Avalara Geo for Communications

Product User Manual

Release: 1711
Document: TM_00206_0018
Date: 11/15/2017

Avalara for Communications - Contact Information	
Address	**Avalara, Inc.** 513 South Mangum Street, Suite 100 Durham, North Carolina, 27701
Toll Free	800-525-8175
Corporate Site	http://communications.avalara.com/
Comms Platform Site	https://communications.avalara.net
Email	communicationsupport@avalara.com

Document Revision History

The Revision History log lists the date and description of the most recent revisions or versions of the document.

Date	Version	Description
10/01/2014	0001	Initial publication/distribution.
10/06/2014	0002	Re-branded for EZtax, Inc.
01/09/2015	0003	Added verbiage to Section 3 regarding address geocoding for US addresses/locations only—not international. (Also inserted references to US addresses in first paragraphs of Sections 5, 6, 7 and 8.)
04/05/2015	0004	Updates specific to 2015Q2r1 release/build for second quarter. Copyright updates throughout. Screenshot updates in Section 7 for EZgeo Direct.
07/02/2015	0005	Updated Location Service Endpoint address in Section 5.4.
07/09/2015	0006	Inserted details regarding valid/invalid street number addresses in Section 3.1.1 and reference to AFC Geo's inability to support geocoding for PO Box address in Section 4.4.
10/22/2015	0007	Changed references from Census Block Group to Census Block, to reflect change in code that returns individual block IDs, instead of block group ID.
01/13/2016	0008	Updates for AFC Geo Direct screenshots and to include census values in Section 7 AFC Geo Direct and in Section 6 AFC Geo SaaS Standard.
02/10/2016	0009	Added clarification in Best Practices (Section 4.4) to use option for Special Tax Districts.
02/22/2016	0010	Added clarification for PO Boxes in Address Parsing (Section 3.1.1) and Best Practices (Section 4.4).
02/26/2016	0011	Avalara branding updates to reflect the transition to the new company and product names have been incorporated into this document. Please **see Appendix A – Avalara Product Names** for specific changes in product references and descriptions.
05/13/2016	0012	Added the following: reference to meaning of PCode = -1 in **Section 5.3.2 Address Location**; **Section 4.7 Secondary Unit** to Best Practices; table added to **Section 4.8 Endpoints**; **Section 5.5 Example SOAP Messages**; fields to AFC Geo SaaS Standard input format in **Section 6.3.1.2 Address Geocoding Format**; reference to Source as internal field in **Section 5.3.1 Input Address**; note regarding geocoding support only for US and US territory addresses in **Section 3.1 Understanding Address Geocoding**; Updated AFC Geo Direct screenshots in **Section 7 AFC Geo Direct**.
08/25/2016	0013	Updated **Section 5.3.1 Input Address** to reflect changes in optional fields; Updated **Section 5.5 Example SOAP Messages** with samples for Basic and Custom binding; Updated **Section 6.3.1.2 Address Geocoding Formats** with default values for options in SaaS Standard; Added Feature/ID as an option for SaaS Standard in **Section 6.3.2.2 Lat/Long Geocoding**; Added new **Section 6.3.3 Error Handling** for SaaS Standard; Added processing options in **Section 7.2.2 Input Address and Options** for Direct and updated AFC Direct screenshots throughout.
11/21/2016	0014	Added new **Section 3.3 Special Tax Jurisdiction and Townships** to provide explanation of the Incorporated flag and how it ties in with special tax jurisdictions and townships. Removed notes from **Section 5.3.2** regarding census data not being available for Florida. Census data is now available for Florida. Updated references to 'special tax districts' to 'special tax jurisdictions' throughout document with exception of the field names in the software.
02/13/2017	0015	Updated release and version numbers.

Date	Version	Description
05/15/2017	0016	Updated **Section 5.3.1 Input Address** to include Option Value 8 for Return Zip+4. Renamed **Section 7 AFC Geo Viewer** and updated content to reflect the transition from AFC Geo Direct to new tool.
08/18/2017	0017	Release and version number updates on the cover.
11/15/2017	0018	Release and version number updates on the cover. Updated Avalara contact information (address and support site). Removed **Appendix A – Avalara Product Names**.

Table of Contents

1. Introduction

1.1 What is Avalara Geo for Communications?

Avalara Geo for Communications (AFC Geo) is a geospatial product which provides assurance that customers are taxed in the appropriate taxing jurisdiction. AFC Geo pinpoints your customers' locations and tax jurisdiction using latitude and longitude. It also validates customers' addresses according to the United States Postal standards and the data is Coding Accuracy Support System (CASS) certified.

AFC Geo offers multiple features such as the ability to:

- Validate customer addresses to eliminate incorrect issues.
- Enhance mailing addresses to ZIP+4 U.S. Postal standards.
- Determine the county where customers are located.
- Determine if addresses are incorporated or unincorporated.
- Identify special tax jurisdiction status including E911, transit, police, school, library and more.

AFC Geo is able to perform these tasks through address scrubbing, address validation and tax jurisdiction assignment (geocoding). It has the ability to drill down to the rooftop-level (based on U.S. Postal standards) which allows users to collect and remit accurate taxes and invoices.

AFC Geo's performance is extremely fast and supports multiple data types, including standard address with Zip+4, FIPS codes, Feature ID's, XY coordinates and PCodes (Avalara's proprietary jurisdiction code).

Additionally, AFC Geo has been certified by the Florida Department of Revenue for address to jurisdiction assignment. This means customers can qualify to receive higher tax collection allowances (.75 percent versus .25 percent) and are granted the hold harmless provision of the Florida Communications Service Tax. Also, the use of CASS validation allows AFC Geo users to obtain a higher degree of accuracy for carrier routes, 5-digit ZIP codes, ZIP+4 codes and delivery point addresses.

Note: AFC Geo only supports geocoding for US, Puerto Rico and other US territory addresses. All addresses in foreign countries (including Canada and Mexico) are not supported and may not return any results.

1.2 Purpose

The purpose of this document is to provide an overview of Avalara's AFC Geo product features and functionality. It also provides high-level information for new users and detailed instruction for working with the different variations of the product.

1.3 Process Overview

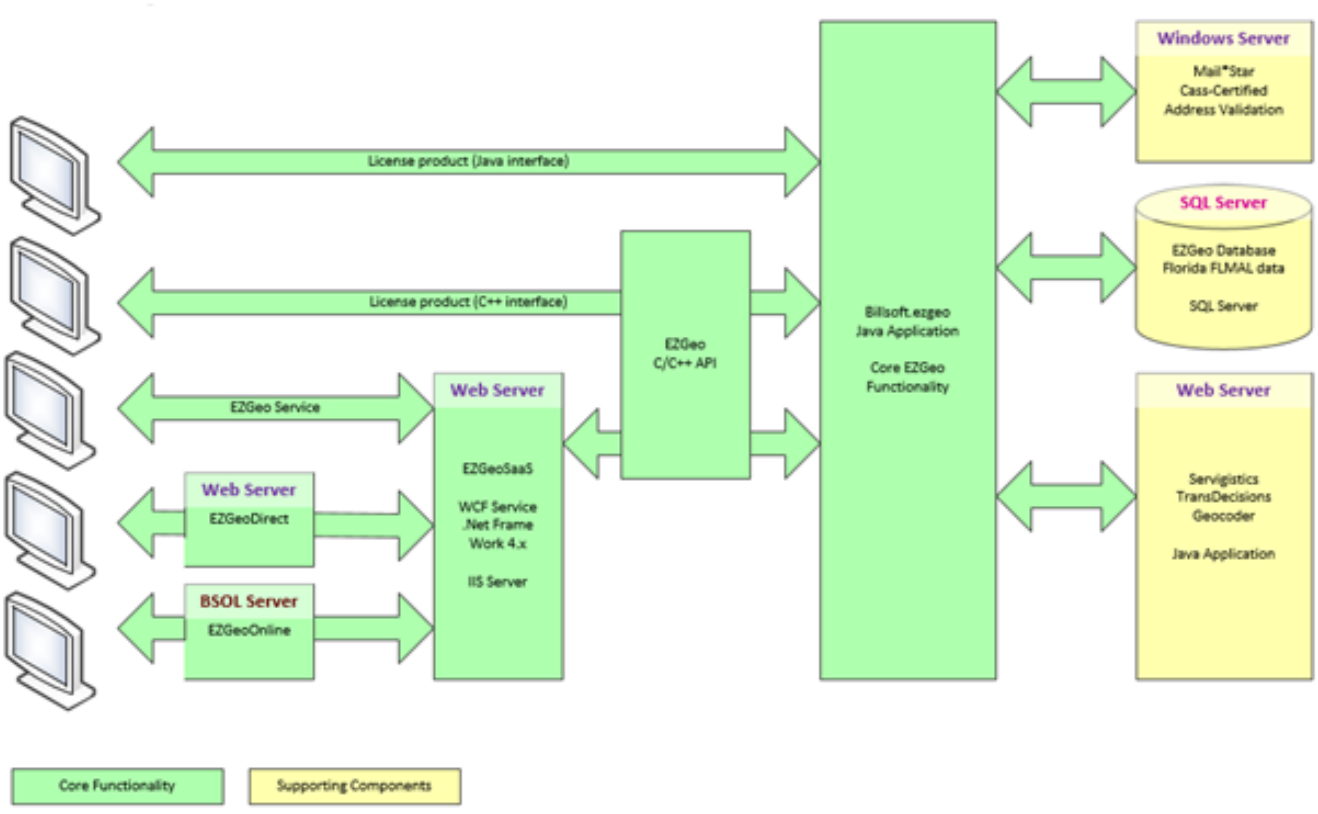
AFC Geo accepts input in the form of addresses or latitude/longitude values and processes (geocodes) them in order to return location and tax jurisdiction information. There are multiple processing methods by which the geocoding can be invoked, depending on client needs. Supporting components are utilized for address validation and matching.

1.3.1 AFC Geo Products

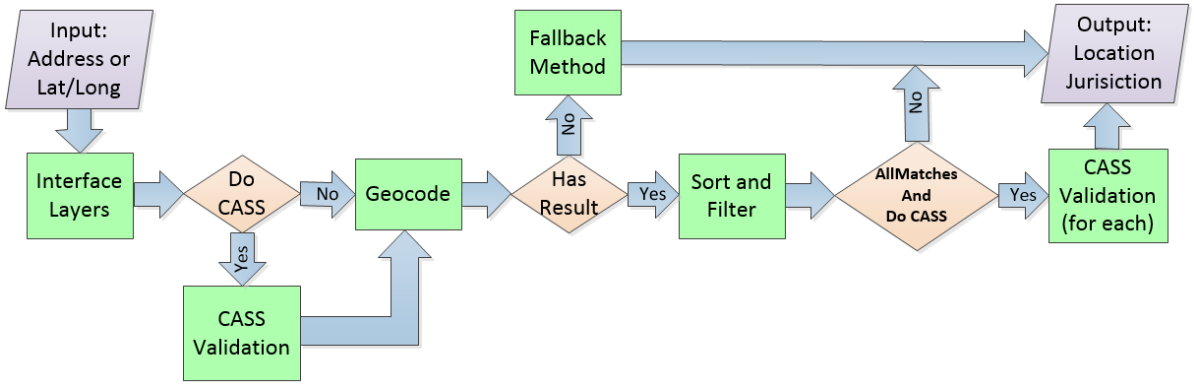
Avalara offers multiple AFC Geo products with different processing methods for its geocoding solution. These products are the following:

- ***AFC Geo SaaS Pro*** *(Software as a Service/Solution)* – hosted, web service solution for cloud-based interfaces that provides real-time processing.
- ***AFC Geo SaaS Standard*** – hosted, FTP batch processing solution.
- ***AFC Geo Direct*** – web site solution that allows one-by-one validation or lookups of addresses
- ***AFC Geo License –*** *site license solution, used onsite or at the client site, with software and address updates provided. Usually recommended for large transaction volumes.*

1.3.2 AFC Geo Architecture



1.3.3 AFC Geo Data Flow Process



The input to the AFC Geo process is either an address or a latitude/longitude pairing. The interface layers accept that input and pass it along for processing.

One of the input options is a Boolean flag indicating if the USPS CASS Validation (Coding Accuracy Support System) should be done. This process will ensure that the input address is as accurate as possible.

Next comes the Geocoding process which uses 3rd party components to return one or more existing locations from their databases. Those locations that score higher than the minimum score given in the input values will be returned.

If no locations are returned, and fallback options were set in the input, then the fallback method will be performed. The details of the fallback options are addressed later in this manual.

If there are matching locations, they are ranked according to their score and filtered based on source. Note that AFC Geo uses two sources for its geocoding: NavTeq and TeleAtlas. AFC Geo will prefer locations from the database which have been most recently updated. In addition, census tract and census block data is also featured and available in the geocoding process and/or results.

If a single result location is requested, the best location will be returned from the geocoding. On the other hand, if all matches above the minimum score were requested, then multiple locations will be returned.

If there are multiple locations returned, and CASS validation was requested, then each of the locations will then go through another validation step.

Finally, the resultant location and jurisdiction information is returned.

