Global-Z
INTERNATIONAL DATA PROCESSING SERVICES

Global Address Verification API Specification

Revision 20211221

Global-Z International
Global Address Verification API Specification

Overview

Global-Z's Global Address Verification service is an automated remotely-hosted web service that provides users with a high-end, real-time, on-demand, record-by-record automated global address verification and standardization solution. The service provides users with a fast, cost-effective way to process, clean and standardize individual international address records in real-time, and utilizes Global-Z's superior proprietary international address hygiene technology, known widely as the industry standard for global address cleansing.

The Address Verification service supports both single-byte and multi-byte name and address data, such as Asian text and other non-Western alphabets.

For users who instead require an automated batch processing service, Global-Z also provides an automated Global Address Verification Batch service.

Applications

Global-Z's address verification service can be integrated into any application that requires verification, standardization or correction of an international address, including:

- eCommerce applications requiring real-time global address verification
- Internet or Intranet-based data capture interfaces or data collection tools
- CRM databases, CDI applications, or global data warehouse applications
- Backend database verification or cleansing processes
- Any other application requiring real-time global address cleansing

Terminology

In this document, the term "outgoing address" means a source address that the client application sends to the Global-Z server. The term "returned address" means a processed address that Global-Z's server returns to the client application.

Throughput Rates

The address verification API utilizes standard HTTP POST over SSL (HTTPS) for passing data between your client application and Global-Z's servers. Addresses can be sent at any time and at any frequency, over one or more concurrent connections. Throughput rates depend on several factors, including Internet bandwidth and traffic, overall system demand, source postal address quality for the given country, and the number of concurrent connections. Under typical conditions, it is expected that an address will be processed and returned within 2 to 3 seconds of submission, assuming submission via an interactive user interface such as a web browser. Transactions automated via an Internet scripting language such as Python or PHP can attain faster average turnaround times, up to 700 to 800 milliseconds, using HTTP keep-alive to avoid re-authentication by the HTTPS server. Actual throughput will vary by record.

The basic service provides a single secure connection to the Global-Z server. Faster throughput times can be attained via multiple concurrent connections, which are available at additional cost. For very high-volume and/or

high-performance requirements, a direct client/server connection can be utilized, such as a dedicated point-to-point or virtual private network (VPN). Please contact your Global-Z sales representative if this is something that your application requires.

Data Transfer

Data transfer utilizes a standard HTTP POST over a secure SSL connection. SSL is the industry standard Internet eCommerce data security layer. Global-Z will provide you with a URL to which to send the data, which unless otherwise specified will be:

https://realtime.globalz.com/api/av

and an SSL certificate for authentication. For security purposes, it is necessary that your sending process utilize a fixed IP address or specific range of IP addresses when using the service, therefore the service does not support applications that use dynamic IPs such as dialup Internet accounts.

Outgoing transactions must be in the form of an HTTP POST. The system supports both the "application/octet-stream" MIME type, and the "application/x-www-form-urlencoded" MIME type. We recommend using the "application/octet-stream" MIME type, however both types are supported.

The maximum input transaction size supported is 5,000 bytes. Input transactions exceeding this size will be rejected by the server.

XML Data Format

Transaction data is sent and returned in XML format, using a defined structure and tag naming convention. See below for detailed outgoing and returned XML data specifications.

Data Encryption

Data encryption is provided by the SSL protocol, therefore it is not necessary to explicitly encrypt the data feed prior to transfer.

Data Encoding

Data must be encoded in a standard encoding, such as UTF-8, ASCII, ANSI Windows 1252, ISO Latin 2, etc. for single-byte data, or UTF-8, Unicode, Shift-JIS, etc. for multi-byte data. The encoding must be identified in the "encoding" attribute of the XML declaration, for example:

```
<?xml version="1.0" encoding="utf-8"?>
```

If the XML declaration does not specify an encoding attribute, UTF-8 is assumed. Failure to specify the correct encoding for the data can result in data corruption.

Although most common encoding standards can be supported, Global-Z recommends the UTF-8 encoding standard for all outgoing and incoming transactions.

Outgoing XML Data Format

The outgoing address data must follow the XML standard, and must contain the following element structure:

```
<Transaction>
  <Input>
    ...input data fields...
  </Input>
</Transaction>
```

where the `Transaction` element is the top level element of the XML document. Input data fields must be tagged with specific field (element) names. Field names are case-sensitive. All input fields are optional. Empty input fields are allowed, however excluding them from the input transaction will minimize transaction size.

