Reference Guide

SmartMover™

Change of Address Processing

MELISSA DATA®

SmartMover Web Service

Reference Guide

Copyright

Trademarks

MELISSA DATA CORPORATION
22382 Avenida Empresa
Rancho Santa Margarita, CA 92688-2112

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

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

Document Code: WSMRFG
Revision number: 120611.058
Last Update: June 11, 2012

Dear Developer,

Thank you for selecting our SmartMover Web Service. This convenient move-update solution will help you reduce undeliverable mail by providing the most current address data available. Using a SOAP toolkit and your programming language of choice, you get the power of NCOALink processing, on-demand, 24/7 - with no software to install or maintain.

- SmartMover Web Service will help you:
- Reduce undeliverable mail
- Lower postage, printing and mail preparation costs
- Speed mail processing and delivery
- Get mail delivered to your intended recipient
- Meet USPS requirements for automation discounts
- Increase response rates and ROI
- Maintain contact with your customers

Melissa Data offers solutions to help businesses of every size achieve the highest level of address quality, maximize response rates, reduce undeliverable mail, and improve customer sales and service. From mailing software, developer tools, to 24/7 data enhancement and hygiene services and mailing lists, we have the tools you need to succeed - all from ONE company. As your partner in data quality, Melissa Data can offer you a one-stop solution for all of your direct marketing and volume mailing needs.

Your feedback is always important to me. Please don't hesitate to contact me with your comments or suggestions to ray@melissadata.com. I look forward to hearing from you.

Best Wishes

Raymond F. Melissa
President

Contents

Using SmartMover Web Service

Create a SmartMover Account

You must first have a Melissa Data sales representative create a SmartMover Account for you. In creating this account, you will need to determine and pay for credits based on the number of records you plan to process. You can call your sales representative and add additional credits to your account at any time but if you exceed your credit threshold, the SmartMover Web Service will stop processing and give you an error. When your account is created, you will receive an email with details about the service and your CustomerID number that you will use to process records.

Send a Processing Acknowledgement Form (PAF)

The USPS requires us to maintain a document for all customers using the SmartMover Web Service. Before you can start processing, you must submit a PAF to Melissa Data.

This can be done online at:

http://www.melissadata.com/user/end_user_paf.aspx

Call the SmartMover Web Service

Once you have your CustomerID, you are ready to start processing. You start by building a SmartMover RequestArray. This RequestArray is made up of your CustomerID, NCOA processing options, and an array of input records. You can fill this array with up to 100 records per call to the RequestArray. Once you build your RequestArray, you call the SmartMover Web Service and send us your RequestArray. We will process your records and send back a ResponseArray with the results. Keep doing this until you finish processing your entire list.

Submitting/Retrieving Data from SmartMover Web Service

Add SmartMover WSDL to Your Project

If you are using Visual Studio.NET, you need to add a web reference to the service to your project. Click on the **Project** menu and select **Add Web Reference...** Enter the following URL on the Add Web Reference dialog box:

Secure: **https://smartmover.melissadata.net/V2b/Smartmover.asmx**
Non-Secure: **http://smartmover.melissadata.net/V2b/Smartmover.asmx**

Name your web reference. For the sample code below, we will name it SmartMoverWS.

If you are not using Visual Studio.NET, see the documentation for your SOAP toolkit to see how to import the SmartMover wsdl.

Create SmartMover Web Service Objects

SmartMover Web Service uses the following objects to process your request and return you the results:

- RequestArray: Array of Records to send to SmartMover
- RequestRecord: An individual record that make up the RequestArray.
- SmartMover: Used to send and receive data
- ResponseArray: Array of Records received back from SmartMover

- ResponseRecord: An individual result record that make up the ResponseArray

Build your RequestArray

First, fill in your CustomerID information as well as what NCOA processing options you would like. Initialize the RequestArray.Record variable as a new array. Then, for each individual input record, create a new RequestRecord and populate the input properties. Add that record into RequestArray.Record until you run out of records or you reach 100 records. Now, put the number of records added to RequestArray.Record into RequestArray.TotalRecords.

Send the RequestArray to SmartMover Web Service

After you have built your RequestArray, call SmartMover.DoSmartMover(reqtArray) and pass in your RequestArray as the parameter. This method will return you a ResponseArray as the result.

Gzip

The SmartMover web service supports the Gzip compression method available with many operating systems and programming languages. This will shrink the size of your tranmission data, lowering your bandwidth usage.

See your programming language documentation and gzip documentation for details.

Parse the ResponseArray

Once you get the ResponseArray back, parse it for the results and update your database. First, check ResponseArray.Fault to see if there was anything wrong with your RequestArray or with the service itself. Now, go though the ResponseArray.Record array and process each individual ResponseRecord and put the results back into your database.

Sample Implementation Code

1. **Create SmartMover Object:**

```
SmartMoverWS.SmartMover sm = new SmartMoverWS.SmartMover();
```

2. **Create A RequestArray Object and fill in the NCOA processing information for this session:**

```
SmartMoverWS.RequestArray reqArray = new SmartMoverWS.RequestArray();
reqArray.CustomerID = 1122334455;
reqArray.ExecutionID = 1;
reqArray.JobID = "ABCDEF"
reqArray.OptAddressParsed = false;
reqArray.OptSmartMoverDetail = false;
reqArray.OptSmartMoverListName = "List12";
reqArray.OptSmartMoverListOwnerFreqProcessing = 12;
reqArray.OptSmartMoverNumberOfMonthsRequested = 48;
reqArray.OptSmartMoverProcessingType =
    SmartMoverWS.ProcessingType.Standard;
```

3. **Initialize the Request.Record array, maximum of 100:**

```
reqArray.Record = new SmartMoverWS.RequestRecord[100];
```

4. **Loop through your records up to 100 at a time:**

```
For count = 0 to 99 Do
```

5. **For each record, create a new RequestRecord, fill it with your input record data, and add to the record array:**

```
SmartMoverWS.RequestRecord reqRecord = new
SmartMoverWS.RequestRecord();
reqRecord.RecordID = "Unique Identified"
reqRecord.Company = "Company Name if available"
reqRecord.FullName = "Full Name if available"
reqRecord.Address1 = "Address1 line"
reqRecord.Address2 = "Address2 line if present"
reqRecord.City = "City"
reqRecord.State = "State"
reqRecord.Zip = "Zip Code"
reqArray.Record[count] = reqRecord
```

6. **After you have added all your records to the Record array, set TotalRequests to the number of records you added:**

```
reqArray.TotalRecords = count + 1;
```

7. **Send the RequestArray to the SmartMover Web Service and get a ResponseArray back:**

```
SmartMoverWS.ResponseArray respArray = sm.DoSmartMover(reqArray);
```

8. **Check the ResponseArray for any faults:**

```
if (respArray.Fault.Code != "")
{
    //Handle error
}
```

9. **Loop through the Record array and retrieve all record results:**

```
For count = 0 to (respArray.TotalRecords - 1) Do
```

10. **For each ResponseRecord inside the Record array, check the Status code to see if the record was moved, standardized, or an error:**

```
if (respArray.Record[x].Address.Result.Status.Code == "0")
{
    //There was an error in the input address
    String ErrorCode = respArray.Record[x].Address.Result.Error.Code;
    String ErrorDescription =
        respArray.Record[x].Address.Result.Error.Description;
}
if (respArray.Record[x].Address.Result.Status.Code == "1")
{
    //This record was a move and the new address is returned
    String NewName = respArray.Record[x].Name.Full;
    String NewCompany = respArray.Record[x].Company.Name;
    String NewAddress1 = respArray.Record[x].Address.Address1;
    String NewSuite = respArray.Record[x].Address.Suite;
    String NewCity = respArray.Record[x].Address.City.Name;
    String NewState = respArray.Record[x].Address.State.Name;
    String NewZip = respArray.Record[x].Address.Zip;
```

```
    String NewPlus4 = respArray.Record[x].Address.Plus4;
    String NewAddress1 = respArray.Record[x].Address.Address1;
}
if (respArray.Record[x].Address.Result.Status.Code == "2")
{
    //This record did not move, but has been verified, standardized,
        and/or corrected
    String StdName = respArray.Record[x].Name.Full;
    String StdCompany = respArray.Record[x].Company.Name;
    String StdAddress1 = respArray.Record[x].Address.Address1;
    String StdSuite = respArray.Record[x].Address.Suite;
    String StdCity = respArray.Record[x].Address.City.Name;
    String StdState = respArray.Record[x].Address.State.Name;
    String StdZip = respArray.Record[x].Address.Zip;
    String StdPlus4 = respArray.Record[x].Address.Plus4;
    String StdAddress1 = respArray.Record[x].Address.Address1;
}
```

RequestArray

The RequestArray object contains information from the user to be processed by the SmartMover Web Service. This includes the fields to identify the customer to the service and set the desired NCOA^{Link} options, in addition to an array of address records to be processed.