2. Getting Started

2.1 Technical/System Requirements

For most of the AFC Geo products, the technical/system requirements are more associated with secondary functionality, rather than AFC Geo specifically. In other words, AFC Geo SaaS Pro requires web services, AFC Geo SaaS Standard requires FTP and AFC Geo Direct requires a web browser. The only product that has explicit requirements is AFC Geo License.

2.1.1 AFC Geo License Software Requirements

Java Runtime Environment version 1.5.0 or higher.

2.1.2 AFC Geo License Hardware Requirements

512 MB RAM (1 GB recommended)

35GB disk space

DVD drive

2.2 Supported Platforms/Configurations

AFC Geo is supported on the following platforms:

- Windows 7, Windows 8, Windows Vista, Windows Server 2008, Windows Server 2012
- Red Hat Linux
- HPUX

3. AFC Geo Features & Functionality

AFC Geo has multiple products that use different methods or avenues through which its functionality can be accessed. Those methods have differences in how they operate with respect to the user or client's application, but, at the core, there is common functionality.

The four different AFC Geo products are as follows:

- AFC Geo SaaS Pro
- AFC Geo SaaS Standard
- AFC Geo Direct
- AFC Geo License

With AFC Geo SaaS Pro, clients can integrate geocoding functions directly into applications by invoking web services API calls across the internet. This method has the most functionality available, but also requires the most responsibility to code the client application. Sample applications are available to provide insight into how this coding can be done.

AFC Geo SaaS Standard is a batch file processing system for geocoding a set of inputs. Input batch files for either addresses or latitude/longitude values are uploaded to an FTP server. Once there, AFC Geo SaaS Standard will process them and produce a results file, which can then be downloaded.

AFC Geo Direct is an interactive web site application that allows for geocoding of one address at a time. It is a fairly simple application, but is useful for a limited number of inputs.

AFC Geo License is a full featured system that is completely hosted at the client's site. It is recommended for large transaction volumes.

Each of these four AFC Geo products will be further described in sections below. However, each of them also has common functionality that is described below in the remainder of this section.

3.1 Understanding Address Geocoding

AFC Geo is the geocoding locator, online solution of AFC that cleans and standardizes US customer addresses and identifies customer locations for taxing purposes. Address geocoding is the process of mapping an address to the street network, i.e., returning a location. The input into this process is a set of one or more strings.

Note: AFC Geo only supports geocoding for US, Puerto Rico and other US territory addresses. All addresses in foreign countries (including Canada and Mexico) are not supported and will not return any results.

3.1.1 Address Parsing

There are multiple input strings for an address to be geocoded. Each string can contain one or more sub-component values of the overall address. Those input strings and their meaning are as follows:

- **StreetAddress** – First line of address, (e.g. '8675 W 96th St')
- **SecondaryUnit** – Second line of address (e.g. 'Suite 220', 'Apt 100')
- **CityStateZip** – Combined values (e.g. 'Overland Park, KS 66212')
- **City** – Individual city name (e.g. 'Overland Park')
- **State** – Individual state abbreviation (e.g. 'KS')
- **Zip** – Individual zip code (e.g. '66212')

There are a number of variations that are allowed with these input strings. First of all, the **City**, **State**, and **Zip** can either be supplied as one combined string, or it can be supplied as individual strings.

You can also put the secondary suite or apartment number within the **StreetAddress**. In fact, the zip code (but not the city and state) can be included within the **StreetAddress**, making a single line address.

The following examples below provide another view on the input address field options.

- **StreetAddress**, **City**, **State**, **Zip**:

8675 W 96th St	Overland Park	KS	66212

- **StreetAddress**, **City**, **State**, **Zip**:

8675 W 96th St, Suite 220	Overland Park	KS	66212

- **StreetAddress**, **SecondaryUnit**, **City**, **State**, **Zip**:

8675 W 96th St	Suite 220	Overland Park	KS	66212

- **StreetAddress**, **CityStateZip**:

8675 W 96th St	Overland Park, KS 66212

- **StreetAddress**:

8675 W 96th St, 66212

(Requires CASS Validation option to be turned on)

There is no need to specify in which option the address fields are being supplied. The geocoding will be able to handle the address in any of the above formats.

Normally, a **StreetAddress** should start with a numeric street number for geocoding to occur. For example, '8675 Main Street' as the **StreetAddress** is acceptable. 'ABC Main Street' is not acceptable and will not produce a valid result. The one exception to this rule is for states that allow a one letter prefix on

the street. For example, G4386 S Saginaw St, Burton MI 48529-2001 is a valid address in Michigan. In this scenario, the letter prefix G will be removed, and the address 4386 S Saginaw St will be geocoded.

The use of a Post Office (PO) Box for tax determination geocoding is not recommended or supported, as PO Boxes are not considered acceptable by tax jurisdictions for tax situsing purposes. For example, StreetAddress 'PO Box 100' or 'P.O. Box 58912' will not be geocoded. Zip codes which are associated with a PO Box are not recommended for situsing because they represent the physical location of the PO Box, not of the situs of the actual transaction. Use of workarounds to circumvent AFC Geo PO Box validation rules is strongly discouraged, as invalid results will be returned for tax determination purposes.

An additional requirement for when a single line address is used, is that the CASS validation option must be turned on. This allows the zip code that is included in the Street Address field to be recognized.

The parser produces one or more interpretations of the input fields, in the form of associating values with database keys. For example, parsing 16 Tech Circle, 01760 as a single-line address will produce the following key/value combination:

Key	Value
Street Number	16
Street Name	Tech Circle
Secondary Unit	N/A
City Name	N/A
State Abbreviation	N/A
Postal Code	01760

AFC Geo uses a parser which is capable of producing multiple interpretations of the ambiguous input text. This is different from parsers used in other geocoders, which can only produce one parsing for the given text. For example, parsing 15 GA HWY 21,ATLANTA,GA,12345 will produce the following two interpretations:

Key	Interpretation 1	Interpretation 2
Street Number	15	15
Street Name	GA HWY 21	GA HWY
Secondary Unit	N/A	21
City Name	Atlanta	Atlanta
State Abbreviation	GA	GA
Postal Code	12345	12345

Indeed, the input address could be interpreted either as number 15 on Georgia Highway 21 or as number 15 on Georgia Highway, apartment 21. Both of these interpretations deserve consideration.

3.1.2 Matching

The input address is used to look up candidate locations within the street database. This process is similar to a traditional database query. At this stage, some of the keys are treated in a "relaxed" manner, in that a strict match is not required to produce a candidate. For example, streets, whose name *sounds like* the input street name, are considered candidates. Therefore, 1 Renee Street, 02134 will match 1 Rena Street, 02134. The algorithm used for matching the sounds is proprietary and is much more powerful than the popular, but relatively primitive, SOUNDEX algorithm.

Note that certain typos may result in street names that do not sound like the correct name. In this case, AFC Geo may not find a match.

If a street has alternate names (aliases) in the database, they will all be considered.

The street number is matched against the address ranges supplied in the vendor's street database. A single street is represented in the street database as a chain of segments called street links. A link is a small segment of the street, usually between two intersections, but sometimes a single block consists of multiple links. Each link is characterized by a range of street numbers on either of its sides. For example, the left side may have the street number range of 2 - 48, and the right side 1 - 47. A precise match occurs when the street number supplied for geocoding falls within the address range of a link. For example, if the street number supplied in the above example is 24, then AFC Geo assumes that the location is approximately in the middle of the link on the left side, and computes the geographic coordinates accordingly.

3.1.2.1 City + State vs. Postal Code

In addition to the mandatory street address, i.e., the house number and street name, a valid address must contain information about the locality. In the US, this information is represented by the city name, state abbreviation, and/or postal code. In many cases, an address can be unambiguously geocoded with only city and state, or only the zip code. In some cases, the client application does not supply a zip code, or the city name, etc. Here is how AFC Geo approaches these situations:

- Initially, the full address, including street, city, state and zip are used to geocode the address. If this produces a score that is above the minimum required score, then that is returned for the results.

- However, if no location is found that scored high enough, then it is possible that some of the input values were incorrect. Therefore, a geocoding with street address and zip code is attempted.

- If it still does not have a result that scores high enough, then it will attempt a geocoding with street address, city and state, but not zip code.

An important implication of this approach is that if *the right city and state, but the wrong postal code are supplied, or vice versa*, the match will still succeed. There are implications for scoring, in that a 100% score will not be achieved, but the correct location will be returned (unless it falls under the scoring threshold). This is because the scoring is relative to the number of input values used in geocoding, and therefore by not including incorrect values, the score can be increased.

One way to look at this process is as producing multiple interpretations, just like during the parsing stage. This, of course, has nothing to do with parsing, and everything to do with the virtual redundancy of city/state versus postal code.

For instance, the address 16 Tech Cir, Natick, MA, 12345 (wrong zip code) will produce the following two interpretations:

Key	Interpretation 1	Interpretation 2
Street Number	16	16
Street Name	Tech Cir	Tech Cir
City Name	Natick	N/A
State Abbreviation	MA	N/A
Postal Code	N/A	12345
Notes	Will match a location in postal code 01760	Will not match any locations

When the results are merged, the one correct location will be returned. If the two interpretations have each produced results, and the client application has used the AllMatches API call, then both sets of results will be scored and returned, and the client application is given an opportunity to select "the right one".

3.1.2.2 Fallback Modes

Some of the keys are normally treated in a strict manner. However, if the query fails to produce a match, AFC Geo enters the so-called fallback mode. The fallback mode can be thought of as relaxing certain criteria. There are two modes that can be independently turned on or off by the client application.

Each of the fallback modes results in a slight decrease of the performance of the query, which is why the application developer is given control over this behavior.

The most useful fallback mode is called Street Number Snapping (also called Number Out Of Range). Either an erroneously entered street number, or outdated street data can result in a situation when none of the address ranges contains the supplied street number. When the Street Number Snapping mode is on, AFC Geo finds the link with the address range closest to the supplied number, and returns that link. The location is "snapped" to the end of the link with the closest street number. For example, geocoding 200 Tech Cir, 01760 with Street Number Snapping on (on the GDT data set) will fall back to 98 Tech Cir, 01760, because 98 is the largest even street number present in the data set.

Another fallback mode is called Postal Code Centroid. If turned on, then when a location has not been found with a score above the minimum, the location of the center of the zip code area will be returned. This, of course, is just an approximation of the location of the address, but it could be sufficient for the application's purpose.

A final fallback mode that is done outside of the geocoding process uses a lookup based on zip code and city name. When this mode is done, the city name must match the lookup table. Therefore, no abbreviations should be used within the city name (e.g. don't use ST when it officially is SAINT). This mode also assigns a special score of 1.01, to indicate that the jurisdiction was assigned by the lookup fallback method, rather than through geocoding.

Note: All fallback modes are approximations, and therefore could produce results that are different than what are expected. For example, the city name lookup method assumes that a location is within the city limits, when the actual location may not be. Fallback methods are only invoked if no match can be found within the street database using normal geocoding methods, which is often caused by inaccurate data in the input address, such as a wrong zip code.

3.1.2.3 Scoring

Finally, all candidate locations are scored according to how well database values match specified input values. The score of 1.0, or 100%, results from an exact match between the inputs and the actual values. The score of 0.0 is a theoretical score that would be produced if nothing matched at all, although such locations never make it to the scoring phase.

An important point here is that the scoring is applied only to the keys that are specified by the calling application. The user is never "punished" by score for *not supplying* a certain key, such as the zip code. For instance, geocoding 16 Tech Cir, Natick, MA succeeds with a 100% score (even though the zip code is not specified). However, if the user does supply a value, then the scoring engine considers it. For instance, geocoding 16 Tech Cir, Natick, MA, 01700 (the zip code is incorrect here) succeeds with a 76% score.

The names, such as city and street names, are matched based on the algorithm known as Edit Distance. Edit Distance is essentially the minimum number of single-character changes, removals or additions required to convert one string into another. The more typos one makes, the larger the edit distance between the typed and the perfect values. The edit distance of 0 produces the perfect score, and so on. For instance, geocoding 1 Great Plaine ave, 02492 (the correct name is Great Plain) succeeds with an 84% score, and 1 Great Plane ave, 02492 results in an even worse, 79% score. Note, by the way, that these reasonable typos result in a relatively small decrease of the score.

The combined score for the location is computed as a weighted average of the individual scores by key.

In cases when the address is not able to be geocoded (i.e. the address is not available in the map data used by AFC Geo), AFC Geo will attempt to use other data sources in order to determine the appropriate taxing jurisdiction for the address. Mainly, AFC Geo will use a combination of the zip code, the zip+4 (if available in the input address), and the location name in order to look up the taxing jurisdiction for that

location. When this lookup succeeds, AFC Geo will return a value of 101% or 1.01. This special code indicates that the geocode data is not available, but the jurisdiction information was able to be retrieved anyway.

Note that a score 1.01 does not have any implications on the CASS validation performed by AFC Geo. The CASS validated address may still be returned in the output depending on whether the CASS validation succeeded or not. Therefore, you should continue to check if the CASS data is available in the output.