Supported input data fields are as follows:

Field Name	Max Length	Description
URN	254	User-defined record identifier, will be supplied back on the returned transaction
Name	254	Full contact name, e.g. “Mr. Brad H. Searles”
Prefix	254	Name prefix, e.g. “Mr”, “Mrs”. If parsed name fields are populated on input, these fields will take precedence over the Name field.
FirstName	254	Given name
MiddleName	254	Middle name
LastName	254	Surname
LastName2	254	Second surname
Suffix	254	Name suffix, e.g. “Jr”, “Sr”
Name_Kanji	254	Full contact name in Kanji writing system
Prefix_Kanji	254	Name prefix in Kanji writing system
FN_Kanji	254	Given name in Kanji writing system
LN_Kanji	254	Surname in Kanji writing system
Title	254	Job title
Business	254	Company name
Business2	254	Second company name or department
Address1	254	Address lines
Address2	254	
Address3	254	
Address4	254	
Address5	254	
City	254	City or town name
State	254	State, province, or county name
Zip	254	Postal code
Country	254	Country Name
Accents	7	Specifies whether accents will be returned in the standardized address:

		YES Standardize in local spelling with accents (e.g. München) NO Standardize in local spelling without accents (e.g. Muenchen) ENGLISH Standardize in English spelling without accents (e.g. Munich)
Casing	5	Specifies what text casing will be used in the returned standardized address: UPPER Standardize to Upper Case MIXED Standardize to Mixed (upper/lower) Case UPU Standardize to UPU Standard (address in mixed case, city and province in upper case)
Language	5	Specifies the language or writing system to use for the returned standardized address: ROMAN Return Roman data LOCAL Return local text data (e.g. Kanji, Cyrillic, etc.) BOTH Return in both Roman and local text INPUT Return in the language of the input data
NameParse	1	Y Parse the name components into granular fields, e.g. parse a prefix value from the FirstName field into the Prefix field. N (default) Return parsed fields as provided on input.
NameSwap	1	Y Swap the first and last names if the derived gender codes suggest the input values were reversed. N (default) Output the first and last names as provided on input, regardless of the derived gender codes.
NameAlias	1	W Convert the first name to a standardized form if a name alias or nickname is provided on input, e.g. standardize “Bob” to “Robert”. The name is standardized to the most common Western name variant. B Standardize the first name to the Base name of historical origin, which is the oldest identified form of a name. Note that many Base names originated in antiquity, and are no longer commonly in use, however this setting provides a useful common origin for name variants. N (default) Leave the first name as provided on input.
NameToneMarks	1	Y Include tone marks in Romanized Chinese names (e.g. ma1, ma2) N (default) Return Romanized Chinese names without tone marks (e.g. ma)

Field names are case-sensitive. All input fields are optional.

IMPORTANT: The above supported field names are guaranteed to be forward-compatible. However, from time to time new fields will be added as functionality is extended. Some options may not be available to all users, depending on account configuration.

Example Outgoing Data

Following is an example of the data payload in the outgoing HTTP POST transaction:

```
<?xml version="1.0" encoding="utf-8"?>
<Transaction>
  <Input>
    <URN>ID12345</URN>
    <Prefix>Mr.</Prefix>
    <FirstName>Junichiro</FirstName>
    <LastName>Koizumi</LastName>
    <Title>President</Title>
    <Business>ABC Company</Business>
    <Address1>1-1-5 Roku-cho</Address1>
    <Address2>Adachi-ku tokyo</Address2>
```


Global-Z

```
    <Country>japan</Country>
    <Accents>Yes</Accents>
    <Casing>Mixed</Casing>
    <Language>Both</Language>
    <NameParse>Y</NameParse>
    <NameSwap>Y</NameSwap>
    <NameAlias>N</NameAlias>
    <NameToneMarks>N</NameToneMarks>
  </Input>
</Transaction>
```

Returned Data Format

The returned address data will follow the XML standard, with the following element structure:

```
<Transaction>
  <ValidationInfo>
    ...return codes...
  </ValidationInfo>
  <RomanAddress>
    ...standardized Roman-text data fields...
  </RomanAddress>
  <LocalAddress>
    ...standardized local-text data fields...
  </LocalAddress>
  <Error>
    ...error message...
  </Error>
</Transaction>
```

Returned fields within each element will be tagged with specific field names. All returned fields are optional except where noted below. Empty fields will not be returned.

