Reference Guide

Personator™

MELISSA DATA®

Personator
Web Service

Reference Guide

Melissa Data Corporation

Copyright

Trademarks

Melissa Data Corporation

22382 Avenida Empresa
Rancho Santa Margarita, CA 92688-2112

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

E-mail: info@MelissaData.com
Internet: www.MelissaData.com

For the most recent version of this document, visit
http://www.melissadata.com/

Document Code: DQTWSPERRG
Revision Number: 23052013.09

Dear Developer,

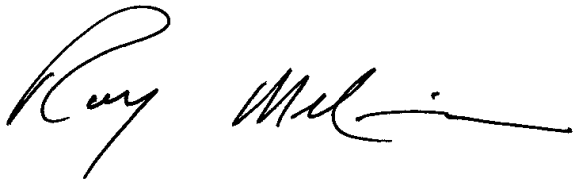
I would like to take this opportunity to thank you for your interest in Melissa Data products and introduce you to the company.

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,



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 5
Pseudocode Request 5
Response Fields 7
Request 7
Response 8
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) 14
Check (Name) 15
Verify (Address, Phone, Email) 16
Append (Address, Phone, Email) 17
Inputs 18
Definition 18
Code 19

Response Details 23

Outputs 23
Columns 23
Default Columns 23
No Group Columns 24
Address Details Group Columns 25
Census Group Columns 26
GeoCode Group Columns 27
Name Details Group Columns 28
Parsed Address Group Columns 29
Parsed Email Group Columns 30
Parsed Phone Group Columns 31
AddressType Results Codes 33
Check (Address) Results Codes 33
Address Error Codes 34
Address Change Codes 34
GeoCoder Results Codes 35

Check (Phone) Results Codes 35
Check (Email) Results Codes 36
Check (Name) Results Codes 37
Verify Results Codes 38
Move (Address) Results Codes 38
Append Results Codes 38
Response Results Codes 39

Appendix 40

Example Requests/Responses 40
REST 40
XML 41

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. 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.

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 ***Move*** action allows you to update your 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 contact records with data returned by the Personator Web Service. The service will return elements based on the selected point of centricity 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***. It must include your customer ID which serves as a key for accessing the service. You also need to select whatever actions you want Personator to execute, which options you want it 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.

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 ***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 ***Move*** action allows you to get the latest move information for an individual or business. It requres that you have at the very least, a person's last name and an address or a business/company name and and 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 ***Append*** action will return elements based on the selected point of centricity 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.

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 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.
- REST

  Uses HTTP GET to accept one input record and returns the response in XML format. 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/>
  </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>
      <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=
```

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":,
  }]
}
```

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 and at least one record.

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

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.

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.

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.

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 centricity 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.

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)

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.

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

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).

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

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)

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

Options

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

Check (Email)

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)

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.
- HypenatedLast: 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)

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)

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 40.

Inputs

Definition

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.
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.

Code

Input	Code	
TransmissionReference	XML/SOAP: REST:	TransmissionReference &t
CustomerID	XML/SOAP: REST:	CustomerID &id
Actions	XML/SOAP: REST:	Actions &act
Columns	XML/SOAP: REST:	Columns &cols
Options	XML/SOAP: REST:	Options &opt
FirstName	XML/SOAP: REST:	FirstName &first
LastName	XML/SOAP: REST:	LastName &last
FullName	XML/SOAP: REST:	FullName &full
CompanyName	XML/SOAP: REST:	CompanyName &comp
AddressLine1	XML/SOAP: REST:	AddressLine1 &a1
AddressLine2	XML/SOAP: REST:	AddressLine2 &a2
City	XML/SOAP: REST:	City &city
State	XML/SOAP: REST:	State &state
PostalCode	XML/SOAP: REST:	PostalCode &postal
Country	XML/SOAP: REST:	Country &ctry
LastLine	XML/SOAP: REST:	LastLine &lastline
Email	XML/SOAP: REST:	Email &email
Phone	XML/SOAP: REST:	Phone &phone

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.