When the snapping is off, no match will be generated. There are links in the data set that do not have an address range assigned. This happens usually when the new street is digitized, but no houses have been built. Later, even when the street is populated, there may be a lag before the address ranges appear in the street data set. Links without address ranges are approached according to the following rules:

- Links without address ranges are not considered unless Street Number Snapping is on.
- If there is even a single street link with an address range, it is preferred to unassigned links.
- Certain streets, especially in new developments, may not have any address ranges assigned. Obviously, in this situation it is impossible to return the closest address range. In this situation, AFC Geo will return an arbitrary segment on the street, to at least get you in the neighborhood.

The other supported fallback is called *Postal Centroid*. It involves reverting to the centroid of a postal code area, when:

- Geocoding failed to produce any match at all;
- The postal code was supplied in the input address;
- The street data set contains at least one street in this postal code.

The centroid is a very coarse approximation of the real location.

3.2 Understanding Lat/Long Geocoding

AFC Geo has the ability to process the latitude/longitude of a particular location and return specific location information along with the Avalara PCode for taxation purposes. However, please make note that lat/long geocoding does not return complete addresses of the input location.

3.3 Special Tax Jurisdictions and Townships

Sometimes areas can be designated as requiring a tax for a specific purpose. These areas are called special tax jurisdictions, and can be applicable for entities such as Fire, Police, Ambulance, Library, Transportation, Economic Development, and Cultural and Science. The boundaries of these special tax jurisdictions can be arbitrary and not related to any other boundary such as city limits.

Also, some states utilize a level of government called Townships, which are in between cities and counties. In other words, townships segment the county into smaller sections. AFC Geo uses special tax jurisdictions to implement taxes at the township level when a township has different taxes than what is at the county level.

Therefore, to handle special tax jurisdictions, AFC Geo extends its standard city/county/state mapping functionality with custom mapped areas that correspond to the areas where those unique taxes are applicable.

When running AFC Geo, the return of special tax jurisdiction results is optional, and there is a Boolean flag that should be used when special tax jurisdiction results are desired. It is a best practice, however, to return special tax jurisdiction data, so normally this option will be turned on.

Special tax jurisdictions can sometimes cover areas outside the city limits of a town, but for tax calculation purposes, they are considered a type of local tax. Previously, that required checking to see if a special tax jurisdiction was applicable and explicitly setting the Incorporated flag going into AFC tax calculation functions. This is because AFC Geo only sets the Incorporated flag based on city limits. However, an enhancement to the tax calculation functions has been made such that it is no longer required to set the Incorporated flag to obtain special tax jurisdiction tax calculations.

Note: The Incorporated flag is used to specify whether the address involved in this transaction is inside or outside of the Local level designated as their location. The tax may or may not be affected by this designator depending upon whether or not the local level has taxes which would apply to the transaction/service type pair.

4. AFC Geo Best Practices

This section contains a list of best practices for using AFC Geo as recommended by Avalara. Each organization must carefully evaluate its own requirements when determining how to use AFC Geo with its applications.

4.1 Minimum Score

Client applications should always specify a minimum score for an address or check the score returned for any matching addresses before using the address. The minimum score must be between 0 and 1, inclusive. However, Avalara recommends not using addresses with a score of less than 0.7 (for 70%). This would allow for a reasonable number of typos when entering the address. This scoring requirement may be increased or decreased to allow for a higher or lower level accuracy, which may vary depending on the quality of the input data.

Note that for different minimum scores in combination with the scoring variations built into the system, AFC Geo can produce some differing scoring results for the same address. For example, if a minimum score of 0.7 is given for an address with an incorrect zip code, then the scoring variation that removes the incorrect zip code could produce a score above 0.7 and be returned. On the other hand, if a minimum score of 0.0 is given, then the incorrect zip code could be included in the full address scoring variation, but return a score that is less than 0.7.

4.2 Street Number Snapping

Avalara recommends that clients do not use street number snapping (Options bit flag of 1) as it almost always results in returning an incorrect street number. In many applications it may be acceptable to enable street number snapping to get the tax jurisdiction (FIPS Code or PCODE) and use the original street number. This would almost always be accurate unless there is a long section of the street with no address range assigned.

4.3 Postal Centroid

This option (Options bit flag of 2) is also not recommended by Avalara because the address returned will have no relationship to the input address other than they will both have the same zip code. In addition to an invalid address, there are also several cases where the tax jurisdiction returned will also be incorrect.

4.4 Special Tax Jurisdictions

It is highly recommended to set the option to return special tax jurisdictions (Options bit flag of 16). Special tax jurisdictions are applicable for taxes associated with fire, police, ambulance, library, roads, economic development, townships, etc. Without this option turned on, jurisdictions for only city or county will be returned.

4.5 Numeric Street Number

Normally, a StreetAddress should start with a numeric street number for geo coding to occur. For example, '8675 Main Street' as the StreetAddress is acceptable. 'ABC Main Street' is not acceptable and will not produce a valid result. The one exception to this rule is for states that allow a one letter prefix on the street. For example, G4386 S Saginaw St, Burton MI 48529-2001 is a valid address in Michigan. In this scenario, the letter prefix G will be removed, and the address 4386 S Saginaw St will be geocoded.

The use of a Post Office (PO) Box for tax determination geocoding is not recommended or supported, as PO Boxes are not considered acceptable by tax jurisdictions for tax situsing purposes. For example, StreetAddress 'PO Box 100' or 'P.O. Box 58912' will not be geocoded. Zip codes which are associated with

a PO Box are not recommended for situsing because they represent the physical location of the PO Box, not of the situs of the actual transaction. Use of workarounds to circumvent AFC Geo PO Box validation rules is strongly discouraged, as invalid results will be returned for tax determination purposes.

4.6 CASS Validation

CASS Validation (Coding Accuracy Support System) is a process by which a USPS database is used to ensure that the geocoded address is as accurate as possible. Therefore, it is a best practice to use this option to increase the accuracy of the results. Please note; however, that if the address is in Florida, CASS Validation will always be performed, no matter what the value of the input flag.

4.7 Secondary Unit

The input address has a field called Secondary Unit for data such as apartment or suite numbers. This level of information does not impact jurisdiction assignment, and therefore is not necessary. However, if it is supplied, it should be placed in the separate Secondary Unit field, rather than being concatenated to the end of the street address. The reason for this is because the Secondary Unit field is not used within geocoding, and therefore will not cause valid addresses to be missed. On the other hand, an apartment number on the street address line can cause the score to be lowered to the point that a valid match is missed.

4.8 Endpoints

AFC Geo SaaS Pro has two sets of endpoints. One set returns a NULL when there is a problem within the geocoding process. The other set of endpoints (ending in /2.0) return exceptions. It is a best practice to use the endpoints that return exceptions and to handle them within your code. Please reference the table below for examples of endpoints.

Binding	URL
Basic HTTP	https://ezgeoasp.billsoft.com/LocatorService.svc/2.0
	https://ezgeoasp.billsoft.com/LocatorService.svc
Custom	https://ezgeoasp.billsoft.com/LocatorService.svc/SSL/2.0
	https://ezgeoasp.billsoft.com/LocatorService.svc/SSL
WS HTTP	http://ezgeoasp.billsoft.com/LocatorService.svc/2.0
	http://ezgeoasp.billsoft.com/LocatorService.svc

5. AFC Geo SaaS Pro

AFC Geo SaaS Pro is an XML web service that provides address verification and assignment of the tax jurisdiction for US addresses. It is used when integrating into other client applications. Function calls are made over the internet, executed at a remote server, and results are returned to the calling program.

AFC Geo SaaS Pro is certified by the Florida Department of Revenue to assign the correct jurisdiction for the Florida Communications Service Tax.

AFC Geo SaaS Pro also provides the option to return a Coding Accuracy Support System (CASS) certified address. CASS certification is the process of normalizing addresses to USPS Publication 28 standards and validating the address against current USPS AMS-II data. All address elements are verified as well as adding ZIP code, ZIP+4, carrier route, delivery point and barcode detail. Customers using CASS certified addresses may be eligible for discounted postal rates. AFC Geo SaaS Pro uses the MailStar CASS product developed by AES Systems to perform the CASS certification. Please note; however, that if the address is in Florida, CASS validation will automatically be performed, no matter the value of the input option flag.

5.1 Using AFC Geo SaaS Pro

AFC Geo SaaS Pro was developed using XML, SOAP 1.1, WSDL and WS-Security so it can be integrated into virtually any application. To use the AFC Geo SaaS Pro web service from your application, you will need to create a proxy stub for your programming language and platform. The proxy stub will encapsulate many of the details of communicating over the Internet between your application and the web service. The proxy stub will contain data types, classes and functions that you will use in your source code to invoke the methods on the web service.

Most programming languages have a toolkit or SDK for generating some or all of the proxy stub. The following is a list of some of the products that may be used to create the proxy stub.

Toolkit or Product Name	Programming Language	Platform
Visual Studio.NET	C# or Visual Basic.NET	Microsoft Windows
.NET Framework SDK	C# or Visual Basic.NET	Microsoft Windows
Systinet Server	Java or C++	See http://www.systinet.com
Apache Axis	Java or C++	See http://ws.apache.org/axis/
GSOAP	C/C++	See http://sourceforge.net/projects/gsoap2

If possible, it is best to use a toolkit or SDK that supports the WS-Security specification. This will make it easier to generate the proxy stub.

5.2 Web Service Methods

The following is a list of web service methods that are supported by AFC Geo SaaS Pro:

GeocodeAddress – accepts a single address as input and returns the address in the street database that best matches the input data.

GeocodeAllMatches – accepts a single address as input and returns all addresses in the street database that matches the input data.

GeocodeLatLong - accepts the latitude and longitude of a location as input and returns jurisdiction information only; it does not return complete addresses.

GetServerTime – returns the current time on the web server. This may be necessary to resolve clock synchronization differences between the web service and the client.

For each match of Input Address, AFC Geo SaaS Pro returns the score of the match, the standardized address, the county the match resides within, the Feature ID, the FIPS code for the match, the Avalara PCode for the match, and whether the match is in an incorporated area. If more than one match is returned, the results will be ordered based on score with the best match coming first. For Latitude/Longitude Geocoding using the GeocodeLatLong method, AFC Geo SaaS Pro returns State, County, City, Feature ID, FIPS Code, Avalara PCode, and the Tax Jurisdiction name.

5.3 Web Service Data Types

5.3.1 Input Address

Class used to contain the input address information and other options which is used to geocode the address.

Properties:

StreetAddress – string containing the street number, street name and optional unit number. In most cases, this value must start with a numeric street number, or the address will not geocode.

CityStateZip – string containing the CityStateZip line of input address; ex – Overland Park, KS 66212. Either this combined field or the individual City, State, and Zip fields can be used.

City – string containing the city name.

State – string containing the two character state abbreviation.

Zip – string containing the postal code.

SecondaryUnit – a string value and is a part of the street address; such as an apartment, suite or any other unit type in USPS publication 28. This information can optionally appear in the StreetAddress field, but it helps the parsing accuracy if it is separated and place in this field.

MinimumScore – a double containing the minimum score that will qualify the address in the street database as a matching address. This value must be between 0 and 1, inclusive. However, a minimum score greater than zero should be used, because lower scores can produce matches that are quite divergent.

Offset – a double value; the distance in meters at which the resulting location is offset from the side of the link. The default is 3 meters.

Options – integer containing a bit mask of additional options to use when matching addresses in the street database.

The options field contains a bitwise combination of any of the following options. To use multiple options, add the option values together for all the options that should be enabled. For example, to enable the return of special tax jurisdictions (bitwise value 16) and census IDs (bitwise value 32) set the option field to 48 (16+32).

Note: The complete list of bitwise values currently supported are provided and listed in the following section below. Additional bitwise values are reserved for future use. Also, please note that geocoding option 8 for returning zip code extensions has been deprecated. There is no need to select this as an option as a zip extension will automatically be returned.

1. Fallback Options

   If a request fails to produce a match, the user can request AFC Geo SaaS Pro use a fallback mode. Specifying a fallback mode relaxes certain criteria and causes a slight decrease in the performance of the request.

   Street Number Snapping (Number Out Of Range). When this mode is on, AFC Geo SaaS Pro finds the location with the closest street number and returns that as a match. This mode is helpful when an incorrect street number has been entered or the location is in a new development and has not been incorporated into the street database yet. Street Number Snapping does not significantly affect the performance of the system.

   Option Value = 1

   Postal Code Centroid. If no match is found, the center of the postal code area is returned.

   Option Value = 2

2. Other Options

Return Zip+4. This option causes the postal code extension field to be populated with the zip+4 value for any matching addresses.

Option Value = 8

Note: Zip+4 is always returned. Even if the 8 is not included in the Option Value totals, the Zip+4 will be returned.

Return Special Tax Jurisdictions. This option causes AFC Geo to look up and return information for special tax jurisdictions, if one applies to the location. Special tax jurisdictions are for things such as libraries and schools, etc.

Option Value = 16

Return Census IDs. This option causes AFC Geo to look up and return information for census areas for the location. It will return the Block ID and the Tract ID for the location.

Option Value = 32

Return Building Zip Code. This option causes AFC Geo to look up and return the zip code specific to a building, if it is different than the street address.