Customer Information and Option Fields

The following fields identify the user to the SmartMover Web Service and set the necessary options to process a list of address records. Except where indicated, all of these fields should be sent as string values.

CustomerID

`RequestArray.CustomerID`

The customerID is a string of characters issued by Melissa Data when you open your SmartMover Web Service account.

PAFId

`RequestArray.PAFID`

The PAF is a string value which identifies the individual list owner when the request is submitted from a broker account.

This value is required for any CustomerID issued to a broker account.

JobID

`RequestArray.JobID`

The JobID is a unique string of characters used to identify a group records submitted with this request. The JobID is returned with the Response Array, making it simple to connect the records returned with the original request.

This JobID is also used to retrieve the various summary reports returned by the Smart Mover Web Service.

ExecutionID

`RequestArray.ExecutionID`

The ExecutionID identifies the thread that you are processing under when using multiple threads. This allows our system to track individual threads in the event that a request times out. If you send the same request immediately after a request times out, using the same ExecutionID, our system will process the 2nd request but it will not count against the total number of records in the NCOA and CASS reports. The default value is 0 if not set.

TotalRecords

`RequestArray.TotalRecords`

This is the number of records contained in the record array.

OptAddressParsed

`RequestArray.optAddressParsed`

This is a boolean value. If set to true, the SmartMover Web Service also returns the address parsed into component parts. If this option is not set, it defaults to false.

OptSmartMoverDetail

`RequestArray.OptSmartMoverDetail`

This is a boolean value. If set to true, the SmartMover Web Service will return the original address, the standardized address and the move address, if any, in separate sections. If set to False, the web service will return the best result, either a standardized address, a moved address or an error message. This option defaults to False.

OptSmartMoverProcessingType

`RequestArray.OptSmartMoverProcessingType`

This field accepts an option from the ProcessingType enumeration.

Value	Option
Standard	Standard Processing Mode requires inquiries in the following order: • **Business** – Match on business name. • **Individual** – Match on first name, middle name, surname and title required. Gender is checked and nickname possibilities are considered. • **Family** – Match on surname only.
IndividualAndBusiness	The NCOA^{Link} customer may choose to omit all “Family” match inquiries and allow only “Individual” and “Business” matches to be acceptable.
Individual	The NCOA^{Link} customer may also choose to omit “Business” match inquiries when processing individual names for mailing lists that contain no business addresses
Business	The NCOA^{Link} customer may choose to process for only “Business” matches when processing a “Business-to-Business” mailing list which contains no residential (Individual or Family) addresses.
Residential	The NCOA^{Link} customer may choose to omit “Business” match inquiries and allow only “Individual” and “Family” matches to be acceptable under Residential Processing Mode.

OptSmartMoverListOwnerFreqProcessing

`RequestArray.OptSmartMoverListOwnerFreqProcessing`

This field accepts an integer value from 1 to 52. This is the number of times per year that the current mailing list is used for mailing. If you use it monthly, enter 12; for quarterly, use 4, etc.

OptSmartMoverNumberofMonthsRequested

`RequestArray.OptSmartMoverNumberofMonthsRequested`

The field accepts an integer value from 6 to 48. This is the number of months back that you want the web service to search for a change of address.

OptSmartMoverListName

`RequestArray.OptSmartMoverListName`

This is the name that identifies the current list. It will be included in the reports that the SmartMover Web Service returns after processing.

RequestArray Record Fields

RequestArray.Record is an array of records containing the addresses to be processed.

Each record within the RequestArray contains some or all of the following fields. The following fields are required for each record:

1. Either **FirstName** and **LastName**, **FullName** or **Company**
2. Address
3. Either **City** and **State**, **ZIP** or **AddressLastLine**

RecordID

`RequestArray.Record[index].RecordID`

This is a unique identifier for this record from your own database. This is not required but it can assist in matching the record in the ResponseArray with the original record.

Company

`RequestArray.Record[index].Company`

If this address is that of a business, include the company name here.

Urbanization

`RequestArray.Record[index].Urbanization`

This field is only used for addresses located in Puerto Rico and is used to break ties between similar addresses in the same ZIP Code.

The urbanization name tells the address checking logic which “neighborhood” to look in if more than one likely address candidate is found.

Address1

`RequestArray.Record[index].Address1`

This is the primary street address. It may also include the suite number.

Address2

`RequestArray.Record[index].Address2`

This may include the suite or mailbox number or an alternate address, such as a P.O. Box.

AddressLastLine

`RequestArray.Record[index].AddressLastLine`

If your database stores City, State and ZIP Code as a single string value, you would pass it to the web service via this field.

Suite

`RequestArray.Record[index].Suite`

If your address stores the suite number separately, pass it to the web service via this field.

PrivateMailBox

`RequestArray.Record[index].PrivateMailbox`

If this address is actually a box in a private mailbox service, and the number is stored separately enter the number here.

City

`RequestArray.Record[index].City`

Pass the full name of the city via this field.

State

`RequestArray.Record[index].State`

Pass either the full name or the two-letter abbreviation for the state via this field.

Zip

`RequestArray.Record[index].Zip`

This could either be a five-digit ZIP Code, the first five digits of a ZIP + 4 or the full nine-digit ZIP + 4.

Plus4

`RequestArray.Record[index].Plus4`

Use this for the last four digits of a ZIP + 4 if not included with Zip field.

Country

`RequestArray.Record[index].Country`

You may pass a country code via this field, but the SmartMover Web Service can only update addresses within the United States.

FullName

`RequestArray.Record[index].FullName`

If this is an individual person's address and the entire name is stored as a single string, pass the string value via this field. The name will be parsed when the record is returned via the ResponseArray.

FirstName

`RequestArray.Record[index].FirstName`

If this is an individual person's address and the first name is stored separately, pass the string value via this field.

MiddleName

`RequestArray.Record[index].MiddleName`

If this is an individual person's address and the middle name is stored separately, pass the string value via this field.

LastName

`RequestArray.Record[index].LastName`

If this is an individual person's address and the last name is stored separately, pass the string value via this field.

NamePrefix

`RequestArray.Record[index].NamePrefix`

This would be a title or honorific such as ”Mr,” ”Miss,” or ”Dr.”

NameSuffix

`RequestArray.Record[index].NameSuffix`

This would be a generational or professional suffix, such as “Jr,” “IV” or “Ph.D.”

ResponseArray

The SmartMover Web Service returns its results in the ResponseArray Object. This object contains any fault codes generated by the request, links to summary reports, and an array containing the results for each record.

Response Fault

The ResponseArray's Fault object returns codes which indicate the cause of a problem with the user's request. If this object's properties are empty then no fault was generated and the request was processed normally. If there is a fault, an error has occurred and the records from the request were not processed.

The Fault object contains four properties which indicate the cause and source of the fault.

Code

`ResponseArray.Fault.Code`

This field contains the unique code for the error that caused fault.

Desc

`ResponseArray.Fault.Desc`

This field contains a detailed description of the cause of the fault.

Source

`ResponseArray.Fault.Source`

This field indicates which component of the SmartMover service generated the fault.

Detail

`ResponseArray.Fault.Detail`

This field contains a further description of the type of error that generated the fault.

Return Values

The possible values of these properties are listed in the table below.