IMPORTANT: Returned element structures and field names described here are guaranteed to be forward-compatible. However, from time to time new fields and structures will be added as functionality is extended. To be forward-compatible, client applications must explicitly allow for future additional field names and structures. Some options may not be available to all users, depending on account configuration.

`ValidationInfo` Element Structure

The `ValidationInfo` returned element is optional, and will be returned on every transaction unless an error occurs, in which case the `Error` element will be returned instead.

Supported `ValidationInfo` fields are as follows:

Field Name	Max Length	Mandatory or Optional	Description
`URN`	254	Optional	Returned unchanged if supplied in the input transaction
`MQCode`	1	Mandatory	Single alpha Mail Quality Code indicating address quality: `A` Valid – postcode correct in input address, address changes or improvements may have been made `B` Valid – postcode appended or modified, other address changes or improvements may have been made `C` Valid – address inferred to be correct based on postcode, address changes or improvements may have been made `D` Invalid – incorrect or ambiguous address information, no address changes were made `G` Formatted – address verified based on format only, no postcode data available for this country `H` Invalid Format – postcode not found in input address, no postcode data available for this country `I` Invalid Format – city or postcode not found in input address,

			no postcode data available for this country J Undeliverable – insufficient address information necessary for delivery K Undeliverable – country could not be identified See the Mail Quality Code Description document for a detailed description and examples of each category. Contact your sales representative to obtain a copy of this document.
NameScore	10	Optional	Customer name validation and standardization score. See the Name Score Specification document for a detailed description of the score.
Country	254	Optional	Standardized Country Name. Value will be omitted if no country name was supplied on the input transaction and a country could not be determined from the input address.
ISOCountry2	2	Optional	ISO 3166-1 Alpha-2 Country Code. Value will be omitted if a destination country could not be successfully identified (MQCode=K).
ISOCountry3	3	Optional	ISO 3166-1 Alpha-3 Country Code. Value will be omitted if a destination country could not be successfully identified (MQCode=K).
ISOSubdivision	10	Optional	ISO 3166-2 Subdivision code (e.g. provinces or states). Not yet implemented for all countries and/or provinces.
Gender	1	Optional	Gender as determined from the input name. If GenderPrefix is known (male or female), Gender will assume the same value, otherwise Gender will assume the same value as GenderFN. Possible values are: M Male F Female U Unknown
GenderPrefix	1	Optional	Gender as determined from the input name prefix. Values are identical as for Gender.
GenderFN	1	Optional	Gender as determined from the input first (given) name. Values are identical as for Gender.
HygieneDetail	10	Optional	Optional reason code(s) returned by the hygiene process for certain countries. See Appendix A for details.
CanadaSERPStatus	1	Optional	Returned only for addresses in Canada: V Valid address C Corrected address N Invalid address
AustraliaDPID	8	Optional	Returned only for valid addresses in Australia (MQCode=A,B,C). Contains Australia Post 8-digit DPID (Delivery Point Identifier).
InputLanguage	1	Mandatory	Indicates the language of the input address: ROMAN Input address was supplied in Roman (Western) text LOCAL Input address was supplied in local-language text
NUTSCode	10	Optional	NUTS (Nomenclature of Territorial Units for Statistics) code supported by some European countries. The NUTS code provides a course geocode designator for addresses. Not yet implemented for all countries or areas.
Language	40	Optional	Specifies the language most likely to be spoken in the geographical area of the address. Supported only for multi-lingual countries where language is determined by geography (e.g. Switzerland, Belgium).
UKGridE	6	Optional	Geocode reference for UK addresses. Specifies the Easting coordinate of the postcode area, based on the National Ordnance Survey National Grid (for UK addresses) or the Irish Grid (for Northern Ireland