Option Value = 256

CassCertify – a Boolean value indicating whether the address should be validated using the AES MailStar CASS Certified address validation application. If set to "true", the service attempts to populate the CassAddress property of the AddressLocation instance (or instances) returned by the GeoCode request. If set to "false", the validation is bypassed, unless the address is in Florida, in which case CASS validation will always be performed. Setting CassCertify to true does not guarantee a CassAddress will be returned. If CassCertify is set to true and the CassAddress returned is null, the address failed Cass validation.

Source - Note there is also a Source field included in the service reference definition for an InputAddress. This is an optional field, and is for internal use only. It should not be populated when making client API calls.

5.3.2 Address Location

Structure or class used to contain a verified address and its tax jurisdiction.

Properties:

NetworkID – string indicating which street database the matching address was found in. (ta = TeleAtlas, nt = NavTeq)

StreetNumber – string containing the street number of the matching address.

StreetName – string containing the street name of the matching address.

SecondaryUnit – string containing the unit number of the matching address.

CityName – string containing the city name of the matching address.

StateName - string containing the two letter state abbreviation of the matching address.

County – string containing the name of the county the matching address is within.

PostalCode – string containing the 5-digit postal code of the matching address.

PostalCodeExtension - string containing the 4-digit postal code extension of the matching address.

Score – double value between 0.0 and 1.0 indicating how good of a match the address is. The higher the score, the better the match.

Latitude – double containing the latitude of the matching address.

Longitude – double containing the longitude of the matching address.

CensusTract – integer containing the ID of the Census Tract, which is a contiguous group of census block groups, geocoded based on census data.

CensusBlockGroup – integer containing the ID of the individual Census Block, which is the smallest area surrounded by streets, geocoded based on census data.

> **NOTE:** Previously, the Census Block Group ID, which is a group of contiguous census blocks, was returned in this field; however, this has been updated to provide a value which reflects finer detail. In order to facilitate backward compatibility in the API, the field name will remain the same.
>
> For example, if the Block Group ID is 01 and the Block ID is 014, the value stored in this field will be the integer value 1014.

PCode – integer containing the Avalara PCode of the matching address. This PCode correlates to the set of taxes that are applicable to the address. In rare circumstances, the value of the

PCode could be returned as -1, which means it is unassigned. This could happen if there is no mapping from its FIPS Code to a PCode. If a PCode of -1 is returned, please contact Avalara support.

TaxJurisdictionName - string containing the jurisdiction name. This is the name of the jurisdiction identified by the PCode property.

SpecialTaxDistrictPCode – integer containing the Avalara PCode of the special tax jurisdiction if the address is within a special tax jurisdiction. Otherwise, this method returns -1. This field is for informational purposes only. For calculating taxes with AFC, the value returned by the PCode should always be used.

SpecialTaxDistrictName – string containing the name of the special tax jurisdiction.

PrimaryJurisdictionPCode – integer. If the location is within a special tax district, this property contains the Avalara PCode of the underlying tax district. This can be used to determine the jurisdiction and taxes for the location if the special tax jurisdiction did not exist. This PCode will always refer to a county or city. It will never be a PCode for a special tax jurisdiction. If the location is not within a special tax jurisdiction, this method has the same value as PCode. This field is for informational purposes only. For calculating taxes with AFC, the value returned by PCode should always be used.

PrimaryJurisdictionName – string. If the location is within a special tax jurisdiction, this property returns the name of the underlying tax district. This can be used to determine the jurisdiction for the location if the special tax jurisdiction did not exist. This will always refer to a county or city. It will never be the name of a special tax jurisdiction. If the location is not within a special tax jurisdiction, this method returns the same value as TaxJurisdictionName. This is the name of the jurisdiction identified by PrimaryJurisdictionPCode. This field is for informational purposes only.

FeatureID – integer The Geographic Names Information System (GNIS) feature id is a unique, permanent geographic feature identifier assigned by the U.S. Board on Geographic Names.

FIPSCode - long containing the FIPS code (Federal Information Processing Standards) of the matching address. FIPS codes are 10 digit numbers in the format of

SSCCCPPPPP where SS = State Code, CCC = County Code and PPPPP = Place Code. If the state code has a leading zero, that will be lost. If the location is in a special tax jurisdiction the FIPS code returned will be assigned by Avalara. Any FIPS codes that start with a "99" in the state location are Avalara, Inc. codes for special tax jurisdictions. This FIPS code or the PCode can be used in the AFC software product to calculate the correct taxes for the jurisdiction.

FIPSPlaceName - string containing the FIPS place name

UnderlyingFIPSCode - long containing the Federal FIPS code of the underlying jurisdiction. This method returns the Federal FIPS code of the underlying tax district. This can be used to determine the Federal FIPS code for the location if the special tax jurisdiction did not exist. This FIPS code will always refer to a county or city. It will never be a FIPS code for a special tax jurisdiction. If the location is not in a special tax jurisdiction, this method returns the same value as FIPS Code. This field is for informational purposes only. For calculating taxes with AFC, the value returned by FIPSCode should always be used.

Incorporated - Boolean indicating if the matching address is within the city limits of a town. When this value is returned as true, then the location is within the city limits. When false, the location is outside the city limits.

CassAddress - CassAddress object that contains the CASS address data if the CassCertify property of the InputAddress object was set to true. If the CassCertify property was set to false or the address could not be certified, this property will be set to null.

- Address - Returns the CASS certified address. Represented in a string array with as many lines of text as required by the standard way of printing the address on an envelope.
- Address Line - The street address portion of the address.
- City State Zip - A combined string with the city, state and zip code. The city, state and zip are separated by commas. The zip and zip+4 (if present) are separated by a hyphen.
- City - The USPS preferred city name.
- State - The state abbreviation.
- Zip - The 5-digit ZIP code.
- Zip4 - Returns the 4-digit ZIP+4. If the ZIP+4 is empty, this means that the address was not certified by CASS and may not be deliverable by the post office.
- Address Quality Flags - A string with the Address Quality Flags identified by the CASS software component. Refer to Appendices A and B in
- MailStar.doc in the docs2006 directory on the CASS DVD for the meaning of this field.
- Carrier Route - The postal carrier route for this address.
- County Code (integer) - The standard USPS code of the US county for this address.
- Delivery Point Validation - Delivery Point Validation data. Refer to maildpvelot.doc in the docs2006 directory of the CASS DVD for information on how to interpret the value of this field.
- Enhanced Line Of Travel - The position of this address in the carrier route walk sequence.
- Reliability (double) - A number, between 0 and 1, indicating the closeness of the match between a location address and USPS standards (higher is better).
- USPS Bar Code - A string of 12 numbers to be printed as a bar code for pre-sorting.

InputAddress – array of Strings; Input Address in an array of two strings. First string contains the street address along with the unit/apt number and the second string contains city, state and zip.

StandarizedAddress – array of Strings; Address matched by AFC Geo in an array of two strings. First string contains the street address along with the unit/apt number and the second string contains city, state and zip.

5.4 Locator Service Endpoint

LocatorService WSDL (secured with WS-Security) – This endpoint is recommended for applications that are developed using Microsoft .NET or other platforms that support WS-Security. The URL for this endpoint is https://EZgeoASP.Billsoft.com/LocatorService.svc.

The LocatorService endpoint uses WS-Security to authenticate requests and ensure the integrity of messages sent to the service. After registering as an AFC Geo SaaS Pro user, clients will be given a user name and password that will be used to access the web service.

The user name and password information will be included in the SOAP request using a UsernameToken with a password digest. The action, message id, reply to, id and timestamp headers and the body of the message should be signed to guarantee message integrity.

The WS-Security Specification is available online at: http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf

5.5 Example SOAP Messages

Geocoding requests through AFC Geo SaaS Pro are made using SOAP messages (Simple Object Access Protocol). Each message is written in XML (Extensible Markup Language), and follows a specific format. A few examples are provided here to help in understanding what should be sent to the service.

Section 5.5.1 contains SOAP examples using Basic HTTP bindings. However, SOAP examples using Custom bindings, which consist of minor changes to the Header section, are provided in Section 5.5.2.

Note also that the order of input fields is important. A requirement of SOAP is that there is a static order of input values, such as fields for inputAddress. If any of the fields is not in the given order, then they will remain unassigned, and will affect final geocoding results.

There is a Source field that is included in the SOAP reference definitions. This field is optional, and is used for internal purposes only, so it should be left out when making client API calls.

5.5.1 Basic Binding

5.5.1.1 GeocodeAddress

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:tem="http://tempuri.org/">
   <s:Header>
      <o:Security s:mustUnderstand="1" xmlns:o="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
         <o:UsernameToken>
            <o:Username>*****</o:Username>
            <o:Password>*****</o:Password>
         </o:UsernameToken>
      </o:Security>
   </s:Header>
       <s:Body>
             <GeocodeAddress xmlns="http://tempuri.org/">
                   <inputAddress
xmlns:a="http://schemas.datacontract.org/2004/07/EZGeoSaaS"
xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
                         <a:CassCertify>true</a:CassCertify>
                         <a:City>Overland Park</a:City>
                         <a:CityStateZip/>
                         <a:MinimumScore>0.7</a:MinimumScore>
                         <a:Offset>3</a:Offset>
                         <a:Options>312</a:Options>
                         <a:SecondaryUnit>Suite 220</a:SecondaryUnit>
                         <a:State>KS</a:State>
                         <a:StreetAddress>8675 West 96th Street</a:StreetAddress>
                         <a:Zip>66212</a:Zip>
                   </inputAddress>
             </GeocodeAddress>
      </s:Body>
</s:Envelope>
```

5.5.1.2 GeocodeAddress Response

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:u="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-
1.0.xsd">
   <s:Header>
      <o:Security s:mustUnderstand="1" xmlns:o="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
         <u:Timestamp u:Id="_0">
            <u:Created>2016-08-08T18:18:10.670Z</u:Created>
            <u:Expires>2016-08-08T18:23:10.670Z</u:Expires>
         </u:Timestamp>
      </o:Security>
   </s:Header>
   <s:Body>
      <GeocodeAddressResponse xmlns="http://tempuri.org/">
         <GeocodeAddressResult
xmlns:a="http://schemas.datacontract.org/2004/07/EZGeoSaaS"
xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
            <a:Alternate>0</a:Alternate>
            <a:CassAddress>
               <a:Address
xmlns:b="http://schemas.microsoft.com/2003/10/Serialization/Arrays">
                  <b:string>8675 W 96TH ST Suite 220</b:string>
```

```
                  <b:string>OVERLAND PARK, KS  66212-3316</b:string>
               </a:Address>
               <a:AddressLine>8675 W 96TH ST Suite 220</a:AddressLine>
               <a:AddressQualityFlags>S4B</a:AddressQualityFlags>
               <a:CarrierRoute>C027</a:CarrierRoute>
               <a:City>OVERLAND PARK</a:City>
               <a:CityStateZip>OVERLAND PARK, KS  66212-3316</a:CityStateZip>
               <a:CountyCode>91</a:CountyCode>
               <a:DeliveryPointValidation>3</a:DeliveryPointValidation>
               <a:EnhancedLineOfTravel>0</a:EnhancedLineOfTravel>
               <a:Reliability>100</a:Reliability>
               <a:State>KS</a:State>
               <a:USPSBarCode>/662123316758/</a:USPSBarCode>
               <a:Zip>66212</a:Zip>
               <a:Zip4>3316</a:Zip4>
            </a:CassAddress>
            <a:CensusBlockGroup>1014</a:CensusBlockGroup>
            <a:CensusTract>51804</a:CensusTract>
            <a:CityName>OVERLAND PARK</a:CityName>
            <a:Country>USA</a:Country>
            <a:County>JOHNSON</a:County>
            <a:ErrorMessage/>
            <a:FeatureID>479210</a:FeatureID>
            <a:FipsCode>2009153775</a:FipsCode>
            <a:FipsPlaceName>OVERLAND PARK</a:FipsPlaceName>
            <a:Incorporated>true</a:Incorporated>
            <a:InputAddress
xmlns:b="http://schemas.microsoft.com/2003/10/Serialization/Arrays">
               <b:string>8675 W 96TH ST</b:string>
               <b:string>OVERLAND PARK,KS 66212</b:string>
            </a:InputAddress>
            <a:Latitude>38.95465893143988</a:Latitude>
            <a:Longitude>-94.685748695942991</a:Longitude>
            <a:NetworkID>nt</a:NetworkID>
            <a:PCode>1248900</a:PCode>
            <a:PostalCode>66212</a:PostalCode>
            <a:PostalCodeExtension>3316</a:PostalCodeExtension>
            <a:PrimaryJurisdictionName>OVERLAND PARK</a:PrimaryJurisdictionName>
            <a:PrimaryJurisdictionPCode>1248900</a:PrimaryJurisdictionPCode>
            <a:Score>1</a:Score>
            <a:SecondaryUnit>Suite 220</a:SecondaryUnit>
            <a:SpecialTaxDistrictName i:nil="true"/>
            <a:SpecialTaxDistrictPCode>-1</a:SpecialTaxDistrictPCode>
            <a:StandardizedAddress
xmlns:b="http://schemas.microsoft.com/2003/10/Serialization/Arrays">
               <b:string>8675 W 96TH ST Suite 220</b:string>
               <b:string>OVERLAND PARK,KS 66212-3316</b:string>
            </a:StandardizedAddress>
            <a:StateName>KS</a:StateName>
            <a:StreetName>W 96TH ST</a:StreetName>
            <a:StreetNumber>8675</a:StreetNumber>
            <a:TaxJurisdictionName>OVERLAND PARK</a:TaxJurisdictionName>
            <a:TimeZone/>
            <a:UnderlyingFipsCode>2009153775</a:UnderlyingFipsCode>
         </GeocodeAddressResult>
      </GeocodeAddressResponse>
   </s:Body>
</s:Envelope>
```

