite number is missing or invalid.
AS03	Non USPS Address Match	US Only. This US address is not serviced by the USPS but does exist and may receive mail through third party carriers like UPS.
AS09	Foreign Address	The address is in a non-supported country.
AS10	CMRA Address	US Only. The address is a Commercial Mail Receiving Agency (CMRA) like a Mailboxes Ect. These addresses include a Private Mail Box (PMB or #) number.
AS13	Address Updated By LACS	US Only. The address has been converted by LACSLink® from a rural-style address to a city-style address.
AS14	Suite Appended	US Only. A suite was appended by SuiteLink™ using the address and company name.
AS15	Apartment Appended	An apartment number was appended by AddressPlus using the address and last name.
AS16	Vacant Address	US Only. The address has been unoccupied for more than 90 days.
AS17	No Mail Delivery	US Only. The address does not currently receive mail but will likely in the near future.

Code	Short Description	Long Description
AS20	Deliverable only by USPS	US Only. This address can only receive mail delivered through the USPS (ie. PO Box or a military address).
AS23	Extraneous Information	Extraneous information not used in verifying the address was found. This has been placed in the ParsedGarbage field.

Address Error Codes

Code	Short Description	Long Description
AE01	Postal Code Error	The Postal Code does not exist and could not be determined by the city/ municipality and state/province.
AE02	Unknown Street	Could not match the input street to a unique street name. Either no matches or too many matches found.
AE03	Component Mismatch Error	The combination of directionals (N, E, SW, etc) and the suffix (AVE, ST, BLVD) is not correct and produced multiple possible matches.
AE04	Non-Deliverable Address	US Only. A physical plot exists but is not a deliverable addresses. One example might be a railroad track or river running alongside this street, as they would prevent construction of homes in that location.
AE05	Multiple Match	The address was matched to multiple records. There is not enough information available in the address to break the tie between multiple records.
AE06	Early Warning System	US Only. This address currently cannot be verified but was identified by the Early Warning System (EWS) as containing new streets that might be confused with other existing streets.
AE07	Missing Minimum Address	Minimum requirements for the address to be verified is not met. Address must have at least one address line and also the postal code or the locality/ administrative area.
AE08	Sub Premise Number Invalid	The thoroughfare (street address) was found but the sub premise (suite) was not valid.
AE09	Sub Premise Number Missing	The thoroughfare (street address) was found but the sub premise (suite) was missing.
AE10	Premise Number Invalid	The premise (house or building) number for the address is not valid.
AE11	Premise Number Missing	The premise (house or building) number for the address is missing.
AE12	Box Number Invalid	The PO (Post Office Box), RR (Rural Route), or HC (Highway Contract) Box number is invalid.
AE13	Box Number Missing	The PO (Post Office Box), RR (Rural Route), or HC (Highway Contract) Box number is missing.
AE14	PMB Number Missing	US Only. The address is a Commercial Mail Receiving Agency (CMRA) and the Private Mail Box (PMB or #) number is missing.

Code	Short Description	Long Description
AE17	Sub Premise Not Required	A sub premise (suite) number was entered but the address does not have secondaries.

Address Change Codes

Code	Short Description	Long Description
AC01	Postal Code Change	The postal code was changed or added.
AC02	Administrative Area Change	The administrative area (state, province) was added or changed.
AC03	Locality Change	The locality (city, municipality) name was added or changed.
AC04	Alternate to Base Change	US Only. The address was found to be an alternate record and changed to the base (preferred) version.
AC05	Alias Name Change	US Only. An alias is a common abbreviation for a long street name, such as "MLK Blvd" for "Martin Luther King Blvd." This change code indicates that the full street name (preferred) has been substituted for the alias.
AC06	Address1/Address2 Swap	Address1 was swapped with Address2 because Address1 could not be verified and Address2 could be verified.
AC07	Address1 & Company Swapped	Address1 was swapped with Company because only Company had a valid address.
AC08	Plus4 Change	US Only. A non-empty plus4 was changed.
AC09	Dependent Locality Change	US Only. The dependent locality (urbanization) was changed.
AC10	Thoroughfare Name Change	The thoroughfare (street) name was changed due to a spelling correction.
AC11	Thoroughfare Suffix Change	The thoroughfare (street) suffix was added or changed, such as from "St" to "Rd."
AC12	Thouroughfare Directional Change	The thoroughfare (street) pre-directional or post-directional was added or changed, such as from "N" to "NW."
AC13	Sub Premise Type Change	The sub premise (suite) type was added or changed, such as from "STE" to "APT."
AC14	Sub Premise Number Change	The sub premise (suite) unit number was added or changed.
AC20	House Number Change	US Only. The house number was changed.