Column	Group	Default Action
AddressExtras (Default)		Check
AddressKey (Default)		Check
AddressLine1 (Default)		Check
AddressLine2 (Default)		Check
City (Default)		Check
CompanyName(Default)		Check
EmailAddress (Default)		Check
NameFull (Default)		Check
PhoneNumber (Default)		Check
PostalCode (Default)		Check
State (Default)		Check
Plus4	(No default group)	
PrivateMailBox	(No default group)	
Suite	(No default group)	
AddressTypeCode	GrpAddressDetails	
CarrierRoute	GrpAddressDetails	
CityAbbreviation	GrpAddressDetails	
CountryCode	GrpAddressDetails	
CountryName	GrpAddressDetails	
DeliveryIndicator	GrpAddressDetails	
DeliveryPointCheckDigit	GrpAddressDetails	
DeliveryPointCode	GrpAddressDetails	
StateName	GrpAddressDetails	
UrbanizationName	GrpAddressDetails	
UTC	GrpAddressDetails	

Column	Group	Default Action
CBSACode	GrpCensus	
CBSADivisionCode	GrpCensus	
CBSADivisionLevel	GrpCensus	
CBSADivisionTitle	GrpCensus	
CBSALevel	GrpCensus	
CBSATitle	GrpCensus	
CensusBlock	GrpCensus	
CensusTract	GrpCensus	
CongressionalDistrict	GrpCensus	
CountyFIPS	GrpCensus	
CountyName	GrpCensus	
PlaceCode	GrpCensus	
PlaceName	GrpCensus	
Latitude	GrpGeocode	
Longitude	GrpGeocode	
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	
AddressPostDirection	GrpParsedAddress	
AddressPreDirection	GrpParsedAddress	

Column	Group	Default Action
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	

Response Details

A response is the result of a request. This consists of returned results codes and parsed, corrected, and/or appended request elements, depeding 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 onlly 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.

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.

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 Results Codes

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

The results 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 results 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 Results 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.
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, Results codes can be cascaded as well to define multiple categories. It is important to read and understand all the results codes to figure out which ones apply to your business needs.

Results Codes

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

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

AddressType Results 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) Results Codes

Address Status Codes

Code	Description
AS01	Address Verified. The address is valid and deliverable.
AS02	Default Address. The default building address was verified but the suite or apartment number is missing or invalid.
AS03	Non-USPS Address. This address is not deliverable by USPS, but it exists.
AS09	Foreign Address. Address is in a non-supported country.
AS10	CMRA Address. Address is a Commercial Mail Receiving Agency (CMRA) like the UPS Store. These addresses include a Private Mail Box (PMB or #) number. (US Only)
AS13	Address Updated By LACS. LACSLink®updated the address from a rural-style (RR 1 Box 2) to a city-style address (123 Main St).
AS14	Suite Appended. SuiteLink® appended a suite number using the address and company name. (US Only)
AS15	Apartment Appended. AddressPlus appended an apartment number using the address and last name.
AS16	Vacant Address. Address has been unoccupied for more than 90 days. (US Only)
AS17	No Mail Delivery. Address does not receive mail at this time. (US Only)
AS18	DPV® Locked Out. DPV processing was terminated due to the detection of what is determined to be an artificially created address. (US Only)
AS20	Deliverable only by USPS. This address can only receive mail delivered through USPS (ie. PO Box or a military address).
AS23	Extraneous information found. Extra information not used in verifying the address was found. They have been placed in the ParsedGarbage field.

Address Error Codes

Code	Description
AE01	Postal Code Error. The ZIP or Postal Code does not exist and could not be determined by the city/municipality and state/province.
AE02	Unknown Street. The street name was not found.
AE03	Component Error. Either the directionals (N, E, SW, etc) or the suffix (AVE, ST, BLVD) are missing or invalid.
AE04	Non-Deliverable. The physical location exists but there are no addresses on this side of the street. (US Only)
AE05	Multiple Match. Input matched to multiple addresses and there is not enough information to break the tie.
AE06	Early Warning System. This address cannot be verified now but will be at a future date. (US Only)
AE07	Minimum Address. The required combination of address/city/state or address/zip is missing.
AE08	Suite/Apartment Invalid. The suite or apartment number is not valid.
AE09	Suite/Apartment Missing. The suite or apartment number is missing.
AE10	House/Building Number Invalid. The address number in the input address is not valid.
AE11	House/Building Number Missing. The address number in the input address is missing.
AE12	Box Number Invalid. The input address box number is invalid.
AE13	Box Number Missing. The input address box number is missing.
AE14	PMB number Missing. The address is a Commercial Mail Receiving Agency (CMRA) and the Private Mail Box (PMB or #) number is missing.

Address Change Codes

Code	Description
AC01	ZIP Code Change. The ZIP Code was changed or added.
AC02	State Change. The State abbreviation was changed or added.
AC03	City Change. The City name was changed or added.
AC04	Address Base Alternate Change. The address was found to be an alternate record and changed to the base (preferred) version.
AC05	Alias Name Change. The street name was changed from a former or nickname street to the USPS preferred street name.
AC06	Address Swapped. Address1 was swapped with Address2 (only Address2 contained a valid address).
AC07	Address1 & Company Swapped. Address1 and Company were swapped (only Company contained a valid address).
AC08	Plus4 Change. A non-empty Plus4 was changed.

