ANCHOR

COMPUTER SOFTWARE

AddressPro REST Web Service

User Guide

Table of Contents

Chapter 1: AddressPro Web Service

Web services extend the World Wide Web infrastructure to provide a means for software to connect to other software applications. Applications access Web services via web protocols and data formats such as HTTP, JSON, and REST, without needing to know how each Web service is implemented.

Unlike SOAP-based Web services, there is no "official" standard for RESTful Web APIs. This is because REST is an architectural style, while SOAP is a protocol. REST is not a standard in itself, but RESTful implementations make use of standards, such as HTTP, URI, JSON, and XML.

Anchor Software's REST based Web service provides the end user the ability to have multiple users and/or systems perform multiple, simultaneous address verification lookups. Users can create an internal company website, or a stand-alone application in any platform and language of their choice to retrieve this address-lookup information. AddressPro supports HTTP and 128-bit encryption HTTPS connectivity.

About AddressPro

Anchor Software's AddressPro system uses the Anchor Coder address-processing engine to perform all USA address-related operation requests,

Client applications on any platform and development language can access AddressPro to perform address lookups and other address-related operations. Sample code to access AddressPro is available in the Appendix section.

Available Operations

These are the operations currently available in Anchor's AddressPro REST/ JSON interface. All operations are implemented with the POST HTTP/ HTTPS method.

- StandardAddress
- CityState
- ZIPCode
- PartialStreet
- AddressLines
- ParsedName
- CasedAddress
- CasedString

StandardAddress

The **StandardAddress** operation returns details for a full address. At most, only one output address will be returned for the given input. If more than one output address is available (i.e. the Coder address-processing engine was not able to choose from a list of multiple candidates that equally match the input), then the **ReturnCode** field will contain **22**. See the **Output Field Descriptions** section of this document for more information on the **ReturnCode** values. The **OutputCasing** parameter can be set to **Upper**, **Lower**, or **Mixed**.

The **StandardAddress** operation uses the following input parameters:

```
{
    "AuthenticationKey": "string",
    "Firm": "string",
    "AddressLine1": "string",
    "AddressLine2": "string",
    "City": "string",
    "State": "string",
    "ZIPCode": "string",
    "Addon": "string",
    "PRUrb": "string",
    "Latitude": "string",
    "Longitude": "string",
    "EnableGeocoding": 0,
    "MaxResults": 0,
    "OutputCasing": "string",
    "ReferenceID": "string"
}
```

The following is the output data structure:

```
{
    "ServiceResult": 0,
    "OperationResult": 0,
    "ErrorDescription": "string",
    "ReferenceID": "string",
    "OutputAddress": {
        "ReturnCode": 0,
        "Firm": "string",
        "AddressLine1": "string",
        "AddressLine2": "string",
        "LastLine": "string",
        "City": "string",
        "State": "string",
        "ZIPCode": "string",
        "Addon": "string",
        "PRUrb": "string",
        "CityAbbreviation": "string",
        "CitySpelled": "string",
```

```
        "CarrierRouteID": "string",
        "DeliveryPoint": "string",
        "CheckDigit": "string",
        "RecordType": "string",
        "StdPriNum": "string",
        "StdPreDir": "string",
        "StdStreetName": "string",
        "StdSuffix": "string",
        "StdPostDir": "string",
        "StdUnit": "string",
        "StdSecNum": "string",
        "StdUnit2": "string",
        "StdSecNum2": "string",
        "PMBDesignator": "string",
        "PMBNumber": "string",
        "MSCDesignator": "string",
        "MSCNumber": "string",
        "Finance": "string",
        "CongressionalDistrict": "string",
        "UniqueZIP": "string",
        "ZIP4CountyNumber": "string",
        "ZIPCountyNumber": "string",
        "ZIPCountyName": "string",
        "Footnotes": "string",
        "DPVA": "string",
        "DPVCMRA": "string",
        "DPVNoStat": "string",
        "DPVVacant": "string",
        "DPVDNA": "string",
        "SuiteLinkFootnote": "string",
        "LACSLinkFootnote": "string",
        "LACSLinkNewAddress": "string",
        "RDI": "string",
        "ProductDate": "string",
        "Latitude": "string",
        "Longitude": "string",
        "GeocodeLevel": "string",
        "CensusTract": "string",
        "CensusBlock": "string",
        "PMSA": "string",
        "MSA": "string",
        "TimeZone": "string",
        "CBSACode": "string",
        "CBSALevel": "string",
        "CBSADivisionCode": "string",
        "CBSADivisionLevel": "string"
    }
```

CityState

The **CityState** operation returns details for an input city and state. Many city names contain more than one ZIP Code. The **RecordCount** field indicates the number of ZIP Codes that are returned for the given city name. A city/ state record, containing data from the USPS City/ State database, is returned for each of these ZIP Codes. The **CityState** operation uses the following input parameters:

```
{
    "AuthenticationKey": "string",
    "City": "string",
    "State": "string",
    "MaxResults": 0,
    "OutputCasing": "string",
    "ReferenceID": "string"
}
```

The following is the output data structure:

> **Note:** For a list of output field descriptions, see the CityState section in Chapter 2.

```
{
    "ServiceResult": 0,
    "OperationResult": 0,
    "ErrorDescription": "string",
    "ReferenceID": "string",
    "Records": {
        "RecordCount": 0,
        "Record": {
            "ZIPCode": "string",
            "CityStateKey": "string",
            "ZIPClassCode": "string",
            "CityName": "string",
            "CityNameAbbrev": "string",
            "FacilityCode": "string",
            "MailingName": "string",
            "PrefCityName": "string",
            "PrefCityStateKey": "string",
            "UniqueZIP": "string",
            "Finance": "string",
            "State": "string",
            "CountyNumber": "string",
            "CountyName": "string"
        }
    }
}
```

ZIPCode

The **ZIPCode** operation returns details for an input ZIP Code. Many ZIP Codes contain more than one city name. The **RecordCount** field in the output data indicates the number of city names that are returned for the given ZIP Code. A city/ state record, containing data from the USPS City/ State database, is returned for each of these city names.

The **ZIPCode** operation uses the following input parameters:

```
{
    "AuthenticationKey": "string",
    "ZIPCode": "string",
    "MaxResults": 0,
    "OutputCasing": "string",
    "ReferenceID": "string"
}
```

The following is the output data structure for the **ZIPCode** operation:

Note: For a list of output field descriptions, see the ZIPCode section in Chapter 2.

```
{
    "ServiceResult": 0,
    "OperationResult": 0,
    "ErrorDescription": "string",
    "ReferenceID": "string",
    "Records": {
        "RecordCount": 0,
        "Record": {
            "ZIPCode": "string",
            "CityStateKey": "string",
            "ZIPClassCode": "string",
            "CityName": "string",
            "CityNameAbbrev": "string",
            "FacilityCode": "string",
            "MailingName": "string",
            "PrefCityName": "string",
            "PrefCityStateKey": "string",
            "UniqueZIP": "string",
            "Finance": "string",
            "State": "string",
            "CountyNumber": "string",
            "CountyName": "string"
        }
    }
}
```

PartialStreet

The **PartialStreet** operation returns a list of candidate street names that match partial street information provided on input. The input to this operation must include a ZIP Code, primary number, and at least one letter for the street name. The user can provide the street information from either a single address line in the input **AddressLine** field (e.g. "123 N M"), or as pre-parsed elements in the input **PriNum**, **PreDir**, and **StreetName** fields (e.g. "123", "N", "M", respectively).

Using Anchor's Address Verifier library, AddressPro will find all of the street names within the input ZIP Code that begin with the partial name provided, and that are also valid for the given input primary number. If a pre-directional value is given, then the list of output street names will also be restricted to those matching the pre-directional value.

Use this operation to reduce the number of keystrokes in a data-entry scenario. Assuming the data-entry operator obtains the ZIP Code of an address first, and then as they are entering the street information, the data-entry application can make calls to **PartialStreet** and receive back a list of candidate streets, possibly even a single street, so that the operator can quickly choose the final street name without having to fully type it in. This can also reduce the likelihood of spelling mistakes during the data-entry process.