5.5.1.3 GeocodeAllMatches

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:tem="http://tempuri.org/">
   <s:Header>
      <o:Security s:mustUnderstand="1" xmlns:o="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
         <o:UsernameToken>
            <o:Username>*****</o:Username>
            <o:Password>*****</o:Password>
         </o:UsernameToken>
      </o:Security>
   </s:Header>
       <s:Body>
             <GeocodeAllMatches xmlns="http://tempuri.org/">
                   <inputAddress
xmlns:a="http://schemas.datacontract.org/2004/07/EZGeoSaaS"
xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
                         <a:CassCertify>true</a:CassCertify>
                         <a:City>Clawson</a:City>
                         <a:CityStateZip/>
                         <a:MinimumScore>0.7</a:MinimumScore>
                         <a:Offset>3</a:Offset>
                         <a:Options>312</a:Options>
                         <a:SecondaryUnit/>
                         <a:State>MI</a:State>
                         <a:StreetAddress>905 E Maple</a:StreetAddress>
                         <a:Zip>48083</a:Zip>
                   </inputAddress>
                   <matchCount>10</matchCount>
             </GeocodeAllMatches>
      </s:Body>
</s:Envelope>
```

5.5.1.4 GeocodeAllMatchesResponse

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:u="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-
1.0.xsd">
   <s:Header>
      <o:Security s:mustUnderstand="1" xmlns:o="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
         <u:Timestamp u:Id="_0">
            <u:Created>2016-08-08T19:00:12.269Z</u:Created>
            <u:Expires>2016-08-08T19:05:12.269Z</u:Expires>
         </u:Timestamp>
      </o:Security>
   </s:Header>
   <s:Body>
      <GeocodeAllMatchesResponse xmlns="http://tempuri.org/">
         <GeocodeAllMatchesResult
xmlns:a="http://schemas.datacontract.org/2004/07/EZGeoSaaS"
xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
            <a:AddressLocation>
               <a:Alternate>0</a:Alternate>
               <a:CassAddress>
                  <a:Address
xmlns:b="http://schemas.microsoft.com/2003/10/Serialization/Arrays">
                     <b:string>905 E MAPLE RD</b:string>
                     <b:string>TROY, MI  48083</b:string>
                  </a:Address>
```

```
                  <a:AddressLine>905 E MAPLE RD</a:AddressLine>
                  <a:AddressQualityFlags>S1:zNCE</a:AddressQualityFlags>
                  <a:CarrierRoute>C099</a:CarrierRoute>
                  <a:City>TROY</a:City>
                  <a:CityStateZip>TROY, MI  48083</a:CityStateZip>
                  <a:CountyCode>125</a:CountyCode>
                  <a:DeliveryPointValidation>64</a:DeliveryPointValidation>
                  <a:EnhancedLineOfTravel>0</a:EnhancedLineOfTravel>
                  <a:Reliability>91</a:Reliability>
                  <a:State>MI</a:State>
                  <a:USPSBarCode>/48083/</a:USPSBarCode>
                  <a:Zip>48083</a:Zip>
                  <a:Zip4/>
               </a:CassAddress>
               <a:CensusBlockGroup>3001</a:CensusBlockGroup>
               <a:CensusTract>197500</a:CensusTract>
               <a:CityName>TROY</a:CityName>
               <a:Country>USA</a:Country>
               <a:County>OAKLAND</a:County>
               <a:ErrorMessage/>
               <a:FeatureID>1615125</a:FeatureID>
               <a:FipsCode>2612580700</a:FipsCode>
               <a:FipsPlaceName>TROY</a:FipsPlaceName>
               <a:Incorporated>true</a:Incorporated>
               <a:InputAddress
xmlns:b="http://schemas.microsoft.com/2003/10/Serialization/Arrays">
                  <b:string>905 E MAPLE RD</b:string>
                  <b:string>TROY,MI 48083</b:string>
               </a:InputAddress>
               <a:Latitude>42.548676318717291</a:Latitude>
               <a:Longitude>-83.1326049289924</a:Longitude>
               <a:NetworkID>nt</a:NetworkID>
               <a:PCode>1833400</a:PCode>
               <a:PostalCode>48083</a:PostalCode>
               <a:PostalCodeExtension/>
               <a:PrimaryJurisdictionName>TROY</a:PrimaryJurisdictionName>
               <a:PrimaryJurisdictionPCode>1833400</a:PrimaryJurisdictionPCode>
               <a:Score>1</a:Score>
               <a:SecondaryUnit/>
               <a:SpecialTaxDistrictName i:nil="true"/>
               <a:SpecialTaxDistrictPCode>-1</a:SpecialTaxDistrictPCode>
               <a:StandardizedAddress
xmlns:b="http://schemas.microsoft.com/2003/10/Serialization/Arrays">
                  <b:string>905 E MAPLE RD</b:string>
                  <b:string>TROY,MI 48083</b:string>
               </a:StandardizedAddress>
               <a:StateName>MI</a:StateName>
               <a:StreetName>E MAPLE RD</a:StreetName>
               <a:StreetNumber>905</a:StreetNumber>
               <a:TaxJurisdictionName>TROY</a:TaxJurisdictionName>
               <a:TimeZone/>
               <a:UnderlyingFipsCode>2612580700</a:UnderlyingFipsCode>
            </a:AddressLocation>
            <a:AddressLocation>
               <a:Alternate>0</a:Alternate>
               <a:CassAddress>
                  <a:Address
xmlns:b="http://schemas.microsoft.com/2003/10/Serialization/Arrays">
                     <b:string>905 E MAPLE RD</b:string>
                     <b:string>TROY, MI  48083</b:string>
                  </a:Address>
                  <a:AddressLine>905 E MAPLE RD</a:AddressLine>
                  <a:AddressQualityFlags>S1NZPK5</a:AddressQualityFlags>
```

```
                  <a:CarrierRoute>C099</a:CarrierRoute>
                  <a:City>TROY</a:City>
                  <a:CityStateZip>TROY, MI  48083</a:CityStateZip>
                  <a:CountyCode>125</a:CountyCode>
                  <a:DeliveryPointValidation>64</a:DeliveryPointValidation>
                  <a:EnhancedLineOfTravel>0</a:EnhancedLineOfTravel>
                  <a:Reliability>92</a:Reliability>
                  <a:State>MI</a:State>
                  <a:USPSBarCode>/48083/</a:USPSBarCode>
                  <a:Zip>48083</a:Zip>
                  <a:Zip4/>
               </a:CassAddress>
               <a:CensusBlockGroup>2001</a:CensusBlockGroup>
               <a:CensusTract>180000</a:CensusTract>
               <a:CityName>TROY</a:CityName>
               <a:Country>USA</a:Country>
               <a:County>OAKLAND</a:County>
               <a:ErrorMessage/>
               <a:FeatureID>1626089</a:FeatureID>
               <a:FipsCode>2612516160</a:FipsCode>
               <a:FipsPlaceName>CLAWSON</a:FipsPlaceName>
               <a:Incorporated>true</a:Incorporated>
               <a:InputAddress
xmlns:b="http://schemas.microsoft.com/2003/10/Serialization/Arrays">
                  <b:string>905 E MAPLE RD</b:string>
                  <b:string>TROY,MI 48083</b:string>
               </a:InputAddress>
               <a:Latitude>42.548072269155561</a:Latitude>
               <a:Longitude>-83.158629478188061</a:Longitude>
               <a:NetworkID>nt</a:NetworkID>
               <a:PCode>1831400</a:PCode>
               <a:PostalCode>48084</a:PostalCode>
               <a:PostalCodeExtension/>
               <a:PrimaryJurisdictionName>CLAWSON</a:PrimaryJurisdictionName>
               <a:PrimaryJurisdictionPCode>1831400</a:PrimaryJurisdictionPCode>
               <a:Score>0.71428573131561279</a:Score>
               <a:SecondaryUnit/>
               <a:SpecialTaxDistrictName i:nil="true"/>
               <a:SpecialTaxDistrictPCode>-1</a:SpecialTaxDistrictPCode>
               <a:StandardizedAddress
xmlns:b="http://schemas.microsoft.com/2003/10/Serialization/Arrays">
                  <b:string>905 W MAPLE RD</b:string>
                  <b:string>TROY,MI 48084</b:string>
               </a:StandardizedAddress>
               <a:StateName>MI</a:StateName>
               <a:StreetName>W MAPLE RD</a:StreetName>
               <a:StreetNumber>905</a:StreetNumber>
               <a:TaxJurisdictionName>CLAWSON</a:TaxJurisdictionName>
               <a:TimeZone/>
               <a:UnderlyingFipsCode>2612516160</a:UnderlyingFipsCode>
            </a:AddressLocation>
         </GeocodeAllMatchesResult>
      </GeocodeAllMatchesResponse>
   </s:Body>
```

5.5.1.5 GeocodeLatLong

```
  </s:Envelope><s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:tem="http://tempuri.org/">
   <s:Header>
```

```
      <o:Security s:mustUnderstand="1" xmlns:o="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
         <o:UsernameToken>
            <o:Username>*****</o:Username>
            <o:Password>*****</o:Password>
         </o:UsernameToken>
      </o:Security>
   </s:Header>
   <s:Body>
      <tem:GeocodeLatLong>
         <tem:latitude>38.956689</tem:latitude>
         <tem:longitude>-94.686252</tem:longitude>
      </tem:GeocodeLatLong>
   </s:Body>
</s:Envelope>
```

5.5.1.6 GeocodeLatLongResponse

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:u="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-
1.0.xsd">
   <s:Header>
      <o:Security s:mustUnderstand="1" xmlns:o="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
         <u:Timestamp u:Id="_0">
            <u:Created>2016-08-08T16:58:54.426Z</u:Created>
            <u:Expires>2016-08-08T17:03:54.426Z</u:Expires>
         </u:Timestamp>
      </o:Security>
   </s:Header>
   <s:Body>
      <GeocodeLatLongResponse xmlns="http://tempuri.org/">
         <GeocodeLatLongResult
xmlns:a="http://schemas.datacontract.org/2004/07/EZGeoSaaS"
xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
            <a:Alternate>0</a:Alternate>
            <a:CassAddress i:nil="true"/>
            <a:CensusBlockGroup>0</a:CensusBlockGroup>
            <a:CensusTract>0</a:CensusTract>
            <a:CityName>OVERLAND PARK</a:CityName>
            <a:Country>USA</a:Country>
            <a:County>JOHNSON</a:County>
            <a:ErrorMessage/>
            <a:FeatureID>479210</a:FeatureID>
            <a:FipsCode>2009153775</a:FipsCode>
            <a:FipsPlaceName>OVERLAND PARK</a:FipsPlaceName>
            <a:Incorporated>true</a:Incorporated>
            <a:InputAddress
xmlns:b="http://schemas.microsoft.com/2003/10/Serialization/Arrays">
               <b:string/>
               <b:string/>
            </a:InputAddress>
            <a:Latitude>38.956689</a:Latitude>
            <a:Longitude>-94.686252</a:Longitude>
            <a:NetworkID>nt</a:NetworkID>
            <a:PCode>1248900</a:PCode>
            <a:PostalCode/>
            <a:PostalCodeExtension/>
            <a:PrimaryJurisdictionName>OVERLAND PARK</a:PrimaryJurisdictionName>
            <a:PrimaryJurisdictionPCode>1248900</a:PrimaryJurisdictionPCode>
            <a:Score>0.999961162858618</a:Score>
```

```
            <a:SecondaryUnit/>
            <a:SpecialTaxDistrictName i:nil="true"/>
            <a:SpecialTaxDistrictPCode>-1</a:SpecialTaxDistrictPCode>
            <a:StandardizedAddress
xmlns:b="http://schemas.microsoft.com/2003/10/Serialization/Arrays">
               <b:string></b:string>
               <b:string>OVERLAND PARK,KS null</b:string>
            </a:StandardizedAddress>
            <a:StateName>KS</a:StateName>
            <a:StreetName/>
            <a:StreetNumber/>
            <a:TaxJurisdictionName>OVERLAND PARK</a:TaxJurisdictionName>
            <a:TimeZone/>
            <a:UnderlyingFipsCode>2009153775</a:UnderlyingFipsCode>
         </GeocodeLatLongResult>
      </GeocodeLatLongResponse>
   </s:Body>
</s:Envelope>
```

5.5.1.7 GetServerTime

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:tem="http://tempuri.org/">
   <s:Header>
      <o:Security s:mustUnderstand="1" xmlns:o="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
         <o:UsernameToken>
            <o:Username>*****</o:Username>
            <o:Password>*****</o:Password>
         </o:UsernameToken>
      </o:Security>
   </s:Header>
   <s:Body>
     <tem:GetServerTime/>
   </s:Body>
</s:Envelope>
```