Code	Description
AC09	Urbanization Change. The Urbanization was changed.
AC10	Street Name Change. The street name was changed.
AC11	Street Suffix Change. The street suffix was changed.
AC12	Street Predirection or Postdirection Change. The street predirection or postdirection was changed.
AC13	Suite Name Change. The suite name was changed.
AC14	Suite Range Change. The secondary unit number was changed or appended.

GeoCoder Results Codes

Code	Description
GS01	Geocoded to Zip+4 or the full 6-digit Postal Code Centroid. Record was coded to the ZIP + 4® centroid (U.S.) or the full 6-digit Postal Code level (Canada).
GS02	Geocoded to Zip+2 Centroid. The submitted information was geocoded down to the 7-digit (zip + plus2) level.
GS03	Geocoded to 5 digit (U.S.) or 3 digit (Canada) postal code Centroid. Record was coded to the 5-digit ZIP Code centroid (U.S.) or the first 3-digit Postal Code level (Canada).
GS05	Geocoded to Rooftop level. The submitted information was geocoded down to the 11 digit (rooftop) level.
GS06	Geocoded to Interpolated Rooftop level. The submitted information was geocoded down to the 11 digit (rooftop) interpolated level.
GE01	Invalid Zip Code entered. The submitted ZIP Code is not in a valid format.
GE02	Zip Code not found. The submitted ZIP Code was not found in the database.

Check (Phone) Results Codes

Code	Description
PS01	Phone Number matched to 10-digit level. The first 10 digits of the phone number have been verified as valid.
PS02	Phone Number matched to 7-digit level. The first 7 digits of the phone number have been verified as valid.
PS03	Corrected Area Code. NewAreaCode contains corrected area code that was changed according to the ZIP Code it falls into.
PS06	Updated Area Code. NewAreaCode contains corrected area code that was changed due to an area code split.
PS07	Phone Number on a Cellular Line. The exchange type of the phone number indicates the number is a cellular number.

Code	Description
PS08	Phone Number on a Land Line. The exchange type of the phone number indicates the number is a land line number.
PS09	Phone Number on a VOIP Line. The exchange type of the phone number indicates the number is a VOIP number.
PS10	Phone Number is a Residential. The phone number belongs to a residence.
PS11	Phone Number is a Business. The phone number belongs to a business.
PS12	Phone Number is a SOHO. 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.
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.
PE06	Bad Zip Code. The input ZIP Code is invalid.

Check (Email) Results Codes

Code	Description
ES01	Valid Email Domain. The email domain name was confirmed as valid by the DatabaseLookup.
ES02	Invalid Email Domain. The email domain name 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, classified as not deliverable by FCC
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).

Check (Name) Results Codes

Code	Description
NS01	Parsing Successful. Name parsing was successful.
NS02	Error Parsing. An error was detected. Please check for error codes.
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 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 — could not accurately genderize.
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 Results Codes

Code	Description
VR01	Individual Name and Address Match
VR02	Individual Name and Phone Match
VR03	Individual Name and Email Match
VR04	Address and Phone Match
VR05	Address and Email Match
VR06	Phone and Email Match
VR07	Organization Name and Address Match
VR08	Organization Name and Phone Match
VR09	Organization Name and Email Match
VS00	Address not found in reference data
VS01	Match made to hisotrical address
VS02	Match made to partial address (secondary range problems)
VS12	Match made to partial name (last name only)
VS22	Match made to partial company name
VS30	Phone not found in reference data
VS31	Match made to historical phone
VS40	Email not found in reference data
VS41	Match made to historical email address

Move (Address) Results Codes

Code	Description
AS12	The submitted record is a move and the latest address is returned in the Address Output fields.

Append Results Codes

Code	Description
DA00	Address Appended
DA10	Name Appended
DA20	Company Appended
DA30	Phone Appended
DA40	Email Appended

Response Results Codes

Response Results Codes are returned in the <RequestResults>

Code	Name	Description
SE01	Web Service Internal Error	The web service experienced an internal error.
GE01	Empty JSON or XML Request Structure	The JSON or XML request structure is empty.
GE02	Empty JSON or XML Request Record Structure	The JSON or XML request record structure is empty.
GE03	Records Per Request Exceeded	The counted records sent more than the number of records allowede 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 JSON or XML Request	The JSON or XML request is invalid.
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.
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=&freeform=&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>
```