GeoPinpoint Suite – Application Program Interface

User Manual
v2010.3

Published Date: August 15, 2010

DMTI Spatial Inc.
15 Allstate parkway,
Suite 400
Markham, Ontario
L3R 5B4
www.dmtispatial.com

Email: info@dmtispatial.com
Phone: (877) 477-3684
Fax: (905) 948-9404

Table of Contents

About DMTI Spatial

DMTI Spatial™ Inc. is Canada's leading Location Intelligence provider. We enable users to understand their customers, optimize resources, realize opportunities, maximize profitability and make more informed decisions through accurate products and innovative thinking.

DMTI Spatial publishes precision built street map, rail and routing data (CanMap®), a detailed water layer, and innovative geocoding and address management software (GeoPinpoint™). In addition, DMTI Spatial publishes a full range of positionally accurate geospatial data products including: enhanced points of interest (EPOI), census data and boundaries, postal geography, topographic maps, and US mapping data. As part of a complete business geographic solution, DMTI Spatial™ offers a wide range of GIS services, consulting, and software training.

Established in 1994, DMTI Spatial is committed to setting the standard within the GIS industry for precision built geo-spatial data and address management services.

At DMTI Spatial, we believe that our true strength comes from working closely with our customers and providing innovative solutions to meet their strategic business objectives. As Canada's premier spatial solutions provider we pride ourselves with having worked with North America's leading organizations to support their mission critical applications.

DMTI Spatial works with large and small organizations representative of a wide variety of industries:

- Agriculture
- Banking/Finance
- Consulting
- Education
- Emergency Services
- Engineering
- Environmental
- Forestry
- Government
- Health
- High Technology
- Insurance
- Manufacturing
- Media
- Mining
- Real Estate
- Retail
- Telecommunications
- Transportation
- Utilities

We are a member of the ESRI Canada Business Partner Program, and winner of the 2001 ESRI Worldwide New Business Partner of the Year Award and the 2005 ESRI Foundation Partner of the Year Award. We are a strategic business partner of MapInfo and winner of the Markham Board of Trade 2000 Award for Entrepreneurship and Innovation. Recipient of The Association of Canadian Map Libraries and Archives (ACMLA) 2002 Certificate of Appreciation.

ESRI Canada
Business Partner

MapInfo
Business Partner

Really Smart Spatial Solutions™

Through the application of its products and services, DMTI Spatial™ has been involved with projects such as: location-based services, logistics planning, emergency dispatch, facilities management, data management, customer care, address management, land base development in support of network planning, and marketing/demographic analysis applications.

DMTI Spatial™ can provide all of the components necessary for the acquisition, implementation, operation and maintenance of a successful GIS system within companies of all sizes. Through its product and service offering, DMTI Spatial™ can provide users with 5 key components:

1. Accurate, detailed, and compatible data
2. Comprehensive maintenance program
3. GIS software
4. Consulting and services
5. Software training

DMTI Spatial™ Product & Service Portfolio

DMTI Spatial's product & service offering includes:

CanMap® - *Digital Map Data for Canada*
- CanMap® Streetfiles
- CanMap® RouteLogistics
- CanMap® Rail
- CanMap® Major Roads and Highways
- CanMap® Parks & Recreation
- CanMap® Water

Satellite Imagery
- Satellite StreetView™

Municipal Amalgamations
- CanMap® Municipality Amalgamation File (MAF)

Business & Recreational Points of Interest
- Enhanced Points Of Interest (EPOI)

GeoPinpoint™ Suite
- Canada's Geocoding Solution
- Modular Architecture
- Windows Standalone Desktop Version
- UNIX, Java Wrapper, ActiveX (DLL Version)

Topographic Data and Base Maps
- Canadian Atlas Map Bundle (CAMB)
- Populated Placenames
- National Topographic Data Base (NTDB)
- 30 & 90m Digital Elevation Models (DEM)
- Clutter Data

Postal Geography - Platinum Postal Code^{OM*} Suite
- Six-Digit Postal Code File (LDU Boundary)
- Enhanced Postal Code File (MEP)
- Forward Sortation Areas (FSA) Boundary File

1996 Census Boundaries & Demographic Data
- Enumeration Area (EA)
- Census Subdivision (CSD)
- Census Division (CD)
- Census Metropolitan Area/Census Agglomeration (CMA/CA)
- Census Tract (CT)
- Federal Electoral Districts (FED)

2001/6 Census Boundaries
- Dissemination Area (DA)
- Census Subdivision (CSD)
- Census Division (CD)
- Census Metropolitan Area/Census Agglomeration (CMA/CA)
- Census Tract (CT)
- Federal Electoral Districts (FED)

GIS Software
- Contour Modeling and Display
- Demographic Profiling and Lifestyle Targeting
- Geocoding and Mapping Software
- Routing and Logistics

Consulting and Services
- Address Management Solutions
- Application Development
- Database Marketing
- Data Conversion and Creation
- Database Scrubbing
- Geocoding Services
- GIS Consulting
- Technical Support

*Postal code is an official mark of Canada Post Corporation

Technical Support, Error Reporting & Product Enhancement Services

DMTI Spatial is committed to building the best products possible for our customers. By using our data every day in your mission critical application you are our best source for product refinement. Please let us know if you have an enhancement request or found an error in any of our products so that we can make the correction for the next release.

This is your opportunity to provide feedback directly to the DMTI Spatial Product Development Team. Please be as specific as possible so that we can improve our products quickly and accurately. To submit an error or request technical assistance please visit:
http://www.dmtispatial.com/en/Resources/TechSupport.aspx

If you have an idea for a new product, or an enhancement request for an existing product, please e-mail:
pm@dmtispatial.com

Contact Information

DMTI Spatial Inc.
15 Allstate Parkway, Suite 400
Markham, Ontario
L3R 5B4 Canada

Telephone: 905-948-2000
Toll Free: 1-877-477-DMTI (3684)
Fax: 905-948-9404

Web Site: www.dmtispatial.com
Error Reporting Service: http://www.dmtispatial.com/helpdesk/index.aspx
Product Enhancement Requests: pm@dmtispatial.com
Technical Support: http://www.dmtispatial.com/helpdesk/index.aspx

Trademarks and Notices

DMTI Spatial is an Authorized User and Distributor of selected Statistics Canada Computer File(s) under Licensing Agreement 6228. No confidential information about an individual, family, household, organization or business has been obtained from Statistics Canada.

Postal Code is an official mark of Canada Post Corporation.

Digital Topographic Data produced under License from Her Majesty the Queen in Right of Canada, with permission of Natural Resources Canada and The Queen in Right of Manitoba.

About GeoPinpoint™ Suite

Overview

The GeoPinpoint Suite software attaches geographic coordinates to records in a client database by means of matching certain database fields against a DMTI proprietary geo-reference database. The geo-reference database is comprised of digital street geometry, street address ranges, postal coordinates, point of interest and other reference databases to ensure that data is “geocoded” as accurately as possible.

When data is “geocoded”, co-ordinates can be transferred into a Geographic Information Systems (GIS) such as MapInfo, ArcInfo, ArcView and other software systems that support the importation of geographic co-ordinate locations.

GeoPinpoint™ Suite positions your data using a powerful and innovative geo-location process called geocoding. GeoPinpoint Suite attaches X and Y coordinates to your facility, customer or prospect address data for map visualization, analysis or location based applications. The GeoPinpoint Suite takes advantage of a new modular design that allows the software to encompass future module enhancements without jeopardizing its performance or usability. Based on the nationwide precision and the robust street address content of CanMap® Streetfiles, GeoPinpoint Suite has been engineered to geocode your data with a high degree of accuracy.

Features

GeoPinpoint Suite has a high degree of flexibility allowing users to customize their geocoding process by defining their “geocoding path”. This flexibility provides a seamless and hierarchical interoperability between GeoPinpoint Suite’s highly developed licensed modules:

- ***Parser:*** Evaluates address information, parses and standardizes data thereby improving address-matching rates. The address-parsing module specifically accommodates Canadian addressing methods and can interpret French addressing standards.
- ***Address Geocoder:*** Evaluates address information and returns an accurate geographic location in the form of longitude and latitude from the geo-reference database. Options are available allowing users to relax matching criteria by street prefix, street type, and street direction.
- ***POI Geocoder:*** Evaluates a Point of Interest name and returns an accurate geographic location in the form of longitude and latitude from the geo-reference database.
- ***Postal Code Geocoder:*** Evaluates address information and returns Postal Code locations in the form of longitude and latitude from the geo-reference database.
- ***Segment Geocoder:*** Evaluates address information and returns the midpoint of the corresponding street segment from the geo-reference database.

To further enhance geocoding, GeoPinpoint Suite is currently supplemented with the following modules (which are available with the appropriate license keys and corresponding geo-reference databases):

- ***Soundex:*** An algorithm that uses fuzzy logic to help geocode street names, municipalities or points of interest, which may suffer from spelling variations.

This manual has been written to provide clients with the information they need to effectively use GeoPinpoint Suite. It is noted, however, every client has unique circumstances that may pose challenges to geocoding in addition to those covered in the GeoPinpoint Suite manual.

Please contact DMTI Spatial for more information or questions regarding your specific requirements.

DMTI Spatial Inc.
15 Allstate Parkway, Suite 400
Markham, Ontario
L3R 5B4 Canada

Telephone:	905-948-2000
Toll Free:	1-877-477-DMTI (3684)
Fax:	905-948-9404
URL:	www.dmtispatial.com
Email:	info@dmtispatial.com

Software Changes

- DMTI has added 3 new geocoding paths for geocoding to 'In-Between' addresses; 34530, 34520, and 34510.

- Zero Address segment geocoder has been added. Function allows users to geocode to streets which are completely non-addressed (i.e.: range for all segments is 0 to 0). The ability to output the total segment length of addresses that geocode to zero addresses has been added as well.

- Inset tolerance function has been added. Function allows users to specify an inset tolerance value so that coordinates falling on or close to intersections are placed more appropriately.

- Return actual address value when geocoding to closest addresss function has been added.

- Opposite Side of street address geocoding function had been added which contains a tolerance search. This function is similar to the closest address function. Return actual address value when geocoding to opposite street function has been added.

- Soundex functionality has been introduced to municipality names.
  - setsearchmunicipalitysoundex()

- Added municipality soundexing function, as well as inputting custom soundex tolerance.

Documentation Changes

- Removed software installation guidelines, now contained within 'GPP_Software_Install_Manual_200x.x_vxx.doc'

- Geocoder.h header file descriptions updated to reflect 'In-Between' geocoding.

- Added 'In-Between Function' to result code listing.

- New result codes were added to the 8th digit (REFINED) to highlight those records that either geocoded By Muni and FSA or where the new Rural Postal Code to PPN function was used

- Added 'In-Between Function' to precision code listing.

Canadian Geography and Data - Summary

GeoPinpoint Suite software utilizes DMTI Spatial Canadian data to provide the most precise geocoded results to its users.

The following Canadian data is found in the geo-reference database by each province:

- CanMap Street Files
- Census Subdivision (CSD) centroids
- Points of Interest (POI)
- Populated Place Names (PPN)
- Platinum Postal Code^{OM*} Suite (PPCS)
- Platinum Postal Code^{OM*} Suite - Forward Sortation Area (FSA) centroids

Refer to section “Geo-reference Database” to learn more information about the role this feature plays in the GeoPinpoint Suite software.

Coordinate Systems

All geo-referenced data used are in Latitude/Longitude decimal degrees.

Basic Canadian Geography

Canada contains 10 province and 3 territories*.

Province Name (English)	Province Name (French)	Abbrev
Alberta	Alberta	AB
British Columbia	Colombie-Britannique	BC
Manitoba	Manitoba	MB
Newfoundland and Labrador	Terre-Neuve et Labrador	NL
New Brunswick	Nouveau-Brunswick	NB
Northwest Territories*	Territoires du Nord-Ouest	NT
Nova Scotia	Nouvelle-Écosse	NS
Nunavut*	Nunavut	NU
Ontario	Ontario	ON
Prince Edward Island	Île-du-Prince-Édouard	PE
Quebec	Québec	QC
Saskatchewan	Saskatchewan	SK
Yukon*	Yukon	YT

CanMap Street Files

The geo-reference database used by GeoPinpoint Suite to geocode all Canadian data is based on the CanMap® street file. The CanMap® street file includes street centerlines and address ranges, and it is updated by DMTI Spatial on an ongoing basis. CanMap® contains street naming for communities down to, and in some cases, under 1,000 population and street addressing for communities down to, and in some cases, under 2,000 population. This data is North America’s #1 choice for Canadian data provides an accurate map fabric for wireless and location based service applications (LBS), market analysis, target marketing, site location analysis, customer service and asset management.

Canadian Geography and Data – Summary (cont'd)

This data consists of the following:

- Over 1.5 million km across Canada
- Street centerline road network, names and address ranges
- Roads look up table including alias name, highway numbers and names, road numbers and more

This information is used to geocode records to address number, street alias and other GeoPinpoint Suite functionality.

Census Subdivision (CSD) centroids

CSDs are municipalities or equivalent areas (e.g.: Indian reserves, Indian settlements and unorganized territories) as determined by provincial legislation. The municipal boundaries are derived from the 1996 Census Subdivision (CSD) boundary files.

In Newfoundland, Nova Scotia and British Columbia, the term CSD also applies to geographic area that have been created by Statistics Canada to assist with reporting of statistical data.[1]

Municipal Centroids

The municipal centroids are located based on shorelined CSD boundaries.

Municipal Aliases

This file is a combination of CSD names, formerly used names (Downsview, Etobicoke), and a total of names collected by observation through the extensive data processing undertaken by DMTI Spatial.

Note: There are two municipality (CSD) values in Canada which occur between 2 provinces.
Flin Flon (Manitoba and Saskatchewan) and Lloydminster (Saskatchewan and Alberta).
Each CSD part occurring in the above province is represented as a separate CSD value.

[1] [1] **Source:** Statistics Canada - Catalogue No. 92-351-UIE, 1996 Census Dictionary, Final Edition Reference. August 1999

Canadian Geography and Data – Summary (cont'd)

Points of Interest (POI)

The Enhanced Points of Interest (EPOI) file is a national database of Canadian business and recreational points of interest. Engineered using CanMap® Streetfiles, each EPOI has been accurately geocoded and precisely placed; two criteria that are fundamental to any successful location sensitive service.

This location enriched point of interest database allows users to see and analyze selected point of interest data in a given geographic area, enabling applications such as wireless location-based services (LBS), Web, Telematics, planning, real estate multiple listing services (MLS), retail site analysis, competitive and market research, intelligent routing, sales territory analysis, business and tourism.

DMTI Spatial currently stores approx ~40,000 POI points in the geo-reference database. The following types of POI data are stored:

- Aerodromes
- Border Crossings and Custom Offices
- Car Rental Agencies
- Hotel Accommodations
- Golf Courses
- Police Stations
- Toll Booths
- Education & Health Care
- Financial Institutions
- Tourist Information

Refer to Appendix 3: Points of Interest Layers for more information.

Canadian Geography and Data – Summary (cont'd)

Populated Place Names (PPN)

The Populated Placenames database includes Cities, Towns, Villages, Communities, Boundaries and other records describing 'Populated' places across Canada.