5.5.1.8 GetServerTimeResponse

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:u="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-
1.0.xsd">
   <s:Header>
      <o:Security s:mustUnderstand="1" xmlns:o="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
         <u:Timestamp u:Id="_0">
            <u:Created>2016-08-08T18:14:56.607Z</u:Created>
            <u:Expires>2016-08-08T18:19:56.607Z</u:Expires>
         </u:Timestamp>
      </o:Security>
   </s:Header>
   <s:Body>
      <GetServerTimeResponse xmlns="http://tempuri.org/">
         <GetServerTimeResult>Aug 8, 2016 1:14:56 PM</GetServerTimeResult>
      </GetServerTimeResponse>
   </s:Body>
</s:Envelope>
```

5.5.2 Custom Binding

5.5.2.1 GeocodeAddress

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
xmlns:tem="http://tempuri.org/"
xmlns:ezg="http://schemas.datacontract.org/2004/07/EZGeoSaaS">
       <soap:Header>
              <UserName xmlns="EZGeo">*****</UserName>
              <Password xmlns="EZGeo">*****</Password>
       </soap:Header>
       <soap:Body>
              <tem:GeocodeAddress>
                     <tem:inputAddress>
                            <ezg:CassCertify>true</ezg:CassCertify>
                            <ezg:City>Overland Park</ezg:City>
                            <ezg:CityStateZip/>
                            <ezg:MinimumScore>0.7</ezg:MinimumScore>
                            <ezg:Offset>3</ezg:Offset>
                            <ezg:Options>312</ezg:Options>
                            <ezg:SecondaryUnit>Suite 220</ezg:SecondaryUnit>
                            <ezg:State>KS</ezg:State>
                            <ezg:StreetAddress>8675 West 96th
Street</ezg:StreetAddress>
                            <ezg:Zip>66212</ezg:Zip>
                     </tem:inputAddress>
              </tem:GeocodeAddress>
       </soap:Body>
</soap:Envelope>
```

5.5.2.2 GeocodeAddressResponse

```
<s:Envelope xmlns:s="http://www.w3.org/2003/05/soap-envelope">
   <s:Body>
      <GeocodeAddressResponse xmlns="http://tempuri.org/">
         <GeocodeAddressResult
xmlns:a="http://schemas.datacontract.org/2004/07/EZGeoSaaS"
xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
            <a:Alternate>0</a:Alternate>
            <a:CassAddress>
               <a:Address
xmlns:b="http://schemas.microsoft.com/2003/10/Serialization/Arrays">
                  <b:string>8675 W 96TH ST Suite 220</b:string>
                  <b:string>OVERLAND PARK, KS  66212-3316</b:string>
               </a:Address>
               <a:AddressLine>8675 W 96TH ST Suite 220</a:AddressLine>
               <a:AddressQualityFlags>S4B</a:AddressQualityFlags>
               <a:CarrierRoute>C027</a:CarrierRoute>
               <a:City>OVERLAND PARK</a:City>
               <a:CityStateZip>OVERLAND PARK, KS  66212-3316</a:CityStateZip>
               <a:CountyCode>91</a:CountyCode>
               <a:DeliveryPointValidation>3</a:DeliveryPointValidation>
               <a:EnhancedLineOfTravel>0</a:EnhancedLineOfTravel>
               <a:Reliability>100</a:Reliability>
               <a:State>KS</a:State>
               <a:USPSBarCode>/662123316758/</a:USPSBarCode>
               <a:Zip>66212</a:Zip>
               <a:Zip4>3316</a:Zip4>
            </a:CassAddress>
            <a:CensusBlockGroup>1014</a:CensusBlockGroup>
```

```
            <a:CensusTract>51804</a:CensusTract>
            <a:CityName>OVERLAND PARK</a:CityName>
            <a:Country>USA</a:Country>
            <a:County>JOHNSON</a:County>
            <a:ErrorMessage/>
            <a:FeatureID>479210</a:FeatureID>
            <a:FipsCode>2009153775</a:FipsCode>
            <a:FipsPlaceName>OVERLAND PARK</a:FipsPlaceName>
            <a:Incorporated>true</a:Incorporated>
            <a:InputAddress
xmlns:b="http://schemas.microsoft.com/2003/10/Serialization/Arrays">
               <b:string>8675 W 96TH ST</b:string>
               <b:string>OVERLAND PARK,KS 66212</b:string>
            </a:InputAddress>
            <a:Latitude>38.95465893143988</a:Latitude>
            <a:Longitude>-94.685748695942991</a:Longitude>
            <a:NetworkID>nt</a:NetworkID>
            <a:PCode>1248900</a:PCode>
            <a:PostalCode>66212</a:PostalCode>
            <a:PostalCodeExtension>3316</a:PostalCodeExtension>
            <a:PrimaryJurisdictionName>OVERLAND PARK</a:PrimaryJurisdictionName>
            <a:PrimaryJurisdictionPCode>1248900</a:PrimaryJurisdictionPCode>
            <a:Score>1</a:Score>
            <a:SecondaryUnit>Suite 220</a:SecondaryUnit>
            <a:SpecialTaxDistrictName i:nil="true"/>
            <a:SpecialTaxDistrictPCode>-1</a:SpecialTaxDistrictPCode>
            <a:StandardizedAddress
xmlns:b="http://schemas.microsoft.com/2003/10/Serialization/Arrays">
               <b:string>8675 W 96TH ST Suite 220</b:string>
               <b:string>OVERLAND PARK,KS 66212-3316</b:string>
            </a:StandardizedAddress>
            <a:StateName>KS</a:StateName>
            <a:StreetName>W 96TH ST</a:StreetName>
            <a:StreetNumber>8675</a:StreetNumber>
            <a:TaxJurisdictionName>OVERLAND PARK</a:TaxJurisdictionName>
            <a:TimeZone/>
            <a:UnderlyingFipsCode>2009153775</a:UnderlyingFipsCode>
         </GeocodeAddressResult>
      </GeocodeAddressResponse>
   </s:Body>
</s:Envelope>
```

5.5.2.3 GeocodeAllMatches

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
xmlns:tem="http://tempuri.org/"
xmlns:ezg="http://schemas.datacontract.org/2004/07/EZGeoSaaS">
       <soap:Header>
              <UserName xmlns="EZGeo">*****</UserName>
              <Password xmlns="EZGeo">*****</Password>
       </soap:Header>
       <soap:Body>
              <tem:GeocodeAllMatches>
                     <tem:inputAddress>
                            <ezg:CassCertify>true</ezg:CassCertify>
                            <ezg:City>Clawson</ezg:City>
                            <ezg:CityStateZip/>
                            <ezg:MinimumScore>0.7</ezg:MinimumScore>
                            <ezg:Offset>3</ezg:Offset>
                            <ezg:Options>312</ezg:Options>
                            <ezg:SecondaryUnit/>
```

```
                        <ezg:State>MI</ezg:State>
                        <ezg:StreetAddress>905 E Maple</ezg:StreetAddress>
                        <ezg:Zip>48083</ezg:Zip>
                    </tem:inputAddress>
                    <tem:matchCount>10</tem:matchCount>
                </tem:GeocodeAllMatches>
        </soap:Body>
</soap:Envelope>
```

5.5.2.4 GeocodeAllMatchesResponse

```
<s:Envelope xmlns:s="http://www.w3.org/2003/05/soap-envelope">
   <s:Body>
      <GeocodeAllMatchesResponse xmlns="http://tempuri.org/">
         <GeocodeAllMatchesResult
xmlns:a="http://schemas.datacontract.org/2004/07/EZGeoSaaS"
xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
            <a:AddressLocation>
               <a:Alternate>0</a:Alternate>
               <a:CassAddress>
                  <a:Address
xmlns:b="http://schemas.microsoft.com/2003/10/Serialization/Arrays">
                     <b:string>905 E MAPLE RD</b:string>
                     <b:string>TROY, MI  48083</b:string>
                  </a:Address>
                  <a:AddressLine>905 E MAPLE RD</a:AddressLine>
                  <a:AddressQualityFlags>S1:zNCE</a:AddressQualityFlags>
                  <a:CarrierRoute>C099</a:CarrierRoute>
                  <a:City>TROY</a:City>
                  <a:CityStateZip>TROY, MI  48083</a:CityStateZip>
                  <a:CountyCode>125</a:CountyCode>
                  <a:DeliveryPointValidation>64</a:DeliveryPointValidation>
                  <a:EnhancedLineOfTravel>0</a:EnhancedLineOfTravel>
                  <a:Reliability>91</a:Reliability>
                  <a:State>MI</a:State>
                  <a:USPSBarCode>/48083/</a:USPSBarCode>
                  <a:Zip>48083</a:Zip>
                  <a:Zip4/>
               </a:CassAddress>
               <a:CensusBlockGroup>3001</a:CensusBlockGroup>
               <a:CensusTract>197500</a:CensusTract>
               <a:CityName>TROY</a:CityName>
               <a:Country>USA</a:Country>
               <a:County>OAKLAND</a:County>
               <a:ErrorMessage/>
               <a:FeatureID>1615125</a:FeatureID>
               <a:FipsCode>2612580700</a:FipsCode>
               <a:FipsPlaceName>TROY</a:FipsPlaceName>
               <a:Incorporated>true</a:Incorporated>
               <a:InputAddress
xmlns:b="http://schemas.microsoft.com/2003/10/Serialization/Arrays">
                  <b:string>905 E MAPLE RD</b:string>
                  <b:string>TROY,MI 48083</b:string>
               </a:InputAddress>
               <a:Latitude>42.548676318717291</a:Latitude>
               <a:Longitude>-83.1326049289924</a:Longitude>
               <a:NetworkID>nt</a:NetworkID>
               <a:PCode>1833400</a:PCode>
               <a:PostalCode>48083</a:PostalCode>
```

```
                <a:PostalCodeExtension/>
                <a:PrimaryJurisdictionName>TROY</a:PrimaryJurisdictionName>
                <a:PrimaryJurisdictionPCode>1833400</a:PrimaryJurisdictionPCode>
                <a:Score>1</a:Score>
                <a:SecondaryUnit/>
                <a:SpecialTaxDistrictName i:nil="true"/>
                <a:SpecialTaxDistrictPCode>-1</a:SpecialTaxDistrictPCode>
                <a:StandardizedAddress
xmlns:b="http://schemas.microsoft.com/2003/10/Serialization/Arrays">
                   <b:string>905 E MAPLE RD</b:string>
                   <b:string>TROY,MI 48083</b:string>
                </a:StandardizedAddress>
                <a:StateName>MI</a:StateName>
                <a:StreetName>E MAPLE RD</a:StreetName>
                <a:StreetNumber>905</a:StreetNumber>
                <a:TaxJurisdictionName>TROY</a:TaxJurisdictionName>
                <a:TimeZone/>
                <a:UnderlyingFipsCode>2612580700</a:UnderlyingFipsCode>
             </a:AddressLocation>
             <a:AddressLocation>
                <a:Alternate>0</a:Alternate>
                <a:CassAddress>
                   <a:Address
xmlns:b="http://schemas.microsoft.com/2003/10/Serialization/Arrays">
                      <b:string>905 E MAPLE RD</b:string>
                      <b:string>TROY, MI  48083</b:string>
                   </a:Address>
                   <a:AddressLine>905 E MAPLE RD</a:AddressLine>
                   <a:AddressQualityFlags>S1NZPK5</a:AddressQualityFlags>
                   <a:CarrierRoute>C099</a:CarrierRoute>
                   <a:City>TROY</a:City>
                   <a:CityStateZip>TROY, MI  48083</a:CityStateZip>
                   <a:CountyCode>125</a:CountyCode>
                   <a:DeliveryPointValidation>64</a:DeliveryPointValidation>
                   <a:EnhancedLineOfTravel>0</a:EnhancedLineOfTravel>
                   <a:Reliability>92</a:Reliability>
                   <a:State>MI</a:State>
                   <a:USPSBarCode>/48083/</a:USPSBarCode>
                   <a:Zip>48083</a:Zip>
                   <a:Zip4/>
                </a:CassAddress>
                <a:CensusBlockGroup>2001</a:CensusBlockGroup>
                <a:CensusTract>180000</a:CensusTract>
                <a:CityName>TROY</a:CityName>
                <a:Country>USA</a:Country>
                <a:County>OAKLAND</a:County>
                <a:ErrorMessage/>
                <a:FeatureID>1626089</a:FeatureID>
                <a:FipsCode>2612516160</a:FipsCode>
                <a:FipsPlaceName>CLAWSON</a:FipsPlaceName>
                <a:Incorporated>true</a:Incorporated>
                <a:InputAddress
xmlns:b="http://schemas.microsoft.com/2003/10/Serialization/Arrays">
                   <b:string>905 E MAPLE RD</b:string>
                   <b:string>TROY,MI 48083</b:string>
                </a:InputAddress>
                <a:Latitude>42.548072269155561</a:Latitude>
                <a:Longitude>-83.158629478188061</a:Longitude>
                <a:NetworkID>nt</a:NetworkID>
                <a:PCode>1831400</a:PCode>
                <a:PostalCode>48084</a:PostalCode>
                <a:PostalCodeExtension/>
                <a:PrimaryJurisdictionName>CLAWSON</a:PrimaryJurisdictionName>
```