The **PartialStreet** operation uses the following input parameters:

```
{
    "AuthenticationKey": "string",
    "AddressLine": "string",
    "PriNum": "string",
    "PreDir": "string",
    "StreetName": "string",
    "Suffix": "string",
    "PostDir": "string",
    "ZIPCode": "string",
    "ReferenceID": "string"
}
```

The following is the output data structure:

Note: For a list of output field descriptions, see the PartialStreet section in Chapter 2.

```
{
    "ServiceResult": 0,
    "OperationResult": 0,
    "ErrorDescription": "string",
    "StdPriNum": "string",
    "ReferenceID": "string",
    "Records": {
        "RecordCount": 0,
        "Record": {
            "StdPreDir": "string",
            "StdStreetName": "string",
            "StdSuffix": "string",
            "StdPostDir": "string"
        }
    }
}
```

AddressLines

The **AddressLines** operation identifies the specific fields of an arbitrary list of input data lines. The specific fields identified are address, name, city, state, ZIP Code, company name and more. The operation returns the various address pieces in known locations for the user to manipulate. In order to perform a request for the above operations the **AddressLines** operation will use the following input parameters:

```
{
    "AuthenticationKey": "string",
    "Line1": "string",
    "Line2": "string",
    "Line3": "string",
    "Line4": "string",
    "Line5": "string",
    "Line6": "string",
    "Line7": "string",
    "Line8": "string",
    "Line9": "string",
    "RemoveCreditCardInfo": 0,
    "ReferenceID": "string"
}
```

The following is the output data structure:


```
{
    "ServiceResult": 0,
    "OperationResult": 0,
    "ErrorDescription": "string",
    "ReferenceID": "string",
    "Output": {
        "Name1": "string",
        "Name2": "string",
        "Name3": "string",
        "Company1": "string",
        "Company2": "string",
        "Company3": "string",
        "Company4": "string",
        "Address1": "string",
        "Address2": "string",
        "Address3": "string",
        "Address4": "string",
        "City": "string",
        "State": "string",
        "Zip": "string",
        "Zip4": "string",
        "CityStateZipCombined": "string",
        "URBData": "string",
        "Email": "string",
        "Web": "string",
        "CreditCard": "string",
        "StatusMessage": "string",
        "StatusCode": "string"
    }
}
```

ParsedName

The **ParsedName** operation parses a full name and returns individual name fields including title, first name, middle name, last name, suffix, and professional suffix. The following is a sample input:

```
{
    "AuthenticationKey": "string",
    "FullNameInput": "string",
    "OutputCasing": "string",
    "ReferenceID": "string"
}
```

The following is the output data structure for the **ParsedName** operation:

```
{
    "ServiceResult": 0,
    "OperationResult": 0,
    "ErrorDescription": "string",
    "ReferenceID": "string",
    "Records": {
        "RecordCount": 0,
        "Record": {
            "Title": "string",
            "FirstName": "string",
            "MiddleName": "string",
            "LastName": "string",
            "SuffixGen": "string",
            "Company": "string",
            "SuffixProf": "string",
            "SuffixTotal": "string"
        }
    }
}
```

CasedAddress

The **CasedAddress** operation performs casing on an input address. The output casing options are **Upper**, **Lower**, or **Mixed**. This feature does not perform address standardization. The following is a sample input:

```
{
    "AuthenticationKey": "string",
    "Firm": "string",
    "AddressLine1": "string",
    "AddressLine2": "string",
    "City": "string",
    "State": "string",
    "ZIPCode": "string",
    "Addon": "string",
    "PRUrb": "string",
    "OutputCasing": "string",
    "ReferenceID": "string"
}
```

The following is the output data structure for the **CasedAddress** operation:

```
{
    "ServiceResult": 0,
    "OperationResult": 0,
    "ErrorDescription": "string",
    "ReferenceID": "string",
    "OutputAddress": {
        "AuthenticationKey": "string",
        "Firm": "string",
        "AddressLine1": "string",
        "AddressLine2": "string",
        "City": "string",
        "State": "string",
        "ZIPCode": "string",
        "Addon": "string",
        "PRUrb": "string",
        "OutputCasing": "string",
        "ReferenceID": "string"
    }
}
```

CasedString

The **CasedString** operation cases a single input string. The output casing options are upper **Upper**, **Lower**, or **Mixed**. This feature does not perform address standardization. The following is a sample input:

```
{
    "AuthenticationKey": "string",
    "String": "string",
    "OutputCasing": "string",
    "ReferenceID": "string"
}
```

The following is the output data structure for the **CasedString** operation:

```
{
    "ServiceResult": 0,
    "OperationResult": 0,
    "ErrorDescription": "string",
    "ReferenceID": "string",
    "Output": "string"
}
```

Chapter 2: Output Field Descriptions

This chapter provides a list of output field descriptions for the following operations:

StandardAddress

Field Name	Description
ServiceResult	This is the result code from the service request. A **0** in this field indicates no error.
OperationResult	This is the result code from the call to the Coder's address-processing function. A **0** in this field indicates no error. If a processing error occurs, this field will contain a non-zero error code.
ErrorDescription	A text description of an error code reported in the **service-result** or **lookup-result** output fields.
ReturnCode	This is the address-processing result code, and contains the basic match level of the input address to the USPS databases. For more details about this code, see the *Address Processing Return Codes* section in the Appendix.
Firm	This is the standardized firm name. If the input address does not match to the Coder database, then this field will contain a copy of the input information.
AddressLine1	This is the standardized address-line 1. If the input address does not match to the Coder database, then this field will contain a copy of the input information.
AddressLine2	This is the standardized address-line 2. If the input address does not match to the Coder database, then this field will contain a copy of the input information.

Field Name	Description
LastLine	This is the standardized last line. This includes the city, state, ZIP Code, and addon code. If the input address does not match to the Coder database, then this field will contain a copy of the input information.
City	This is the standardized city name. If the input address does not match to the Coder database, then this field will contain a copy of the input information.
State	This is the standardized two-character state code. If the input address does not match to the Coder database, then this field will contain a copy of the input information.
ZIPCode	This is the standardized ZIP Code. If the input address does not match to the Coder database, then this field will contain a copy of the input information.
Addon	This is the standardized addon code. If the input address does not match to the Coder database, then this field will contain a copy of the input information.
PRUrb	This is the standardized Puerto Rico urbanization name. If the input address does not match to the Coder database, then this field will contain a copy of the input information.
CityAbbreviation	This is the abbreviated form of the standardized city name, if one exists. City names that are longer than 13 characters will have an abbreviated name.
CitySpelled	This is the spelled-out form of the standardized city name.
CarrierRouteID	This is the USPS Carrier Route ID.
DeliveryPoint	This is the two-digit Delivery Point code. The Delivery Point code, when appended to the nine-digit ZIP+4 code, comprises the complete 11-digit USPS Delivery Point Barcode.
CheckDigit	This is the check digit for the 11-digit Delivery Point Barcode.
RecordType	This is the USPS ZIP+4 record-type field that comes from the ZIP+4 record that was matched to. Possible values are: • **S** – Street record • **H** – High-rise record • **F** – Firm record • **R** – Rural Route (or Highway Contract) record • **P** – PO Box record • **G** – General Delivery record

Field Name	Description
StdPriNum	This is the primary number from the standardized address.
StdPreDir	This is the street pre-directional from the standardized address.
StdStreetName	This is the street name from the standardized address.
StdSuffix	This is the street suffix from the standardized address.
StdPostDir	This is the street post-directional from the standardized address.
StdUnit	This is the unit designator (e.g. APT, STE) from the standardized address.
StdSecNum	This is the secondary number from the standardized address.
StdUnit2	This is the second unit designator from the standardized address.
StdSecNum2	This is the second secondary number from the standardized address.
PMBDesignator	This is the Private Mail Box (PMB) designator from the standardized address.
PMBNumber	This is the Private Mail Box (PMB) number from the standardized address.
MSCDesignator	This is the Mail Stop Code (MSC) designator from the standardized address.
MSCNumber	This is the Mail Stop Code (MSC) number from the standardized address.
Finance	This is the USPS Finance number for the standardized address.
CongressionalDistrict	This is the US Congressional District for the standardized address.
UniqueZIP	A **Y** in this field indicates that the standardized ZIP Code is a Unique ZIP Code.
ZIP4CountyNumber	This is the county number from the USPS ZIP+4 file for the standardized address. This county number reflects the physical county where the address resides.
ZIPCountyNumber	This is the county number for the ZIP Code from the standardized address. This county number is for the county where the delivering post office resides, which may not necessarily be the county where the standardized address is located.