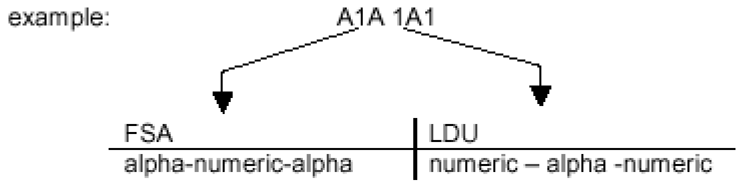
Based on the Canadian Geographic Names Database (CGNDB) as well as the toponymic data from the National Topographic Database (NTDB), this data product has been enhanced by more accurately aligning the features in relationship to CanMap ® Streetfiles. Each point is flagged with a "Precison" code, which categorizes the feature based on its positional accuracy. This makes PPN information a cost effective data product for macro level geocoding and mapping applications. This accurate placement of data points is due to CanMap Streetfiles positional accuracy and additional research and verification

Platinum Postal Code^{OM*} Suite (PPCS)

The **CanMap Six Digit Postal Code** product is a precision-based point file of postal codes across Canada. This Six Digit Postal Code product contains over 800,000 postal code points positioned to the most representative address. The majority of the points are located to the specific address (in the case of large apartment buildings or office towers) or the most representative address [where the same FSALDU (forward sortation area local delivery unit) services multiple addresses]. This degree of positional accuracy is made possible because of the postal codes are geocoded using CanMap® street map data.

A Postal Code, otherwise know as – 6 Digit Postal Code, or "FSA LDU", is defined and maintained by Canada Post for the sorting and delivery of mail. Postal Codes are also widely used as a means of geocoding databases for the purposes of location analysis, demographic analysis, and other types of geographical enabled analyses.

A postal code is comprised of an FSA LDU, or in other words - a Forward Sortation Area and a Local Delivery Unit. The characters are arranged in the form "ANA NAN" where "A" represents an alphabetic character and "N" represents a numeric character (i.e. L3R 9T8). The first character of a postal code is allocated in alphabetic sequence from East to West across Canada and denotes a province, territory or a major sector found entirely within the boundaries of a province.[1]



Rural postal codes can be distinguished from urban postal codes because the second character is "0" (zero). One rural postal code, with multiple positions, may represent several small rural towns.1

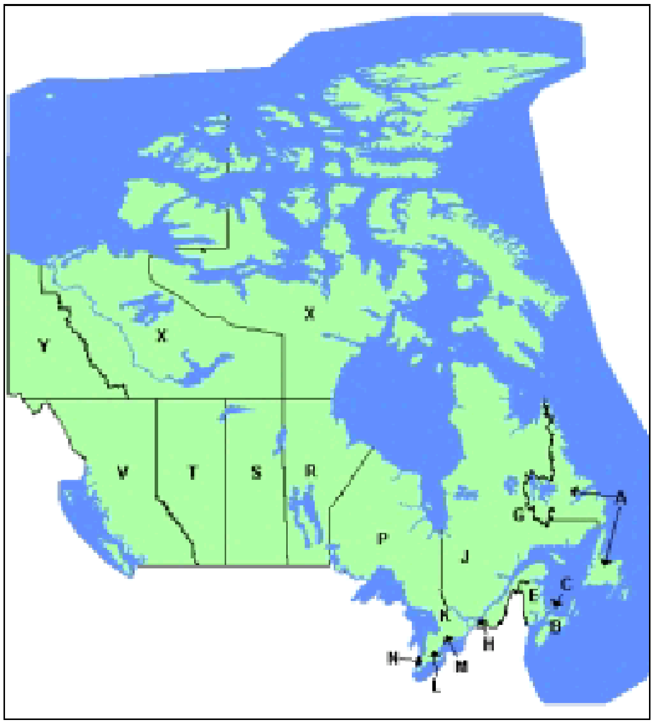
[1] **Source:** Understanding Postal Code Files – Multiple and Unique Enhanced Postal Code Files v.s. the Six Digit Postal Code Files, DMTI Spatial whitepaper, 2003

Canadian Geography and Data – Summary (cont'd)

Platinum Postal Code^{OM*} Suite - Forward Sortation Area (FSA) Boundaries

The first three characters of a postal code represent the Forward Sortation Area (FSA) indicating a geographic area in an urban or rural area. The first character of the Forward Sortation Area identifies one of the 18 major geographic areas, provinces or districts.

First Letter of FSA	Geographic Area
A	Newfoundland & Labrador
B	Nova Scotia
C	Prince Edward Island
E	New Brunswick
G	Quebec (east)
H	Québec (metropolitan Montréal)
J	Quebec (west)
K	Ontario (east)
L	Ontario (central)
M	Ontario (metropolitan Toronto)
N	Ontario (southwest)
P	Ontario (northern)
R	Manitoba
S	Saskatchewan
T	Alberta
V	British Columbia
X	Northwest Territories/Nunavut
Y	Yukon Territory



The second numeric character (numerals 0-9) of the Forward Sortation Area Boundary identifies either an urban postal code or a rural postal code. Rural postal code are represented by the numeral 0 (zero) for example, (A0A) and are serviced by rural route drivers and/or postal outlets. An urban postal code is represented by the numerals 1 to 9 for example, (E2J) and are generally serviced by letter carrier or community mailboxes.

The third character of the Forward Sortation Area segment (E2J) in conjunction with the first two characters, describes an area of a city or town or other geographic area.[1]

Platinum Postal Code^{OM*} Suite - Local Delivery Unit (LDU)

The last three characters represent the Local Delivery Unit (LDU) identifying a specific business or residential point of delivery located within a Forward Sortation Area.

FSA boundaries may include multi-polygon regions, for example two or more polygons forming one region/entity reflecting the complexity inherent in FSA geography. Generally FSA boundaries conform to streets, administrative boundaries and other physical features within CanMap ® products.[1]

[1] Source: CanMap Postal Geography. Feb 2004.

Geo-reference Database

Source Data

The geo-reference database is a proprietary database supplied by DMTI Spatial that must be used in the GeoPinpoint Suite. The geo-reference database is released quarterly, in line with the updates to the CanMap Streetfiles.

The geo-reference database contains the following data:

- CanMap Street Files
- CSD centroids
- Points of Interest (POI)
- Populated Place Names (PPN)
- Platinum Postal Code^{OM*} Suite - (FSA centroids)

For a more detailed explanation of this data, refer to above section “Canadian Geography and Data - Summary”

Version Changes

For each major version of the GeoPinpoint Suite software (e.g.: v3.x, v4.x, v5.x, v6.x), significant geo-reference database changes have occurred as a result of added functionality to the software or database structure. Therefore, it is important to use the most current version of the geo-reference database to take advantage of these new additions.

Location

The Geo-reference folder is located where the software was installed. Refer to above installation steps for more information. Do not select any of the corresponding folder under the Georef folder (e.g.: Ab, Com, etc) as the software will then be unable to read the contents of the geo-reference database.

Defining Data Sources *(cont'd)*

Software Versions

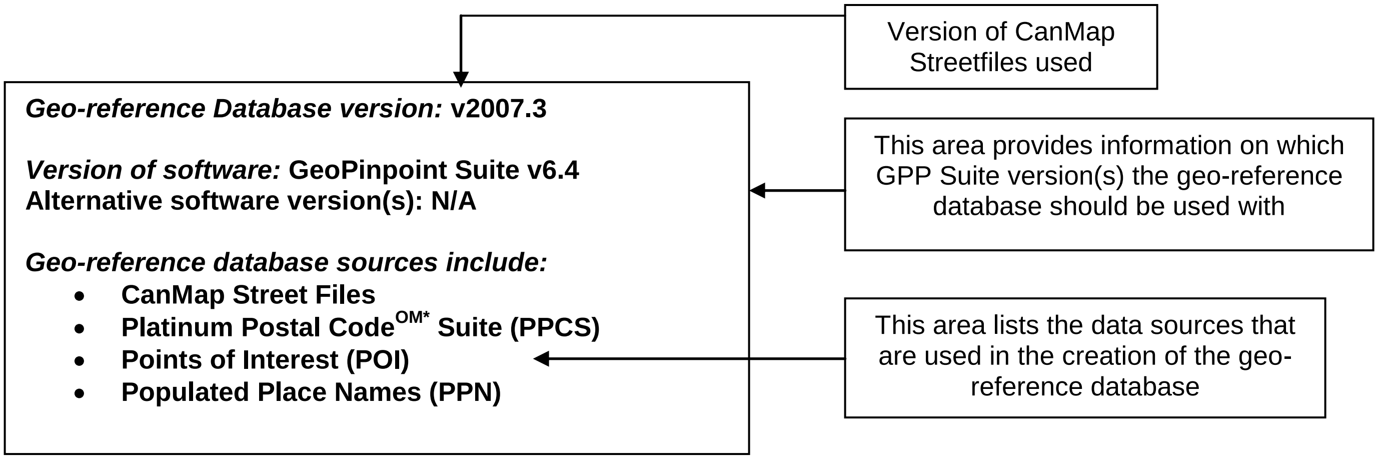
The following table outlines the software versions and the correct geo-reference database versions they can interact with:

Release Date	GeoPinpoint Suite Version	Geo-reference version (version.txt)
Nov-03	v5.0	v7.3
Feb-04	v5.1	v8.0
May-04	v5.2	v8.1
Aug-04	v5.2	v8.2
Nov-04	v5.4	v8.3
Feb-05	v5.4	v2005.1
May-05	v5.4	v2005.2
Aug-05	v5.4	v2005.3
Nov-05	BETA v2006.1	v2005.4
Feb-06	BETA v2006.1	v2006.1
May-06	BETA v2006.2	v2006.2
Aug-06	BETA v2006.3	v2006.3
Nov-06	v6.3	v2006.4
Feb-07	v6.4	v2007.1
May-07	v6.4	v2007.2
Aug-07	v6.4	v2007.3
Nov-07	v6.4	v2007.4
Feb-08	v6.4	v2008.1

Version Text File

The version.txt is stored under the Georef folder (e.g.: C:\Georef)

Sample version.txt contents for GeoPinpoint Suite v6.4:



***Geo-reference Database version:* v2007.3**

***Version of software:* GeoPinpoint Suite v6.4**
Alternative software version(s): N/A

Geo-reference database sources include:

- **CanMap Street Files**
- **Platinum Postal Code^{OM*} Suite (PPCS)**
- **Points of Interest (POI)**
- **Populated Place Names (PPN)**

Version of CanMap Streetfiles used

This area provides information on which GPP Suite version(s) the geo-reference database should be used with

This area lists the data sources that are used in the creation of the geo-reference database

Defining The Geocoding Path

The geocoding functions are organized in a hierarchal order. They are listed in order from most precise (*Address Geocoder*) to least precise (*Boundary Geocoder*). Refer to section "*Geocoder.h Header File Values*".

Geocoding precision depends on two things:

1) The input data that is being geocoded
2) The level of geocoding precision the user wishes to achieve

Refer to *Appendix 2: Interpretation of Precision Code* for more information on precision within GeoPinpoint Suite.

First, input data is important to consider when trying to achieve a certain level of geocoding precision (e.g.: To address). For example, if a user was to select the address geocoder by municipality – the input record must contain the following address components (unparsed or parsed) in order to geocode:

- Address Number
- Street Name
- Street Type
- Street Direction
- Municipality
- Province

Refer to the geocoding functions below to obtain more information on what is expected by each geocoding module when they geocode input data.

For those times when the address information is incorrect or missing for the address geocoder to find a match, the GeoPinpoint Suite software allows the users to select functions which will assist in the geocoding process. For example, the record has the address number and street name but is missing the street type – in this situation, a relax type function may be warranted such as the '*Relax on Street typ*e' function can be used.

Second, the user may determine that they only need to geocode their data to postal code centroid, as this level of precision satisfies their project data requirements. For example, a user may have a database with address records but only wishes to geocode to postal code. As long, as postal code data exists within the database – the software will return the user-defined precision to the user.

Note: If all of the geocoding functions are initialized – GeoPinpoint Suite will follow the geocoding sequence outlined in Appendix 5: Geocoding Sequence.

Each geocoder (address, POI, postal code, segment) module has functions that either geocode to *By Municipality* or *By FSA.* These types of functions are common to all of the geocoder modules listed below.

Defining The Geocoding Path *(cont'd)*

Geocoding *By Municipality and FSA…*

Each geocoder module has a function <*By Municipality and FSA*> (see Address Geocoder | By Municipality and FSA in the example above). These functions, depending on the geocoder module used will:

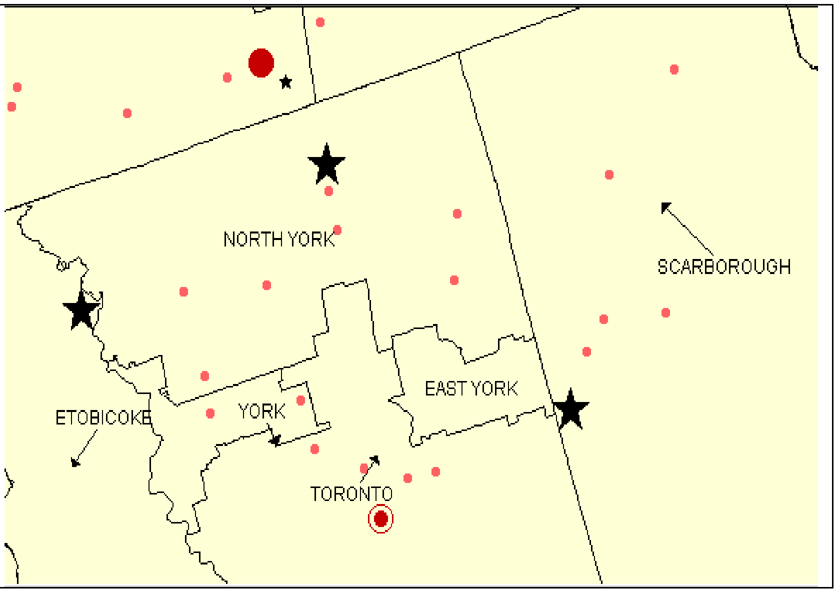
1. Accept the input data
2. Search the geo-reference database
3. Geocode the record after confirmation of its existence in the input municipality and FSA boundary.

This is an important function because it helps to properly geocode those records, which use municipality values, which are a MAF value (i.e.: one muni value is equivalent to many muni value). An example of a MAF value is Toronto, which is equivalent to 6 CSD values (Toronto, North York, Etobicoke, East York, York and Scarborough).

Example Input record:

Address	28 Byng Ave
Municipality	Toronto
Province	Ontario
Postal Code	

In this example, the address 28 Byng Ave, Toronto, ON (i.e.: large stars) is found in the 1996 CSDs of Etobicoke, North York and Scarborough. This address can only be geocoded to the correct location when the Address By Municipality and FSA or Address By FSA geocoder functions are used.



Defining The Geocoding Path *(cont'd)*

Geocoding By Municipality…

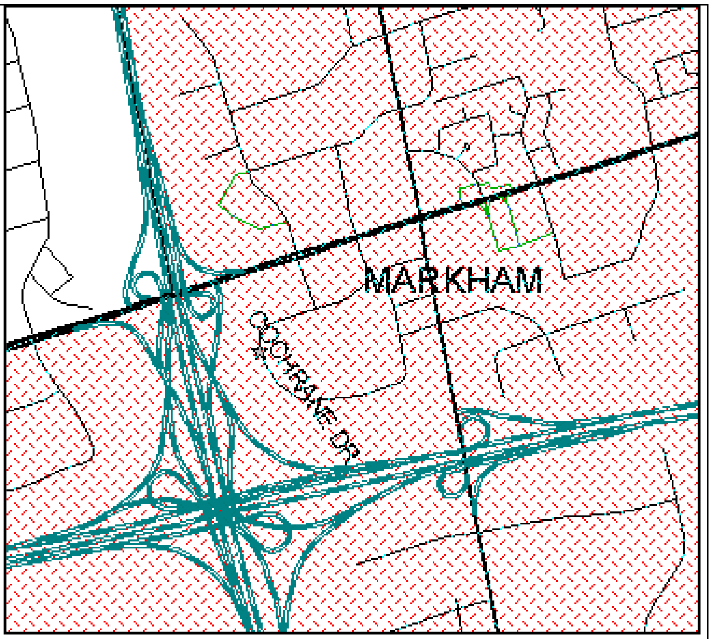
Each geocoder module has a function <*By Municipality*> (see Address Geocoder | By Municipality in the example above). These functions, depending on the geocoder module used will:

1. Accept the input data
2. Search the geo-reference database
3. Geocode the record after confirmation of its existence in the input municipality boundary.