			addresses). The value is specified in meter units using numerical values only (the optional National Grid alpha references are mapped to numerical values). Grid references are provided at 100-meter resolution. Note that some UK addresses do not support geocode information, e.g. postboxes, Isle of Man, Jersey, and Guernsey.
UKGridN	6	Optional	Specifies the Northing coordinate of the postcode area.
Latitude	13	Optional	Specifies the Latitude coordinate of the address.
Longitude	13	Optional	Specifies the Longitude coordinate of the address.
GeocodeResolution	5	Optional	Resolution of the geocode (latitude/longitude) coordinates returned: 1.0.0 Premise level 2.0.0 Approximate premise level 3.0.0 Street level 5.0.0 Rural street level (rural route and street) 6.0.0 Rural route level 8.0.0 Postcode level 9.0.0 City / town level
Warning	1024	Optional	Returns any warning messages generated by errors on input or output, e.g. invalid characters, encoding errors, etc.

`RomanAddress` and `LocalAddress` Element Structure

The `RomanAddress` and `LocalAddress` returned elements are optional. The `RomanAddress` element will only be returned if the input `Language` parameter requests Roman-text data (`ROMAN`, `INPUT`, or `BOTH`), or if local text is not supported for the given country. The `LocalAddress` element will only be returned if the input `Language` parameter requests local-text data (`LOCAL`, `INPUT`, or `BOTH`), and only if local text is supported for the given country (e.g. Cyrillic in Russia, Kanji in Japan, etc.).

Supported returned fields in the `RomanAddress` and `LocalAddress` elements are as follows:

Field Name	Max Length	Mandatory or Optional	Description
Name	254	Optional	Full contact name, e.g. “Mr. Brad H. Searles”
Prefix	254	Optional	Name prefix, e.g. “Mr”, “Mrs”
FirstName	254	Optional	Given name
MiddleName	254	Optional	Middle name
LastName	254	Optional	Surname
LastName2	254	Optional	Second surname
Suffix	254	Optional	Name suffix, e.g. “Jr”, “Sr”
Name_Kanji	254	Optional	Full contact name in Kanji writing system, if applicable
Prefix_Kanji	254	Optional	Name prefix in Kanji writing system
FN_Kanji	254	Optional	Given name in Kanji writing system
LN_Kanji	254	Optional	Surname in Kanji writing system
Title	254	Optional	Job title
Business	254	Optional	Company name
Business2	254	Optional	Second company name or department

Parsed Output Fields			
`Address1`	254	Optional	Address lines, excluding City, State and Zip
`Address2`	254	Optional	
`Address3`	254	Optional	
`Address4`	254	Optional	
`Address5`	254	Optional	
`City`	254	Optional	City or town name
`State`	254	Optional	State, province, or county name
`Zip`	254	Optional	Zip or postal code
Formatted Output Fields			
`Formatted1`	254	Optional	Formatted address lines, including City, State and Zip
`Formatted2`	254	Optional	
`Formatted3`	254	Optional	
`Formatted4`	254	Optional	
`Formatted5`	254	Optional	
`Formatted6`	254	Optional	
`Formatted7`	254	Optional	
`Formatted8`	254	Optional	
Analysis Fields (see section below entitled Producing a Complete Address from the Returned Data)			
`HouseNo`	254	Optional	House or building number, e.g. “10” in “10 Main Street”
`HouseSufx`	254	Optional	House or building number suffix, unit, suite, etc., e.g. “Unit 5” in “10 Main Street, Unit 5”
`StreetName`	254	Optional	Street name
`StreetName2`	254	Optional	Secondary street name, e.g. Dependent Thoroughfare in UK
`Building`	254	Optional	Building name, where applicable
`POBox`	254	Optional	PO box number, e.g. “100” in “PO Box 100”
`Floor`	254	Optional	Floor number, where applicable, e.g. “3” in “Floor 3”
`Lane`	254	Optional	Lane designator, e.g. “5” in “Lane 5”, used primarily in Taiwan and China
`Alley`	254	Optional	Alley designator, e.g. “12” in “Alley 12”, used primarily in Taiwan
`Sector`	254	Optional	Sector designator, e.g. “2” in “Sector 2”, used primarily in Tawian
`SubCity`	254	Optional	Sub-city area name, e.g. Dependent Locality in UK, Colonia in Mexico, “ku” in Japan, Bairro in Brazil, etc.
`AdminArea`	254	Optional	Administrative area name, where applicable.
`StateName`	254	Optional	State, province, or county name, where applicable. For addresses where the state or province name is not used on a mailing address by convention, the `State` field will be empty, whereas the `StateName` field will contain the state name or abbreviation.