Field Name	Description
ZIPCountyName	This is the county name for the ZIP Code from the standardized address. This county name is for the county where the delivering post office resides, which may not necessarily be the county where the standardized address is located.
Footnotes	This field contains the Coder footnote codes for the standardized address. Each code is two characters long and gives details about the quality of the address match that was made. Multiple codes will appear in this field as a continuous string of characters. For more details about this code, see the *Footnote Codes* section in the Appendix.
DPVA	This is the USPS DPV **A** flag, which represents the delivery-point-validation level of the standardized address. The possible values are: • **Y** – All standardized address components were delivery-point validated. • **S** – The primary number was validated, but the secondary number was not. • **D** – The primary number was validated, and the DPV system determined that the input address is missing a secondary number. • **N** – None of the standardized address components were delivery-point validated. • **Blank** – The DPV feature is turned off or the DPV lookup was not attempted.
DPVCMRA	This is the USPS DPV **CMRA** flag. A **Y** in this field indicates that the address is a Commercial Mail Receiving Agency. CMRA addresses normally have a Private Mail Box (PMB) number associated with them.
DPVNoStat	This is the USPS DPV **No Statistics** flag. A **Y** in this field indicates that the standardized address does not have any statistics in the USPS DPV database.
DPVVacant	This is the USPS DPV **Vacant** flag. A **Y** in this field indicates that the standardized address is currently vacant.
DPVDNA	This is the DPV **Door Not Accessible** flag. A **Y** in this field indicates that the carrier delivering to the address cannot knock on a door for mail delivery, or cannot physically access the residence/ building.

Field Name	Description
SuiteLinkFootnote	This is the USPS **SuiteLink footnote** field. The possible values are: • **A** – The standardized address was processed through the USPS SuiteLink database and a match was made. The secondary information in the address was updated to reflect the information from the SuiteLink database. • **00** – The standardized address was processed through the USPS SuiteLink database and a match was not made. • **Blank** – The standardized address was not processed through the USPS SuiteLink database.
LACSLinkFootnote	This is the USPS **LACSLink Return Code** field. The possible values are: • **A** – The standardized address matched to a record in the LACSLink database, and it was updated to reflect the new LACSLink information. • **00** – The standardized address did not match to a record in the LACSLink database. • **09** – The standardized address matched to a record in the LACSLink database, but the old address is a high-rise default, and no new LACSLink information could be provided. • **14** – The standardized address matched to a record in the LACSLink database, but an error occurred while attempting to build the new address. • **92** – The standardized address matched to a LACSLink record, but the standardized address had a secondary number and the LACSLink record does not. The standardized address was updated to reflect the new LACSLink information.

Field Name	Description
LACSLinkNewAddress	This is the USPS **LACSLink Indicator** field. The possible values are: • **Y** – The standardized address was processed through the USPS LACS^{Link} database, and LACS^{Link} information was found. Check the **LACSLinkFootnote** field for details about the match. • **S** – The standardized address was processed through the USPS LACS^{Link} database, and LACS^{Link} information was found, but only after the secondary information was removed. Check the **LACSLinkFootnote** field for details about the match. • **N** – The standardized address was processed through the USPS LACS^{Link} database, and LACS^{Link} information was not found. • **Blank** – LACS^{Link} processing did not occur.
RDI	This is the **Residential Delivery Indicator**. A **Y** in this field indicates that the address is a residence.
ProductDate	This is the release date of the USPS ZIP+4 database used to process the input address.
Latitude	This is the latitude value for the standardized address. See the **GeocodeLevel** field below for information on the accuracy of this value.
Longitude	This is the longitude value for the address. See the **GeocodeLevel** field below for information on the accuracy of this value.
GeocodeLevel	This is the accuracy level of the latitude and longitude values for the standardized address. The possible values are: • **3** – ZIP+4 centroid • **5** – ZIP Code centroid
CensusTract	This is the US Census Tract for the standardized address.
CensusBlock	This is the US Census Block for the standardized address.
MSA	This is the US Office of Management and Budget Metropolitan Statistical Area code for the standardized address. This 4-digit code represents a geographic area of at least 50,000 people. This field is now defunct, and was replaced by the **CBSACode** below. While this code is still being returned, it is no longer being updated by the US Office of Management and Budget.

Field Name	Description
PMSA	This is the US Office of Management and Budget Primary Metropolitan Statistical Area code for the standardized address. The 4-digit PMSA represents a smaller geographic area within a metropolitan statistical area with a population of at least 1 million people. This field is now defunct, and was replaced by the **CBSADivisionCode** below. While this code is still being returned, it is no longer being updated by the US Office of Management and Budget.
TimeZone	This is the time zone for the standardized address.
CBSACode	This is the US Census Core Based Statistical Area code for the standardized address.
CBSALevel	This is the US Census Core Based Statistical Area level for the standardized address.
CBSADivisionCode	This is the US Census Core Based Statistical Area Division code for the standardized address.
CBSADivisionLevel	This is the US Census Core Based Statistical Area Division level for the standardized address.
FIPSState	This is the 2-digit state FIPS code.
FIPSCounty	This is the 5-digit county FIPS code.
ReferenceID	If the **ReferenceID** field of the input data is populated, then that value is returned in this output field. This is a convenient way to correlate output data to input data when operations are called asynchronously.

CityState and ZIPCode

Field Name	Description
ServiceResult	This is the result code from the service request. A **0** in this field indicates no error.
OperationResult	This is the result code from the call to the Coder's address-processing function. A **0** in this field indicates no error. If a processing error occurs, this field will contain a non-zero error code.
ErrorDescription	A text description of an error code reported in the **service-result** or **lookup-result** output fields.
RecordCount	This is the number of returned City/ State records. Each record matches the input data that was given.
ZIPCode	This is the ZIP Code.
CityStateKey	This is the USPS City/ State key for the city name in this record.
ZIPClassCode	This is the USPS ZIP Classification Code. The possible values are: • **Blank** – Non-special ZIP Code • **M** – APO/FPO Military ZIP Code • **P** – PO Box ZIP Code • **U** – Unique ZIP Code
CityName	This is the city name.
CityNameAbbrev	This is the abbreviated city name. An abbreviated name is present if the city name is greater than 13 characters long.
FacilityCode	This is the USPS Facility Type Code, which describes the relationship of the city name to the ZIP Code. The possible values are: • **B** – Branch • **C** – Community Post Office • **N** – Non-Postal Community Name, Former Postal Facility, or Place Name • **P** – Post Office • **S** – Station • **U** – Puerto Rico Urbanization

Field Name	Description
MailingName	This is the USPS Mailing Name Indicator. The USPS often recognizes more than one city name for a given ZIP Code, but not all of those city names are allowed for use in an address that has been standardized through the USPS ZIP+4 database. A **Y** in this field indicates that the city name may be used for mailing.
PrefCityName	This is the USPS Preferred City Name. **Preferred** is often misunderstood in this context. This field is effectively the **default** city name for the given ZIP Code. An address matching within this ZIP Code may standardize to an alternate city name.
PrefCityStateKey	This is the USPS City/ State key that is associated with the Preferred City Name.
UniqueZIP	This is the Unique ZIP Indicator. A **Y** in this field indicates that the name in the **CityName** field is not actually a city, but the name of the organization that owns this Unique ZIP Code.
Finance	This is the USPS Finance number.
State	This is the two-character state code.
CountyNumber	This is the default county number for the ZIP Code. It is possible that some address deliveries in this ZIP Code physically reside in a different county.
CountyName	This is the default county name for the ZIP Code. It is possible that some addresses deliveries in this ZIP Code physically reside in a different county.
ReferenceID	If the **ReferenceID** field of the input data is populated, then that value is returned in this output field. This is a convenient way to correlate output data to input data when operations are called asynchronously.