Example Input record:

Address	625 Cochrane Drive
Municipality	Markham
Province	Ontario
Postal Code	L3R9R9

In this example, the address 625 Cochrane Dr is geocoded within the municipality boundary or polygon of Markham, Ontario.



Defining The Geocoding Path *(cont'd)*

Municipality Aliasing

The geocoder modules <*By Municipality*> functions also use an internal process called "Municipality Aliasing", which searches an alias table stored in the geo-reference database. This table is a combination of CSD names, formerly used names (e.g.: Downsview, Etobicoke), and a total of names collected by observation through the extensive data processing undertaken by DMTI Spatial. The purpose of this alias table is to allow users to enhance their geocoding match rate by providing links between municipality values where one value may be known as another. This table is updated on a semi-annual basis.

The input municipality value of Toronto can either mean one of two things:

1. The user knows/believes that the address occurs within the municipality of Toronto
2. The user has mistaken the value of Toronto to represent the true municipality value of a smaller division of Toronto (e.g.: Etobicoke)

This table provides examples of possible municipality values which are sometimes used interchangeably with the value Toronto.

Input Municipality Values	Possible Municipality Alias Values
Toronto	East York
	North York
	Scarborough
	Etobicoke
	York
	Metropolitan Toronto

By looking at the above table, we can see that Toronto is used sometimes instead of Etobicoke when trying to geocode a record. This may be because they are not from the area or only know of Toronto as a general municipality value.

When GeoPinpoint Suite looks up this address in the geo-reference database, it will first search for the information that falls within the original input municipality value ("Toronto") and any alias values that are related to Toronto. In this example, if Toronto is not found to contain this address information, two municipality results are returned to the software: Scarborough and Dryden. This street can also be found in Dryden but because it does not have a municipality alias relationship to Toronto, it will not be considered as a match by the software.

There are times where a record may geocode incorrectly to a different municipality value other than what was intended by the user. A reason for this is that the address number and street do not occur in that particular municipality.

Therefore, the information does not exist in Toronto but was instead found to exist in other municipalities. When GeoPinpoint Suite Windows encounters multiple solutions (i.e.: same address occurring in related municipalities), it will return the first value that it finds. Records which geocode using municipality alias can be identified for the user via the results codes (i.e.: 5th digit of result code = 6). Refer to *Appendix 2 – Interpretation of Result Code (Rcode).*

Defining The Geocoding Path *(cont'd)*

Identifying records which geocoded using the municipality alias functionality can allow the user to:

- Examine those records whose municipality values may have been entered incorrectly OR
- Communicate a missing street segment and/or range to DMTI Spatial via their error reporting website (http://www.dmtispatial.com/helpdesk/index.html)

Defining The Geocoding Path *(cont'd)*

Geocoding *By FSA…*

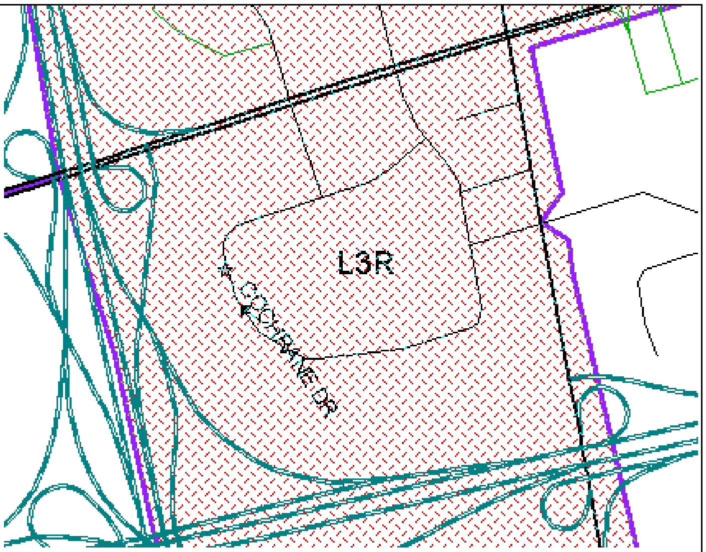
Each of the geocoder modules also has a function <*By FSA*> (see Address Geocoder | By FSA in the example above). These functions, depending on the geocoder module used will:

1. Accept the input data
2. Search the geo-reference database
3. Geocode the record after confirmation of its existence in the input FSA boundary. GeoPinpoint Suite will take the FSA value (e.g.: L3R) from the postal code (e.g.: L3R9R9).

Example Input record:

Address	625 Cochrane Drive
Municipality	Markham
Province	Ontario
Postal Code	**L3R**9R9

In this example, the address 625 Cochrane Dr is geocoded within the municipality boundary or polygon of Markham, Ontario.

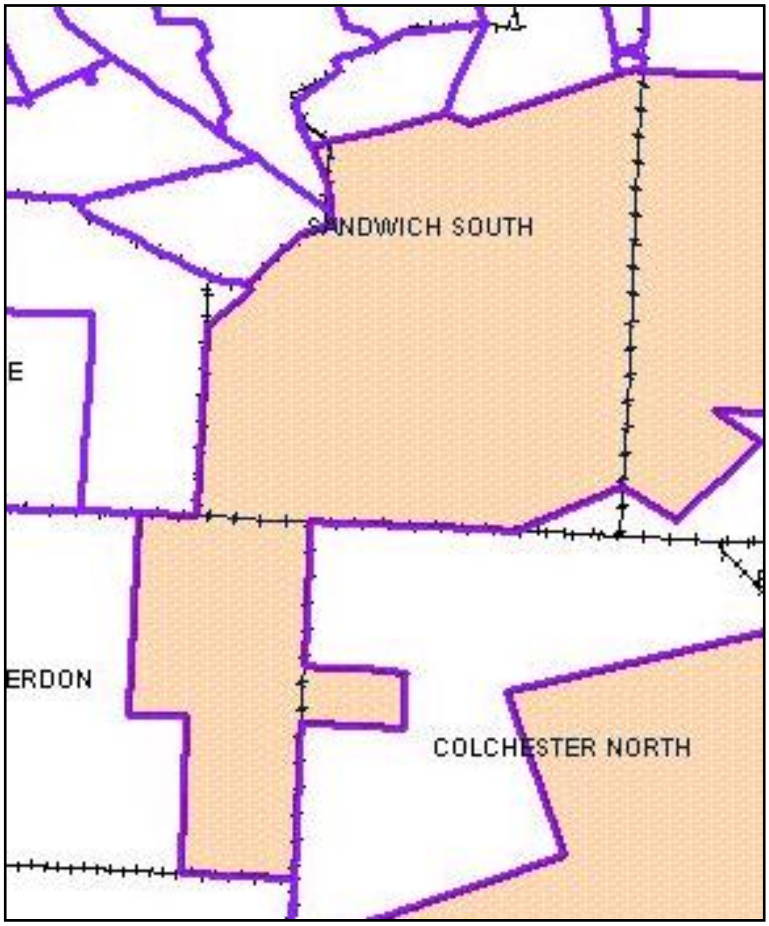


Defining The Geocoding Path *(cont'd)*

Using geocoding functions *By FSA* also provides another method for geocoding when the following scenario is encountered:

- Sometimes an FSA encompasses two or more municipalities in some rural areas. For example, this would allow the user to find a street, which may pass between two municipalities.

Here is an example of an FSA boundary which contains more than one municipality:



Defining The Geocoding Path *(cont'd)*

Address Geocoder

The Address Geocoder is able to geocode the following types of data:

- Unparsed address data
- Parsed address data
- Intersection data

The Address Geocoder matches address input data against the information stored within the geo-reference database.

This section of the document is useful when encountering data which is missing address information such as street type or street direction.

Address Geocoder | By Municipality and FSA | Segment Data Model: This function is used to geocode address information using the input municipality name and FSA value as the boundaries for geocoding.

Unparsed data

Street	***Muni***	***Prov***	***Postal Code***
625 Cochrane Dr	**Markham**	ON	**L3R**9R9
20 Water St N	**Kitchener**	ON	**N2H** 5A5

Parsed data

Address Number	***Street Name***	***Street Type***	***Street Direction***	***Muni***	***Prov***	***Postal Code***
625	Cochrane	Dr		**Markham**	ON	**L3R**9R9
20	Water	St	N	**Kitchener**	ON	**N2H** 5A5

Intersection data

Street	***Muni***	***Prov***	***Postal Code***
YONGE ST && YORKVILLE AVE	**Toronto**	ON	**M4W**

Defining The Geocoding Path *(cont'd)*

Address Geocoder | By Municipality | Segment Data Model: This function is used to geocode address information using the input municipality name as the boundary for geocoding.

Examples of records which can be geocoded using this function are:

Unparsed data

Street	***Muni***	***Prov***	***Postal Code***
625 Cochrane Dr	**Markham**	ON	
20 Water St N	**Kitchener**	ON	

Parsed data

Address Number	***Street Name***	***Street Type***	***Street Direction***	***Muni***	***Prov***	***Postal Code***
625	Cochrane	Dr		**Markham**	ON	
20	Water	St	N	**Kitchener**	ON	

Intersection data

Street	***Muni***	***Prov***	***Postal Code***
YONGE ST && YORKVILLE AVE	**Toronto**	ON	

Address Geocoder | By FSA | Segment Data Model: This function is used to geocode to an address point using the FSA (Forward Sortation Area) value from the postal code as the boundary for geocoding.

Examples of records which can be geocoded using this function are:

Unparsed

Street	***Muni***	***Prov***	***Postal Code***
625 Cochrane Dr		ON	**L3R**9R9
20 Water St N		ON	**N2H** 5A5

Parsed

Address Number	***Street Name***	***Street Type***	***Street Direction***	***Muni***	***Prov***	***Postal Code***
625	Cochrane	Dr			ON	**L3R**9R9
20	Water	St	N		ON	**N2H** 5A5

Intersection

Street	***Muni***	***Prov***	***Postal Code***
YONGE ST && YORKVILLE AVE		ON	**M4W**

Defining The Geocoding Path *(cont'd)*

Address Geocoder and Scrubber functionality

GeoPinpoint Suite offers a Scrubber function (under license) which examines the data and outputs the geocoded results (x, y, rcode, prescode) to the input database. In order to receive the best possible Scrubber results when using the Address Geocoder, ensure that there is an input postal code for each record (if available).

Defining The Geocoding Path *(cont'd)*

Point Of Interest (POI) Geocoder

The Point of Interest (POI) Geocoder is able to geocode the following types of data:

- POI data

When geocoding POI data in GeoPinpoint Suite, the data can either be geocoded unparsed or parsed. The POI data value (e.g.: CN Tower) can be stored in the unparsed address column or in the parsed street name column. Refer to the examples given for the POI Geocoder below, for more details.

The Input Specifications section (see below) will also outline how the data should be prepared, before using the POI Geocoder to geocode POI data.

POI Geocoder | Use Whole Name | To POI Point By Municipality and FSA: This function is used to geocode to Point of Interest locations by matching the whole POI name in the geo-reference database. The input municipality and FSA are used as the boundaries for geocoding.

Examples of records which can be geocoded using this function are:

Unparsed

Geoaddress	Muni	Prov	Postal Code
CN Tower	**TORONTO**	ON	**M5V** 2T6

Parsed

Address Number	*Street Name*	*Street Type*	*Street Direction*	*Muni*	*Prov*	*Postal Code*
	CASINO NIAGARA			**NIAGARA FALLS**	ON	**L2G** 3K6

POI Geocoder | Use Whole Name | To POI Point By Municipality: This function is used to geocode to Point of Interest locations by matching the whole POI name in the geo-reference database. The input municipality is used as the boundary for geocoding.

Examples of records which can be geocoded using this function are:

Unparsed

Geoaddress	Muni	Prov	Postal Code
CN Tower	**TORONTO**	ON	

Parsed

Address Number	*Street Name*	*Street Type*	*Street Direction*	*Muni*	*Prov*	*Postal Code*
	CASINO NIAGARA			**NIAGARA FALLS**	ON	

Defining The Geocoding Path *(cont'd)*

POI Geocoder | Use Whole Name | To POI Point By FSA: This function is used to geocode to Point of Interest locations by matching the whole POI name in the geo-reference database. The FSA of the postal code is by GeoPinpoint Suite to select the appropriate FSA boundary for geocoding.

Examples of records which can be geocoded using this function are:

Unparsed

Geoaddress	Muni	Prov	Postal Code
CN Tower		ON	**<u>M5V</u>** 2T6

Parsed

Address Number	*Street Name*	*Street Type*	*Street Direction*	*Muni*	*Prov*	*Postal Code*
	CASINO NIAGARA				ON	**<u>L2G</u>** 3K6

Note: GeoPinpoint Suite currently only stores aerodrome (airport) aliases for Whole Alias / Partial Alias functions.

POI Geocoder | Use Whole Alias | To POI Point By Municipality and FSA: This function is used to geocode to Point of Interest locations by matching the whole POI alias in the geo-reference database. The input municipality and FSA value are used as the boundaries for geocoding.

The example we will use for 'Use Whole Alias' will be **Guelph Airport** where the alias is known as **Guelph Air Park**.

Unparsed

Geoaddress	Muni	Prov	Postal Code
Guelph Air Park	**<u>Guelph</u>**	ON	**<u>N1H</u>**6H8

Parsed

Address Number	Street Name	Street Type	Street Direction	Muni	Prov	Postal Code
	Guelph Air Park			**<u>Guelph</u>**	ON	**<u>N1H</u>**6H8

Defining The Geocoding Path *(cont'd)*

POI Geocoder | Use Whole Alias | To POI Point By Municipality: This function is used to geocode to Point of Interest locations by matching the whole POI alias in the geo-reference database. The input municipality is used as the boundary for geocoding.

The example we will use for 'Use Whole Alias' will be **Guelph Airport** where the alias is known as **Guelph Air Park**.

Unparsed

Geoaddress	Muni	Prov	Postal Code
Guelph Air Park	**Guelph**	ON	

Parsed

Address Number	Street Name	Street Type	Street Direction	Muni	Prov	Postal Code
	Guelph Air Park			**Guelph**	ON	

POI Geocoder | Use Whole Alias | To POI Point By FSA: This function is used to geocode to Point of Interest locations by matching the whole POI alias in the geo-reference database. The FSA of the postal code is by GeoPinpoint Suite to select the appropriate FSA boundary for geocoding.

The example we will use for 'Use Whole Alias' will be **Guelph Airport** where the alias is known as **Guelph Air Park**.

Unparsed

Geoaddress	Muni	Prov	Postal Code
Guelph Air Park		ON	**N1H**6H8

Parsed

Address Number	Street Name	Street Type	Street Direction	Muni	Prov	Postal Code
	Guelph Air Park				ON	**N1H**6H8

POI Geocoder | Use Partial Name | To POI Point By Municipality and FSA: This function is used to geocode to Point of Interest locations by checking to see if the input POI name matches partial POI name in the geo-reference database. The input municipality and FSA value are used as the boundaries for geocoding.

The example we will use for 'Use Partial Name' will be UNIVERSITY OF WESTERN ONTARIO - BRESCIA COLLEGE where the Partial Name value is Brescia College.

Unparsed

Geoaddress	Muni	Prov	Postal Code
BRESCIA COLLEGE	**London**	ON	**N6G**1H2

Parsed

Address Number	Street Name	Street Type	Street Direction	Muni	Prov	Postal Code
	BRESCIA COLLEGE			**London**	ON	**N6G**1H2

Defining The Geocoding Path *(cont'd)*

POI Geocoder | Use Partial Name | To POI Point By Municipality: This function is used to geocode to Point of Interest locations by checking to see if the input POI name matches partial POI name in the geo-reference database. The input municipality is used as the boundary for geocoding.

The example we will use for 'Use Partial Name' will be UNIVERSITY OF WESTERN ONTARIO - BRESCIA COLLEGE where the Partial Name value is Brescia College.