Code	Desc	Source	Detail
WSC00	Your request could not be processed due to an unexpected internal configuration error. Please contact our sale representative.	SmartMover	InternalException
WSE00	Your request could not be processed due to an unexpected error. Please retry your request.	SmartMover	InternalException

Code	Desc	Source	Detail
WSE03	Months to allow for NCOALink product ranges from 6 to 48.	SmartMover	Customer Request Setting
WSE05	Records array is empty, un-allocated, or out of bounds.	SmartMover	Customer Request Setting
WSE13	Requesting records must in the range of 1 and 100	SmartMover	Customer Request Setting
WSE14	Field does not exist in summary report table.	SmartMover	Customer Request Setting
WSE15	OptSmartMoverProcessing-Type is empty or incorrect type	SmartMover	Customer Request Setting
WSE16	OptSmartMoverListOwner-FreqProcessing ranges from 1 to 52.	SmartMover	Customer Request Setting
WSE17	OptSmartMoverListName cannot contain over 30 characters.	SmartMover	Customer Request Setting
WSE18	OptSmartMoverListName cannot contain any of the following characters \/*:?""<>|	MoveObject Component	Internal error. Contact Melissa Data.
WSE21	PAF's customer does not exist.	SmartMover	Customer Request Setting
WSE23	Bad request stream or incorrect request XML type.	SmartMover	Customer Request Setting
WES24	Invalid link	SmartMover	Customer Request Setting
WSE26	CustomerID is not valid	SmartMover	CustomerID is not valid, customerID is not enabled, or customer's PAF is not on file.
WSE28	JobID too long.	MoveObject Component	JobID cannot contain over 50 characters.

Code	Desc	Source	Detail
WSE29	Invalid characters in JobID	SmartMover	JobID cannot contain any of the following characters \/*:?"<> |, non-printing characters or spaces.

Total Records

`ResponseArray.TotalRecords`

This property returns the total number of records processed. It should match the TotalRecords field in the original request.

JobID

`ResponseArray.JobID`

This field returns the JobID submitted with the request array, making it simple to matched the records returned with the original request.

Record

`ResponseArray.Record[index]`

The ResponseArray returns an array of records, one for each submitted, containing the processed address information and the name or company information.

RecordID

`ResponseArray.Record[index].RecordID`

This returns the unique identifier submitted with the original request. Use this field to match the current record with the corresponding record in your own database.

Address

See the following section for a breakdown of the address information returned by each record.

Name

See the section beginning on page 49 for a breakdown of the name information returned by each record.

Company

```
ResponseArray.Record[index].Company.Name
```

Returns a string value containing the company name for this record as passed via the RequestArray.

Results

```
ResponseArray.Record[index].Results.Code
ResponseArray.Record[index].Results.Description
```

The Code field returnd a comma-delimted set of four-character codes indicating the address verification and move status of the current record. The Description field returns a comma-delimited list of the matching explanations for each code.

Address Check Result Codes

Code	U.S. Only	Meaning	Details
		Status Result Codes	
AS01		Address Matched to Postal Database	Street Address is valid and deliverable. Check AE08 and AE09 for full deliverability.
AS02		Street Address Match	Address street matched to USPS database but a suite was missing or invalid.
AS09		Foreign Postal Code Detected	Postal Code from a non-supported foreign country detected. A US or Canadian Postal Code can also return this error if the US or Canadian data files are not initialized.
AS10	Y	Address Matched to CMRA	Address belongs to a Commercial Mail Receiving Agency (CMRA) like The UPS Store®.
AS13	Y	Address has been Updated by LACS^{Link}	Address has been converted by LACS$^{Link®}$ from a rural-style address to a city-style address.
AS14	Y	Suite Appended by Suite^{Link}	A suite was appended by Suite$^{Link™}$ using the address and company name.
AS15	Y	Suite Appended by AddressPlus	A suite was appended by AddressPlus using the address and last name.

Code	U.S. Only	Meaning	Details
AS16		Address is vacant	Address has been unoccupied for 90 days or more.
AS17		Alternate delivery	Address does not receive mail at this time.
AS18	Y	DPV Error	Call 1-800-Melissa Tech Support for assistance.
AS20	Y	This address is deliverable by USPS only.	Alternate carriers such as UPS and Fed Ex do not deliver to this address.
AS22	Y	No suggestions.	No suggested alternatives were found.
AS23		Extraneous Information Found	Information found in input street address that was not used for verification. This information was returned by the GetParsedGarbage function.
		Error Result Codes	
AE01		Zip Code Error	The Postal Code does not exist and could not be determined by the city/municipality and state/province.
AE02		Unknown Street Error	An exact street name match could not be found and phonetically matching the street name resulted in either no matches or matches to more than one street name.
AE03		Component Mismatch Error	Either the directionals or the suffix field did not match the post office database, or there was more than one choice for correcting the address.
AE04		Non-Deliverable Address Error	The physical location exists but there are no homes on this street. One reason might be railroad tracks or rivers running alongside this street, as they would prevent construction of homes in this location.

Code	U.S. Only	Meaning	Details
AE05		Multiple Match Error	Address matched to multiple records. More than one record matches the address and there is not enough information available in the input address to break the tie between multiple records.
AE06	Y	Early Warning System Error	This address has been identified in the Early Warning System (EWS) data file and should be included in the next postal database update.
AE07		Missing Minimum Address Input Error	Minimum required input of address/city/state or address/zip not found.
AE08		Suite Range Invalid Error	The input street address was found but the input suite number was not valid.
AE09		Suite Range Missing Error	The input street address was found but a required suite number is missing.
AE10		Primary Range Invalid Error	The street number in the input address was not valid.
AE11		Primary Range Missing Error	The street number in the input address was missing.
AE12	Y	PO, HC, or RR Box Number Invalid Error	The input address PO, RR or HC number was invalid.
AE13	Y	PO, HC, or RR Box Number Missing Error	The input address is missing a PO, RR, or HC Box number.
AE14	Y	CMRA Secondary Missing Error	Address Matched to a CMRA Address but the secondary (Private mailbox number) is missing.
AE15		Demo Mode	Address Object is in demo mode and the input address is not supported in this mode. Demo mode only validates Nevada addresses.
AE16		Expired Database	The database has expired. Please update with a fresh database.

Code	U.S. Only	Meaning	Details
AE17		Suite Range Extraneous Error	A suite number was entered but no suite information found for primary address.
AE19	Y	FindSuggestion time-out	Time allotted to FindSuggestion was exceeded.
AE20	Y	Suggestions disabled.	Cannot offer suggestion. The SetCassEnable function must be set to false and the DPV data path must be set in order to use FindSuggestion.
		Change Codes	
AC01		ZIP Code Change	The five-digit ZIP Code™ was added or corrected based on the city and state names.
AC02		State Change	The state name was corrected based on the combination of city name and ZIP Code.
AC03		City Change	The city name was added or corrected based on the ZIP Code.
AC04		Base/Alternate Change	Some addresses have alternate names, often chosen by the owner or resident for clarity or prestige. This change code indicates that the address from the official, or “base,” record has been substituted for the alternate.
AC05		Alias Change	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 has been substituted for the alias.

Code	U.S. Only	Meaning	Details
AC06		Address1/Address2 Swap	The value passed to SetAddress could not be verified, but SetAddress2 was used for verification. The value passed to the SetAddress function will be returned by the GetAddress2 function.
AC07		Address1/Company Swap	A Company name was detected in address line 1 and moved to the GetCompany function.
AC08		Plus4 Change	A non-empty Plus4 was changed.
AC09		Urbanization Change	The Urbanization was changed.
AC10		Street Name Change	The street name was changed due to a spelling correction.
AC11		Suffix Change	The street name suffix was corrected, such as from “St” to “Rd.”
AC12		Street Directional Change	The street pre-directional or post-directional was corrected, such as from “N” to “NW.”
AC13		Suite Name Change	The unit type designator for the secondary address was changed, such as from “STE” to “APT.”