PartialStreet

Field Name	Description
ServiceResult	This is the result code from the service request. A **0** in this field indicates no error.
OperationResult	This is the result code from the call to the Coder's address-processing function. A **0** in this field indicates no error. If a processing error occurs, this field will contain a non-zero error code.
ErrorDescription	A text description of an error code reported in the **service-result** or **lookup-result** output fields.
StdPriNum	This is the primary number that was given in the input data. This is provided in the output fields as a convenience for constructing a full address line using the fields in the returned address records below.
RecordCount	This is the number of returned street-name records. Each record matches the input ZIP Code, primary number, partial street name, and the input pre-directional if one was given.
StdPreDir	This is the street pre-directional for a street-name record.
StdStreetName	This is the street (primary) name for a street-name record.
StdSuffix	This is the street suffix from for a street-name record.
StdPostDir	This is the street post-directional for a street-name record.
ReferenceID	If the **ReferenceID** field of the input data is populated, then that value is returned in this output field. This is a convenient way to correlate output data to input data when operations are called asynchronously.

AddressLines

Field Name	Description
ServiceResult	This is the result code from the service request. A **0** in this field indicates no error.
OperationResult	This is the result code from the call to the processing function. A **0** in this field indicates no error. If a processing error occurs, this field will contain a non-zero error code.
ErrorDescription	A text description of an error code reported in the **ServiceResult** or **OperationResult** output fields.
Name1	This contains an identified Name component, if found.
Name2	This contains an identified Name component, if multiple names are found.
Name3	This contains an identified Name component, if found.
Company1	This contains an identified Company name component, if found.
Company2	This contains an identified Company name component, if multiple names are found.
Company3	This contains an identified Company name component, if multiple names are found.
Company4	This contains an identified Company name component, if multiple names are found.
Address1	This contains an identified Address Line 1 component, if identified. **Note:** This will combine Lines 1 and 2 if they are found on different lines.
Address2	This contains an identified Address Line 2 component, if identified.
Address3	This contains an identified Address Line 3 component, if identified.
Address4	This contains an identified Address Line 4 component, if identified.
City	This contains the spelled-out form of the standardized city name, if identified.
State	This contains the State name, if identified.
Zip	This contains the ZIP Code, if identified.

Field Name	Description
Zip4	This contains the ZIP+4, if identified.
CityStateZipCombined	This contains a combined form of the City, State, and Zip.
URBData	This contains the Puerto Rico urbanization name, if identified.
Email	This contains the e-mail address, if found.
Web	This contains the web address, if found.
CreditCard	This contains the credit card number, if found. If the user does not wish to have credit card information returned in the output data, then set the input field **RemoveCreditCardInfo** to **1**.
ReferenceID	If the **ReferenceID** field of the input data is populated, then that value is returned in this output field. This is a convenient way to correlate output data to input data when operations are called asynchronously.

ParsedName

Field Name	Description
ServiceResult	This is the result code from the service request. A **0** in this field indicates no error.
OperationResult	This is the result code from the call to the processing function. A **0** in this field indicates no error. If a processing error occurs, this field will contain a non-zero error code.
ErrorDescription	A text description of an error code reported in the **ServiceResult** or **OperationResult** output fields.
RecordCount	When multiple names option is used, this will indicate how many identified records are being returned.
Title	This contains the Title used in the name, if any is identified.
FirstName	This contains the First name, if any is identified.
MiddleName	This contains the Middle name, if any is identified.
LastName	This contains the Last name, if any is identified.
SuffixGen	This contains the Generational Suffix, if any is identified.
Company	This contains the Company name, if any is identified.
SuffixProf	This contains the Professional Suffix, if any is identified.
SuffixTotal	This contains all Suffixes, if any are identified, as comma-separated values.
ReferenceID	If the **ReferenceID** field of the input data is populated, then that value is returned in this output field. This is a convenient way to correlate output data to input data when operations are called asynchronously.

CasedAddress

Field Name	Description
ServiceResult	This is the result code from the service request. A **0** in this field indicates no error.
OperationResult	This is the result code from the call to the processing function. A **0** in this field indicates no error. If a processing error occurs, this field will contain a non-zero error code.
ErrorDescription	A text description of an error code reported in the **ServiceResult** or **OperationResult** output fields.
Firm	This contains the Firm name in the correct case
AddressLine1	This contains the Address Line 1 in the correct case.
AddressLine2	This contains the Address Line 2 in the correct case.
City	This contains the City name in the correct case.
State	This contains the State name in the correct case.
ZIPCode	This contains the ZIP Code as sent in the Input.
Addon	This contains the Addon as sent in the Input.
PRUrb	This contains the Puerto Rico urbanization name in the correct case.
ReferenceID	If the **ReferenceID** field of the input data is populated, then that value is returned in this output field. This is a convenient way to correlate output data to input data when operations are called asynchronously.

CasedString

Field Name	Description
Output	This contains the input string returned in the correct case requested, and is empty if any error is encountered.

Appendix

This section will provide additional information for the following:

Address Processing Return Codes

Return Code	Description
32	**Default Match** A match was made to a single ZIP4 record; however, there was incorrect or missing information in the input address. This can happen when a match is made to a high-rise record, but the secondary number (suite, apartment, etc.) was either missing or incorrect. This can also happen when a match is made to a rural route record, but the BOX number was missing or incorrect.
31	**Single Match** A match was made to a single ZIP4 record.
22	**Multiple Match** More than 1 address record was made that equally matched the input address. AddressPro was not able to choose which record was correct.
21	**No Match** A full lookup attempt was made, but no address record was found that was close to the input.
13	Internal processing error.
11	**Invalid Last Line** The last line components (city, state, ZIP code) could not be resolved to find a section of the country to search. No lookup attempt was made.
10	**Unable to Parse Address** No lookup was attempted because the address could not be parsed.

Footnote Codes

Footnote	Description
AA	**USPS ZIP+4** - Record Matched to the ZIP+4 File This footnote is set any time that a record is matched. This is the equivalent of having a return code of 31 or 32.
A1	**USPS ZIP+4** - Record Not Matched to the ZIP+4 File This footnote is set every time that a record does not match to the ZIP4 file, but makes it through basic parse and last line validation. This is the equivalent of having a return of 21 or 22.
A2	**Alias Street Name Matched** This footnote is set every time a match is made to an alias street name.
A3	**Alternate Record Matched on the ZIP+4 File** This footnote is set every time a match occurs to an alternate ZIP4 record. An alternate record contains a different presentation a street name for a base ZIP4 record. The add-on code is the same for the alternate and the base record.
A4	**Match to Small Town Default** A small-town default match does not actually match to an existing ZIP4 record, but gives back a return code of **31** and an add-on code of **9999**. This condition exists when the input ZIP code is flagged as a Small Town and only accepts **GENERAL DELIVERY** type addresses, but a street-style or PO BOX address was presented on input. Since there are no actual street or PO BOX deliveries, all addresses in this ZIP code will get a default add-on of 9999.
A5	**Match to Unique ZIP Default** A Unique ZIP code is a ZIP code set up by an institution (like a university) or a business that handles its own mail delivery. The USPS collects the mail at the post office, and the institution picks it up and distributes it internally. Even though it is possible to have ZIP4 records for Unique ZIP code, most institutions have addressing schemes that are not fully populated in the database. Therefore, when an input Unique ZIP code is found, and the input address is not found, a return code of **31** is given back, along with a default add-on of **0001**.