Unparsed

Geoaddress	Muni	Prov	Postal Code
BRESCIA COLLEGE	**London**	ON	

Parsed

Address Number	Street Name	Street Type	Street Direction	Muni	Prov	Postal Code
	BRESCIA COLLEGE			**London**	ON	

POI Geocoder | Use Partial Name | To POI Point By FSA: This function is used to geocode to Point of Interest locations by checking to see if the input POI name matches partial POI name in the geo-reference database. The FSA of the postal code is by GeoPinpoint Suite to select the appropriate FSA boundary for geocoding.

The example we will use for 'Use Partial Name' will be UNIVERSITY OF WESTERN ONTARIO - BRESCIA COLLEGE where the Partial Name value is Brescia College.

Unparsed

Geoaddress	Muni	Prov	Postal Code
BRESCIA COLLEGE			**N6G**1H2

Parsed

Address Number	Street Name	Street Type	Street Direction	Muni	Prov	Postal Code
	BRESCIA COLLEGE				ON	**N6G**1H2

POI Geocoder | Use Partial Alias | To POI Point By Municipality and FSA: This function is used to geocode to Point of Interest locations by checking to see if the input POI name matches partial POI alias in the geo-reference database. The input municipality and FSA value are used as the boundaries for geocoding.

The example we will use for 'Use Partial Alias' will be LONDON INTERNATIONAL AIRPORT where the Partial Name alias is LONDON INTERNATIONAL.

Unparsed

Geoaddress	Muni	Prov	Postal Code
LONDON INTERNATIONAL	**LONDON**	ON	**N5V**1A1

Parsed

Address Number	Street Name	Street Type	Street Direction	Muni	Prov	Postal Code
	LONDON INTERNATIONAL			**LONDON**	ON	**N5V**1A1

Defining The Geocoding Path *(cont'd)*

POI Geocoder | Use Partial Alias | To POI Point By Municipality: This function is used to geocode to Point of Interest locations by checking to see if the input POI name matches partial POI alias in the geo-reference database. The input municipality is used as the boundary for geocoding.

The example we will use for 'Use Partial Alias' will be LONDON INTERNATIONAL AIRPORT where the Partial Name alias is LONDON INTERNATIONAL.

Unparsed

Geoaddress	Muni	Prov	Postal Code
LONDON INTERNATIONAL	**LONDON**	ON	

Parsed

Address Number	Street Name	Street Type	Street Direction	Muni	Prov	Postal Code
	LONDON INTERNATIONAL			**LONDON**	ON	

POI Geocoder | Use Partial Alias | To POI Point By FSA: This function is used to geocode to Point of Interest locations by checking to see if the input POI name matches partial POI alias in the geo-reference database. The FSA of the postal code is by GeoPinpoint Suite to select the appropriate FSA boundary for geocoding.

The example we will use for 'Use Partial Alias' will be LONDON INTERNATIONAL AIRPORT where the Partial Name alias is LONDON INTERNATIONAL.

Unparsed

Geoaddress	Muni	Prov	Postal Code
LONDON INTERNATIONAL		ON	**N5V**1A1

Parsed

Address Number	Street Name	Street Type	Street Direction	Muni	Prov	Postal Code
	LONDON INTERNATIONAL				ON	**N5V**1A1

POI Geocoder | Use POI Type | To POI Point By Municipality and FSA: This function is used to geocode to Point of Interest locations by matching the POI type in the geo-reference database. The input municipality name and FSA are used as the boundaries for geocoding.

Example

NAME	POI Type
Guelph Airport	CNC4

Unparsed

Geoaddress	Muni	Prov	Postal Code
CNC4	**Guelph**	ON	**N1H**6H8

Parsed

Address Number	Street Name	Street Type	Street Direction	Muni	Prov	Postal Code
	CNC4			**Guelph**	ON	**N1H**6H8

Defining The Geocoding Path *(cont'd)*

POI Geocoder | Use POI Type | To POI Point By Municipality: This function is used to geocode to Point of Interest locations by matching the POI type in the geo-reference database. The input municipality name is used as the boundary for geocoding.

Example

NAME	POI Type
Guelph Airport	CNC4

Unparsed

Geoaddress	Muni	Prov	Postal Code
CNC4	**Guelph**	ON	

Parsed

Address Number	Street Name	Street Type	Street Direction	Muni	Prov	Postal Code
	CNC4			**Guelph**	ON	

POI Geocoder | Use POI Type | To POI Point By FSA: This function is used to geocode to Point of Interest locations by matching the POI type in the geo-reference database. The FSA of the postal code is by GeoPinpoint Suite to select the appropriate FSA boundary for geocoding.

Example:

NAME	POI Type
Guelph Airport	CNC4

Unparsed

Geoaddress	Muni	Prov	Postal Code
CNC4		ON	**N1H**6H8

Parsed

Address Number	Street Name	Street Type	Street Direction	Muni	Prov	Postal Code
	CNC4				ON	**N1H**6H8

Note: For more examples of POI Type values – refer to Appendix 4: Points of Interest Layers

POI Geocoder and Soundex functionality

GeoPinpoint Suite offers a Soundex function under license to help match POI data that maybe dirty or missing information and geocode these records. The Soundex function once turned on will work by converting the input POI name to a Soundex key and comparing this value to POI data Soundex keys which are stored in the geo-reference database.

Defining The Geocoding Path *(cont'd)*

Postal Code Geocoder

The Postal Code Geocoder is able to geocode the following types of data:

- ❖ Postal Code data

Hint! GeoPinpoint Suite can geocode postal code data with or without provincial information (e.g.: ON for Ontario)

Postal Code Geocoder | Use Postal Code | To Postal Code Point: This function is used to geocode to a postal code point using the input municipality name as the boundary. This function is based on postal code data, therefore a postal code field is required as part of the input information.

Examples of records which can be geocoded using this function are:

Unparsed

Geoaddress	Muni	Prov	Postal Code
			M5V3C9
		ON	**M5V3C9**
625 Cochrane Dr	Markham	ON	**L3R9R9**
625 Cochrane Dr		ON	**L3R9R9**

Parsed

Address Number	*Street Name*	*Street Type*	*Street Direction*	*Muni*	*Prov*	*Postal Code*
						M5V3C9
					ON	**M5V3C9**
625	Cochrane	Dr		Markham	ON	**L3R9R9**
625	Cochrane	Dr			ON	**L3R9R9**

Defining The Geocoding Path *(cont'd)*

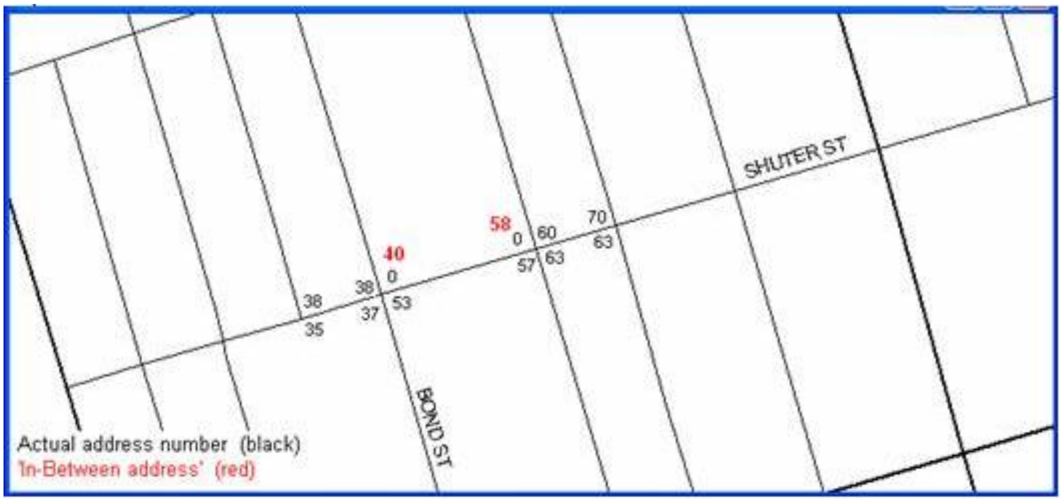
Segment Geocoder

The Segment Geocoder is able to geocode the following types of data:

- **Address data**
- **Zero-address streets**

Segment Centroid functions

The **Segment Geocoder | Use Address | To In-Between Addresses** function should be used when the user wants to geocode to an unaddressed address range on a CanMap segment, that is also located between segments with known address ranges



'In-Between Address' geocoding allows GeoPinpoint users to geocode to potential addresses even before they are added to CanMap streetfiles. This function provides users with high-precision infill geocoding, which may or may not exist in the real world. The In-Between function takes priority over the closest address and opposite-side of the street functions.

The **Segment Geocoder | Use Address | To In-Between Addresses | By Municipality and FSA** should be used when the user wants to geocode to an 'In-Between Address' using the input municipality name and FSA as the boundaries for geocoding.

The **Segment Geocoder | Use Address | To In-Between Addresses | By Municipality** should be used when the user wants to geocode to an 'In-Between Address' using the input municipality name as the boundary for geocoding.

The **Segment Geocoder | Use Address | To In-Between Addresses | By FSA** should be used when the user wants to geocode to an 'In-Between Address' using the input FSA as the boundary for geocoding.

The **Segment Geocoder | Use Address | To Segment Centroid** function should be used when the user wants to geocode to the centroid point of the street block that contains the address number.

Defining The Geocoding Path *(cont'd)*

Segment Geocoder | Use Address | To Segment Centroid | By Municipality and FSA: This function is used to geocode to the centre point of a street segment using the input municipality name and FSA as the boundaries for geocoding.

Examples of records which can be geocoded using this function are:

Unparsed

Geoaddress	Muni	Prov	Postal Code
9 GRENADIER DR	**HAMILTON**	Ontario	**L8T**4C7

Parsed

Address Number	Street Name	Street Type	Muni	Prov	Postal Code
9	GRENADIER	DR	**HAMILTON**	Ontario	**L8T**4C7

Segment Geocoder | Use Address | To Segment Centroid | By Municipality: This function is used to geocode to the centre point of a street segment using the input municipality name as the boundary for geocoding.

Examples of records which can be geocoded using this function are:

Unparsed

Geoaddress	Muni	Prov	Postal Code
9 GRENADIER DR	**HAMILTON**	Ontario	

Parsed

Address Number	Street Name	Street Type	Muni	Prov	Postal Code
9	GRENADIER	DR	**HAMILTON**	Ontario	

Segment Geocoder | Use Address | To Segment Centroid | By FSA: This function is used to geocode to the centre point of a street segment using the FSA portion of the input postal code as the boundary for geocoding.

Examples of records which can be geocoded using this function are:

Unparsed

Geoaddress	Muni	Prov	Postal Code
9 GRENADIER DR		Ontario	**L8T**4C7

Parsed

Address Number	Street Name	Street Type	Muni	Prov	Postal Code
9	GRENADIER	DR		Ontario	**L8T**4C7

Defining The Geocoding Path *(cont'd)*

Street Segments functions

The **Segment Geocoder | Use Address | To Street Segments** function should be used an address string was unable to be geocoded using the address geocoder. One possible reason for this could be a missing street number value.

To demonstrate this – we can use the example of the record: Bay St. If Bay St has 8 segments with different address ranges associated with each respectively, the Street Segments function would return the segment centroid of the first segment found in the geo-reference database.

Segment Geocoder | Use Address | To Street Segments | By Municipality and FSA: This function is used to geocode to a series of street segments using the input municipality name and FSA as the boundaries for geocoding. For window's desktop version, only the first segment is returned as the result. The API version is able to let the user retrieve all of the segments GeoPinpoint Suite geocoded to.

Examples of records which can be geocoded using this function are:

Unparsed

Geoaddress	Muni	Prov	Postal Code
GRENADIER DR	**HAMILTON**	Ontario	**L8T**4C7

Parsed

Address Number	Street Name	Street Type	Muni	Prov	Postal Code
	GRENADIER	DR	**HAMILTON**	Ontario	**L8T**4C7

Segment Geocoder | Use Address | To Street Segments | By Municipality: This function is used to geocode to a series of street segments using the input municipality name as the boundary for geocoding. For window's desktop version, only the first segment is returned as the result. The API version is able to let the user retrieve all of the segments GeoPinpoint Suite geocoded to.

Examples of records which can be geocoded using this function are:

Unparsed

Geoaddress	Muni	Prov	Postal Code
GRENADIER DR	**HAMILTON**	Ontario	

Parsed

Address Number	Street Name	Street Type	Muni	Prov	Postal Code
	GRENADIER	DR	**HAMILTON**	Ontario	

Defining The Geocoding Path *(cont'd)*

Segment Geocoder | Use Address | To Street Segments | By FSA: This function is used to geocode a series of street segments using the FSA portion of the input postal code as the boundary for geocoding. For window's desktop version, only the first segment is returned as the result. The API version is able to let the user retrieve all of the segments GeoPinpoint Suite geocoded to.

Examples of records which can be geocoded using this function are:

Unparsed

Geoaddress	Muni	Prov	Postal Code
GRENADIER DR		Ontario	**L8T**4C7

Parsed

Address Number	Street Name	Street Type	Muni	Prov	Postal Code
	GRENADIER	DR		Ontario	**L8T**4C7

Zero Street Segments functions

The **Segment Geocoder | Use Address | To Zero Street Segments** function should be used to geocode to streets which do not contain any addressing on their segments. Therefore, the entire street and its associated segments all have address ranges of zero (0). This function is useful for geocoding to those new segments found in new subdivisions but have not yet been updated with addressing.

NOTE: The coordinate will be placed to the centroid of the zero-addressed segment

Segment Geocoder | Use Address | To Zero Street Segments | By Municipality and FSA: This function is used to geocode to a series of street segments using the input municipality name and FSA as the boundaries for geocoding. For window's desktop version, only the first segment is returned as the result. The API version is able to let the user retrieve all of the segments GeoPinpoint Suite geocoded to.

Segment Geocoder | Use Address | To Zero Street Segments | By Municipality: This function is used to geocode to a series of street segments using the input municipality name as the boundary for geocoding. For window's desktop version, only the first segment is returned as the result. The API version is able to let the user retrieve all of the segments GeoPinpoint Suite geocoded to.

Segment Geocoder | Use Address | To Zero Street Segments | By FSA: This function is used to geocode a series of street segments using the FSA portion of the input postal code as the boundary for geocoding. For window's desktop version, only the first segment is returned as the result. The API version is able to let the user retrieve all of the segments GeoPinpoint Suite geocoded to.

If the user selects the geocoding path, Segment Geocoder >> Use Address >> To Zero Street Segments, they will be prompted to add a column to their database called 'TotalDistZeroStSegs(m)', after the start button is selected in the geocode tab. If a record geocodes to an unaddressed street (and its associated street segments), the field will be populated with the total distance of all the unaddressed segments for that street name; not to be confused with the individual segment length for the particular segment that it gets geocoded to

Defining The Geocoding Path *(cont'd)*

Boundary Geocoder

The Boundary Geocoder is able to geocode the following types of data:

- FSA data
- PPN data
- Municipal centroid data

Boundary | Use FSA | To FSA Centroid: This function is used to geocode to the centroid of the FSA area using the FSA portion of the input postal code as the boundary. This function is based on postal code data, therefore a postal code field (FSA LDU) is required as part of the input information.

Examples of records which can be geocoded using this function are:

Unparsed

Geoaddress	Muni	Prov	Postal Code
		ON	**M5V**3C9
625 Cochrane Dr	Markham	ON	**L3R**9R9
625 Cochrane Dr		ON	**L3R**9R9

Parsed

Address Number	Street Name	Street Type	Muni	Prov	Postal Code
				Ontario	**L8T**4C7
625	Cochrane	Dr	Markham	ON	**L3R**9R9
625	Cochrane	Dr		ON	**L3R**9R9

Boundary | Use Municipality | To PPN Points: This function is used to geocode to a Populated Place Name (PPN) point using the input municipality name.