Move Result Codes

Code	Meaning	Detail
CS01	Move with New Address	Submitted record is a move and a new moved to address provided
CS02	Standardized	Submitted record is not a move but was standardized
CS03	Move Input Requirements not Satisfied	Submitted record is matched to change of address file but did not satisfy all requirements to produce a moved to address
CS04	Move but no New Address	Submitted record is a move but could not provide moved to address
CS10	Individual Move	Submitted record is classified as a Individual

Code	Meaning	Detail
CS11	Family Move	Submitted record is classified as a Family
CS12	Business Move	Submitted record is classified as a Business
CS13	Daily Delete	Submitted record is a Daily Delete address. COA with this address is pending deletion from the master file and that no mail may be forwarded from this address.
CM01	COA Match	COA Match: An Individual, Business or Family
CM02	Foreign Move	FoundCOA:Foreign Move
CM03	Move Left no Address	FoundCOA:Move Left No Address (MLNA)
CM04	Box Closed	FoundCOA:Box Closed No Order (BCNO)
CM05	Cannot Match Secondary	Cannot MatchCOA: Street Address with Secondary
CM06	DPBC Ambiguous	FoundCOA: New 11-digit DPBC is Ambiguous
CM07	Conflicting Middle Name	Cannot MatchCOA: Conflicting Directions: Middle Name Related
CM08	Conflicting Gender	Cannot MatchCOA: Conflicting Directions: Gender Related
CM09	Conflicting Instructions	Cannot MatchCOA: Other Conflicting Instructions
CM10	Cannot Match High-rise Default	Cannot MatchCOA: High-rise Default
CM11	Cannot Match Rural Default	Cannot MatchCOA: Rural Default
CM12	Insufficient Name	Cannot MatchCOA: Individual Match: InsufficientCOA Name for Match
CM13	Middle Name Test Failed	Cannot MatchCOA: Middle Name Test Failed
CM14	Gender Test Failed	Cannot MatchCOA: Gender Test Failed

Code	Meaning	Detail
CM15	Cannot Convert Address	Found COA: New Address Would Not Convert at Run Time
CM16	Individual Name Insufficient	Cannot MatchCOA: Individual Name Insufficient
CM17	Secondary Descrepancy	Cannot MatchCOA: Secondary Number Discrepancy
CM18	Other Insufficient Name	Cannot MatchCOA: Other Insufficient Name
CM19	Cannot Match General Delivery	Cannot MatchCOA: General Delivery
CM20	Move not ZIP+4 Coded	Found COA: New Address not ZIP+4 coded
CM21	Conflicting Directions	Cannot MatchCOA: Conflicting Directions after rechaining
CM24	Secondary Dropped from COA	COA Match: Secondary Number Dropped fromCOA
CM25	Secondary Dropped from Input	COA Match: Secondary Number Dropped from input address

Using Result Codes: Coding for the present and the future

Over a year ago, Melissa Data introduced a new concept know as Result codes. These are four-character codes (two letters followed by two numbers), delimited by commas, which indicate status and errors generated by the most recent request to an object or service. An Address Object result code for a coded address record might look something like this: “AC03, AC11, AS01, AS15.” Instead of looking at multiple properties and methods to determine the status or error of a record, you can simply look at the output of the Results property. Currently there are close to 50 possible result codes for Address Object alone. This section will dive into the best way to use these codes in your application now and in the future, focusing specifically on Address Object.

Best Practice #1: Read all the Result codes, but you won't use them all.

The first step to understanding how to use Result codes is to know each code, individually. Having said that, understanding all the codes does not mean you will use all of them. You will likely only ever use a few. We have many different codes that indicate many different statuses or errors. A code may be important to one person but not another. For example, the AS20 codes means the address is deliverable only by the USPS, like a PO Box or a military address. This would not be important for you if you already delivery using USPS or don't deliver at all, but it would be important if you delivery using a third party carrier, like UPS.

Best Practice #2: Determine what a "good" record means.

The ultimate goal of using Result codes is to determine what to do with the record you have.

To do so, first determine what a "good" record is. In most cases, it will simply involve the AS01 or AS02 codes. For example, if you want your "good" record to be all addresses verified as fully deliverable, you would use:

```
if(Results.Contains("AS01")) { //good record}
```

If you want all fully deliverable addresses but also addresses that have missing/invalid suites, you would use:

```
if(Results.Contains("AS01") or Result.Contains("AS02") { //good record}
```

In more complex cases when you want to take more factors in account, add more code to your "good" record filter. For example, if you want all records that have a fully deliverable address or records that have an invalid suite but also a 10-digit verified phone number, you would write:

```
If(Results.Contains("AS01") or (Result.Contains("AS02") and
    Result.Contains("PS01"))
```

This filter introduces the Result codes for Phone Object, which behaves the same way as Address Object Result codes logically. Having said this, you can have more than one "good" filter. It is possible to cascade them in a "good," "okay," and then "bad", in the same fashion as a switch statement. Once you have your "good" record filter, all the other records will naturally fall into the "bad" category.

Best Practice #3: Result codes will change. Code for it.

Since the inception of Result codes, the number of possible codes has doubled. Melissa Data is always innovating and adding new information and enrichments. You will not be able to know exactly what new codes may be introduced in the future, but we can still account for them. So, as we see in Best Practice #2, always use the `String.Contains()` or an equivalent function when detecting for codes, so re-

ordering and future additions will not affect your current code. Also, have all records that do not pass your filter become a “bad” record. This allows for future codes to be added without records being lost if you don't specifically filter for them.

Like many things, the best way to learn how to use Result codes is to actually try and use them. See what Result codes are produced by different types of addresses, and how your code handles them. For an up-to-date online reference of all Result codes available and examples to produce each code, visit here: **http://www.melissadata.com/tech/ResultCodes.asp**.

ResponseArray Address Fields

This section details the address properties returned by each record in the ResponseArray.

Some are identical in name and purpose to the matching fields in the RequestArray. For information on these fields, see the section on submitting records with the RequestArray, beginning on page 9.

Address Result

The address result returns any status and error codes returned by processing the current address.

Error

```
ResponseArray.Record[index].Address.Result.Error.Code
ResponseArray.Record[index].Address.Result.Error.Description
```

These fields contain the code and description for any error generated by the current address record.

Code	Description
0	Update Address Error
C	Canadian ZIP Code
M	Multiple Matches
R	Range Error
U	Unknown Street
X	Non-Deliverable Address
T	Component Error
W	Early Warning Address
Z	Invalid ZIP/Postal Code

Status

`ResponseArray.Record[index].Address.Result.Status.Code`
`ResponseArray.Record[index].Address.Result.Status.Description`

These fields contain the code and description for the status of the returned address record. Generally this code indicates whether or not the record was updated with move data, just standardized, or contained errors.

Code	Description
0	Error
1	Move
2	Standardized

Address Type

Most addresses have a type code associated with them. Some ZIP codes also have a type code which indicate that the ZIP Code has a special purpose, such as being assigned to a military facility.

Address

```
ResponseArray.Record[index].Address.Type.Address.Code
ResponseArray.Record[index].Address.Type.Address.Description
```

These fields return the address type code and the accompanying description.

Code	Type
F	Firm or Company
G	General Delivery
H	Highrise or Business Complex
P	PO Box
R	Rural Route
S	Street or Residential

ZIP

```
ResponseArray.Record[index].Address.Type.Zip.Code
ResponseArray.Record[index].Address.Type.Zip.Description
```

These fields return the ZIP type code and the accompanying description.

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

Flags

Flags return indicators about the current status of the returned address.

Vacant

`ResponseArray.Record[index].Address.Flags.Vacant`

Value	Explanation
Y	The submitted address is currently vacant.
N	The submitted address is not currently vacant.

Address Fields

The following fields include the standardized or updated contents of the fields submitted with the RequestArray. Other fields are populated during address checking or during the NCOA^{Link} processing itself.

Fields with no description are identical to the those described in the RequestArray section. See page 9 for more details.

Urbanization

`ResponseArray.Record[index].Address.Urbanization`

This field returns the value passed to the Urbanization field in the Request Array. See page 13 for more informaion.