Footnote	Description
A6	**Match to a High-Rise Default Alternate Record** This footnote is set when a High-rise Default Alternate style address match occurs. This condition exists when an input address contains the secondary number (an apartment, suite, etc.) along with a building name. No primary number or street name is given. **For Example:** 456 FEDERAL BUILDING This address will be turned back into a street-style presentation. 123 MAIN ST STE 456
A7	Address rejected due to EWS
A8	**Possible Dual Address Detected** This footnote is set when a dual address may be present, such as: RR 3 BOX 123 456 MAIN ST Since there are 2 possible addresses available, the Coder will not choose which one to parse and return. This address construct results in a **21** return code.
A9	**DPV used to Break ZIP4 Multiple Response** In some cases when a multiple-response is found during ZIP4 address processing, the DPV engine can be used to break the tie. During this process, all the candidate records are queried against the DPV database. If one and only one record is found to have a positive DPV confirmation, then this record will be chosen for the match, and a single response code is given back from the address lookup process. This footnote is turned on under this condition.
AC	ZIP+4 Level county name is different than the ZIP Code level
AN	A match was made to an **ND** (Non-Deliverable) address. This type of match will result in a return code of **31**, but no DPV codes will set.
AO	An input address was matched to an **O**-type Alias, and the alias name was kept in the output.
AP	A match was made to a **P**-type (Preferred) Alias.
AQ	The address found is in a Unique ZIP code, and the record information given back is based off of the ZIP+4 code, rather than the actual address record that was matched. **For Example:** If an input address matches a Street record, but the input address contained a ZIP+4 code that corresponds to a Firm record, then the Firm-record information is returned.

Footnote	Description
AU	Unique ZIP default information returned.
BB	**USPS DPV** - DPV match, all components A DPV match occurred using all the components of the address (primary and secondary number). This footnote is the equivalent of a **Y** in the DPV A flag.
CC	**USPS DPV** - DPV match primary number, not secondary number A DPV match occurred with the primary number only. A secondary number was present on the input address, but was not validated by DPV. This footnote is the equivalent of an **S** in the DPV A flag.
DD	**Corrected City Name and/or State Information** The city or the state was changed.
D1	Input city name was non-mailing, corrected. The input city name was a non-mailing name for the matched ZIP code. The output address contains the corrected city name.
D2	ZIP+4 State code different than ZIP Code default This footnote indicates that the state code on a matched ZIP4 record differs from the state code assigned to the ZIP code of the output address. This occurs when a physical address exists in one state, but the mail is delivered to that address from another state. In some geographical situations, it is better for a post office in one state to deliver mail to addresses that are in a neighboring state. In these situations, the output address will still show the state code of the delivering post office.
EE	**Corrected Primary Address** (E1 or E2 is on) The primary address line was corrected.
E1	**Corrected Primary Address Component** A directional or suffix address component was added, deleted, or changed.
E2	**Corrected Primary Street Name** Set if the street name spelling was changed.
E3	**Primary Address Line Standardized** Set if any portion of the primary address line was standardized.
E4	A questionable address standardization was made. This can happen when an input address looks like 123 MAIN ST BOX 123 (a possible dual address), and it is standardized to 123 MAIN ST # 123.
E5	The first letter of the street name has been added or changed.

Footnote	Description
E6	Address information that was not used was removed from the main address lines.
E7	**Removed City and / or State Information** that is found at the end of the address line. This information is detected as a duplicate of the actual city / state found in the input **City / State** fields.
FF	**Corrected Secondary Address Data** Set when the secondary number or unit designator was changed. Examples of allowable secondary number changes are reversing alpha-numerics, and adding or deleting dashes: A5 => 5A A5 => A-5
FS	Secondary information appended to the output address after a Firm match. The input address did not have secondary information present, but the matched Firm record did. The USPS allows matching to a Firm record that contains secondary information when the input does not, assuming the Firm names match.
F1	**USPS ZIP+4** - Military match A match was made to a ZIP Code that has been determined to be a Military ZIP Code.
G1	**USPS ZIP+4** - General Delivery match A match was made to a General Delivery ZIP+4 record.
G4	**Out of Range Alias Match** A match occurred to an alias street name, but the primary number did not match the allowable range for the alias street name. This is a no-match condition and issues a return code of **21**.
II	**Firm and Address Swapped** The firm line and the address line were swapped to make the match.
I1	**PR Urb** moved from primary address line A **PR Urbanization** was found in the primary address line and moved to the Urbanization line. This will only happen if the Urbanization name is appended to the end of the primary address, like “A18 CALLE ESPANA URB VISTA BELLA.” One of the **U*** footnotes will also be set, depending on the validation level of the Urb name that was found.

Footnote	Description
I2	**PR Urb** moved from secondary address line An Urbanization name was found on the secondary address line and moved to the output **PR Urb** field. One of the **U*** footnotes will also be set, depending on the validation level of the **Urb** name that was found. This behavior will occur if the primary address is located in the input field addr1 and the secondary address is located in addr2, or vice versa.
HH	**ZIP Code Changed** The input ZIP code was changed.
H1	**ZIP Code Added** No ZIP code was in the input address, and one was added.
JJ	**Failure to Match Last Line of Address** The last line was not able to be matched. This is equivalent to the return code **11**.
J1	**City, State, and ZIP Code Agree** This is set when an address does not make a match (return code <31), but the input city, state, and ZIP code all agree. This currently only applies to when the input city name is correctly spelled.
KK	Multiple Match in Primary Address Field
K1	Multiple Match due to Missing/Incorrect Directional(s)
K2	Multiple Match due to Missing/Incorrect Suffix
K3	Multiple Match due to Missing/ Incorrect ZIP Code
K4	Multiple Match due to Missing/ Incorrect City name
K5	Multiple Match due to Missing/ Incorrect PR Urbanization name
K6	Multiple Match due to overlapping primary numbers in different Carrier Routes
LL	A match was made to the LACS^{Link} database.

Footnote	Description
LM	Indicates a condition where a LACS^{Link} match was made, but the new-side address could not be ZIP+4 confirmed. In this situation, the final address will be the result of the original address lookup, not the result of the LACS^{Link} new-side lookup. In such cases, the LACS^{Link} output fields will still be populated with the LACS^{Link} match information. Since a LACS^{Link} lookup is performed when the input address does not ZIP+4 confirm, the new LM footnote may be set in any match condition (single-response, default, multiple-response, or no-match). A LACS^{Link} new-side address that does not ZIP+4 confirm is a very rare condition. However, it may happen from time to time. Often this is a result of the new-side ZIP+4 being invalid in the database currently being used, and will most likely become valid in a future database release.
MA	**ZIP4** - Missing Street Number A primary number is missing from the input address.
MB	**ZIP4** – Street Name Not Found The input street name could not be found in the ZIP+4 data file.
MC	**ZIP4** - No Such Primary Number This is set when the input street name was found in the ZIP+4 file, but the input primary number could not be found with the given street name.
MD	**ZIP4** - Firm Name Not Matched
ME	An address was found, but it was not matched due to a violation of the Cardinal Point rule.
M1	**USPS ZIP+4** - Missing Street Number A primary number is missing from the input address, and the input address was a street, high-rise, or firm type.
M3	**USPS DPV** - No Such Primary Number This is set when DPV cannot validate the primary number, even though the address matched to a valid ZIP+4 range.
NA	**ZIP4** - Missing Secondary Address Number A match was made to a ZIP+4 High-rise record, but no secondary number was present on the input address.

Footnote	Description
NB	**ZIP4** - Secondary Number not found A match was made to a ZIP+4 record, but the input secondary number could not be found.
NC	**Multiple Match in Secondary Address Field** Set when a 'multiple response' condition occurs with high-rise specific records, and one cannot be picked. A high-rise default or street level record is returned. **For Example:** Input address is '123 MAIN ST # 4' and both APT 4 and STE 4 are present.
N1	**USPS DPV** - Missing Secondary Address Number for a high-rise address The input address was confirmed in DPV for the primary number, but the secondary number was not present on the input address. This happens when the address matched a high-rise-default ZIP+4 record, so a secondary number is expected. This footnote is the equivalent of a **D** in the DPV A flag.
P1	**USPS ZIP+4** - Missing RR/HC Box Number For rural-route style addresses, the BOX number was missing on input.
P2	**RR/HC Box Number not Found** Currently not being set.
P3	**USPS DPV** - Invalid RR/ HC/ PO BOX number.
Q1	Missing PO Box Number
Q2	PO Box Number not found
RR	**USPS DPV** - CMRA match This means that the input address is a CMRA business. This footnote is equivalent to a **Y** in the DPV C flag.
R1	**USPS DPV** - CMRA, no PMB number This footnote is turned on when there is a **Y** in the DPV C flag (the address is a CMRA business), but that the input address is missing a PMB number.
S1	**Seasonal Record Information Present** This footnote indicates that the address has seasonal delivery and receives mail only during part of the year.
S2	SuiteLink information added to the address.