Examples of records which can be geocoded using this function are:

Unparsed

Geoaddress	Muni	Prov	Postal Code
	Bobcaygeon	ON	
	Flamborough	ON	

Parsed

Address Number	Street Name	Street Type	Muni	Prov	Postal Code
			Bobcaygeon	ON	
			Flamborough	ON	

Note: Province information has to be provided with each record otherwise the record will not geocode to PPN point. FSA information should be provided to help the software distinguish between same name PPN values (e.g.: Mount Pleasant which occurs seven times in Ontario).

Defining The Geocoding Path *(cont'd)*

Boundary | Use Municipality | To Municipal Centroid: This function is used to geocode to a municipal centroid point using the input municipality name.

Examples of records which can be geocoded using this function are:

Unparsed

Geoaddress	Muni	Prov	Postal Code
	Toronto	ON	
	Vancouver	BC	

Parsed

Address Number	Street Name	Street Type	Muni	Prov	Postal Code
			Toronto	ON	
			Vancouver	BC	

Note: Province information has to be provided with each record otherwise the record will not geocode to Municipal centroid. FSA information should be provided to help the software distinguish between same name muni values (e.g.: Hamilton which occurs twice in Ontario – once as a city and as a township).

Sample Application Program Interface

Every effort has been made to maintain backward compatibility with the previous Version 3.x GeoPinpoint API. If you are a user of the v3.x API, a recompilation may be all that is required to migrate to the new GeoPinpoint Suite API v5.x.

However, in order to take full advantage of the new API version and benefit from the new features, we highly recommend that our clients modify their code to use the new API functions instead of keep using the API functions from Version 3.x. Some of the API functions are contradictory between the two versions and therefore cannot be used interchangeably as detailed in their descriptions seen below.

Programming Steps

A sample API to perform the geocoding is described below.

Step 1: Declare a Geocoder Class Instance

In C++: #include “Geocoder.h”
CGeoCoder ComGeoCoder;

In VB and other ActiveX compliant languages, make reference to “dmtiGeocoder 4.0 Type Library” and instantiate a Geocoder object.

Step 2: Initialize the Geocoder

Geocoder.Initialize(char* georefPath, int ProcessMode, int Offset)

The geocoder must be initialized before use. The initialization function is responsible for the following:

1. Sets the path of the Georef directory;
2. Set the processing mode (0 for INTERACTIVE mode, 1 for BATCH mode). This parameter is now obsolete and has been kept for backward compatibility only.
3. Sets the default Offset value (The distance that the point needs to be offset from the street centreline).

Default values are assumed if Initialize() is not called. (geoRefPath is set to the current working directory, ProcessMode is set to INTERACTIVE, and Offset is set to 10).

Step 3: Set Input Street Information

Geocoder.setInput(char* streetName, char* MuniName, char* Prov, char* PostalCode)

This function sets the input street information. NULL values and empty string (“”) are allowed.

Input an unparsed address string into **streetName.**

**Geocoder .setParsedInput(char* streetNum, char* streetPrefix,
char* streetName, char* streetType, char* streetDir,
char* suite, char* MuniName, char* Prov, char* PostalCode);**

This function sets the parsed input street information. NULL values and empty strings (“”) are allowed.

Sample Application Program Interface *(cont'd)*

Step 4: Set Processing Options

Geocoder.setRelaxOnType()

Enables relaxed searching for street types. Please see Glossary document for relax definition.

Geocoder.setRelaxOnDir()

Enables relaxed searching for street directions.

Geocoder.setIntersectionDelimiter()

This function sets a string value to be used in distinguishing the street intersection input. By Default, its value is "&&".

Step 5: Define Geocoding Path

Geocoding path defines how the geocoders should work together to perform the requested task.

Use the following function to turn on each intended Geocoding choice,

Geocoder.setGeocodingChoiceOn(int msgCode)

And use the following function to turn off the Geocoding choice:

Geocoder.setGeocodingChoiceOff(int msgCode)

Both functions should be called after the Initialize() function has been called as in Step 2.

The second method does not need to be used, if the geocoding choice will not change during each geocoding session. If the above operations are not performed, then the geocoding process will perform by using the default geocoding path defined internally. The current default geocoding path attempts to geocode to Address points interpolated from street segments.

The values of the msgCode are defined in the Geocoder.h header file. These values determine which geocoder will be used and ultimately tasks it can perform.

Step 6: Perform Geocoding

Geocoder.Geocoding(int ProcessingFlag)

This is the main function to perform the geocoding task. In normal caseS, pass ProcessingFlag = 0 to this function.

ProcessingFlag is obsolete, but kept for compatibility purpose. It is highly recommended to pass 0 for this argument and use Step 3 to control how the Geocoding process should be performed. This flag still has its original meanings with previous version in order to keep the compatibility but any definition in geocoding path choices in Step 3 will be combined with these options if they are used:

Sample Application Program Interface *(cont'd)*

Step 7: Get Results

```
Geocoder.getOutput(&addressID, stdStreetName, StdMuniName, &Lon, &Lat,
&InterpolationCode, &stSegID, streetWholeName, &streetNum,  streetPrefix,
streetName, streetType, streetDir,  suite,  MuniName, Prov, PostalCode,
&resultCode, RangeNumber);
```

This function needs to be called to retrieve the geocoding results. The result code will indicate which level of geocoding was performed. Call this function successively in order to retrieve all of the geocoded results. A return value of -1 indicates failure, and values of zero or greater indicate how many results are still yet to be retrieved

```
Geocoder.setResultPosAtStart()
```

Use this function to set the current result to be the first of the matches, and the calling of the getOutput() function can start from the beginning of the geocoding matches.

GeoPinpoint Suite ActiveX Sample Note

The ActiveX Sample provided by DMTI Spatial is an application containing the following functionality found in the GeoPinpoint Suite ActiveX API.

Geocoding functionality and options not included in this ActiveX Sample can be found by referencing certain sections in the GeoPinpoint Suite API manual.

1. **Geocode an address**

   Example: Address: 20 Bay St
   City: Toronto
   Province: on

   Press the Geocoding button - This will return the latitude/longitude of this address point.

2. **Geocode an intersection**

   Example: Address: Yonge St && Finch Ave East
   City: North York
   Province: on

   Press the Geocoding button - This will return the latitude/longitude of the intersection point.

   Press the Next button - 2 results returned

3. **Geocode a postal code**

   Example: Postal Code: M5V3C9
   Select *Fallback to Postal* checkbox

   Press the Geocoding button - This will return the latitude/longitude of the postal code point.

4. **Determine range for street segment**

   Example: Address: Bay St
   City: Toronto
   Province: on
   Select *Search Segment* checkbox

   Press the Geocoding button - This will return the latitude/longitude of the street segment centroid.

   The range number is also returned for the first street segment for Bay St.

   Press the Next button to get the rest of the ranges for Bay St.

GeoPinpoint Suite ActiveX Sample Note *(cont'd)*

The GeoPinpoint Suite ActiveX sample also **contains relax** options**:**

```
'Set Falling back to Postal Code
If ChkSearchPostalCode.Value = 1 Then
    ret = objGeocoder.setFallbackPostalCode()
End If
```

Other relax options can be used in a similar way

```
'ret = objGeocoder.setRelaxOnDir()
'ret = objGeocoder.setRelaxOnType()
```

If you have a GeoPinpoint license for Soundex/Scrubber ActiveX, the following code has been added to the sample that accompanies the software

```
'Set Search POI Name by Soundex
If chkSoundex.Value = 1 Then
        ret = objGeocoder.setSearchSoundex()
        'For POI Geocoder
        processFlag = 4
End If
```

Geocoder.h Header File Values

This table gives definitions for those macros that are located in the georef.h file that is utilize by GeoPinpoint Suite API.

	Definition
#define AD_TO_SEGPNT_BY_MUNI_AND_FSA 39130	This function is used to geocode to an address point using the input municipality name and FSA as the boundaries for geocoding.
#define AD_TO_PNT_BY_MUNI 39125	Not yet implemented.
#define AD_TO_SEGPNT_BY_MUNI 39120	This function is used to geocode to an address point using the input municipality name as the boundary for geocoding.
#define AD_TO_PNT_BY_FSA 39115	Not yet implemented.
#define AD_TO_SEGPNT_BY_FSA 39110	This function is used to geocode to an address point using the FSA portion of the input postal code as the boundary for geocoding.
#define POI_BY_WHOLENAME_BY_MUNI_AND_FSA 37530	This function is used to geocode to Point of Interest locations by matching the whole POI name in the geo-reference database. The input municipality and FSA are used as the boundaries for geocoding.
#define POI_BY_WHOLENAME_BY_MUNI 37520	This function is used to geocode to Point of Interest locations by matching the whole POI name in the geo-reference database. The input municipality is used as the boundary for geocoding.
#define POI_BY_WHOLENAME_BY_FSA 37510	This function is used to geocode to Point of Interest locations by matching the whole POI name in the geo-reference database. The FSA portion of the input postal code is used as the boundary for geocoding.
#define POI_BY_WHOLEALIAS_BY_MUNI_AND_FSA 37430	This function is used to geocode to Point of Interest locations by matching the whole POI alias in the geo-reference database. The input municipality and FSA are used as the boundaries for geocoding. *Note: Sometimes a POI can be identified by a commonly used alias.*
#define POI_BY_WHOLEALIAS_BY_MUNI 37420	This function is used to geocode to Point of Interest locations by matching the whole POI alias in the geo-reference database. The input municipality is used as the boundary for geocoding. *Note: Sometimes a POI can be identified by a commonly used alias.*

Geocoder.h Header File Values *(cont’d)*

#define POI_BY_WHOLEALIAS_BY_FSA 37410	This function is used to geocode to Point of Interest locations by matching the whole POI alias in the geo-reference database. The FSA portion of the input postal code is used as the boundary for geocoding. *Note: Sometimes a POI can be identified by a commonly used alias.*
#define POI_BY_PARTIALNAME_BY_MUNI_AND_FSA 37330	This function is used to geocode to Point of Interest locations by checking to see if the input POI name matches partial POI name in the geo-reference database. The input municipality and FSA are used as the boundaries for geocoding. *Note: If a POI has several words in its name, you can search for it by inputting only part of its name.*
#define POI_BY_PARTIALNAME_BY_MUNI 37320	This function is used to geocode to Point of Interest locations by checking to see if the input POI name matches partial POI name in the geo-reference database. The input municipality is used as the boundary for geocoding. *Note: If a POI has several words in its name, you can search for it by inputting only part of its name.*
#define POI_BY_PARTIALNAME_BY_FSA 37310	This function is used to geocode to Point of Interest locations by checking to see if the input POI name matches partial POI name in the geo-reference database. The FSA portion of the input postal code is used as the boundary for geocoding. *Note: If a POI has several words in its name, you can search for it by inputting only part of its name.*
#define POI_BY_PARTIALALIAS_BY_MUNI_AND_FSA 37230	This function is used to geocode to Point of Interest locations by checking to see if the input POI name matches partial POI alias in the geo-reference database. The input municipality and FSA are used as the boundaries for geocoding. *Note: If a POI has several words in its alias, you can search for it by inputting only part of its alias.*
#define POI_BY_PARTIALALIAS_BY_MUNI 37220	This function is used to geocode to Point of Interest locations by checking to see if the input POI name matches partial POI alias in the geo-reference database. The input municipality is used as the boundary for geocoding. *Note: If a POI has several words in its alias, you can search for it by inputting only part of its alias.*
#define POI_BY_PARTIALALIAS_BY_FSA 37210	This function is used to geocode to Point of Interest locations by checking to see if the input POI name matches partial POI alias in the geo-reference database. The FSA portion of the input postal code is used as the boundary for geocoding. *Note: If a POI has several words in its alias, you can search for it by inputting only part of its alias.*

Geocoder.h Header File Values *(cont'd)*

#define POI_BY_POICODE_BY_MUNI_AND_FSA 37130	This function is used to geocode to Point of Interest locations by matching the POI type in the geo-reference database. The input municipality name and FSA are used as the boundaries for geocoding.
#define POI_BY_POICODE_BY_MUNI 37120	This function is used to geocode to Point of Interest locations by matching the POI type in the geo-reference database. The input municipality name is used as the boundary for geocoding.
#define POI_BY_POICODE_BY_FSA 37110	This function is used to geocode to Point of Interest locations by matching the POI type in the geo-reference database. The FSA portion of the input postal code is used as the boundary for geocoding.
#define PCODE_PC_TO_PCPNT 35230	This function is used to geocode to a postal code point using the input municipality name as the boundary. This function is based on postal code data, therefore a postal code field is required as part of the input information.
#define PCODE_PC_TO_SEGS 35220	Not yet implemented.
#define PCODE_PC_TO_MUNINAME 35215	Not yet implemented.
#define PCODE_PC_TO_MUNIID 35210	Not yet implemented.
#define SEGMENT_AD_TO_SEGS_INB_BY_MUNI_AND_FSA 34530	This function is used to geocode to a street segment where the address range is not known, but is inferred from the neighbouring addressed segments, while using municipality name and FSA as the boundaries for geocoding.
#define SEGMENT_AD_TO_SEGS_INB_BY_MUNI 34520	This function is used to geocode to a street segment where the address range is not known, but is inferred from the neighbouring addressed segments, while using municipality name as the boundary for geocoding.
#define SEGMENT_AD_TO_SEGS_INB_BY_FSA 34510	This function is used to geocode to a street segment where the address range is not known, but is inferred from the neighbouring addressed segments, while using FSA as the boundary for geocoding.
#define SEGMENT_AD_TO_SEGCENTER_BY_MUNI_AND_FSA 33340	This function is used to geocode to the centre point of a street segment using the input municipality name and FSA as the boundaries for geocoding.
#define SEGMENT_AD_TO_SEGCENTER_BY_MUNI 33335	This function is used to geocode to the centre point of a street segment using the input municipality name as the boundary for geocoding.
#define SEGMENT_AD_TO_SEGCENTER_BY_FSA 33330	This function is used to geocode to the centre point of a street segment using the FSA portion of the input postal code as the boundary for geocoding.
#define SEGMENT_AD_TO_PCODE_BY_MUNI 33325	Not yet implemented.

Geocoder.h Header File Values *(cont'd)*

#define SEGMENT_AD_TO_PCODE_BY_FSA 33320	Not yet implemented.
#define SEGMENT_AD_TO_SEGS_BY_MUNI_AND_FSA 33318	This function is used to geocode to a series of street segments using the input municipality name as the boundary for geocoding. For window's desktop version, only the first segment is returned as the result.
#define SEGMENT_AD_TO_SEGS_BY_MUNI 33315	This function is used to geocode to a series of street segments using the input municipality name as the boundary for geocoding. For window's desktop version, only the first segment is returned as the result.
#define SEGMENT_AD_TO_SEGS_BY_FSA 33310	This function is used to geocode a series of street segments using the FSA portion of the input postal code as the boundary for geocoding. For window's desktop version, only the first segment is returned as the result.
#define SEGMENT_AD_TO_SEGS_ZERO_BY_MUNI_AND_FSA 33308	This function is used to geocode to the centroid of a street which contains segments which are all completely zero addressed (i.e.: address range is 0 to 0). The muni and FSA boundaries are used for geocoding.
#define SEGMENT_AD_TO_SEGS_ZERO_BY_MUNI 33305	This function is used to geocode to the centroid of a street which contains segments which are all completely zero addressed (i.e.: address range is 0 to 0). The municipality boundary is used for geocoding.
#define SEGMENT_AD_TO_SEGS_ZERO_BY_FSA 33300	This function is used to geocode to the centroid of a street which contains segments which are all completely zero addressed (i.e.: address range is 0 to 0). The FSA boundary is used for geocoding.
#define BNDRY_FSA_TO_FSACENTER 31130	This function is used to geocode to the centroid of the FSA area using the FSA portion of the input postal code as the boundary.
#define BNDRY_MUNI_TO_PPN 31125	This function is used to geocode to a Populated Place Name (PPN) point using the input municipality name. FSA information should be provided to help the software distinguish between same name PPN values (e.g., Mount Pleasant which occurs seven times in Ontario).
#define BNDRY_MUNI_TO_MUNICENTER 31120	This function is used to geocode to a municipal centroid point using the input municipality name. FSA information should be provided to help the software distinguish between same name muni values (e.g., Hamilton which occurs twice in Ontario – once as a city and as a township).