Address1

`ResponseArray.Record[index].Address.Address1`

This is the standardized or updated version of the street address in the Address field of the RequestArray. This may include the contents of Address2 if the Address Check process had to swap the two fields to verify the address. This field does not contain any Suite or Private Mail Box information that might have been passed via the Address1 field in the RequestArray. Those are returned via the Suite and PrivateMailBox fields below.

Address2

`ResponseArray.Record[index].Address.Address2`

This is the standardized or updated version of the street address in the Address2 field of the RequestArray. This may include the contents of Address1 if the Address Check process had to swap the two fields to verify the address. This field does not contain any Suite or Private Mail Box information that might have been passed via the Address1 field in the RequestArray. Those are returned via the Suite and PrivateMailBox fields below.

Suite

`ResponseArray.Record[index].Address.Suite`

This field may have been populated by suite information from the Address or Address2 fields of the corresponding record in the RequestArray.

PrivateMailBox

`ResponseArray.Record[index].Address.PrivateMailBox`

This field returns the private mail box number associated with a CMRA (Commercial Mail Receiving Agency).

CMRAs are private businesses that provide a mailing address and “post office” box for their customers.

Mail is delivered by the Postal Service to the CMRA, which then distributes the mail to the customer’s private mail box.

City

```
ResponseArray.Record[index].Address.City.Name
ResponseArray.Record[index].Address.City.Abbreviation
```

If the city name is more than 28 characters long, the Abbreviation will contain the official USPS abbreviation for that city.

State

```
ResponseArray.Record[index].Address.State.Name
ResponseArray.Record[index].Address.State.Abbreviation
```

These fields return the full name of the state in the returned address as well as the standard two-letter abbreviation.

Zip

```
ResponseArray.Record[index].Address.Zip
```

This will contain only the five-digit ZIP Code, even if the full ZIP + 4 was passed via the Zip field in the RequestArray.

Plus4

```
ResponseArray.Record[index].Address.Plus4
```

This field returns the last four digits of a nine-digit ZIP + 4 code.

Carrier Route

```
ResponseArray.Record[index].Address.CarrierRoute
```

The first character of this property is always alphabetic, and the last three characters are numeric. For example, “R001” or “C027” would be typical carrier routes. The alphabetic letter indicates the type of delivery associated with this address.

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

LACSLink

```
ResponseArray.Record[index].Address.LacsLink.LacsStatusCode
ResponseArray.Record[index].Address.LacsLink.LacsReturnCode
```

The LACS^{Link} service flags records that have undergone a conversion to city-style street addresses and automatically updates them. These fields indicate whether this update has taken place and also the level of matching between the current address and the USPS LACS^{Link} database.

LACS Status Code

Returns “Y” if the LACS^{Link} updated the address, “N” otherwise.

LACS Return Code

The return code indicates the level of match between the current address and the LACSLink database.

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

Delivery Point Code

```
ResponseArray.Record[index].Address.DeliveryPointCode
```

The delivery point code contains the 10th and 11th digits of the POSTNET barcode, usually the last two digits of the street number.

Delivery Point Check Digit

```
ResponseArray.Record[index].Address.DeliveryPointCheckDigit
```

The delivery point code contains the 12th digit of the POSTNET barcode.

DPV

DPV enables the Post Office to validate that a submitted address actually corresponds to a real, deliverable address. DPV processing is now required for CASS Certified address checking software like Melissa Data's MAILERS+4 and Address Object.

Footnotes

`ResponseArray.Record[index].Address.DPV.Footnotes`

The DPV footnotes indicate the level of matching between the current address and the USPS's DPV database. The footnote may be up to six characters long, containing a combination of the codes below.

CMRA

`ResponseArray.Record[index].Address.DPV.CMRA`

Returns a "Y" if the address is a mailbox at a Commercial Mail Receiving Agency, such as the UPS store, "N" otherwise. This field will be blank if the DPV Footnote returned is "F1," "G1," or "U1."

Address Status

`ResponseArray.Record[index].Address.DPV.AddressStatus`

Returns the address status description that corresponds to the DPV Footnotes.

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

Country

```
ResponseArray.Record[index].Address.Country.Name
ResponseArray.Record[index].Address.Country.Abbreviation
```

Returns the full name and standard abbreviation of the country in which the address is located. Currently, only Canada and the United States are supported, although SmartMover Web Service will only update the U.S. addresses.

Address Key

```
ResponseArray.Record[index].Address.AddressKey
```

For each record in the Response Array, returns a string value containing a unique key for the current address.

Move and Return Codes

The Move Type and Return Codes indicate the type of address matching between the submitted address and the NCOA^{Link} database.

Move Type

```
ResponseArray.Record[index].Address.Move.Type.Code
ResponseArray.Record[index].Address.Move.Type.Description
```

The Move Type indicates the type of address record that was matched.

Code	Description
F	Family
B	Business
I	Individual

Return Code

```
ResponseArray.Record[index].Address.Move.Return.Code
ResponseArray.Record[index].Address.Move.Return.Description
```

The return code indicates the level of matching between the current record and the NCOA^{Link} database.

Code	Description
A	COA Match - A Business, Individual or Family
00	No COA Match
01	COA Match - Foreign Move
02	COA Match - Moved Left No Address
03	COA Match - PO Box Closed No Forwarding Address
04	Cannot Match COA – Street Address with Secondary
05	COA Match - A New Address cannot be provided
06	Cannot Match COA – Middle Name Conflict
07	Cannot Match COA – Gender Conflict
08	Cannot Match COA – Conflicting Instructions
09	Cannot Match COA – Highrise Default
10	Cannot Match COA – Rural Default
11	Cannot Match COA – Individual Move – Insufficient COA Name
12	Cannot Match COA – Middle Name Test Failed
13	Cannot Match COA – Gender Test Failed
14	COA Match - New Address would not convert
15	Cannot Match COA – Individual Name Insufficient
16	Cannot Match COA – Secondary Number Discrepancy
17	Cannot Match COA – Other Insufficient Name
18	Cannot Match COA – General Delivery
19	Found COA – New Address not ZIP+4 or DPV Confirm
20	Cannot Match COA – Conflicting Directions
66	Daily Delete
91	COA Match - Secondary Number Dropped from COA

Code	Description
92	COA Match - Secondary Number Dropped from Input

Effective Date

This field returns the effective date of the move as a string value in the format "YYYYMM."

```
ResponseArray.Record[index].Address.Move.EffectiveDate
```

Parsed Address Fields

If address parsing is turned on when the RequestArray is submitted, the ResponseArray will return the following fields.

StreetName

`ResponseArray.Record[index].Address.Parsed.StreetName`

Returns just the name portion of the street address, minus the street number, suffix and any directionals.

AddressRange

`ResponseArray.Record[index].Address.Parsed.AddressRange`

Returns just the numeric portion of the street address as a string value.

Suffix

`ResponseArray.Record[index].Address.Parsed.Suffix`

Returns the street name suffix, such as “Rd,” St,” “Blvd,” and so on.

Direction

Pre

`ResponseArray.Record[index].Address.Parsed.Direction.Pre`

Returns any directional abbreviation that comes before the street name. “100 North Main Street” would return “N.”

Post

`ResponseArray.Record[index].Address.Parsed.Direction.Post`

Returns any directional abbreviation that comes after the street name. An address on Park Ave South would return “S.”

Suite

Range

`ResponseArray.Record[index].Address.Parsed.Suite.Range`

Returns only the numeric portion of suite number as a string value.

Name

`ResponseArray.Record[index].Address.Parsed.Suite.Name`

Returns standardized text, such as “STE” or “Unit,” that is part of the suite number.

PrivateMailBox

Range

`ResponseArray.Record[index].Address.Parsed.PrivateMailBox.Range`

If the address is located at a CMRA (Commercial Mail Receiving Agency), the numeric portion is returned here as a string value.

Name

`ResponseArray.Record[index].Address.Parsed.PrivateMailBox.Name`

This field returns the non-numeric portion of a private mailbox number, either “#” or “PMB”

Garbage

`ResponseArray.Record[index].Address.Parsed.Garbage`

This field returns any text that could not be identified as belonging in any of the above fields.