Footnote	Description
S3	Address matched to STOP[Link].
S4	**Input Secondary Changed with Suite[Link]** This is set when an input address contains secondary information and makes a match to Suite[Link], and the original secondary information is changed based on the Suite[Link] record. If space allows, the original secondary input information is moved to the **Output Address Line 2** field.
UA	No **PR Urbanization** was given with the input address, but an Urbanization was found in the matched ZIP4 record. This Urbanization was returned on output.
UB	The input **PR Urbanization** was verified to be valid according to the ZIP4 record that was matched.
UC	The input **PR Urbanization** was not verified with the matched ZIP4 record, because the matched record contained a blank Urbanization field. The input Urbanization was retained in the output address.
UD	The input **PR Urbanization** was not verified with the matched ZIP4 record, because the matched record contained a different Urbanization name. The Urbanization from the matched ZIP4 record was returned with the output address.
U1	**USPS ZIP+4** - Unique ZIP Code Match A match was made to a ZIP Code that has been determined to be a Unique ZIP Code. See footnote **A5** for more information on Unique ZIP Codes.
X1	**IntelliZIP Match** This footnote indicates that a match was made using **IntelliZIP** logic. If a match is not made through normal address matching procedures, and if the input address contains a 9-digit ZIP code, then a reverse 9-digit lookup is performed. If the ZIP+4 record corresponding to the input 9-digit ZIP code closely matches the input address, then a match will be returned and this footnote will be set.

Footnote	Description
Z0	**ZIPMOVE Match** A match was made to a ZIP Move address. ZIP Move represents a collection of addresses that have been affected by a ZIP code realignment (i.e. the boundaries for a ZIP code were redrawn). When this happens, some street segments will move from one ZIP code to another. This can cause problems when an input address contains the old ZIP code, and the address matching software tries to change the address to keep it in the old ZIP code instead of moving to a new ZIP code. To help out, the USPS keeps a separate file of these situations that are used by address matching software. This footnote is set when the input address contains the old ZIP code, and it was changed to the new ZIP code. An exact match must be made to the address in order for this to happen.
Z1	**ZIPMOVE No Match Due to Component Change** The best match available was to a ZIP Move address, but the input address was changed to get there. This is not allowed, so a no-match condition is made with a return code of **21**.
Z2	**ZIPMOVE No Match Due to Invalid New ZIP+4** This represents a data inconsistency with the USPS ZIP4 and ZIP Move files. The address was no-matched with a return code of **21**.

Sample Code for C#

The following C# sample code demonstrates how to call the StandardAddress and ZIPCode AddressPro REST operations. This code uses the Newtonsoft JSON NuGet package for handling JSON data structures.

```
using System;
using System.Text;
using System.IO;
using System.Net;
using Newtonsoft.Json;

namespace AddressProRESTTestCS
{
    // Input/ output classes for JSON serialize/ deserialize.
    class StandardAddressInput
    {
        public string AuthenticationKey { get; set; }
        public string Firm { get; set; }
        public string AddressLine1 { get; set; }
        public string AddressLine2 { get; set; }
        public string City { get; set; }
        public string State { get; set; }
        public string ZIPCode { get; set; }
        public string Addon { get; set; }
        public string PRUrb { get; set; }
        public string Latitude { get; set; }
        public string Longitude { get; set; }
        public int EnableGeocoding { get; set; }
        public int MaxResults { get; set; }
        public string OutputCasing { get; set; }
        public string ReferenceID { get; set; }
    }

    class OutputAddress
    {
        public string ReturnCode { get; set; }
        public string Firm { get; set; }
        public string AddressLine1 { get; set; }
        public string AddressLine2 { get; set; }
        public string LastLine { get; set; }
        public string City { get; set; }
        public string State { get; set; }
        public string ZIPCode { get; set; }
        public string Addon { get; set; }
        public string PRUrb { get; set; }
        public string CityAbbreviation { get; set; }
        public string CitySpelled { get; set; }
        public string CarrierRouteID { get; set; }
        public string DeliveryPoint { get; set; }
```

```
        public string CheckDigit { get; set; }
        public string RecordType { get; set; }
        public string StdPriNum { get; set; }
        public string StdPreDir { get; set; }
        public string StdStreetName { get; set; }
        public string StdSuffix { get; set; }
        public string StdPostDir { get; set; }
        public string StdUnit { get; set; }
        public string StdSecNum { get; set; }
        public string StdUnit2 { get; set; }
        public string StdSecNum2 { get; set; }
        public string PMBDesignator { get; set; }
        public string PMBNumber { get; set; }
        public string MSCDesignator { get; set; }
        public string MSCNumber { get; set; }
        public string Finance { get; set; }
        public string CongressionalDistrict { get; set; }
        public string UniqueZIP { get; set; }
        public string ZIP4CountyNumber { get; set; }
        public string ZIPCountyNumber { get; set; }
        public string ZIPCountyName { get; set; }
        public string Footnotes { get; set; }
        public string DPVA { get; set; }
        public string DPVCMRA { get; set; }
        public string DPVNoStat { get; set; }
        public string DPVVacant { get; set; }
        public string DPVDNA { get; set; }
        public string SuiteLinkFootnote { get; set; }
        public string LACSLinkFootnote { get; set; }
        public string LACSLinkNewAddress { get; set; }
        public string RDI { get; set; }
        public string ProductDate { get; set; }
        public string Latitude { get; set; }
        public string Longitude { get; set; }
        public string GeocodeLevel { get; set; }
        public string CensusTract { get; set; }
        public string CensusBlock { get; set; }
        public string PMSA { get; set; }
        public string MSA { get; set; }
        public string TimeZone { get; set; }
        public string CBSACode { get; set; }
        public string CBSALevel { get; set; }
        public string CBSADivisionCode { get; set; }
        public string CBSADivisionLevel { get; set; }
    }

    class StandardAddressOutput
    {
        public int ServiceResult { get; set; }
        public int OperationResult { get; set; }
        public string ErrorDescription { get; set; }
        public string ReferenceID { get; set; }
        public OutputAddress OutputAddress { get; set; }
    }
```

```
class ZIPCodeInput
{
    public string AuthenticationKey { get; set; }
    public string ZIPCode { get; set; }
    public string ReferenceID { get; set; }
}

class ZIPCodeRecord
{
    public string ZIPCode { get; set; }
    public string CityStateKey { get; set; }
    public string ZIPClassCode { get; set; }
    public string CityName { get; set; }
    public string CityNameAbbrev { get; set; }
    public string FacilityCode { get; set; }
    public string MailingName { get; set; }
    public string PrefCityName { get; set; }
    public string PrefCityStateKey { get; set; }
    public string UniqueZIP { get; set; }
    public string Finance { get; set; }
    public string State { get; set; }
    public string CountyNumber { get; set; }
    public string CountyName { get; set; }
}

class ZIPCodeRecords
{
    public int RecordCount { get; set; }
    public ZIPCodeRecord[] Record { get; set; }
}

class ZIPCodeOutput
{
    public int ServiceResult { get; set; }
    public int OperationResult { get; set; }
    public string ErrorDescription { get; set; }
    public string ReferenceID { get; set; }
    public ZIPCodeRecords Records { get; set; }
}

class AddressProRESTTest
{
    static string SendPostRequest(string URi, string PostData)
    {
        var request = (HttpWebRequest)WebRequest.Create(URi);
        var responseValue = string.Empty;

        request.Method = "POST";
        request.ContentType = "text/json";

        var encoding = new UTF8Encoding();
        var post_bytes = Encoding.GetEncoding("iso-8859-1").GetBytes(PostData);
        request.ContentLength = post_bytes.Length;

        using (var writeStream = request.GetRequestStream())
```

```
        {
            writeStream.Write(post_bytes, 0, post_bytes.Length);
        }

        using (var response = (HttpWebResponse)request.GetResponse())
        {
            if (response.StatusCode != HttpStatusCode.OK)
            {
                var message = String.Format("Request failed. Received HTTP {0}",
                                            response.StatusCode);
                throw new ApplicationException(message);
            }

            // grab the response
            using (var responseStream = response.GetResponseStream())
            {
                if (responseStream != null)
                {
                    using (var reader = new StreamReader(responseStream))
                    {
                        responseValue = reader.ReadToEnd();
                    }
                }
            }
        }
        return responseValue;
    }

    static void Main(string[] args)
    {
        string post_data;
        string response_value;
        string addresspro_url = ""; // Put your AddressPro REST URL here.
        string my_auth_key = ""; // Put your AddressPro authentication key here.

        StandardAddressInput sa = new StandardAddressInput();
        sa.AuthenticationKey = my_auth_key;
        sa.AddressLine1 = "400 Chisholm Pl Ste 300";
        sa.City = "Plano TX";
        sa.ReferenceID = "Anchor's TX Office";

        post_data = JsonConvert.SerializeObject(sa);
        response_value = SendPostRequest(addresspro_url + "/AddressPro/v1/StandardAddress",
                         post_data);

        //Console.WriteLine("{0}", response_value);

        Console.WriteLine("StandardAddress Call:");
        StandardAddressOutput sao =
            JsonConvert.DeserializeObject<StandardAddressOutput>(response_value);
        if (sao.ServiceResult != 0 || sao.OperationResult != 0)
        {
            Console.WriteLine("ServiceResult: {0}", sao.ServiceResult);
            Console.WriteLine("OperationResult: {0}", sao.OperationResult);
            Console.WriteLine("ErrorDescription: {0}", sao.ErrorDescription);
```

```
            }
            else
            {
                Console.WriteLine("ReferenceID: {0}", sao.ReferenceID);
                Console.WriteLine("Return code: {0}", sao.OutputAddress.ReturnCode);
                Console.WriteLine("Footnotes: {0}", sao.OutputAddress.Footnotes);
                Console.WriteLine("{0}", sao.OutputAddress.AddressLine1);
                Console.WriteLine("{0}", sao.OutputAddress.LastLine);
            }
            Console.WriteLine("");

            ZIPCodeInput zc = new ZIPCodeInput();
            zc.AuthenticationKey = my_auth_key;
            zc.ZIPCode = "38104";
            zc.ReferenceID = "Test ZIP Code";

            post_data = JsonConvert.SerializeObject(zc);
            response_value = SendPostRequest(addresspro_url + "/AddressPro/v1/ZIPCode",
                                             post_data);

            //Console.WriteLine("{0}", response_value);

            Console.WriteLine("ZIPCode Call:");
            ZIPCodeOutput zco = JsonConvert.DeserializeObject<ZIPCodeOutput>(response_value);
            if (zco.ServiceResult != 0 || zco.OperationResult != 0)
            {
                Console.WriteLine("ServiceResult: {0}", zco.ServiceResult);
                Console.WriteLine("OperationResult: {0}", zco.OperationResult);
                Console.WriteLine("ErrorDescription: {0}", zco.ErrorDescription);
            }
            else
            {
                Console.WriteLine("ReferenceID: {0}", zco.ReferenceID);
                Console.WriteLine("Record count: {0}", zco.Records.RecordCount);
                for (int i = 0; i < zco.Records.RecordCount; i++)
                {
                    Console.WriteLine("  {0}: {1} {2}", i + 1, zco.Records.Record[i].CityName,
                                      zco.Records.Record[i].State);
                }
            }
            Console.WriteLine("");
        }
    }
}
```