Geocoder.h Header File Values *(cont'd)*

Note: The 'Segment_AD_To_Segs_Zero' definitions have the ability to output the total meter distance of zero addressed street segments for that particular street based on the geocoding area. The distance value associated with these segments will be placed into the output parameter 'RangeNumber'. This value is expressed in meters (m).

If the functions listed as '**Not yet implemented'** are used the software will use a default geocoding function instead.

Example: If using the msgcode #define AD_TO_PNT_BY_MUNI 39125,
GPP Suite will instead use the function #define AD_TO_SEGPNT_BY_MUNI 39120.

These are the default geocoding functions for each geocoder:

Geocoder	Function
Address geocoder	AD_TO_SEGPNT_BY_MUNI V5.x: 39120
POI geocoder	POI_BY_WHOLENAME_BY_MUNI V5.x: 37520
Postal Code geocoder	PCODE_PC_TO_PCPNT V5.x: 35230
Segment geocoder	SEGMENT_AD_TO_SEGCENTER_BY_MUNI V5.x: 33335

API Functions

GeoPinpoint Suite Version 5.x uses the following API functions to perform geocoding.

1. Object Creation and Initialization Functions
2. Address Input Functions
3. Geocoding Path Definition functions
4. Option Functions
5. Main Geocoding Functions
6. Output Functions

A list of all the functions with the purpose, the syntax, some remarks, and an example are detailed in the next five sections.

Note: Obsolete functions from Version 3 are also listed for backward compatibility reason.

If you are using any of the following API functions from the GeoPinpoint Suite library:

```
Geocoder.setGeocodingChoiceOn(int msgCode)
Geocoder.setGeocodingChoiceOff(int msgCode)
Geocoder.setGeocodingPath(char* pathStr)
```

You should not use obsolete Version 3 API functions at the same time to define the Geocoding options, and vice versa. The mixed usage between the new API functions in GeoPinpoint Suite, and the obsolete functions in version 3.x (which are replaced by these functions) will possibly cause abnormal behavior in the geocoding process.

Important Notes to remember:

- Each instance of GeoPinpoint Suite requires approximately **512K** bytes stack memory. Please make sure that in the custom application, a minimum of such memory should be allocated to each instance of GeoPinpoint. For example, in running GeoPinpoint Suite in a Java environment, the user needs to run "**Java –ss512K**", instead of "Java".

- GeoPinpoint Suite requires intensive file access activities. However, on some operating systems, file handles are restricted to a limited number (e.g.: Sun-Solaris Unix). In our development, we have overcome the hard limit so that such limitation will not be a constraint on how many instances of GeoPinpoint Suite can be created on one machine. However, in the multi-threading environment where GeoPinpoint Suite will have many instances running concurrently, the soft limit still needs to be increased to a safe level. On Sun-Solaris, the command is "ulimit -n #####", say "ulimit -n 3000". This number can be determined by assuming each instance of GeoPinpoint Suite requires roughly 40 file handles, so please take into account other programs running on the same machine that could possibly compete with it.

API Functions *(cont'd)*

Object Creation and Initialization Functions

CGeoCoder, ComGeoCoder

Purpose

Creates a geocoder object.

Syntax

In C++:

```
#include "Geocoder.h"
CGeoCoder  myGeoCoder;
```

In VB:

```
Dim myGeoCoder As DMTIGEOCODERLib.ComGeoCoder
Set myGeocoder = New DMTIGEOCODERLib.ComGeoCoder
```

Remarks

In VB and other ActiveX compliant languages, make reference to "dmtiGeocoder 4.0 Type Library" and instantiate a Geocoder object.

Compatibility: This is identical to Version 3.x.

Geocoder.Initialize(char* georefPath, int ProcessMode, int Offset)

Purpose

Initializes the geocoder object.

Syntax

In C++:

```
myGeocoder.Initialize(georefPath, ProcessMode, Offset);
```

In VB:

```
myGeocoder.Initialize georefPath, ProcessMode, Offset
```

In Java:

```
myGeocoder.Initialize(georefPath, ProcessMode, Offset);
```

Remarks

Return type: Integer,   0: Failure;   1: Success.

Arguments:

georefPath: directory pointing to the Georef top directory;
ProcessMode: 0 for INTERACTIVE mode, 1 for BATCH mode. This parameter is now obsolete and has been kept for backward compatibility only;
Offset: the distance, in metres, by which the address should be offset from the street centreline. This value must be positive and between 0 and 20 metres. If no value is specified – 20 metres will be substituted.

This function must be called prior to calling setGeocodingChoiceOn() or setGeocodingPath()

API Functions *(cont'd)*

Example

myGeocoder.Initialize("d:\\data\\georef", 0, 5)

Compatibility: This is identical to Version 3.x.

Address Input Functions

Geocoder.setInput(char* streetName, char* MuniName, char* Prov, char* PostalCode)

Purpose

Inputs the unparsed address information to GeoPinpoint Suite for geocoding.

Syntax

In C++:

```
myGeocoder.setInput(streetName, MuniName, Prov, PostalCode);
```

In VB:

```
myGeocoder.setInput streetName, MuniName, Prov, PostalCode
```

In Java

```
myGeocoder.setInput(streetName, muniName, prov, postalCode);
```

Remarks

Return type: Integer, 0: Failure; 1: Success.

Arguments:

streetName: The address input;

Note: For POI geocoding – enter the POI name into the streetName parameter

MuniName: The municipal name;

Prov: The province name;

PostalCode: The postal code.

This function also sets the variable ProcessParsedAddress to 0 to indicate that the input address has not been parsed before.

Note: *In the case of un-parsed data, concatenate the pre-direction value with the street address string (e.g.: SOUTH FRONT ST* | *TORONTO* | *ON)*

Example

myGeocoder.setInput("5800 Yonge St", "Toronto", "On", "M2N 3E4");

Compatibility: This is identical to Version 3.x.

API Functions *(cont'd)*

Geocoder.setParsedInput(char* streetNum, char* streetPrefix, char* streetName, char* streetType, char* streetDir, char* suite, char* MuniName, char* Prov, char* PostalCode)

Purpose

Inputs parsed address information to GeoPinpoint Suite for geocoding.

Syntax

In C++:

```
myGeocoder.setParsedInput(streetNum, streetPrefix, streetName, streetType,
streetDir, suite, MuniName, Prov, PostalCode)
```

In VB:

```
myGeocoder.setParsedInput streetNum, streetPrefix, streetName, streetType,
streetDir, suite, MuniName, Prov, PostalCode
```

In Java:

```
myGeocoder.setParsedInput (streetNum, streetPrefix, streetName, streetType,
streetDir, suite, muniName, prov, postalCode);
```

Remarks

Return type: Integer, 0: Failure; 1: Success.

Arguments:

streetNum: The street number;
streetPrefix: The street prefix;
streetName: The street name;
Note: For POI geocoding – enter the POI name into the streetName parameter
streetType: The street type;
streetDir: The street direction;
suite: The suite number;
MuniName: The municipal name;
Prov: The province name;
PostalCode: The postal code.

This function sets the variable ProcessParsedAddress to 1. By default, all the argument values will be null.

If input data contains a parsed out pre-direction field, to handle it at the API coding level for parsed address input - user needs to concatenate the pre-direction value w*ith the street name value and treat both as the parsed street name field*

e.g. SOUTH | FRONT | ST → SOUTH FRONT | ST;
EAST | BEAVER CREEK | DR | WEST → EAST BEAVER CREEK | DR | WEST

Example

```
myGeocoder.setParsedInput("5800", "", "Yonge", "ST", "", "", "Toronto", "ON", "M2N
3E4")
```

Compatibility: This is identical to Version 3.x.

API Functions *(cont'd)*

Geocoding Path Definition Functions

Geocoder.setGeocodingPath(char * pathStr)

Purpose

Sets the geocoding path at one time. The format of the path Str is the numeric message code separated by "|", something like "34120 | 32230 |31120".

Syntax

In C++:

myGeocoder.setGeocodingPath (pathStr);

In VB:

myGeocoder.setGeocodingPath pathStr

In Java:

myGeocoder.setGeocodingPath(pathStr);

Remarks

MsgCodes that can be used to compose this path string manually are defined in Geocoder.h header file.

Note: Even if the msgcodes are entered in the wrong hierarchy order (e.g.: postal code then address) the function will sort the msgcodes internally and still adhere to a hierarchy (address → POI → postal code → etc…)

Arguments:

pathStr: This is the numeric message code.

Example

myGeocoder.setGeocodingPath ("34120 | 32230 | 31120");

Compatibility: Refer to section "Geocoder.h Header File Values" for additional notes on msgcodes (default functions).

Please ensure that the **Geocoder.Initialize** function is called before using this function (Refer to section "Object Creation and Initialization Functions").

API Functions *(cont'd)*

Geocoder.setGeocodingChoiceOn(int msgCode)

Purpose

This function enables geocoding choices. Continuous calling of this function will define the operations to be performed for the geocoding process. Message code definitions are given in Geocoder.h header file.

Syntax

In C++:

```
myGeocoder.setGeocodingChoiceOn (msgCode);
```

In VB:

```
myGeocoder.setGeocodingChoiceOn msgCode
```

In Java:

```
myGeocoder.setGeocodingChoiceOn(msgCode);
```

Remarks

MsgCode values are defined in the Geocoder.h header file. Call setGeocodingChoiceOn() repeatedly to enable desired choices.

Example

```
myGeocoder.setGeocodingChoiceOn (AD_TO_SEGPNT_BY_MUNI);
```

Compatibility:

Please ensure that the **Geocoder.Initialize** function is called before using this function (Refer to section “Geocoder.h Header File Values” for additional notes on msgcodes).

Geocoder.setGeocodingChoiceOff(int msgCode)

Purpose

Sets the geocoding choice off. This is called because the previously defined Geocoding path needs to be modified without destroying the Geocoder instance.

Syntax

In C++: myGeocoder.setGeocodingChoiceOff (msgCode);

In VB: myGeocoder.setGeocodingChoiceOff msgCode

In Java: myGeocoder.setGeocodingChoiceOff(msgCode);

Remarks

MsgCode are defined in Geocoder.h header file.

Example

```
myGeocoder.setGeocodingChoiceOff (AD_TO_PNT_BY_MUNI);
```

Compatibility

Refer to section “Geocoder.h Header File Values” for additional notes on msgcodes

API Functions *(cont'd)*

Geocoder.ClearGeocodingPath()

Purpose

Clear the geocoding path set before. This function can be used when a new geocoding path is intended to be used.

Syntax

```
In C++: myGeocoder.ClearGeocodingPath ();
In VB:   myGeocoder.ClearGeocodingPath

In Java: myGeocoder.ClearGeocodingPath();
```

Remarks

No arguments are required.

Example

```
myGeocoder.ClearGeocodingPath ();
```

Compatibility

API Functions *(cont'd)*

Option Functions

Geocoder.setRelaxOnType()

Purpose

Sets the flag to relax on street type matching so that if the type is missing or wrong, GeoPinpoint Suite can still geocode the address correctly.

Syntax

In C++: myGeocoder.setRelaxOnType();

In VB: myGeocoder.setRelaxOnType

In Java: myGeocoder.setRelaxOnType();

Remarks

Return type: Integer, 0: Failure; 1: Success.

Example

myGeocoder.setRelaxOnType();

Geocoder.setRelaxOnTypeOff()

Purpose

Unsets the relax on street type flag so that the type will not be relaxed after calling this method.

Syntax

In C++: myGeocoder.setRelaxOnTypeOff();

In VB: myGeocoder.setRelaxOnTypeOff

In Java: myGeocoder.setRelaxOnTypeOff();

Remarks

No Arguments.

Example

myGeocoder.setRelaxOnTypeOff();

Geocoder.setRelaxOnDir()

Purpose

Sets the flag to relax on street direction matching so that if the street direction is missing or wrong, GeoPinpoint Suite will attempt to geocode the address correctly.

Syntax

In C++:

myGeocoder.setRelaxOnDir();

In VB:

myGeocoder.setRelaxOnDir

In Java:
myGeocoder.setRelaxOnDir();

Remarks

Return type: Integer, 0: Failure; 1: Success.

Example

myGeocoder.setRelaxOnDir();

Geocoder.setRelaxOnDirOff()

Purpose

Unsets the relax on the street direction flag so that the street direction will not be relaxed after calling this method

Syntax

In C++: myGeocoder.setRelaxOnDirOff();

In VB: myGeocoder.setRelaxOnDirOff

In Java: myGeocoder.setRelaxOnDirOff();

Remarks

No Arguments.

Example

myGeocoder.setRelaxOnDirOff();

Geocoder.setRelaxOnPrefix()

Purpose

Sets the flag to relax on street prefix matching so that if the street prefix is missing or wrong, GeoPinpoint Suite will attempt to geocode the address correctly.

Syntax

In C++: myGeocoder.setRelaxOnPrefix();

In VB: myGeocoder.setRelaxOnPrefix

In Java: myGeocoder.setRelaxOnPrefix();

Remarks

Return type: Integer, 0: Failure; 1: Success.

Example

myGeocoder.setRelaxOnPrefix();

API Functions *(cont'd)*

Geocoder.setRelaxOnPrefixOff()

Purpose

Unsets the relax on street prefix flag so that the prefix will be part of search criteria if not null.

Syntax

In C++: myGeocoder.setRelaxOnPrefixOff();

In VB:myGeocoder.setRelaxOnPrefixOff

In Java: myGeocoder.setRelaxOnPrefixOff();

Remarks

No Arguments.

Example

myGeocoder.setRelaxOnPrefixOff();

Geocoder.setIntersectionDelimiter(char *Delimiter)

Purpose

Sets or changes the delimiter used to distinguish intersections.

Note: There is no actual function for processing intersections rather the user passes an unparsed intersection string to the Street Name parameter of the setInput function and GPP Suite will attempt to geocode this intersection. (e.g.: Finch Ave E && Yonge St). If you have a valid address and an intersection in your address record field, GeoPinpoint will geocode to intersection. For example, if you have "5600 Yonge St && Finch Ave" in your address field, GeoPinpoint will recognize the intersection delimiter first and geocode to intersection.

Syntax

In C++: myGeocoder.setIntersectionDelimiter (Delimiter);

In VB: myGeocoder.setIntersectionDelimiter Delimiter

In Java: myGeocoder.setIntersectionDelimiter (Delimiter);

Remarks

Arguments:

Delimiter: The delimiter is a character string that will be used for distinguishing street intersections. ***By default, its value is "&&".***

Example

myGeocoder.setIntersectionDelimiter ("@");

API Functions *(cont'd)*

Geocoder.setGeocodeByStAlias()

Purpose

Sets the flag to geocode to the street alias if the address level geocoding fails.

This option allows GeoPinpoint Suite to search the geo-reference database for alternative street names for streets that have more than one correct identifying name. For example, a road may be known as Broadway Avenue, and Highway 9. While the name Broadway Avenue may be in the geo-reference database street field, the target database table may have a record with the address on Highway 9. By selecting this option, GeoPinpoint Suite is able to make a match by finding Highway 9 in the geo-reference database Street Alias field.

Note: The Geocode to Street Alias option now includes the previously separate option Geocode
to Former Street Name. When GeoPinpoint Suite now searches the Street Alias field in
the geo-reference database, it will also search the Former Street Name field.

Syntax

In C++: myGeocoder.setGeocodeByStAlias();

In VB: myGeocoder.setGeocodeByStAlias

In Java: myGeocoder.setGeocodeByStAlias();

Remarks

Return type: Integer, 0: Failure; 1: Success.