`Error` Element Structure

The `Error` returned element is optional, and will only be returned if an error occurs.

Supported `Error` fields are as follows:

Field Name	Max Length	Mandatory or Optional	Description
ErrorMesg	254	Mandatory	Error message

Example Returned Data

Following is an example of the data payload in the returned HTTP POST transaction:

```
<?xml version="1.0" encoding="utf-8" ?>
<Transaction>
  <ValidationInfo>
    <URN>ID12345</URN>
    <MQCode>B</MQCode>
    <Country>JAPAN</Country>
    <ISOCountry2>JP</ISOCountry2>
    <Gender>M</Gender>
    <GenderPrefix>M</GenderPrefix>
    <GenderFN>U</GenderFN>
    <InputLanguage>ROMAN</InputLanguage>
  </ValidationInfo>
  <RomanAddress>
    <Name>Mr. Junichiro Koizumi</Name>
    <Prefix>Mr.</Prefix>
    <FirstName>Junichiro</FirstName>
    <LastName>Koizumi</LastName>
    <Title>President</Title>
    <Business>ABC Company</Business>
    <Address1>1-1-5 Roku-Cho</Address1>
    <Address2>Adachi-Ku</Address2>
    <City>Tokyo</City>
    <Zip>121-0073</Zip>
    <Formatted1>1-1-5 Roku-Cho</Formatted1>
    <Formatted2>Adachi-Ku</Formatted2>
    <Formatted3>Tokyo 121-0073</Formatted3>
    <HouseNo>1-1-5</HouseNo>
    <StreetName>ROKU-CHO</StreetName>
    <SubCity>ADACHI-KU</SubCity>
  </RomanAddress>
  <LocalAddress>
    <Name>Mr. Junichiro Koizumi</Name>
    <Prefix>Mr.</Prefix>
    <FirstName>Junichiro</FirstName>
    <LastName>Koizumi</LastName>
    <Title>President</Title>
    <Business>ABC Company</Business>
    <Address1>六町1-1-5</Address1>
    <City>足立区</City>
    <State>東京都</State>
```

```
    <Zip>121-0073</Zip>
    <Formatted1>121-0073</Formatted1>
    <Formatted2>東京都足立区六町1-1-5</Formatted2>
    <HouseNo>1-1-5</HouseNo>
    <StreetName>六町</StreetName>
    <SubCity>足立区</SubCity>
  </LocalAddress>
</Transaction>
```

Producing a Complete Address from the Returned Data

The returned data structure includes the standardized address data in multiple formats, so care must be taken when mapping the returned fields to a final address format. Two alternate sets of fields are provided for this purpose: the Parsed Output Fields and the Formatted Output Fields. To produce a complete output address, use either the Parsed Output Fields or the Formatted Output Fields, but not both. Each set of fields contains a complete copy of the address, so using both will result in duplication of the address.

The Analysis Fields are not intended for address output, and must not be used for this purpose. See below for more information.

Parsed Output Fields

The following fields define the full output address:

Returned Fields	Example of Returned Fields	Complete Address in the Correct Format
Name	Mr. Brad H. Searles	Mr. Brad H. Searles Vice President ABC Company Accounting Office Suite 5 3 rue Sainte-Catherine O Montreal QC H2X 1Z5 CANADA
Title	Vice President	
Business	ABC Company	
Business2	Accounting Office	
Address1	Suite 5	
Address2	3 rue Sainte-Catherine O	
Address3		
Address4		
Address5		
City	Montreal	
State	QC	
Zip	H2X 1Z5	
Country	CANADA	

Note that the City, State and Zip fields are provided as separate values, to support applications that require these values in separate fields. Your application is responsible for formatting these values in the correct sequence for the given country, as shown in the complete address example in the right column above. An alternative is to use the Formatted Output Fields, described below.

Formatted Output Fields

The following fields define the full output address:

Returned Fields	Example of Returned Fields
Name	Mr. Brad H. Searles
Title	Vice President
Business	ABC Company
Business2	Accounting Office
Formatted1	Suite 5
Formatted2	3 rue Sainte-Catherine O
Formatted3	Montreal QC H2X 1Z5
Formatted4	
Formatted5	
Formatted6	
Formatted7	
Formatted8	
Country	CANADA

Note that the City, State and Zip fields are contained within the Formatted lines, already formatted in the correct sequence for the given country.