Sample Code for PHP

The following PHP sample code demonstrates how to call the StandardAddress, CityState, and ZIPCode AddressPro REST operations. This code uses the CURL PHP package for REST Web Service calls.

```
<HTML>
<HEAD>
<title>AddressPro Demo Domestic USA</title>
</HEAD>

<BODY>

<table width="75%" border="3" align="center" cellpadding="0" cellspacing="0">
  <tr>
    <td bgcolor="#FFFFFF" style="padding: 5px;" >

AddressPro Address Lookup</P>

<script>
    function ClearForm() {
        document.FullLookup.in_firm.value = "";
        document.FullLookup.in_addr.value = "";
        document.FullLookup.in_city.value = "";
        document.FullLookup.in_state.value = "";
        document.FullLookup.in_zip.value = "";
    }
</script>

<FORM id=FullLookup name=FullLookup action="index.php" method=post>

<TABLE width="75%" border=0 align="center" cellPadding=1 cellSpacing=1>
  <TR>
    <TD>Firm</TD>
    <TD><INPUT name="in_firm" size=60 value="<?php echo $_POST["in_firm"]; ?>"></TD>
  </TR>
  <TR>
    <TD>Address</TD>
    <TD><INPUT name="in_addr" size=60 value="<?php echo $_POST["in_addr"]; ?>"></TD>
  </TR>
  <TR>
    <TD>City/ State/ ZIP</TD>
    <TD><INPUT name="in_city" size=40 value="<?php echo $_POST["in_city"]; ?>">
        <INPUT name="in_state" size=2 value="<?php echo $_POST["in_state"]; ?>">
        <INPUT name="in_zip" size=10 value="<?php echo $_POST["in_zip"]; ?>">
    </TD>
  </TR>
  <TR>
    <TD height="52" colspan="2" valign="bottom">
      <input id=submit1 type=submit value="Standard lookup" name=submit1>
      <input id=submit2 type=submit value="City/ State lookup" name=submit2>
      <input id=submit3 type=submit value="ZIP Lookup" name=submit3>
      <input id=clear type=button value=Clear name=clear onClick="ClearForm();">
    </TD>
  </TR>
</TABLE>

<P> </P>

</FORM>
```

```php
<?php

//*************************************************************************
//*************************************************************************
function GetRESTCUrl($url, $page, $data, $calltype)
{
    global $m_error_message;

    $full_url="$url/$page";

    $ch = curl_init();
    // Return data as output string, not printed to console.
    curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);

    if($calltype == "POST")
    {
        curl_setopt($ch, CURLOPT_CUSTOMREQUEST, "POST");
        if(isset($data) && $data != "" )
        {
            $json_input=json_encode($data);
            curl_setopt($ch, CURLOPT_POSTFIELDS, $json_input);
            curl_setopt($ch, CURLOPT_HTTPHEADER, array(
                'Content-Type: application/json','Content-Length: ' . strlen($json_input)));
        }
    }

    curl_setopt($ch, CURLOPT_URL, $full_url);
    $result=curl_exec($ch);
    $response_code=curl_getinfo($ch, CURLINFO_HTTP_CODE);
    curl_close($ch);

    if($response_code!=200)
    {
        $m_error_message="Got bad response from server: $response_code";
        return "";
    }

    return $result;
}

//*************************************************************************
//*************************************************************************
function PrintTableAddressField($result, $field)
{
    global $m_show_empty_fields;

    if( !isset($result) ||
        !isset($result->{"OutputAddress"}) ||
        !isset($result->{"OutputAddress"}->{$field})
      )
    {
        return;
    }
```

```
    if($m_show_empty_fields==0 && $result->{"OutputAddress"}->{$field} == "")
    {
        return;
    }

    print "<TR>";
    print "<TD>" . $field . "</TD>";
    print "<TD>" . $result->{"OutputAddress"}->{$field} . "</TD>";
    print "</TR>\n";
}

//*****************************************************************************
//*****************************************************************************
function PrintTableFullLookupOutput()
{
    global $m_fulllookup_output;

    print "<TABLE align=center cellSpacing=0 cellPadding=1 width=\"75%\" border=1
           bordercolor=#000099 >\n";
    PrintTableAddressField($m_fulllookup_output, "ReturnCode");
    PrintTableAddressField($m_fulllookup_output, "Firm");
    PrintTableAddressField($m_fulllookup_output, "AddressLine1");
    PrintTableAddressField($m_fulllookup_output, "LastLine");
    PrintTableAddressField($m_fulllookup_output, "CityAbbreviation");
    PrintTableAddressField($m_fulllookup_output, "CitySpelled");
    PrintTableAddressField($m_fulllookup_output, "CarrierRouteID");
    PrintTableAddressField($m_fulllookup_output, "DeliveryPoint");
    PrintTableAddressField($m_fulllookup_output, "CheckDigit");
    PrintTableAddressField($m_fulllookup_output, "ZIPCountyName");
    PrintTableAddressField($m_fulllookup_output, "ZIPCountyNumber");
    PrintTableAddressField($m_fulllookup_output, "ZIP4CountyNumber");
    PrintTableAddressField($m_fulllookup_output, "RecordType");
    PrintTableAddressField($m_fulllookup_output, "Footnotes");

    PrintTableAddressField($m_fulllookup_output, "CongressionalDistrict");
    PrintTableAddressField($m_fulllookup_output, "Finance");
    PrintTableAddressField($m_fulllookup_output, "DPVA");
    PrintTableAddressField($m_fulllookup_output, "DPVCMRA");
    PrintTableAddressField($m_fulllookup_output, "DPVNoStat");
    PrintTableAddressField($m_fulllookup_output, "DPVVacant");
    PrintTableAddressField($m_fulllookup_output, "DPVDNA");
    PrintTableAddressField($m_fulllookup_output, "StdPriNum");
    PrintTableAddressField($m_fulllookup_output, "StdPreDir");
    PrintTableAddressField($m_fulllookup_output, "StdStreetName");
    PrintTableAddressField($m_fulllookup_output, "StdSuffix");
    PrintTableAddressField($m_fulllookup_output, "StdPostDir");
    PrintTableAddressField($m_fulllookup_output, "StdUnit");
    PrintTableAddressField($m_fulllookup_output, "StdSecNum");
    PrintTableAddressField($m_fulllookup_output, "SuiteLinkFootnote");
    PrintTableAddressField($m_fulllookup_output, "LACSLinkFootnote");
    PrintTableAddressField($m_fulllookup_output, "RDI");
    PrintTableAddressField($m_fulllookup_output, "Latitude");
    PrintTableAddressField($m_fulllookup_output, "Longitude");
    PrintTableAddressField($m_fulllookup_output, "GeocodeLevel");
```

```
    print "</TABLE>\n";
}