Example

myGeocoder.setGeocodeByStAlias();

Geocoder.setGeocodeByStAliasOff()

Purpose

Unsets the geocode to the street alias flag so that the search on street alias will no longer be performed after calling this method.

Syntax

In C++: myGeocoder.setGeocodeByStAliasOff();

In VB: myGeocoder.setGeocodeByStAliasOff

In Java: myGeocoder.setGeocodeByStAliasOff();

Remarks

No Arguments.

Example

myGeocoder.setGeocodeByStAliasOff();

API Functions *(cont'd)*

Geocoder.setRefineAddressByPostalCode()

Purpose

Sets the option to refine the Address selection by postal code. This option will be used to apply to the multiple result situation, and refine the final results.

Syntax

In C++: myGeocoder.setRefineAddressByPostalCode();

In VB: myGeocoder.setRefineAddressByPostalCode

In Java: myGeocoder.setRefineByPostalCode();

Remarks

No Argument.

Example

myGeocoder.setRefineAddressByPostalCode();

Geocoder.setRefineAddressByPostalCodeOff()

Purpose

Unsets the refine address level geocoding by using the postal code centroid flag. This function will no longer be performed after calling this method.

Syntax

In C++: myGeocoder.setRefineAddressByPostalCodeOff();

In VB: myGeocoder.setRefineAddressByPostalCodeOff

In Java: myGeocoder.setRefineByPostalCodeOff();

Remarks

No Arguments.

Example

myGeocoder.setRefineAddressByPostalCodeOff();

API Functions *(cont'd)*

Geocoder.GetMuniIDbyPostalCode()

Purpose

Sets the flag to allow the use of the municipal ID (MuniID) found by the postal code. If this option is set, and the address level geocoding fails, it will attempt to geocode the address using the municipal ID found in the postal code table.

Syntax

In C++: myGeocoder.GetMuniIDbyPostalCode();

In VB: myGeocoder.GetMuniIDbyPostalCode

In Java: myGeocoder.setMuniIDByPostalCode();

Remarks

Return type: Integer, 0: Failure; 1: Success.

Example

myGeocoder.GetMuniIDbyPostalCode();

Geocoder.SetMuniIDbyPostalCodeOff()

Purpose

Unsets the use of the municipal ID found by the postal code flag so that this process will no longer be performed after calling this method.

Syntax

In C++: myGeocoder.setMuniIDbyPostalCodeOff();

In VB: myGeocoder.setMuniIDbyPostalCodeOff

In Java: myGeocoder.setMuniIDByPostalCodeOff();

Remarks

No Arguments.

Example

myGeocoder.setMuniIDbyPostalCodeOff();

API Functions *(cont'd)*

Geocoder.setOffset(long offset)

Purpose

Sets the offset value to be used for offsetting the address from the street centerline.

Syntax

In C++: myGeocoder.setOffset(offset);

In VB: myGeocoder.setOffset offset

In Java: myGeocoder.setOffset(offset);

Remarks

Return type: Integer, 0: Failure; 1: Success.

Arguments:
Offset: To specify the distance of the offset for the address from the street centerline. Please note, this function will only accept a positive value (negative values will be made positive)

If the user enters a value which is greater than 50, the offset value will automatically become 50. This is done to achieve a higher precision.

Example

myGeocoder.setOffset(offset);

Geocoder.setInset(long inset)

Purpose

Sets the inset value to be used for offsetting the address from the street centerline.

Syntax

In C++:myGeocoder.setInset(inset);

In VB:myGeocoder.setInset inset

In Java:myGeocoder.setInset(inset);

Remarks

Return type: Integer, 0: Failure; 1: Success.

Arguments:
Inset: To specify the distance of the inset for the address from the nearest street that is perpendicular to the input street name value. Please note, this function will only accept a positive value (negative values will be made positive).

If the user enters a value which is greater than 50, the inset value will automatically become 50. This is done to achieve a higher precision.

Example

myGeocoder.setInset(inset);

API Functions *(cont'd)*

Geocoder.setOptClosestHouseNumDifferenceLimit (int val)

Purpose

Sets the closest address tolerance value to be used for geocoding the closest address (same side of the street) if the exact address number fails.

To return the actual address number that is geocoded, this value is written to the StNum output parameter.

If this option has been set, then it will take precedence over '**setSearchOppositeStreet**'.

Syntax

In C++: myGeocoder.setOptClosestHouseNumDifferenceLimit (val);

In VB: myGeocoder.setOptClosestHouseNumDifferenceLimit val

In Java: myGeocoder.setOptClosestHouseNumDifferenceLimit (val);

Remarks

Return type: Integer, 0: Failure; 1: Success.

Val: Specify the maximum difference of the house numbers allowed. An attempt will be made to geocode to an address within this maximum difference. For example, if the exact address is not found, then an attempt to geocode to the address number + 2 is made. If still unsuccessful, then an attempt to geocode to the address number - 2. A sequence of attempts is made (+2, -2, +4, -4, +6, -6, ... , +val, -val) until the record is successfully geocoded or the range has been exhausted.

Example

myGeocoder.Geocoder.setOptClosestHouseNumDifferenceLimit (2);

Geocoder.setStripPrefixFromStName ()

Purpose

Sets the option to parse out the street name prefix from the street name input. This option will be only effective when geocoding parsed form of addresses.

Syntax

In C++: myGeocoder.setStripPrefixFromStName ();

In VB: myGeocoder.setStripPrefixFromStName

In Java: myGeocoder.setStripPrefixFromStName ();

Remarks

No arguments.

Example

myGeocoder.setStripPrefixFromStName ();

API Functions *(cont'd)*

Geocoder.setStripPrefixFromStNameOff ()

Purpose

Unsets the option to parse out the street name prefix from the street name input. This option will be only effective when geocoding parsed form of addresses. This function was introduced as some street names which contain prefixes are represented as two separate columns in the Georef, therefore the prefix has to be stripped out of the street name.

Syntax

In C++: myGeocoder.setStripPrefixFromStNameOff ();

In VB: myGeocoder.setStripPrefixFromStNameOff

In Java: myGeocoder.setStripPrefixFromStNameOff ();

Remarks

No arguments.

Example

myGeocoder.setStripPrefixFromStNameOff ();

Geocoder.setSearchOppositeStreet()

Purpose

Sets the flag to geocode on opposite side of street if failing to search the exact match.

Sets the flag to geocode to the opposite side of the street. If geocoding fails for the input address number, this option will attempt to geocode to the opposite side of the street by +/- one value to the address number.

Input Address	Geocoded Address
625 Cochrane Dr	624 and 626 Cochrane Dr

To return the actual address number that is geocoded, this value is written to the StNum output parameter.

If the option **setOptClosestHouseNumDifferenceLimit** has been set, then it will take precedence over **setSearchOppositeStreet**.

Syntax

In C++:myGeocoder.setSearchOppositeStreet();

In VB:myGeocoder.setSearchOppositeStreet

In Java:myGeocoder.setSearchOppositeStreet();

Remarks

Return type: Integer, 0: Failure; 1: Success.

Example

myGeocoder.setSearchOppositeStreet ();

API Functions *(cont'd)*

Geocoder.setSearchOppositeStreetOff()

Purpose

Unsets the opposite side street flag so that the geocoder will not search opposite street side after calling this method.

Syntax

In C++:myGeocoder.setSearchOppositeStreetOff();

In VB:myGeocoder.setSearchOppositeStreetOff

In Java:myGeocoder.setSearchOppositeStreetOff();

Remarks

Return type: Integer, 0: Failure; 1: Success.

Example

myGeocoder.setSearchOppositeStreetOff ();

Geocoder.setOppositeSideHouseNumDifferenceLimit (int val);

Purpose

Sets the address tolerance value to be used for geocoding the closest address (**opposite** side of the street) if the exact address number fails.

If this option has been set along with the closest address, the latter function will take precedence before utilizing the opposite side function.

Syntax

In C++: myGeocoder.setOppositeSideHouseNumDifferenceLimit (val);

In VB: myGeocoder.setOppositeSideHouseNumDifferenceLimit val;

In Java: myGeocoder.setOppositeSideHouseNumDifferenceLimit (val);

Remarks

Return type: Integer, 0: Failure; 1: Success.

Val: Specify the maximum difference of the house numbers allowed. An attempt will be made to geocode to an address within this maximum difference. For example, if the exact address is not found, then an attempt to geocode to the address number + 1 is made. If still unsuccessful, then an attempt to geocode to the address number - 1. A sequence of attempts is made (+1, -1, +3, -3, +5, -5, ... , +val, -val) until the record is successfully geocoded or the range has been exhausted.

Example

myGeocoder.Geocoder. setOppositeSideHouseNumDifferenceLimit (1);

API Functions *(cont'd)*

Geocoder.setSearchSoundex()
Geocoder.setSoundexScore() [int;score]
Geocoder.setSearchStreetNameSoundex()
Geocoder.setSearchMunicipalitySoundex()

Purpose

Sets the flag to search POI name by Soundex algorithm if the original POI name fails for Geocoding.

The Soundex module for GeoPinpoint Suite Windows/ActiveX is a new function for release to clients. This module is an algorithm which uses fuzzy logic to help geocode POI names, Street names, and Municipality names which suffer from spelling variations, abbreviations, difference in case and/or are incomplete. The Soundex algorithm attempts to match an incorrectly spelled name to it's respective spelling in the Georef.

Currently the Soundex module for GeoPinpoint Suite will perform the following:

- Help geocode to POI names
- Ability to match to street names and municipalities (StName / MuniName)
- Set thresholds for Soundex matching
- Include the Soundex score as part of the output
- Match one or multiple POI names simultaneously

Examples of Soundex:

Original word	Input word	Output word
Toronto City Hall	Toronto City Hall	Toronto City Hall
Toronto City Hall	City Hall of Toronto	Toronto City Hall
Toronto City Hall	Holl of Sity Toronto	Toronto City Hall
Konberg	Comperg	Konberg

Syntax

In VB:myGeocoder.setSearchSoundex

Remarks

Return type: Integer, 0: Failure; 1: Success.

Example

myGeocoder.setSearchSoundex();

Geocoder.setSearchSoundexOff()

Purpose

Unsets the flag so that the Soundex Search will not be implemented.

Syntax

In VB:myGeocoder.setSearchSoundexOff

Remarks

Return type: Integer, 0: Failure; 1: Success.

Example

myGeocoder.setSearchSoundexOff();

API Functions *(cont'd)*

Geocoder.setPostalCodePrecisionCode(char *PrecisionCode)

Purpose

Sets the postal code precision code at one time. This function is used for postal code geocoder. Only the postal code records which precision codes have been specified will be geocoded. The format of the path PrecisionCode is the numeric code separated by "|", something like "100|200|300".

Syntax

In C++: MyGeocoder.setPostalCodePrecisionCode(char *PrecisionCode)

In VB: MyGeocoder.setPostalCodePrecisionCode PrecisionCode

In Java: MyGeocoder.setPostalCodePrecisionCode(PrecisionCode)

Remarks

There are five types of postal precision codes:

CanMap Street High Precision	100
CanMap Street Low Precision	200
LDU Centroid	300
FSA Centroid	400
PPN Centroid	500

Arguments

PrecisionCode: This is the numeric code(100,200,300,400,500).

Example

MyGeocoder.setPostalCodePrecisionCode("100|200|300");

Geocoder.setPostalCodePrecisionCodeOff()

Purpose

Unsets the search postal code precision code information flag so that this process will no longer be performed after calling this method.

Syntax

In C++:MyGeocoder.setPostalCodePrecisionCodeOff();

In VB:MyGeocoder.setPostalCodePrecisionCodeOff;

In Java:MyGeocoder.setPostalCodePrecisionCodeOff();

Remarks

No Arguments;

Example

MyGeocoder. setPostalCodePrecisionCodeOff();

API Functions *(cont'd)*

Geocoder.setSearchPPNMuniFSAForRuralPostalCode()

Purpose

Function is used when PPN, Muni and FSA centroid geocoders have already been selected. Instead of automatically geocoding to a rural FSA centroid as dictated by the geocoding hierarchy, the record is first geocoded to PPN or if it fails to municipality centroid. If both of these fail to geocode, the record is geocoded to FSA centroid. This function is important because sometimes a PPN or muni centroid is more precise than a rural FSA centroid.

Syntax

In C++: MyGeocoder.setSearchPPNMuniFSAForRuralPostalCode();

In VB: MyGeocoder.setSearchPPNMuniFSAForRuralPostalCode;

In Java: MyGeocoder.setSearchPPNMuniFSAForRuralPostalCode();

Example

MyGeocoder.setSearchPPNMuniFSAForRuralPostalCode()

Geocoder.setSearchPPNMuniFSAForRuralPostalCodeOff()

Purpose

Unsets this function which alters the geocoding hierarchy to allow records which contain rural FSA values to geocode to PPN or municipality centroid.

Syntax

In C++: MyGeocoder.setSearchPPNMuniFSAForRuralPostalCodeOff()

In VB: MyGeocoder.setSearchPPNMuniFSAForRuralPostalCodeOff;

In Java: MyGeocoder.setSearchPPNMuniFSAForRuralPostalCodeOff();

Remarks

No Arguments;

Example

MyGeocoder.setSearchPPNMuniFSAForRuralPostalCodeOff();

API Functions *(cont'd)*

Main Geocoding Functions

Geocoder.Geocoding(int ProcessingFlag)

Purpose

Performs geocoding. This function can be used when the user wishes to geocode only 1 type of data (e.g.: address or POI or postal code)

Syntax

In C++: myGeocoder.Geocoding(ProcessingFlag);

In VB: myGeocoder.Geocoding ProcessingFlag

In Java: myGeocoder.geocoding(ProcessingFlag);

Remarks

Return type: Integer, Total Number of Matches or 0.

ProcessingFlag: Obsolete argument, always pass 0 for newly written software and use new API to define geocoding path.

If this function is used, the processing flag (from 1 to 5) will take precedence over setGeocodingPath() (refer to section "Geocoding Path Definition Functions"). "

Example

myGeocoder.Geocoding(0);

Compatibility

Function is the same, but the argument is obsolete. Kept for compatibility. The obsolete flag values are explained in the following and are suggested not to use them but using the setGeocodingChoiceOn() to replace them.

Flag values

GEOCODING TO ADDRESS POINTS (0):. This has been taken care of by the default geocoding path.

POSTALCODE_CENTROID_ONLY (1): This flag value indicates the geocoding process will be getting postal code centroids only.

MUNICIPAL_CENTROID_ONLY(2): This flag value indicates the geocoding process will be getting Municipal centroids only;

FSA_CENTROID_ONLY(3): This flag value indicates the geocoding process will be getting the FSA entroids only;

POI_ONLY (4): This flag value indicates that the geocoding process will be getting only points of interest centroids.

PPN_CENTROID_ONLY (5): This flag value indicates that the geocoding process will be getting only populated place names.

API Functions *(cont'd)*

Output Functions

C++/VB

Geocoder.getOutput(long* addressID, char* stdStreetName, char *StdMuniName, double* Lon, double* Lat, long * InterpolationCode, long* stSegID, char* streetWholeName, char* streetNum, char* streetPrefix, char* streetName, char* streetType, char* streetDir, char* suite, char* MuniName, char* Prov, char* PostalCode, long* resultCode, char* rangeNumber)

Java

JDmtiGeoCoder.GeoCoderOutput getOutput();

The inner class JDmtiGeoCoder.GeoCoderOutput has the following definition:

```
public static class GeoCoderOutput {
        public int addressID;
        public String stdStreetName;
        public String stdMuniName;
        public double longitude;
        public double latitude;
        public int interpolationCode;
        public int stSegID;
        public String StreetWholeName;
        public String StNum;
        public String StPrefix;
        public String StName;
        public String StType;
        public String StDir;
        public String Suite;
        public String MuniName;
        public String Prov;
        public String PostalCode;
        public String RangeNumber;
        public int resultCode;
        public int returnCode;
};
```

API Functions *(cont'd)*

Purpose

Retrieves the geocoded results.