Move (Address) Result Codes

Code	Short Description	Long Description
AS12	Record Move	The record moved to a new address.

GeoCoder Result Codes

Code	Short Description	Long Description
GS01	Geocoded to Street Level	The record was geocoded to the street (thoroughfare) level (Zip+4 for US, full postal code for CA).
GS02	Geocoded to Neighborhood Level	The record was geocoded to the neighborhood level (Zip+2 for US).
GS03	Geocoded to City Level	The record was geocoded to the city (locality) level (ZIP centroid for US, 3-digit postal code for CA).
GS04	Geocoded to State Level	The record was geocoded to the state (administrative area) level.
GS05	Geocoded to Rooftop Level	The record was geocoded to the rooftop level.
GS06	Geocoded to Interpolated Rooftop Level	The record was geocoded to the rooftop level using interpolation (educated estimations using street coordinates).
GS10	Wire Center Lat/Long	The latitude and longitude are based off of the wire center of the phone number.
GE01	Invalid Postal Code	The submitted postal code is not in a valid format.
GE02	Postal Code Not Found	The submitted postal code was not found in the database.

Check (Phone) Result Codes

Code	Short Description	Long Description
PS01	10-Digit Match	The first 10-digits of the phone number have been verified as valid.
PS02	7-Digit Match	The first 7-digits of the phone number has been verified as valid.
PS03	Corrected Area Code	NewAreaCode contains corrected area code that was changed according to the postal code it falls into.
PS06	Updated Area Code	The area code was changed due to an area code split. The updated code is located within NewAreaCode.
PS07	Cellular Line	The exchange type of the phone number indicates the number is a cellular number.
PS08	Land Line	The exchange type of the phone number indicates the number is a land line number.
PS09	VOIP Line	The exchange type of the phone number indicates the number is a VOIP number.
PS10	Residential Number	The phone number belongs to a residence.
PS11	Business Number	The phone number belongs to a business.
PS12	SOHO Number	The phone number belongs to a small office or home office.
PE01	Bad Area Code	The area code does not exist in our database or contains non-numbers.
PE02	Blank Phone Number	The phone number is blank.

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

Check (Email) Result Codes

Code	Short Description	Long Description
ES01	Valid Email Domain	The email domain name was confirmed as valid by either the DatabaseLookup or MXLookup.
ES02	Invalid Email Domain	The email domain name was either not located by MXLookup or was located on the list of invalid domains.
ES03	Unverified Email Domain	The domain name was not confirmed as valid by either DatabaseLookup, but was not found on the list of invalid domain names.
ES04	Mobile Email Address	The domain name was identified as a mobile email address and classified as not deliverable by the FCC.
ES05	Disposable Domain	The domain name of the submitted email was identified as a disposable domain. For example: DODGEIT.COM
ES06	Spamtrap Domain	The domain name of the submitted email was identified as a spamtrap domain. Mailing to one of these domains could result in the sender being blacklisted. For example: LAUNCH.COM.
ES10	Syntax Changed	The syntax of the submitted email address was changed.
ES11	Top Level Domain Changed	The top level domain of the submitted email address was changed.
ES12	Domain Changed (Spelling)	The domain of the submitted email address was corrected for spelling.
ES13	Domain Changed (Update)	The domain of the submitted email address was updated due to a domain name change.
EE01	Syntax Error	There is a syntax error in the submitted email address.
EE02	Top Level Domain Not Found	The top level domain of the submitted email address was not found.
EE03	Mail Server Not Found	The mail server (domain) of the submitted email address was not found.
EE04	Invalid Mailbox Name	An invalid mailbox name was detected (IE: noreply). To configure invalid mailbox names, review mdEmailConfig.ini.