```
               <a:PrimaryJurisdictionPCode>1831400</a:PrimaryJurisdictionPCode>
               <a:Score>0.71428573131561279</a:Score>
               <a:SecondaryUnit/>
               <a:SpecialTaxDistrictName i:nil="true"/>
               <a:SpecialTaxDistrictPCode>-1</a:SpecialTaxDistrictPCode>
               <a:StandardizedAddress
xmlns:b="http://schemas.microsoft.com/2003/10/Serialization/Arrays">
                  <b:string>905 W MAPLE RD</b:string>
                  <b:string>TROY,MI 48084</b:string>
               </a:StandardizedAddress>
               <a:StateName>MI</a:StateName>
               <a:StreetName>W MAPLE RD</a:StreetName>
               <a:StreetNumber>905</a:StreetNumber>
               <a:TaxJurisdictionName>CLAWSON</a:TaxJurisdictionName>
               <a:TimeZone/>
               <a:UnderlyingFipsCode>2612516160</a:UnderlyingFipsCode>
            </a:AddressLocation>
         </GeocodeAllMatchesResult>
      </GeocodeAllMatchesResponse>
   </s:Body>
</s:Envelope>
```

5.5.2.5 GeocodeLatLong

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
xmlns:tem="http://tempuri.org/"
xmlns:ezg="http://schemas.datacontract.org/2004/07/EZGeoSaaS">
       <soap:Header>
              <UserName xmlns="EZGeo">*****</UserName>
              <Password xmlns="EZGeo">*****</Password>
       </soap:Header>
       <soap:Body>
              <tem:GeocodeLatLong>
                     <tem:latitude>38.956689</tem:latitude>
                     <tem:longitude>-94.686252</tem:longitude>
              </tem:GeocodeLatLong>
       </soap:Body>
</soap:Envelope>
```

5.5.2.6 GeocodeLatLongResponse

```
<s:Envelope xmlns:s="http://www.w3.org/2003/05/soap-envelope">
   <s:Body>
      <GeocodeLatLongResponse xmlns="http://tempuri.org/">
         <GeocodeLatLongResult
xmlns:a="http://schemas.datacontract.org/2004/07/EZGeoSaaS"
xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
            <a:Alternate>0</a:Alternate>
            <a:CassAddress i:nil="true"/>
            <a:CensusBlockGroup>0</a:CensusBlockGroup>
            <a:CensusTract>0</a:CensusTract>
            <a:CityName>OVERLAND PARK</a:CityName>
            <a:Country>USA</a:Country>
            <a:County>JOHNSON</a:County>
            <a:ErrorMessage/>
            <a:FeatureID>479210</a:FeatureID>
            <a:FipsCode>2009153775</a:FipsCode>
            <a:FipsPlaceName>OVERLAND PARK</a:FipsPlaceName>
            <a:Incorporated>true</a:Incorporated>
```

```
            <a:InputAddress
xmlns:b="http://schemas.microsoft.com/2003/10/Serialization/Arrays">
               <b:string/>
               <b:string/>
            </a:InputAddress>
            <a:Latitude>38.956689</a:Latitude>
            <a:Longitude>-94.686252</a:Longitude>
            <a:NetworkID>nt</a:NetworkID>
            <a:PCode>1248900</a:PCode>
            <a:PostalCode/>
            <a:PostalCodeExtension/>
            <a:PrimaryJurisdictionName>OVERLAND PARK</a:PrimaryJurisdictionName>
            <a:PrimaryJurisdictionPCode>1248900</a:PrimaryJurisdictionPCode>
            <a:Score>0.999961162858618</a:Score>
            <a:SecondaryUnit/>
            <a:SpecialTaxDistrictName i:nil="true"/>
            <a:SpecialTaxDistrictPCode>-1</a:SpecialTaxDistrictPCode>
            <a:StandardizedAddress
xmlns:b="http://schemas.microsoft.com/2003/10/Serialization/Arrays">
               <b:string></b:string>
               <b:string>OVERLAND PARK,KS null</b:string>
            </a:StandardizedAddress>
            <a:StateName>KS</a:StateName>
            <a:StreetName/>
            <a:StreetNumber/>
            <a:TaxJurisdictionName>OVERLAND PARK</a:TaxJurisdictionName>
            <a:TimeZone/>
            <a:UnderlyingFipsCode>2009153775</a:UnderlyingFipsCode>
         </GeocodeLatLongResult>
      </GeocodeLatLongResponse>
   </s:Body>
</s:Envelope>
```

5.5.2.7 GetServerTime

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
xmlns:tem="http://tempuri.org/">
   <soap:Header/>
   <soap:Body>
      <tem:GetServerTime/>
   </soap:Body>
</soap:Envelope>
```

5.5.2.8 GetServerTimeResponse

```
<s:Envelope xmlns:s="http://www.w3.org/2003/05/soap-envelope">
   <s:Body>
      <GetServerTimeResponse xmlns="http://tempuri.org/">
         <GetServerTimeResult>Aug 8, 2016 1:44:26 PM</GetServerTimeResult>
      </GetServerTimeResponse>
   </s:Body>
</s:Envelope>
```

6. AFC Geo SaaS Standard

AFC Geo SaaS Standard is used for geocoding multiple US addresses within a batch input file.

6.1 Accessing the AFC Geo FTP Site

To initiate an AFC Geo SaaS Standard batch, your input must be uploaded to the AFC Geo FTP server. Once complete, the results file can then be downloaded.

Please be aware that web browsers (e.g. Microsoft Internet Explorer) may not be used for uploading and downloading your files to the AFC Geo FTP site. You must use an FTP client application to transfer files to and from the AFC Geo FTP site.

Windows users:

Microsoft does not currently include convenient FTP client software in its operating systems. It is assumed you either have a third-party FTP client application such as *WS_FTP or FileZilla*, or that you are comfortable accessing FTP sites using command-line syntax.

Linux users:

If you are using a Linux system to transfer your data files, you can use any number of free FTP clients to contact the AFC Geo FTP site, (such as *WXFTP*), or you can use command-line syntax.

6.2 Logging into the AFC Geo FTP Site

To log on to the Avalara FTP site, the following information is needed:

HOSTNAME/ADDRESS: *ftp.billsoft.com*

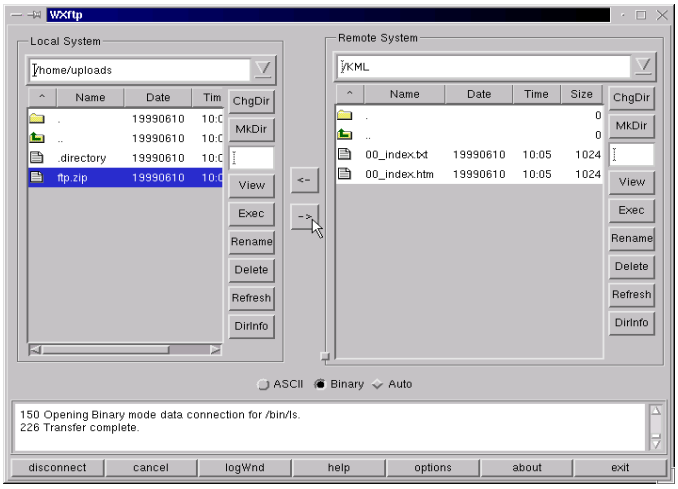
USER ID: Company's login User ID, which is available from the Avalara Technical Support contact.

PASSWORD: Company's login password, which is available from the Avalara Technical Support contact.

6.3 Transferring Files to and From the FTP Site

Once connected to the AFC Geo FTP site, you should change to the directory that has the same name as your company's three-character code. You are now ready to send your file for processing (see figure 15, which shows a 3rd party FTP client application).

When you have successfully connected to the EZgeo FTP site and changed to your company's file-transfer directory, you can send your txt file for processing. You can ignore the index files.



When the file has completed processing, you can download the output (or "result" file) from the AFC Geo FTP site. The result file will have a similar name to the .txt file you uploaded, except it will contain an under score and the character "R" - indicating it is a result file. For example, if the uploaded txt file contained a file called *KML02172014AA.txt or KML02172014AA.ltl*, the result file (for downloading) would be called *KML02172014AA_R.txt* or *KML02172014AA_R.ltl*

6.3.1 Preparing Files for Upload

When you upload your data file for processing by AFC Geo, the AFC Geo FTP server begins automatic processing of your file. However, in order to provide for automatic processing, you ***must*** be certain to name your file correctly. Please observe the following guidelines when preparing your file for upload:

- The file must be named correctly.
- The file must be in the correct file format.
- The file must be in plain text.

If more than one file is sent in the current day, you should change the variable portion of the filename in order to differentiate the files and prevent previous file uploads from being overwritten.

See sections below for more details on these guidelines.

6.3.1.1 Naming Files

Uploaded files MUST be named using the following naming convention:

Filename Format: ABCMMDDYYYYVV.EXT

Where:

- The filename should begin with the company's three-letter code (e.g. ABC).
- The 4th and 5th characters in the filename should be the current month (e.g., *02* for February).
- The 6th and 7th characters in the filename should be the current numerical day (e.g., 17 for the 17th day of the current month)
- The 8th, 9th, 10th and 11th characters in the filename should be the current year (e.g., 2014)
- The remaining characters before the extension should be variable (e.g., AA, 11 or A1). This is a unique identifier for each file uploaded. This prevents files from being overwritten.
- The filename would then be finished by a .TXT for Plain text format for Address file and .LTL for Latitude/Longitude file.

Examples:

Address Files

For company ABC and receipt of a file on the 17th February, 2014, the file name would be - ABC02172014AA.txt

Latitude/Longitude Files

For company ABC and receipt of a file on the 17th February, 2014, the file name would be - ABC02172014AA.ltl

6.3.1.2 Address Geocoding Format

When requesting address information via AFC Geo SaaS Standard, please provide a plain text document containing the following details:

- **Unique Identifier** – You may use any numerical identifier you choose (e.g., customer number, account number, etc.)
- **Street address**
- **City**
- **State abbreviation**
- **5 digit Zip or 5 digit Zip +4**
- **Minimum Score** (optional, default of 0.7 if no value is provided)

- **Offset** (optional, default of 3 if no value is provided)
- **Cass certify flag** (optional, default of true if no value is provided)
- **Options** (optional, default of 312: Zip+4, Special Tax Dist, Census, Bldg Zip if no value is provided)

See **Section 5.3.1** for a detailed description of the input fields.

This must be formatted using comma separation. Please format using one line per requested address. The preferred format is to have both Carriage Return (0x0D) and Line Feed (0x0A) for line termination. The format will look like this:

1,8675 west 96th St suite 220,Overland Park, KS,66212
2,1600 Pennsylvania Ave,Washington DC,DC,20500,0.7,3,TRUE,312

6.3.1.3 Lat/Long Geocoding Format

The input file for Lat/Long Geocoding contains the following details:

- **Unique Identifier** – You may use any numerical identifier you choose (e.g., customer number, account number, etc.)
- **Latitude**
- **Longitude**

This must be formatted using comma separation. Please format using one line per requested address. The preferred format is to have both Carriage Return (0x0D) and Line Feed (0x0A) for line termination. The format will look like this:

312254, 34.08, -117.89
376009, 34.08, -87.06

6.3.2 Understanding Output Files

6.3.2.1 Address Geocoding

The Address Geocoding result file format is as follows:

This file contains a comma separated plain text document with the results of your upload to the AFC Geo SaaS Standard service. This file format contains the following columns:

- **Unique ID (from input file)**
- **Score (1 equals 100% reliability)**
- **PCode**
- **Jurisdiction name**
- **FIPS code**
- **Unincorporated (1 for true or 0 for false)**

- County
- Latitude
- Longitude
- Street Address
- City
- State
- Zip Code
- CASS Street Address
- CASS City
- CASS State
- CASS Zip code
- FeatureID*
- Census Tract ID*
- Census Block ID*

**Return of this data is optional. Clients must contact communicationsupport@avalara.com in order to activate this functionality.*

Below is an example of a single-line record in the output file of Address Geocoding, without optional output values:

1,1,1248900,OVERLAND PARK,2009153775,0,JOHNSON,38.9546476806009,-94.6859671231543,8675 W 96TH ST STE 220,OVERLAND PARK,KS,66212-3382,8675 W 96TH ST STE 220,OVERLAND PARK,KS,66212-3382

Below is an example of a single-line record in the output file of Address Geocoding, with optional output values:

2,0.761904776096344,534000,WASHINGTON,1100150000,0,WASHINGTON DC,38.8986939338077,-77.0359746164679,1600 PENNSYLVANIA AVE NW ,WASHINGTON,DC,20502-0003,1600 PENNSYLVANIA AVE NW,WASHINGTON,DC,20500-0003,531871,6202,1031

6.3.2.2 Lat/Long Geocoding

The Lat/Long Geocoding result file format is as follows:

This file contains a comma separated plain text document with the results of your upload to the AFC Geo SaaS Standard service. This file format contains the following columns:

- Unique ID (from input file)
- Latitude
- Longitude
- Country Name
- State Name
- County Name
- Jurisdiction Name
- PCode

- FeatureID*

**Return of this data is optional. Clients must contact communicationsupport@avalara.com in order to activate this functionality.*

Below is an example of a single-line record in the output file of Lat/Long Geocoding, without optional output values:

312254,34.08,-117.89,USA,CA,LOS ANGELES,COVINA,299700

Below is an example of a single-line record in the output file of Lat/Long Geocoding, with optional output values:

376009,34.08,-87.06,USA,AL,CULLMAN,CULLMAN COUNTY,24100,161547

6.3.3 Error Handling

Records with invalid data will generate a line in the output file indicating there is an error with that one entry, but all valid records are processed and reported as normal. There is also a log file that is created that provides more detail for each record that has an error. With this information, clients will be able to obtain valid results, determine what is incorrect as well as be able to change any records with errors and reprocess them.

Below is an example of a single-line error record.

1234,ERROR - Invalid input data - Minimum Score (KS) is not a valid double

This error would be generated if the street data inadvertently included an embedded comma, thereby throwing off the comma separated value fields.

NOTE: At this time, AFC Geo Standard does not support using quotes to try to embed commas (e.g. "123 Main Street, Apt 100"). AFC Geo Standard does not include the Secondary Unit field; therefore, it is best to simply leave out the apartment or suite number in the Street Address field.

7. AFC Geo Viewer (AFC Geo Direct)

AFC Geo Viewer provides US address verification and assignments of tax jurisdictions through input of a street address or latitude and longitude coordinates.

Note: The AFC Geo Viewer is accessible on the **Geo Viewer** page of the **AFC Customer Portal** https://communications.avalara.net. Please contact communicationsupport@avalara.com in order to obtain access and login credentials for this tool.

8. AFC Geo License

AFC Geo License is a full featured system that is completely hosted at the client's site, with software and US address updates provided. It is recommended for large transaction volumes. AFC Geo License is similar to AFC Geo SaaS Pro in that the geocoding is integrated into client software, but all functionality is local, rather than executed over the internet, and there are no transaction charges.

8.1 Using AFC Geo License

The main interface to AFC Geo is called billsoft.EZgeo.Locator. The JavaDoc for this class and all other AFC Geo classes can be found in the docs directory under EZGEO_HOME. The Locator object takes in one or more addresses or one or more intersections and returns one or more matches for each location, depending on your preference. For each match, AFC Geo returns the score of the match, the standardized address, the county the match resides in, the FIPS code for the match, the Avalara PCode for the match, whether the match is in an incorporated area, the census block, and the census tract. If more than one match is returned, the best match will appear first in the list, the next best will appear next in the list, and so on.

8.1.1 Example

This sample Java code geocodes a single address by initializing the Locator object and sending in the address through the Locator's geocode function. The results are then printed out and the Locator exit function is called.