//*************************************************************************
//*************************************************************************
function ProcessFullLookup()
{
    global $m_webservice_location;
    global $m_default_auth_key;
    global $m_fulllookup_output;

    $fulllookup_status=0;
    $input_fields=array('AuthenticationKey'=>$m_default_auth_key,
                        'Firm'=>          $_POST["in_firm"],
                        'AddressLine1'=> $_POST["in_addr"],
                        'City'=>          $_POST["in_city"],
                        'State'=>         $_POST["in_state"],
                        'ZIPCode'=>       $_POST["in_zip"],
                        'MaxResults'=>    0,
                        );

    $result=GetRESTCUrl($m_webservice_location, "AddressPro/V1/StandardAddress",
                        $input_fields,
                        "POST");
    if($result=="")
    {
        $fulllookup_status=-1;
    }
    else
    {
        $fulllookup_status=1;
        $m_fulllookup_output=$result;
        $m_fulllookup_output=json_decode($m_fulllookup_output);
    }

    return $fulllookup_status;
}

//*************************************************************************
//*************************************************************************
function PrintTableHeaderCTST()
{
    print "<TABLE align=center cellSpacing=0 cellPadding=1 width=\"75%\" border=1
           bordercolor=#000099 >\n";
    print "  <TR>\n";
    print "    <TH>ZIP
Code</TH><TH>Name</TH><TH>Preferred</TH><TH>Abbrev</TH><TH>State</TH><TH>Finance</TH><TH>Cou
nty</TH><TH>LL Key</TH><TH>PLL Key</TH><TH>County
Num.</TH><TH>ZCC</TH><TH>FTC</TH><TH>MNI</TH><TH>UZI</TH>\n";
    print "  </TR>\n";
}

//*************************************************************************
//*************************************************************************
function PrintTableFieldCTST($x)
```

```
{
    print "<TD>" . $x . "</TD>";
}

//*****************************************************************************
//*****************************************************************************
function PrintTableRecordCTST($record)
{
    print "  <TR>\n";
    PrintTableFieldCTST($record->{"ZIPCode"});
    PrintTableFieldCTST($record->{"CityName"});
    PrintTableFieldCTST($record->{"PrefCityName"});
    PrintTableFieldCTST($record->{"CityNameAbbrev"});
    PrintTableFieldCTST($record->{"State"});
    PrintTableFieldCTST($record->{"Finance"});
    PrintTableFieldCTST($record->{"CountyName"});
    PrintTableFieldCTST($record->{"CityStateKey"});
    PrintTableFieldCTST($record->{"PrefCityStateKey"});
    PrintTableFieldCTST($record->{"CountyNumber"});
    PrintTableFieldCTST($record->{"ZIPClassCode"});
    PrintTableFieldCTST($record->{"FacilityCode"});
    PrintTableFieldCTST($record->{"MailingName"});
    PrintTableFieldCTST($record->{"UniqueZIP"});
    print "  </TR>\n";
}

//*****************************************************************************
//*****************************************************************************
function PrintTableCTST($output)
{
    $n_recs=$output->{"Records"}->{"RecordCount"};
    print "Number of records: " . $n_recs . "<BR>\n";

    PrintTableHeaderCTST();
    if($n_recs==1)
    {
        PrintTableRecordCTST($output->{"Records"}->{"Record"});
    }
    else if($n_recs>1)
    {
        for($i=0; $i<$n_recs; $i++)
        {
            PrintTableRecordCTST($output->{"Records"}->{"Record"}[$i]);
        }
    }
    print "</TABLE>\n";
}

//*****************************************************************************
//*****************************************************************************
function ProcessCityStateLookup()
{
    global $m_webservice_location;
    global $m_default_auth_key;
    global $m_ctst_output;
```

```
    $ctst_status=0;
    $input_fields=array('AuthenticationKey'=>$m_default_auth_key,
                        'City'=>          $_POST["in_city"],
                        'State'=>         $_POST["in_state"],
                        'MaxResults'=>    0,
                       );

    $result=GetRESTCUrl($m_webservice_location, "AddressPro/V1/CityState", $input_fields,
                        "POST");
    if($result=="")
    {
        $ctst_status=-1;
    }
    else
    {
        $m_ctst_output=$result;
        $m_ctst_output=json_decode($m_ctst_output);
        $ctst_status=1;
    }

    return $ctst_status;
}


//*************************************************************************
//*************************************************************************
function ProcessZIPLookup()
{
    global $m_webservice_location;
    global $m_default_auth_key;
    global $m_zip_output;

    $zip_status=0;
    $input_fields=array('AuthenticationKey'=>$m_default_auth_key,
                        'ZIPCode'=>       $_POST["in_zip"],
                        'MaxResults'=>    0,
                       );

    $result=GetRESTCUrl($m_webservice_location, "AddressPro/V1/ZIPCode", $input_fields,
                        "POST");
    if($result=="")
    {
        $zip_status=-1;
    }
    else
    {
        $zip_status=1;
        $m_zip_output=$result;
        $m_zip_output=json_decode($m_zip_output);
    }

    return $zip_status;
}
```

```
//*************************************************************************
//*************************************************************************

$m_show_empty_fields=0;
$m_error_message="";
$m_fulllookup_output="";
$m_ctst_output="";
$m_zip_output="";
$m_webservice_location=""; // Put your AddressPro Web Service URL here.
$m_default_auth_key=""; // Put your AddressPro authentication key here.

if($_POST["submit1"] != "")
{
    $result=ProcessFullLookup();
    if($result==1)
        PrintTableFullLookupOutput();
    else
        print $m_error_message;
}
else if($_POST["submit2"] != "")
{
    $result=ProcessCityStateLookup();
    if($result==1)
        PrintTableCTST($m_ctst_output);
    else
        print $m_error_message;
}
else if($_POST["submit3"] != "")
{
    $result=ProcessZIPLookup();
    if($result==1)
        PrintTableCTST($m_zip_output);
    else
        print $m_error_message;
}

?>

    </td>
  </tr>
</table>

</BODY>
</HTML>
```

Index