Analysis Fields

The Analysis Fields (HouseNo, HouseSufx, etc.) are supplied for analysis purposes only. Analysis Fields are populated only when the relevant values were successfully identified in the address. However, there may be important information in the address that is not included in any of the Analysis Fields, therefore these fields cannot be used to produce a complete address. For any address elements that could not be recognized, but are still necessary for a complete address, that information will be present in the Parsed Output Fields and the Formatted Output Fields, but will not appear in any of the Analysis Fields. This is possible even for valid or corrected addresses. For example:

Returned Fields	Example of Returned Fields
Address1	Ambassador Building, Tower 2
Address2	Suite 5
Address3	3 rue Sainte-Catherine O
Address4	
Address5	
City	Montreal
State	QC
Zip	H2X 1Z5
Country	CANADA
HouseNo	3
HouseSufx	Suite 5
StreetName	rue Sainte-Catherine O

Note in the above example that the building name and related information are not present anywhere in the Analysis Fields, but are present in the Address lines. The building-related information was not recognized because it is not part of the standard postcode database for the given country, but the information is still an important part of the complete address. In some situations, these unrecognized values can include critically

important data such as the house number and street name, which are absolutely necessary to form a complete address, yet may not be explicitly identified and parsed by the address verification process.

Attempting to create a full address using the Analysis Fields will result in critical information being omitted from the address.

Error Handling

If an address is processed successfully, the processed address will be returned by the Global-Z server via a response to the client's HTTP POST transaction. However it is possible that a processing error can occur. Errors fall into two general categories: connection errors, and data errors. Connection errors can occur if transmission of the HTTP data across the Internet fails, for example due to network problems. Data errors can occur if the outgoing data stream does not adhere to the agreed-upon data format and/or encoding.

Connection errors caused by some types of network problems cannot be detected by the Global-Z server, therefore connection errors should be handled directly by the HTTP POST function being called by your client application. Data errors will be reported by the Global-Z processing system via the `Error` element in the returned XML transaction, and an error message will be returned in the `ErrorMesg` field.

In rare cases, an error condition may occur because the SaaS service itself is unavailable, such as during scheduled maintenance periods. In this situation, the HTTP POST will receive a response from the server, however the returned transaction will be an HTTP error string rather than an XML transaction. To handle this possibility, the client application should expect to receive XML data only if HTTP status code 200 (OK) is returned. For all other status codes, no XML data will be returned.

If an error occurs, regardless of the type of error, no address information will be returned to the calling application. Therefore it is your application's responsibility to store a copy of the original address information on the client side, in the event that an error condition is encountered.

Persistent Sessions

For best performance, the client application should create a persistent connection to the server which should be used for all transactions ongoing, rather than creating a new session for each transaction. This avoids increased latency caused by the overhead required to re-authenticate each time a new session is started. However, this assumes that transactions are being processed regularly. If transactions are only submitted intermittently, after a period of no activity the session will time out and the connection will be closed by the server. In this event, your client application will then receive a connection error at the next attempted transaction.

For best performance, if no records are submitted by your application in a span of approximately three minutes, it is recommended that your application close the connection to the server, and then open a new connection when your application is ready to submit the next record.

Tips to Improve Address Verification Results

- Include a country name on the input transaction. If a country name is not included, the address verification process will attempt to determine the country from the input address. However the success rate is significantly improved and the error rate decreased when the country name is explicitly included in the input transaction.
- Parsing address values into individual fields is not necessary, but can improve verification rates in some situations. For example, it isn't necessary to ensure that the city or town name, province name, and zip or postal code are mapped into their individual fields on the input transaction, as the address

verification process will attempt to find that information even if it is present in an address line. However, providing that information in individual fields where possible will often improve verification rates and reduce the probability of unrecognized “noise” values being left behind in the address lines after standardization.