Detailed Response Address Fields

If the OptSmartMoverDetail property is set to TRUE, the SmartMover Web Service returns three addresses: The original submitted address, a standardized version of the original address and the updated move address, if any.

The field names and their usage are identical to those returned without the Detail option turned on. Only the structure is slightly different. For more information on the fields, see the preceding section.

Original

The original address is returned exactly as passed to the RequestArray, so the Original address includes those fields, including the LastLine field.

```
ResponseArray.Record[index].Address.Original.Urbanization
ResponseArray.Record[index].Address.Original.Address1
ResponseArray.Record[index].Address.Original.Address2
ResponseArray.Record[index].Address.Original.Suite
ResponseArray.Record[index].Address.Original.PrivateMailBox
ResponseArray.Record[index].Address.Original.City.Name
ResponseArray.Record[index].Address.Original.City.Abbreviation
ResponseArray.Record[index].Address.Original.State.Name
ResponseArray.Record[index].Address.Original.State.Abbreviation
ResponseArray.Record[index].Address.Original.Zip
ResponseArray.Record[index].Address.Original.Plus4
ResponseArray.Record[index].Address.Original.Country
```

Standardized

The Standardized address is the original address, DPV coded and standardized using USPS addressing standards.

```
...Address.Standardized.Results
...Address.Standardized.Urbanization
...Address.Standardized.Address1
...Address.Standardized.Address2
...Address.Standardized.Suite
...Address.Standardized.PrivateMailBox
...Address.Standardized.City.Name
...Address.Standardized.City.Abbreviation
...Address.Standardized.State.Name
...Address.Standardized.State.Abbreviation
...Address.Standardized.Zip
...Address.Standardized.Plus4
...Address.Standardized.CarrierRoute
...Address.Standardized.Lacs
...Address.Standardized.LacsLink.LacsStatusCode
...Address.Standardized.LacsLink.LacsReturnCode
...Address.Standardized.DeliveryPointCode
```

```
...Address.Standardized.DeliveryPointCheckDigit
...Address.Standardized.DPV.Footnote
...Address.Standardized.DPV.CMRA
...Address.Standardized.DPV.AddressStatus
...Address.Standardized.Type.Address.Code
...Address.Standardized.Type.Address.Description
...Address.Standardized.Type.Zip.Code
...Address.Standardized.Type.Zip.Description
...Address.Standardized.Country.Name
...Address.Standardized.Country.Abbreviation
...Address.Standardized.Parsed.StreetName
...Address.Standardized.Parsed.AddressRange
...Address.Standardized.Parsed.Suffix
...Address.Standardized.Parsed.Direction.Pre
...Address.Standardized.Parsed.Direction.Post
...Address.Standardized.Parsed.Suite.Range
...Address.Standardized.Parsed.Suite.Name
...Address.Standardized.Parsed.PrivateMailBox.Range
...Address.Standardized.Parsed.PrivateMailBox.Name
...Address.Standardized.Parsed.Garbage
```

Moved

The moved address fields are populated with the updated address when there is a change of address match with the NCOA^{Link} database and the address is updated.

```
...Address.MovedTo.Urbanization
...Address.MovedTo.Address1
...Address.MovedTo.Address2
...Address.MovedTo.Suite
...Address.MovedTo.PrivateMailBox
...Address.MovedTo.City.Name
...Address.MovedTo.City.Abbreviation
...Address.MovedTo.State.Name
...Address.MovedTo.State.Abbreviation
...Address.MovedTo.Zip
...Address.MovedTo.Plus4
...Address.MovedTo.CarrierRoute
...Address.MovedTo.Lacs
...Address.MovedTo.LacsLink.LacsStatusCode
...Address.MovedTo.LacsLink.LacsReturnCode
...Address.MovedTo.DeliveryPointCode
...Address.MovedTo.DeliveryPointCheckDigit
...Address.MovedTo.DPV.Footnote
...Address.MovedTo.DPV.CMRA
...Address.MovedTo.DPV.AddressStatus
...Address.MovedTo.Type.Address.Code
...Address.MovedTo.Type.Address.Description
...Address.MovedTo.Type.Zip.Code
...Address.MovedTo.Type.Zip.Description
```

```
...Address.MovedTo.Country.Name
...Address.MovedTo.Country.Abbreviation
...Address.MovedTo.Parsed.StreetName
...Address.MovedTo.Parsed.AddressRange
...Address.MovedTo.Parsed.Suffix
...Address.MovedTo.Parsed.Direction.Pre
...Address.MovedTo.Parsed.Direction.Post
...Address.MovedTo.Parsed.Suite.Range
...Address.MovedTo.Parsed.Suite.Name
...Address.MovedTo.Parsed.PrivateMailBox.Range
...Address.MovedTo.Parsed.PrivateMailBox.Name
...Address.MovedTo.Parsed.Garbage
```

ResponseArray Name Fields

The SmartMover Web Service uses the accompanying name data to match records for NCOA^{Link} updates. If the full name is submitted as a single string, the Web Service parses it before processing and returns the parsed name. We will also attempt to gender-match the name before returning it.

Name Result

The Name Result returns any status and error codes caused by processing the current name and address. The values returned by these fields will be identical to what is returned by Address.Result.

Error

```
ResponseArray.Record[index].Name.Result.Error.Code
ResponseArray.Record[index].Name.Result.Error.Description
```

These fields contain the code and description for any error generated by the current record.

For a list of possible values, see the table on page 32.

Status

```
ResponseArray.Record[index].Name.Result.Status.Code
ResponseArray.Record[index].Name.Result.Status.Description
```

These fields contain the code and description for the status of the returned name and address record. Generally this code indicates whether or not the record was updated with move data or just standardized.

For a list of possible values, see the table on page 33.

Name Fields

Most of the name fields returned by the ResponseArray correspond exactly to the name fields submitted to the RequestArray. For more information on the specific fields, see page 9.

Full

```
ResponseArray.Record[index].Name.Full
```

Returns the full name as submitted to the FullName field of a record in the RequestArray.

First

```
ResponseArray.Record[index].Name.First
```

Returns the first name as submitted to the FirstName field of a record in the RequestArray.

Middle

`ResponseArray.Record[index].Name.Middle`

Returns the middle name as submitted to the MiddleName field of a record in the RequestArray.

Last

`ResponseArray.Record[index].Name.Last`

Returns the last name as submitted to the LastName field of a record in the RequestArray.

Prefix

`ResponseArray.Record[index].Name.Prefix`

Returns the name prefix as submitted to the NamePrefix field of a record in the RequestArray.

Suffix

`ResponseArray.Record[index].Name.Suffix`

Returns the name suffix as submitted to the NameSuffix field of a record in the RequestArray.

Get Summary Report Link Service

The Get Summary Report Link Service will return URL links to web pages that contain the CASS 3553 and NCOA^{Link} forms based on the customer ID and JobID.

You can also request the information used to generate the following report. See “Get CASS Summary Service” on page 55 and “Get NCOALink Summary Report Service” on page 61.

Using the Summary Report Link Service

Getting a Summary Report Link from the SmartMover is simple process.

1. **Create SmartMover Object:**

```
SmartMoverWS.SmartMover sm = new SmartMoverWS.SmartMover();
```

2. **Create a RespNCOASummaryReport Object and call the GetSummaryReportLink method, passing the customer ID number and JobID as shown.**

```
SmartMoverWS.RespNCOALinkReportLink SummaryLinks =
    sm.GetSummaryReportLink (customerID, JobID);
```

RespNCOASummaryReport Object properties

The GetSummaryReportLink method populates the following fields in the RespNCOA-SummaryReport Object, named SummaryLinks in the example above.