```
public class TestAddress {

        public static void main(String[] args) {

                Locator loc = null;
                try {
                        // The "EZgeo.home" property can be set here
                        // or with the java -DEZgeo.home=D:\billsoft\EZgeo argument
                        // Windows
                        System.setProperty("EZgeo.home", "D:\\billsoft\\EZgeo");
                        // Linux
                        // System.setProperty("EZgeo.home", "//billsoft//EZgeo");

                        loc = new Locator();

                        // Create an inputAddress object to hold the input address
                        InputAddress = new InputAddress();

                        // Request the address to be CASS certified BEFORE EZgeo processing

                        // false turns off cass
                        inputAddress.setCassCertify(false);
```

```
                inputAddress.setOptions(InputAddress.RETURN_CUSTOM_AREAS);

                // Put a two line address into the input object
                //inputAddress.setAddressLine("10150 W 87th St");
                //inputAddress.setCityStateZip("overland park,ks,66212");


                // geocode a single address returning best match
                {
                inputAddress.setAddressLine("10100 W 87th St");
            inputAddress.setCityStateZip("Overland park KS");

                inputAddress.setOptions(280) ;
                AddressLocation addrLoc = loc.geocodeAddress(inputAddress);

                // Print the results
                printAddressLocation(addrLoc);
                }

        } catch (EZgeoException ex) {
                System.out.println(ex.getMessage());
                System.out.println("");
                System.out.println("stack trace:");
                ex.printStackTrace();
        }
        try
        {
                loc.exit();
        }
        catch(Exception ex)
        {
                System.out.println(ex.getMessage());
        }
        System.out.println("\r Done: " + new Date());
        System.exit(0);
}

static void printAddressLocation(AddressLocation addrLoc) {
        if (addrLoc == null)
        {
                System.out.println("\rMatch is null");
        }
        else
        {
                System.out.println("\r** Input Address ****\r");
                String[] inAddr = addrLoc.getInputAddress();
                System.out.println(inAddr[0]);
                System.out.println(inAddr[1]);

                // Print some CASS data
                System.out.println("\r** CASS data****\r");
```

```
                        if(addrLoc.getCassAddress() != null)
                        {
                                System.out.println(addrLoc.getCassAddress().getAddressLine());
                                System.out.println(addrLoc.getCassAddress().getCityStateZip());
                                System.out.println(addrLoc.getCassAddress().getCity());
                                System.out.println(addrLoc.getCassAddress().getState());
                                System.out.println(addrLoc.getCassAddress().getZip());
                                System.out.println(addrLoc.getCassAddress().getZip4());
                        }
                        // Print some EZgeo data
                        System.out.println("\r** EZgeo data****\r");
                        String[] stdAddr = addrLoc.getStandardizedAddress();
                        System.out.println(stdAddr[0]);
                        System.out.println(stdAddr[1]);
                        System.out.println("\rScore: " + addrLoc.getScore());
                        if (addrLoc.isIncorporated())
                                System.out.println("Incorporated");
                        else
                                System.out.println("Unincorporated");
                        System.out.println("County: " + addrLoc.getCounty());
                        System.out.println("PCode: " + addrLoc.getPCode());
                        System.out.println("Jurisdiction Name: " + addrLoc.getPrimaryJurisdictionName());
                        System.out.println("FIPS Code: " + addrLoc.getFIPSCode());
                        System.out.println("Latitude: " + addrLoc.getLatitude());
                        System.out.println("Longitude: " + addrLoc.getLongitude());
                        System.out.println("Network ID: " + addrLoc.getNetworkId());
                        if(addrLoc.getSpecialTaxDistrictName()!= "")
                                System.out.println("STD name: "     + addrLoc.getSpecialTaxDistrictName());
                }
        }
}
```

8.1.2 Running AFC Geo

The Maximum Heap Size of the Java Virtual Machine must be at least the page pool size plus 100 MB per street database. In addition, the system property EZgeo.home must be set to the EZGEO_HOME environment variable.

The following examples assume a page pool size of 50 MB and the use of two street databases.

Example (Windows): java –Xmx250m –DEZgeo.home=%EZGEO_HOME% EZgeoTest

Example (UNIX): java –Xmx250m –DEZgeo.home=$EZGEO_HOME -DEZgeo.options=3 EZgeoTest

9. Troubleshooting Scenarios & Frequently Asked Questions

9.1 Common Troubleshooting Scenarios

9.1.1 AFC Geo SaaS Pro

I'm not getting AFC Geo SaaS Pro programming to work

The programming required to integrate AFC Geo SaaS Pro into a client program has a number of things that must be done. A few of the standard items that need to be checked are the following:

- Endpoints and bindings
- Username / Password
- SOAP Message
- IP Address registered with Avalara

To aid in developing the client programming, Avalara can provide sample programs to help provide an example of how it can be done. In addition, the following link provides information about AFC Geo SaaS Pro, including WSDL links.

http://EZgeoasp.billsoft.com/locatorservice.svc

Your Username and Password will be assigned to you by Avalara support. The service is also restricted to specific IP addresses. Therefore, you must provide Avalara with the IP address that you will be using, so it can be placed in the list of approved clients.

Finally, it is often beneficial to be able to capture the specific SOAP message that is being sent to AFC Geo SaaS Pro, so that it can be examined in order to determine if it contains everything that is required. To do that capture, you will need a utility program. One such program is Fiddler.

I'm getting an exception error

There are a number of potential exception errors that can be sent back from AFC Geo. The information about each individual exception is given in the following section.

I'm not getting any exception errors

There are two sets of endpoints for AFC Geo. The endpoints with /2.0 at the end are the ones that will send back an exception when there is an error. The other endpoints will simply return a null result.

9.1.2 AFC Geo SaaS Standard

I uploaded my file quite a while ago, and I still don't see any result file. What's wrong?

The most likely reason for this is an incorrect filename or an incorrect file format. The file must be in the correct format for the system to process it. Likewise, the file must be named correctly for the system to recognize it (see the appropriate sections within this manual regarding the naming of files). The filename can be especially tricky: the actual .TXT or .LTL file must be named according to the afore-mentioned file-naming convention.

Solution: If, after verifying your file is both formatted and named correctly, you are still having problems - call or email your Avalara Technical Support contact.

I've connected to the AFC Geo FTP server, but I cannot view the contents of my directory.

If you're unable to change directories, or if you are unable to list the contents of your transfer directory or transfer files to/from the FTP server, you are most likely being denied access to the AFC Geo SaaS Standard server by a firewall or proxy server.

Solution: You should contact your network administrator or Internet service provider and ask them to reserve the address block of ftp.BillSoft.com as trusted addresses for AFC Geo's FTP server.

9.1.3 AFC Geo Direct

I get a browser error saying Web Page Cannot Be Found - HTTP 404

Solution: Check the URL to ensure you are entering it correctly. It should be http://EZgeodirect.billsoftservices.com/EZgeoDirectLocator.aspx . If you have the URL correct, contact Avalara support for assistance.

I am unable to log in

Solution: Double check your user name and password. The log in is not case sensitive. If you believe you have your log in correct, please contact Avalara support for assistance.

9.1.4 AFC Geo License

AFC Geo License is not working

Setting up AFC Geo License has numerous installation and configuration issues. Working with Avalara Support to resolve these issues will be required.

9.1.5 Common

Why did I get two different scores for the same address on two different requests?

AFC Geo has many input options, and the one that is most likely associated with this situation is the Minimum Score. If you make two separate geocoding requests with two different minimum scores, then

it is possible to receive results with different scores, even though the address is the same. This is because AFC Geo has scoring options that use different subsets of the address input values, which can produce different scores. This is done to address the possibility of incorrect input values, such as a wrong zip code or misspelled city name.

Solution: Make sure that the Minimum Score is consistent, and not defaulting to a value of zero.

9.2 AFC Geo Exceptions

AFC Geo SaaS Pro has the option of returning exceptions when there are errors within the procedure call. The format of those errors is as follows.

Error <MsgID>-<SeqID> - <Message Text>

Where:

<MsgID> Code that is unique to this type of error.

<SeqID> Sequence number within an Avalara database log. This ID can be used by Avalara support people in order to get details on how resolve the issue.

<MessageText> Standard text for this type of error.

The following is a list of the standard exceptions that can be returned by AFC Geo SaaS Pro.

MsgID MessageText / Description

E0001 Could not geocode address. Check InputAddress values.

This is a general purpose error related to input address values not being supplied. The address must include a street address, and the street address should start with a street number that should normally be all numeric. The address must also include either a zip code or city and state.

E0002 Minimum score must be between 0 and 1, inclusive.

The minimum score cannot be negative or greater than one. A score of 1 indicates 100% match.

E0003 The EZgeo.home property is not defined. Use System.setProperty.

AFC Geo needs to know the directory where it has been installed. The System.setProperty function can be used to make that assignment within the client code.

E0004 Invalid Configuration: Config values not set in Config.txt

The AFC Geo configuration file has a standard set of options that must be set. This error indicates that one of the configuration entries that are expected are not in the file.

E0005 Invalid Street database. Must be 1=NT, 2=TA, 3=Both.

The input value indicating which street database must be 1 for NavTeq only, 2 for TeleAtlas only, or 3 to use both.

E0101 User not found.

The user name supplied for login was not found.

E0102 User account is inactive.

The user account was found, but has been inactivated.

E0103 Unable to geocode with CASS Certify enabled. Contact Avalara Support.

The CASS certification service is not operational. This service must be restarted in order to geocode with that option. As an alternative, however, geocoding can be done if the CASS certification option is turned off.

E9998 Server Setup/Operation error. Contact Avalara Support.

This error occurs due to some setup or operation error on the Avalara side of the process. Please contact Avalara in order to resolve the issue.

E9999 An internal error occurred. Contact Avalara Support.

This is a general purpose error message for situations in which the specific error condition is undefined. Please contact Avalara support in order to report and resolve the issue.

10. Support

Avalara strives to provide the very best in customer support. If you should need assistance or further information, please call us at 1-800-525-8175, (913) 859- 9674, or send an email message to communicationsupport@avalara.com.

If you are calling with an urgent matter outside of the hours of 8 am to 5 pm (Central time) Monday through Friday, you can leave a message on your Avalara Technical Support contact's voicemail at (913) 221-7320.