Check (Name) Result Codes

Code	Short Description	Long Description
NS01	Parsing Successful	Name parsing was successful.
NS02	Error Parsing	An error was detected. Please check for a name error code.
NS03	First Name Spelling Corrected	The spelling in the first name field was corrected.
NS04	First Name 2 Spelling Corrected	The spelling in the second first name field was corrected.
NS05	First Name 1 Found	FirstName1 was found in our census table of names. Very likely to be a real first name.
NS06	Last Name 1 Found	LastName1 was found in our census table of names. Very likely to be a real last name.
NS07	First Name 2 Found	FirstName2 was found in our census table of names. Very likely to be a real first name.
NS08	Last Name 2 Found	LastName2 was found in our census table of names. Very likely to be a real last name.
NE01	Unrecognized Format	Two names were detected but the FullName string was not in a recognized format.
NE02	Multiple First Names Detected	Multiple first names were detected and could not be accurately genderized.
NE03	Vulgarity Detected	A vulgarity was detected in the name.
NE04	Suspicious Word Detected	The name contained words found on the list of nuisance names, such as "Mickey Mouse."
NE05	Company Name Detected	The name contained words normally found in a company name.
NE06	Non-Alphabetic Character Detected	The named contained a non-alphabetic character.

Verify Result Codes

Code	Short Description	Long Description
VR01	Individual and Address Match	The individual name and address match.
VR02	Individual and Phone Match	The individual name and phone match.
VR03	Individual and Email Match	The individual name and email match.
VR04	Address and Phone Match	The address and phone match.
VR05	Address and Email Match	The address and email match.
VR06	Phone and Email Match	The phone and email match.
VR07	Organization and Address Match	The organization name and address match.
VR08	Organization and Phone Match	The organization name and phone match.

Code	Short Description	Long Description
VR09	Organization and Email Match	The organization name and email match.
VR10	Organization and Individual Match	The organization name and individual name match.
VS00	Address Not Found	An address was not found in the reference data.
VS01	Historical Address Match	A match was made to a historical address.
VS02	Partial Address Match	A match was made to a partial address. This could be due to matching the street address but not to the suite.
VS12	Partial Last Name Match	A match was made to the last name only.
VS13	Partial First Name Match	A match was made to the first name only.
VS22	Partial Company Name Match	A match was made to a partial company name.
VS30	Phone Not Found	A phone number was not found in the reference data.
VS31	Historical Phone Match	A match was made to a historical phone number.
VS40	Email Not Found	An email address was not found in the reference data.
VS41	Historical Email Address	A match was made to a historical email address.

Append Result Codes

Code	Short Description	Long Description
DA00	Address Appended	An address was changed or appended.
DA01	City/State Append from Phone	A city or state was appended from a phone number wire center.
DA10	Name Appended	A full name was changed or appended.
DA20	Company Appended	A company name was changed or appended.
DA30	Phone Appended	A phone number was changed or appended.
DA40	Email Appended	An email address was changed or appended.

Response Result Codes

Response Results Codes are returned in the <RequestResults>

Code	Short Description	Long Description
SE01	Web Service Internal Error	The web service experienced an internal error.
GE01	Empty Request Structure	The SOAP, JSON, or XML request structure is empty.
GE02	Empty Request Record Structure	The SOAP, JSON, or XML request record structure is empty.

Code	Short Description	Long Description
GE03	Records Per Request Exceeded	The counted records sent more than the number of records allowed per request.
GE04	Empty CustomerID	The CustomerID is empty.
GE05	Invalid CustomerID	The CustomerID is invalid.
GE06	Disabled CustomerID	The CustomerID is disabled.
GE07	Invalid Request	The SOAP,JSON, or XML request is invalid.
GE08	Invalid CustomerID for Product	The CustomerID is invalid for this product.
GE20	Verify Not Activated	The Verify package was requested but is not active for the Customer ID.
GE21	Append Not Activated	The Append package was requested but is not active for the Customer ID.
GE22	Move Not Activated	The Move package was requested but is not active for the Customer ID.
GW01	Expiring License	The license will expire within 2 weeks.

Appendix

Example Requests/Responses

REST

Request

```
https://personator.melissadata.net/v3/WEB/ContactVerify/
doContactVerify?t=Sample&id=[CUSTOMERID]&act=Check&cols=
&opt=CentricHint:Auto;AdvancedAddressCorrection:Off&first=&last=
&full=&comp=melissa%20data&a1=22382%20avenida%20empresa&a2=
&city=rancho%20santa%20margarita&state=ca&postal=92688&ctry=
&lastlines=&ff=&email=&phone=9498583000&reserved=
```