Fault Fields

If there was any error in calling GetSummaryReportLink, the web service will return a code indicating the cause and source of the error. For more information on these fields and a list of the possible values for these fields, see page 18.

```
SummaryLinks.Fault.Code
SummaryLinks.Fault.Description
SummaryLinks.Fault.Source
SummaryLinks.Fault.Detail
```

Summary Report Link Fields

These fields return string values containing the URLs of the CASS report and NCOA^{Link} summary.

```
SummaryLinks.SummaryReport.NCOALink
SummaryLinks.SummaryReport.CASS
```

Get CASS Summary Service

The Get CASS Summary Report Service returns all of the information from processing your list that would be required to fill out a CASS form 3553 form. To use this service, simply call GetCASS_SummaryReport and pass in your Customer ID and JobID. The Customer ID will identify who you are and the JobID will identify which processed list you are retrieving. This service will return a RespCASSReport, which contains all the CASS information for your list in individual fields.

The SmartMover Web Service also provides GetReportSummaryLink, which generates a link to the completed CASS form. You can use this service, however, if you wish to track the information described below.

Using the Get CASS Summary Service

Getting a CASS Summary from the SmartMover is simple process.

1. **Create SmartMover Object:**

```
SmartMoverWS.SmartMover sm = new SmartMoverWS.SmartMover();
```

2. **Create A CASSReport Object and call the GetCASS_SummaryReport method, passing the customer ID number and JobID as shown.**

```
SmartMoverWS.RespCASSReport CASSreport =
    sm.GetCASS_SummaryReport(customerID, JobID);
```

RespCASSReport Object properties

The GetCASS_SummaryReport method populates the following fields in the RespCASSReport Object, named CASSreport in the example above.

Fault Fields

If there was any error in calling GetCASS_SummaryReport, the web service will return a code indicating the cause and source of the error. For more information on these fields and a list of the possible values for these fields, see page 18.

```
CASSReport.Fault.Code
CASSReport.Fault.Description
CASSReport.Fault.Source
CASSReport.Fault.Detail
```

CASS Software Information

These fields return the information from section A of CASS Form 3553, which includes the publisher name, software name, version number and configuration of the software used to address check the submitted addresses. For the SmartMover Web Service, this will always be Melissa Data's Address Object, which supports all of our web services.

CertifiedCompany

```
CASSReport.CASS.Software.CertifiedCompany
```

This will always return "Melissa Data."

CASSVersion

```
CASSReport.CASS.Software.CassVersion
```

This will return the name and version number of the current Address Object being used by the SmartMover Web Service.

Configuration

```
CASSReport.CASS.Software.Configuration
```

This will normally return the string "STD" for "standard." Address Object has only one configuration.

CASS List Information

These fields return information for section B of CASS Form 3553, which includes the name of the list, number of records, date of processing and the identity of the list owner.

ProcessorName

`CASSReport.CASS.List.ProcessorName`

This field will be blank because you will need to provide your own name on the CASS form.

CurrentDate

`CASSReport.CASS.List.CurrentDate`

This field returns a string value containing the date on which the list or lists were processed.

DBDate

`CASSReport.CASS.List.DBDate`

This field returns a string value containing the revision date of the database which was used to verify the addresses.

ListName

`CASSReport.CASS.List.ListName`

This field returns the name of list as it was passed to the RequestArray object when the list was submitted to the web service.

NumLists

`CASSReport.CASS.List.NumLists`

This field returns a string value displaying the number of mailing lists covered by the current CASS form. This value should always be “1.”

TotalRecCount

`CASSReport.CASS.List.TotalRecCount`

This field returns a string showing the number of records processed by the SmartMover Web Service under the current JobID.

CASS Output Fields

These fields return the information necessary to fill out section C of CASS Form 3553, which presents the total number of records verified or coded under certain categories and the range of dates during which those records can be used with this CASS form before they have to be verified again.

Zip4Count

`CASSReport.CASS.Output.Zip4Count`

This field returns a long integer value containing the number of records that were DPV coded by the Web Service.

Zip4Period

`CASSReport.CASS.Output.Zip4Period`

This field returns a string value showing the range of dates during which the above records can be used for mailing under this CASS form without being DPV verified again.

FiveDigitCount

`CASSReport.CASS.Output.FiveDigitCount`

This field returns a long integer value indicating the number of records that were successfully coded to the five-digit ZIP Code level.

FiveDigitPeriod

`CASSReport.CASS.Output.FiveDigitPeriod`

This field returns a string value showing the range of dates during which the above records can be used for mailing under this CASS form without being coded again.

CRRTCount

`CASSReport.CASS.Output.CRRTCount`

This field returns a long integer value indicating the number of records that were successfully coded to the Carrier Route level.

CRRTPeriod

`CASSReport.CASS.Output.CRRTPeriod`

This field returns a string value showing the range of dates during which the above records can be used for mailing under this CASS form without being coded again.

LotCount

`CASSReport.CASS.Output.LotCount`

This field returns a long integer value indicating the number of records that were successfully eLot coded.

LotPeriod

`CASSReport.CASS.Output.LotPeriod`

This field returns a string value showing the range of dates during which the above records can be used for mailing under this CASS form without being coded again.

CASS Mailer Fields

These fields return a series of string values containing the contact information for the party who will be using this list for conducting a mailing. This is used for Section D of CASS Form 3553.

```
CASSReport.CASS.Mailer.MailingName
CASSReport.CASS.Mailer.MailingCompany
CASSReport.CASS.Mailer.MailingAddress
CASSReport.CASS.Mailer.MailingCity
```

```
CASSReport.CASS.Mailer.MailingState
CASSReport.CASS.Mailer.MailingZIP
```

CASS QSS Fields

The Qualitative Statistical Summary fields return long integer values containing counts of the number of the fields that fall under certain statistical categories.

```
CASSReport.CASS.QSS.HRDefault
CASSReport.CASS.QSS.HRExact
CASSReport.CASS.QSS.RRDefault
CASSReport.CASS.QSS.RRExact
CASSReport.CASS.QSS.LACS
CASSReport.CASS.QSS.EWS
CASSReport.CASS.QSS.DPV
CASSReport.CASS.QSS.RDI
CASSReport.CASS.QSS.SuiteLink
```

Get NCOA^{Link} Summary Report Service

The Get NCOA^{Link} Summary Report Service returns all of the information from processing your list that would be required to fill out a summary report. To use this service, simply call GetNCOALink_SummaryReport and pass in your Customer ID and JobID. The Customer ID will identify who you are and the JobID will identify which processed list you are retrieving. This service will return a RespCASSReport, which contains all the CASS information for your list in individual fields.

The SmartMover Web Service also provides GetReportSummaryLink, which generates a link to the completed summary report. You can use this service, however, if you wish to track the information described below.

Using the NCOA^{Link} Summary Service

Getting a NCOA^{Link} Summary from the Smart Mover is simple process.

1. **Create SmartMover Object:**

```
SmartMoverWS.SmartMover sm = new SmartMoverWS.SmartMover();
```

2. **Create A CASSReport Object and call the GetCASS_SummaryReport method, passing the customer ID number and JobIDshown.**

```
SmartMoverWS.RespNCOALinkReportLink NCOAReport=
    sm.GetNCOALink_SummaryReport(customerID, job);
```

RespNCOALinkReportLink Object properties

The GetNCOALink_SummaryReport method populates the following fields in the RespNCOALinkReportLink Object, named NCOAReport in the example above.