- Where possible, supply the input address lines in the sequence expected for the given country’s address format and language. Even if the input address does not contain values in their correct fields, supplying the address lines in the correct top-to-bottom and left-to-right order will generally improve results. Address lines are expected in Western top-down left-to-right sequence when submitting addresses in Roman (Western) text, and bottom-up sequence when submitting address lines in Asian text. Note that the sequence of elements in the XML structure is irrelevant. This comment applies to the sequence of information in the Address1 through Address5 lines only.
- Avoid repeating data values unnecessarily in the address. For example, including the state or province name both in an Address line as well as in the State field can create ambiguity that may adversely impact the address verification process, or can result in “noise” or residual data being left behind in the standardized Address lines.
- Avoid populating both the Name field and the individual parsed name fields (Prefix, FirstName, etc.). These two distinct sets of fields are provided in order to allow for flexibility in the input mapping, however populating both sets can create ambiguity for the name parsing process. If both sets of fields are populated on the input transaction, the verification process will overwrite one of the sets on the returned transaction.
- Invalid addresses (MQCode D, H, I, J, K) are generally not modified by the address standardization process. However some values may be identified and moved into their correct locations, e.g. a company name moved from an Address line to the Business field. Every attempt has been made to prevent unnecessary removal of data from any address, whether a valid, invalid or undeliverable address. The expectation is that some value may be added to an address, even if it is found to be invalid. However users should exercise their own discretion in which changes to accept in any address or in any category of addresses.

Reporting

Global-Z will provide a monthly processing summary report indicating the number of addresses submitted and processed. We can provide samples of these reports for your reference.

Support and System Error Notification

In order to ensure high availability, Global-Z’s processing system utilizes redundant co-located networks on independent Internet backbones. The system is monitored and supported by on-call maintenance engineers 24/7/365. Notification of system outages is reported to internal maintenance staff via call center, email, and pager. If your application requires direct notification of system outages, this can be accommodated.

Immediate engineer support is available for system-related issues. Response time for data processing questions or other processing matters is provided within one business day.

Archiving and Data Retention

Global-Z does not retain any copies or archives of individual addresses processed.

System Testing and Deployment Process

We recommend that prior to production implementation, several phases of testing be performed to ensure that both sides of the interface are functioning as designed. Following are the testing steps that we recommend prior to production launch:

- Initial Data Audit – we recommend that you provide us with a representative sample of your data in its source format so that we can analyze the data and assess initial data quality. This process allows us to identify unique aspects of your data which may need to be addressed prior to submitting the data to the address cleaning process.

- Mapping Audit – once any data-specific issues have been addressed, we suggest that you provide us with a similar data sample formatted in the XML data structure and encoding that will be used for ongoing transactions. This will allow us to identify any data quality issues that arise due to mapping of the data from the source format to the transaction data stream.

- Interface Test – after data mapping integrity has been confirmed, we suggest an initial test of the processing interface. We will provide you with an IP address and SSL certificate for testing, which you can use for processing test transactions to verify that your interface is working correctly. Because this phase is designed to test the interface only at this stage, throughput rates will not necessarily be comparable to production rates.

- Interface Acceptance – after successful interface testing, we will provide you with the IP address and SSL certificate for the production-ready process.

- User Acceptance Testing – you will have the opportunity to test the system at production rates, in order to achieve internal user acceptance on your end.

- User Acceptance and Deployment – upon user acceptance, the system will be deployed to production.

Appendix A – `HygieneDetail` Values

Following are the possible values returned in the `ValidationInfo.HygieneDetail` field:

Country	Valid Codes and Meanings	
AUSTRALIA	0	Correct
	1	Postcode couldn't be matched
	2	State couldn't be matched
	3	Locality couldn't be matched
	4	Street Name couldn't be matched
	5	Ambiguous
	6	Street Number couldn't be matched
	7	Postal Type couldn't be matched
	8	Postal Number couldn't be matched
	9	Phantom Primary Point
	10	Unit Number couldn't be matched
	12	Level Type couldn't be matched
	14	Lot Number couldn't be matched
	18	Primary Point
	20	Parser Warning