Response

```
<Response xmlns:i="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://schemas.datacontract.org/2004/07/WcfServiceMD.
mdContactVerify">
  <Records>
    <ResponseRecord>
      <AddressExtras></AddressExtras>
      <AddressKey>92688211282</AddressKey>
      <AddressLine1>22382 Avenida Empresa</AddressLine1>
      <AddressLine2></AddressLine2>
      <City>Rancho Santa Margarita</City>
      <CompanyName>Melissa Data</CompanyName>
      <EmailAddress></EmailAddress>
      <NameFull></NameFull>
      <PhoneNumber>9498583000</PhoneNumber>
      <PostalCode>92688-2112</PostalCode>
      <RecordExtras></RecordExtras>
      <RecordID>1</RecordID>
      <Reserved></Reserved>
      <Results>AS01, PS02, PS08</Results>
      <State>CA</State>
    </ResponseRecord>
  </Records>
  <TotalRecords>1</TotalRecords>
```

```

    <TransmissionReference>Sample</TransmissionReference>
    <TransmissionResults></TransmissionResults>
    <Version>3.0.50</Version>
</Response>

```

XML

Request

```

<Request>
  <TransmissionReference>Sample</TransmissionReference>
  <CustomerID>[CUSTOMERID]</CustomerID>
  <Actions>Check;Verify</Actions>
  <Columns>GrpParsedAddress,GrpAddressDetails,</Columns>
  <Options>CentricHint:Address;AdvancedAddressCorrection:On</
Options>
  <Records>
    <RequestRecord>
      <RecordID>0</RecordID>
      <FirstName/>
      <LastName/>
      <FullName/>
      <CompanyName>melissa data</CompanyName>
      <AddressLine1>22382 avenida empresa</AddressLine1>
      <AddressLine2/>
      <City>rancho santa margarita</City>
      <State>ca</State>
      <PostalCode>92688</PostalCode>
      <Country/>
      <LastLine/>
      <EmailAddress/>
      <PhoneNumber/>
      <FreeForm/>
      <Reserved/>
    </RequestRecord>
  </Records>
</Request>

```

Response

```

<Response xmlns:i="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://schemas.datacontract.org/2004/07/WcfServiceMD.
mdContactVerify">

```

```
<Records>
  <ResponseRecord>
    <AddressDeliveryInstallation> </AddressDeliveryInstallation>
    <AddressExtras></AddressExtras>
    <AddressHouseNumber>22382</AddressHouseNumber>
    <AddressKey>92688211282</AddressKey>
    <AddressLine1>22382 Avenida Empresa</AddressLine1>
    <AddressLine2></AddressLine2>
    <AddressLockBox></AddressLockBox>
    <AddressPostDirection></AddressPostDirection>
    <AddressPreDirection></AddressPreDirection>
    <AddressPrivateMailboxName></AddressPrivateMailboxName>
    <AddressPrivateMailboxRange></AddressPrivateMailboxRange>
    <AddressRouteService></AddressRouteService>
    <AddressStreetName>Avenida Empresa</AddressStreetName>
    <AddressStreetSuffix></AddressStreetSuffix>
    <AddressSuiteName></AddressSuiteName>
    <AddressSuiteNumber></AddressSuiteNumber>
    <AddressTypeCode>S</AddressTypeCode>
    <CarrierRoute>C059</CarrierRoute>
    <City>Rancho Santa Margarita</City>
    <CityAbbreviation>Rcho Sta Marg</CityAbbreviation>
    <CompanyName>Melissa Data</CompanyName>
    <CountryCode>US</CountryCode>
    <CountryName>United States of America</CountryName>
    <DeliveryIndicator>B</DeliveryIndicator>
    <DeliveryPointCheckDigit>1</DeliveryPointCheckDigit>
    <DeliveryPointCode>82</DeliveryPointCode>
    <EmailAddress></EmailAddress>
    <NameFull></NameFull>
    <PhoneNumber></PhoneNumber>
    <PostalCode>92688-2112</PostalCode>
    <RecordExtras></RecordExtras>
    <RecordID>0</RecordID>
    <Reserved></Reserved>
    <Results>AS01,VR01</Results>
    <State>CA</State>
    <StateName>California</StateName>
    <UTC>-08:00</UTC>
    <UrbanizationName></UrbanizationName>
  </ResponseRecord>
</Records>
```

```
<TotalRecords>1</TotalRecords>  
<TransmissionReference>Sample</TransmissionReference>  
<TransmissionResults></TransmissionResults>  
<Version>3.0.50</Version>  
</Response>
```