Fault Fields

If there was any error in calling GetNCOALink_SummaryReport, the web service will return a code indicating the cause and source of the error. For more information on these fields and a list of the possible values for these fields, see page 18.

```
NCOAReport.Fault.Code
NCOAReport.Fault.Description
NCOAReport.Fault.Source
NCOAReport.Fault.Detail
```

NCOA^{Link} Customer Info Fields

These fields return string values containing the contact information for the owner of the mailing list submitted to the SmartMover Web Service.

```
NCOAReport.NCOALink.Customer_Info.PAF_Cutomer_ID
NCOAReport.NCOALink.Customer_Info.Title
NCOAReport.NCOALink.Customer_Info.PAF_Name
NCOAReport.NCOALink.Customer_Info.Company_Name
NCOAReport.NCOALink.Customer_Info.List_Name
NCOAReport.NCOALink.Customer_Info.Phone
```

NCOA^{Link} Processing Options Fields

Theses field return string values containing the processing options used by the SmartMover Web Service when processing the list submitted under the current JobID.

```
NCOAReport.NCOALink.Processing_Options.Processing_Category
NCOAReport.NCOALink.Processing_Options.Class_Of_Mail
NCOAReport.NCOALink.Processing_Options.Date_NCOALink_Began
NCOAReport.NCOALink.Processing_Options.Date_List_Returned
NCOAReport.NCOALink.Processing_Options.Pre_Process
NCOAReport.NCOALink.Processing_Options.Concurrent_Processes
NCOAReport.NCOALink.Processing_Options.Post_Processing
NCOAReport.NCOALink.Processing_Options.Obtaining_final_results
NCOAReport.NCOALink.Processing_Options.Standard_output_Returned_flag
NCOAReport.NCOALink.Processing_Options.Matching_Logic_Applied_flag
NCOAReport.NCOALink.Processing_Options.Data_Returned_flag
```

NCOA^Link Results Fields

These fields return string values containing counts of the records processed and the number matched to ZIP + 4, DPV and NCOA^Link databases.

```
NCOAReport.NCOALink.Results.Number_of_Records_Proccessed
NCOAReport.NCOALink.Results.Records_Matched
NCOAReport.NCOALink.Results.Records_ZIP4_Coded
NCOAReport.NCOALink.Results.Records_DPV_Confirmed
NCOAReport.NCOALink.Results.Total_Moved
```

NCOA^Link Move Results Fields

These fields return string value containing counts of the number of moved records that fall under each type of movie, Individual, Family, and Business.

```
NCOAReport.NCOALink.Move_Results.Individual_Move
NCOAReport.NCOALink.Results.Family_Move
NCOAReport.NCOALink.Results.Business_Move
```

NCOA^Link Return Codes Fields

These fields return string values counting the number of records that fall under each NCOALink Return Code.

```
NCOAReport.NCOALink.Return_Codes.codeA
NCOAReport.NCOALink.Return_Codes.code00
NCOAReport.NCOALink.Return_Codes.code01
NCOAReport.NCOALink.Return_Codes.code02
NCOAReport.NCOALink.Return_Codes.code03
NCOAReport.NCOALink.Return_Codes.code04
NCOAReport.NCOALink.Return_Codes.code05
NCOAReport.NCOALink.Return_Codes.code06
NCOAReport.NCOALink.Return_Codes.code07
NCOAReport.NCOALink.Return_Codes.code08
NCOAReport.NCOALink.Return_Codes.code09
NCOAReport.NCOALink.Return_Codes.code10
NCOAReport.NCOALink.Return_Codes.code11
NCOAReport.NCOALink.Return_Codes.code12
NCOAReport.NCOALink.Return_Codes.code13
NCOAReport.NCOALink.Return_Codes.code14
NCOAReport.NCOALink.Return_Codes.code15
NCOAReport.NCOALink.Return_Codes.code16
NCOAReport.NCOALink.Return_Codes.code17
NCOAReport.NCOALink.Return_Codes.code18
NCOAReport.NCOALink.Return_Codes.code19
NCOAReport.NCOALink.Return_Codes.code20
NCOAReport.NCOALink.Return_Codes.code66
NCOAReport.NCOALink.Return_Codes.code77
NCOAReport.NCOALink.Return_Codes.code91
NCOAReport.NCOALink.Return_Codes.code92
```

For an explanation of these codes, see the table below.

Code	Definition
A	COA Match - The input record matched to a business, individual or family type master file record. A new address could be furnished.
00	No Match - The input record COULD NOT BE matched to a master file record. A new address could not be furnished.
01	Found COA: Foreign Move - The input record matched to a business, individual or family type master file record but the new address was outside USPS delivery area.
02	Found COA: Moved Left No Address (MLNA) - The input record matched to a business, individual or family type master file record and the new address was not provided to USPS.
03	Found COA: Box Closed No Order (BCNO) - The Input record matched to a business, individual or family type master file record which contains an old address of PO BOX that has been closed without a forwarding address provided.
04	Cannot match COA: Street Address with Secondary - In the STANDARD mode utilizing Family matching logic the input record matched to a family record type on the master file with an old address that contained secondary information which obtained a ZIP+4 street level match. The input record does not contain secondary information. This address match situation requires individual name matching logic to obtain a match and individual names do not match.
05	Found COA: New 11-digit DPBC is Ambiguous - The input record matched to a business, individual or family type master file record. The new address on the master file record could not be converted to a deliverable address because the DPBC represents more than one delivery point.
06	Cannot Match COA: Conflicting Directions: Middle Name Related - There is more than one COA (individual or family type) record for the match algorithm and the middle names or initials on the COAs are different. Therefore, a single match result could not be determined.
07	Cannot Match COA: Conflicting Directions: Gender Related -There is more than one COA (individual or family type) record for the match algorithm and the genders of the names on the COAs are different. Therefore, a single match result could not be determined.
08	Cannot Match COA: Other Conflicting Instructions - The input record matched to two master file (business, individual or family type) records. The two records in the master file were compared and due to differences in the new addresses, a match could not be made.

Code	Definition
09	Cannot Match COA: High-rise Default - The input record matched to a family record on the master file from a High-rise address ZIP+4 coded to the building default. This address match situation requires individual name matching logic to obtain a match and individual names do not match.
10	Cannot Match COA: Rural Default - The input record matched to a family record on the master file from a Rural Route or Highway Contract Route address ZIP+4 coded to the route default. This address situation requires individual name matching logic to obtain a match and individual names do not match.
11	Cannot Match COA: Individual Match: Insufficient COA Name for Match - There is a master file (individual or family type) record with the same surname and address but there is insufficient name information on the master file record to produce a match using individual matching logic.
12	Cannot Match COA: Middle Name Test Failed - The input record matched to an individual or family record on the master file with the same address and surname. However, a match cannot be made because the input name contains a conflict with the middle name or initials on the master file record.
13	Cannot Match COA: Gender Test Failed - The input record matched to a master file (individual or family type) record. A match cannot be made because the gender of the name on the input record conflicts with the gender of the name on the master file record.
14	Found COA: New Address Would Not Convert at Run Time - The input record matched to a master file (business, individual or family type) record. The new address could not be converted to a deliverable address.
15	Cannot Match COA: Individual Name Insufficient - There is a master file record with the same address and surname. A match cannot be made because the input record does not contain a first name or contains initials only.
16	Cannot Match COA: Secondary Number Discrepancy - The input record matched to a street level individual or family type record. However, a match is prohibited based on I of the following reasons: 1) There is conflicting secondary information on the input and master file record; 2) the input record contained secondary information and matched to a family record that does not contain secondary information. In item 2, this address match situation requires individual name matching logic to obtain a COA match and individual names do not match.
17	Cannot Match COA: Other Insufficient Name - The input record matched to an individual or family master file record. The input name is different or not sufficient enough to produce a match.

Code	Definition
18	Cannot Match COA: General Delivery - The input record matched to a family record on the master file from a General Delivery address. This address situation requires individual name matching logic to obtain a match and individual names do not match.
19	Found COA: New Address not ZIP+4 coded - There is a change of address on file but the new address cannot be ZIP+4 coded and therefore there is no 11?digit DPBC to store or return.
20	Cannot Match COA: Conflicting Directions after re-chaining - Multiple master file records were potential matches for the input record. The master file records contained different new addresses and a single match result could not be determined.
66	Daily Delete - The input record matched to a business, individual or family type master file record with an old address that is present in the daily delete file. The presence of an address in the daily delete file means that a COA with this address is pending deletion from the master file and that no mail may be forwarded from this address.
91	COA Match: Secondary Number dropped from COA – The input record matched to a master file record. The master file record had a secondary number and the input address did not. Please Note: This return code is derived from Individual matching logic only. If this return code is achieved then no other matching attempts are permitted regardless of the PROCESSING mode.
92	COA Match: Secondary Number Dropped from input address – The input record matched to a master file record, but the input address had a secondary number and the master file record did not. The record is a ZIP + 4 street level match. Please Note: This return code is derived from individual matching logic only. If this return code is achieved then no other matching attempts are permitted regardless of the PROCESSING mode.