Appendix B – Example Python Code

Following is an example of Python code showing a sample HTTP POST transaction, assuming UTF-8 encoding, and the application/octet-stream MIME type:

```
# -*- coding: UTF-8 -*-
import httplib

# Client certificate is in the current directory
CLIENT_CERT = [specify certificate file name provided at account setup]
GZ_HOST = 'realtime.globalz.com'
URI = '/api/av'

# Create a secure connection
conn = httplib.HTTPSConnection(
 GZ_HOST,
 key_file = CLIENT_CERT,
 cert_file = CLIENT_CERT )

# Create the header
header = {"Content-type": "application/octet-stream", "Accept": "identity"}

# Create the delimited input
addressdata = '<?xml version="1.0" encoding="utf-8"?>'+\
        '<Transaction>'+\
          '<Input>'+\
            '<URN>ID12345</URN>'+\
            '<Prefix>Mr.</Prefix>'+\
            '<FirstName>Junichiro</FirstName>'+\
            '<LastName>Koizumi</LastName>'+\
            '<Title>President</Title>'+\
            '<Business>ABC Company</Business>'+\
            '<Address1>1-1-5 Roku-cho</Address1>'+\
            '<Address2>Adachi-ku tokyo</Address2>'+\
            '<Country>japan</Country>'+\
            '<Accents>Yes</Accents>'+\
            '<Casing>Mixed</Casing>'+\
            '<Language>Both</Language>'+\
          '</Input>'+\
        '</Transaction>'

# POST it
conn.request('POST', URI, addressdata, header)

# Wait for response
response = conn.getresponse()
ret_dat = response.read()

# Display input and output
print 'in-->' + addressdata
print '\nout->' + ret_dat

# Close the connection
conn.close()
```

Appendix C – Example PHP Code

Following is an example of PHP code showing a sample HTTP POST transaction, assuming UTF-8 encoding, and the application/octet-stream MIME type:

```
<?php

        # Client certificate is in the current directory
        $client_cert = [specify certificate file name provided at account setup] ;
        $gz_host = "https://realtime.globalz.com" ;
        $uri = "/api/av" ;

        # Create the delimited input
        $addressdata = '<?xml version="1.0" encoding="utf-8"?>' .
                '<Transaction>' .
                  '<Input>' .
                    '<URN>ID12345</URN>' .
                    '<Prefix>Mr.</Prefix>' .
                    '<FirstName>Junichiro</FirstName>' .
                    '<LastName>Koizumi</LastName>' .
                    '<Title>President</Title>' .
                    '<Business>ABC Company</Business>' .
                    '<Address1>1-1-5 Roku-cho</Address1>' .
                    '<Address2>Adachi-ku tokyo</Address2>' .
                    '<Country>japan</Country>' .
                    '<Accents>Yes</Accents>' .
                    '<Casing>Mixed</Casing>' .
                    '<Language>Both</Language>' .
                  '</Input>' .
                '</Transaction>' ;

        # Create the header
        $header = array("MIME-Version: 1.0",
                "Content-type: application/octet-stream",
                "Content-length: ".strlen($addressdata),
                "Connection: close");

        # Set Curl Options to allow SSL connection with certificate and host verification
        $ch = curl_init();
        curl_setopt($ch, CURLOPT_FAILONERROR, TRUE) ;
        curl_setopt($ch, CURLOPT_POST, TRUE) ;
        curl_setopt($ch, CURLOPT_URL,$gz_host . $uri) ;
        curl_setopt($ch, CURLOPT_RETURNTRANSFER, TRUE) ;
        curl_setopt($ch, CURLOPT_TIMEOUT, 10) ;
        curl_setopt($ch, CURLOPT_SSLCERT, $client_cert) ;
        curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, FALSE) ;
        curl_setopt($ch, CURLOPT_SSL_VERIFYHOST, TRUE) ;
        curl_setopt($ch, CURLOPT_HTTPHEADER, $header) ;
        curl_setopt($ch, CURLOPT_POSTFIELDS, $addressdata) ;

        # Execute the Curl Transaction
        $data_return = curl_exec($ch);
        if (curl_errno($ch)) {
                # Curl encountered an error, handle appropriately
```

```
	} else {
		curl_close($ch);
		if (strlen($data_return) == 0){
			# Record has timed out, handle appropriately
		}else{
			if( !strstr($data_return,"</Transaction>") ){
				# Process has returned an error, handle appropriately
			}
		}
	}

	# Display input and output
	echo "Input --> " . htmlentities($addressdata) . "<br>\n" ;
	echo "Output --> " . htmlentities($data_return) . "<br>\n" ;

?>
```

Global-Z International, Inc.
PO Box 1524
Bennington, VT 05201
800.447.3978 Toll Free
802.445.1011 Phone
802.445.1016 Fax
info@globalz.com
www.globalz.com