Syntax

In C++:

```
myGeocoder.getOutput(addressID, stdStreetName, StdMuniName, &Lon,  &Lat,
& InterpolationCode,  &stSegID, streetWholeName, streetNum, streetPrefix,
streetName,  streetType,  streetDir,  suite,  MuniName, Prov, PostalCode,
&resultCode, RangeNumber);
```

In VB:

```
myGeocoder.getOutput addressID, stdStreetName, StdMuniName, Lon, Lat,
InterpolationCode, stSegID, streetWholeName, streetNum,
streetPrefix,streetName, streetType, streetDir, suite, MuniName, Prov,
PostalCode, resultCode, RangeNumber
```

In Java:

```
JDmtiGeoCoder.GeoCoderOutput myOutput = new
JDmtiGeoCoder.GeoCoderOutput();
myOutput = myGeocoder.getOutput();
```

Remarks

Return type: Integer, -1: last output/No Output; >0: Current result; 0: No output

addressID: Outputs internal address ID value;

- **atdStreetName**: Outputs standardized street name without type and direction;
- **StdMuniName**: Outputs standardized municipal name;
- **Lon**: Outputs the longitude value of the geocoded address;
- **Lat**: Outputs the latitude value of the geocoded address;
- **InterpolationCode**: Outputs the interpolation precision code for the address. Value 0 means an interpolated address; Value 20 means an address coincides with an endpoint on the street segment; Value 21 means an address belongs to a segment with an address range of only this one point (i.e. if you had a street segment where the same address value was at both ends, the point is moved to the middle of the segment);
- **stSegID**: Outputs the street segment ID;
- **streetWholeName**: Outputs the original name from the database;
- **streetNum**: Outputs the street number of the address;
- **streetPrefix**: Outputs the street prefix;
- **streetName**: Outputs the street name that is not standardized;
- **streetType**: Outputs the street type;
- **streetDir**: Outputs the street direction;
- **suite**: Outputs the suite number;
- **MuniName**: Outputs the municipal name that is associated with the range number;
- **Prov**: Outputs the province abbreviation;
- **PostalCode**: Outputs the postal code;
- **resultCode**: Outputs the result code;
- **RangeNumber**: Outputs start and end address number separated by dash if it is geocoded to street segments. If contains suffix "L" if it is on left side of street, and "R" if it is on right side of street.

API Functions *(cont'd)*

The result code will indicate which level of geocoding was performed. Call this function successively in order to retrieve all of the geocoded results. A return value of -1 indicates failure, and values of zero or greater indicate how many results are still yet to be retrieved

The following parameters will return French accents as part of the output if they are found to exist in the geo-reference database

Output Parameter	French Accents?
streetWholeName	Y
MuniName	Y

Example

```
myGeocoder.getOutput(addressID, stdStreetName, StdMuniName, &Lon,  &Lat, &
InterpolationCode,  &stSegID, streetWholeName, streetNum, streetPrefix, streetName,
streetType,  streetDir,  suite,  MuniName, Prov, PostalCode, &resultCode,
rangeNumber);
```

API Functions *(cont'd)*

C++/VB

Geocoder.getOutput2(long addressID, char* stdStreetName; char *StdMuniName, double* Lon, double* Lat, long * InterpolationCode, long* stSegID, char* outPreDir, char* outPreType, char* outStName, char* outSufType, char* outSufdir, char* outPOIname, char* inStNum, char* inStPrefix, char* inStName, char* inStType, char* inStDir, char* inSuite, char* inMuniName, char* inProv, char* inPostalCode, char* rangeNumber, long* resultCode)

Java

JDmtiGeoCoder.GeoCoderOutput2 getOutput2();

The inner class JDmtiGeoCoder.GeoCoderOutput2 has the following definition:

```
public  static class GeoCoderOutput2 {
        public int addressID;
        public String stdStreetName;
        public String stdMuniName;
        public double longitude;
        public double latitude;
        public int interpolationCode;
        public int stSegID;
        public String outPreDir;
        public String outPreType;
        public String outStName;
        public String outSufType;
        public String outSufdir;
        public String outPOIname;
        public String inStNum;
        public String inStPrefix;
        public String inStName;
        public String inStType;
        public String inStDir;
        public String inSuite;
        public String inMuniName;
        public String inProv;
        public String inPostalCode;
        public String RangeNumber;
        public int resultCode;
        public int returnCode;
};
```

API Functions *(cont'd)*

Purpose

To retrieve the geocoded results in a form that the street found from database is in a parsed format.

Syntax

In C++:

```
myGeocoder.getOutput2(&addressID, stdStreetName; StdMuniName, &Lon, &Lat,
&InterpolationCode,  &stSegID, outPreDir, outPreType, outStName,  outSufType,
outSufdir, outPOIname, inStNum, inStPrefix, inStName, inStType, inStDir, inSuite,
inMuniName, inProv, inPostalCode, resultCode, rangeNumber);
```

In VB:

```
myGeocoder.getOutput2 addressID, stdStreetName; StdMuniName, Lon, Lat,
InterpolationCode, stSegID, outPreDir, outPreType, outStName,  outSufType,
outSufDir, outPOIname, inStNum, inStPrefix, inStName, inStType, inStDir, inSuite,
inMuniName, inProv, inPostalCode, resultCode, rangeNumber
```

In Java:

```
JDmtiGeoCoder.GeoCoderOutput2 myOutput = new
JDmtiGeoCoder.GeoCoderOutput2();
myOutput = myGeocoder.getOutput2();
```

Remarks

- **addressID**: Outputs internal address ID value;
- **stdStreetName**: Outputs standardized street name without type and direction;
- **stdMuniName**: Outputs the municipal name that is associated with the range number;
- **Lon**: Outputs the longitude value of the geocoded address;
- **Lat**: Outputs the latitude value of the geocoded address;
- **InterpolationCode**: Outputs the interpolation precision code for the address Value 0 means an interpolated address; Value 20 means an address coincides with an endpoint on the street segment; Value 21 means an address belongs to a segment with an address range of only this one point (i.e. if you had a street segment where the same address value was at both ends, the point is moved to the middle of the segment)
- **stSegID**: Outputs the street segment ID;
- **outPreDir**: Outputs the Prefix street direction of the matched result;
- **outPreType**: Outputs the Prefix street type of the matched result;
- **outStName**: Outputs the street name (no type and dir) of the matched result;
- **outSufType**: Outputs the Suffix street type of the matched result;
- **outSufDir**: Outputs the Suffix street direction of the matched result;
- **outPOIName**: Outputs the POI name of the matched result (if using POI geocoder);
- **inStNum**: Outputs the parsed street number of the address;
- **inStPrefix**: Outputs the parsed street prefix;
- **inStName**: Outputs the parsed street name that is not standardized;
- **inStType**: Outputs the parsed street type;
- **inStDir**: Outputs the parsed street direction;
- **inSuite**: Outputs the parsed suite number;
- **inMuniName**: Outputs the original municipal name;
- **inProv**: Outputs the province abbreviation;
- **inPostalCode**: Outputs the postal code;
- **resultCode**: Outputs the result code;
- **rangeNumber**: Outputs start and end address number separated by dash if it is geocoded to street segments. If contains suffix "L" if it is on left side of street, and "R" if it is on right side of street.

API Functions *(cont'd)*

The result code will indicate which level of geocoding was performed. Call this function successively in order to retrieve all of the geocoded results. A return value of -1 indicates failure, and values of zero or greater indicate how many results are still yet to be retrieved

The following parameters will return French accents as part of the output if they are found to exist in the geo-reference database

Output Parameter	French Accents?
streetWholeName	Y
StdMuniName	Y
outPreDir	Y
outPreType	Y
outStName	Y
outSufType	Y
outSufDir	Y

Example

```
myGeocoder.getOutput3(&addressID, stdStreetName; StdMuniName, &Lon, &Lat,
&InterpolationCode,  &stSegID, outPreDir, outPreType, outStName,  outSufType,
outSufdir, outPOIname, inStNum, inStPrefix, inStName, inStType, inStDir, inSuite,
inMuniName, inProv, inPostalCode, resultCode, rangeNumber);
```

API Functions *(cont'd)*

C++/VB

Geocoder.getOutput3(long* addressID, char* stdStreetName, char *StdMuniName, double* Lon, double* Lat, long * InterpolationCode, long* stSegID, char* outPreDir, char* outPreType, char* outStName, char* outSufType, char* outSufdir, char* outPOIname, char* inStNum, char* inStPrefix, char* inStName, char* inStType, char* inStDir, char* inSuite, char* inMuniName, char* inProv, char* inPostalCode, char* RangeNumber, long* resultCode, double* fromDistance, double* toDistance, char* IntersectedStartStreet, char* IntersectedEndStreet);

Java

JDmtiGeoCoder.GeoCoderOutput3 getOutput3();

The inner class JDmtiGeoCoder.GeoCoderOutput3 has the following definition:

```
public  static class GeoCoderOutput3 {
        public int addressID;
        public String stdStreetName;
        public String stdMuniName;
        public double longitude;
        public double latitude;
        public int interpolationCode;
        public int stSegID;
        public String outPreDir;
        public String outPreType;
        public String outStName;
        public String outSufType;
        public String outSufdir;
        public String outPOIname;
        public String inStNum;
        public String inStPrefix;
        public String inStName;
        public String inStType;
        public String inStDir;
        public String inSuite;
        public String inMuniName;
        public String inProv;
        public String inPostalCode;
        public String RangeNumber;
        public int resultCode;
        public int returnCode;
        public double fromDistance;
        public double toDistance;
        public String intersectedStartStreet;
        public String intersectedEndStreet;
};
```

API Functions *(cont'd)*

Purpose

To retrieve the geocoded results in both Geographic coordinate systems and Linear reference system representation.

Syntax

In C++:

```
myGeocoder.getOutput3(&addressID, stdStreetName; StdMuniName, &Lon, &Lat,
&InterpolationCode,  &stSegID, outPreDir, outPreType, outStName,  outSufType,
outSufdir, outPOIname, inStNum, inStPrefix, inStName, inStType, inStDir, inSuite,
inMuniName, inProv, inPostalCode, rangeNumber, resultCode, fromDistance,
toDistance, IntersectedStartStreet, IntersectedEndStreet);
```

In VB:

```
myGeocoder.getOutput3 addressID, stdStreetName; StdMuniName, Lon, Lat,
InterpolationCode, stSegID, outPreDir, outPreType, outStName,  outSufType,
outSufDir, outPOIname, inStNum, inStPrefix, inStName, inStType, inStDir, inSuite,
inMuniName, inProv, inPostalCode, rangeNumber, resultCode, fromDistance,
toDistance, IntersectedStartStreet, IntersectedEndStreet
```

In Java:

```
 JDmtiGeoCoder.GeoCoderOutput3 myOutput = new
 JDmtiGeoCoder.GeoCoderOutput3();
myOutput = myGeocoder.getOutput3();
```

Remarks

- **addressID**: Outputs internal address ID value;
- **stdStreetName**: Outputs standardized street name without type and direction;
- **stdMuniName**: Outputs the municipal name that is associated with the range number;
- **Lon**: Outputs the longitude value of the geocoded address;
- **Lat**: Outputs the latitude value of the geocoded address;
- **InterpolationCode**: Outputs the interpolation precision code for the address Value 0 means an interpolated address; Value 20 means an address coincides with an endpoint on the street segment; Value 21 means an address belongs to a segment with an address range of only this one point (i.e. if you had a street segment where the same address value was at both ends, the point is moved to the middle of the segment)
- **stSegID**: Outputs the street segment ID;
- **outPreDir**: Outputs the Prefix street direction of the matched result;
- **outPreType**: Outputs the Prefix street type of the matched result;
- **outStName**: Outputs the street name (no type and dir) of the matched result;
- **outSufType**: Outputs the Suffix street type of the matched result;
- **outSufDir**: Outputs the Suffix street direction of the matched result;
- **outPOIname**: Outputs the POI name of the matched results (if using POI geocoder);
- **inStNum**: Outputs the parsed street number of the address;
- **inStPrefix**: Outputs the parsed street prefix;
- **inStName**: Outputs the parsed street name that is not standardized;
- **inStType**: Outputs the parsed street type;
- **inStDir**: Outputs the parsed street direction;
- **inSuite**: Outputs the parsed suite number;
- **inMuniName**: Outputs the original municipal name;
- **inProv**: Outputs the province abbreviation;
- **inPostalCode**: Outputs the postal code;

API Functions *(cont'd)*

- **rangeNumber**: Outputs start and end address number separated by dash if it is geocoded to street segments. If contains suffix "L" if it is on left side of street, and "R" if it is on right side of street.
- **resultCode**: Outputs the result code;
- **fromDistance**: Outputs the distance in meters from the intersected start street;
- **toDistance**: Outputs the distance in meters to the intersected end street;
- **IntersectedStartStreet**: Outputs the intersected street whole name on the starting point side of the referenced street;
- **IntersectedEndStreet**: Outputs the intersected street whole name on the end point side of the referenced street;

The result code will indicate which level of geocoding was performed. Call this function successively in order to retrieve all of the geocoded results. A return value of -1 indicates failure, and values of zero or greater indicate how many results are still yet to be retrieved.

The following parameters will return French accents as part of the output if they are found to exist in the geo-reference database

Output Parameter	French Accents?
streetWholeName	Y
StdMuniName	Y
outPreDir	Y
outPreType	Y
outStName	Y
outSufType	Y
outSufDir	Y

Example

myGeocoder.getOutput3(&addressID, stdStreetName; StdMuniName, &Lon, &Lat, &InterpolationCode, &stSegID, outPreDir, outPreType, outStName, outSufType, outSufdir, outPOIname, inStNum, inStPrefix, inStName, inStType, inStDir, inSuite, inMuniName, inProv, inPostalCode, rangeNumber, resultCode, fromDistance, toDistance, IntersectedStartStreet, IntersectedEndStreet);

Compatibility

The street segment rangeNumber is divided into left and right sides.

API Functions *(cont'd)*

Geocoder.setResultPosAtStart()

Purpose

Resets the result record location to the beginning of the geocoding results.

Syntax

In C++: myGeocoder.setResultPosAtStart();

In VB: myGeocoder.setResultPosAtStart

In Java: myGeocoder.setResultPosAtStart();

Remarks

Return type: Integer, 0: Failure; 1: Success.

No argument. This function is called in order to get the geocoded results once more by using the getOutput() function.

Example

myGeocoder.setResultPosAtStart();

Geocoder.ExplainResultCode(int ResultCode, char* pchInterpretResultCode)

Purpose

Retrieves an explanation string from the digital result code.

Syntax

In C++: myGeocoder.ExplainResultCode (ResultCode);

In VB: myGeocoder.ExplainResultCode ResultCode

In Java: myGeocoder.explainResultCode(ResultCode);

Remarks

Return type: Integer, 0: Failure; 1: Success.

Argument:

Result Code: The numeric result code obtained by the getOutput() function.

Example

myGeocoder.ExplainResultCode (ResultCode);

API Functions *(cont'd)*

Geocoder.getVersion (char* Version)

Purpose

Retrieves a version number from the API library.

Syntax

In C++: myGeocoder.getVersion (version);

In VB: myGeocoder.getVersion version

In Java: myGeocoder.getVersion (version);

Remarks

Return type: Integer, 0: Failure; 1: Success.

Argument:
version: The character string for storing returned versions string. In a language the length of string has to be specified, this "version" string has to be 10 characters or longer.

Example

myGeocoder.getVersion (version);

Geocoder.getPrecisionCode(int resultCode, int InterpolationCode)

Purpose

Generate the precision code for the geocoding results.

Syntax

In C++: myGeocoder.getPrecisionCode(resultCode, InterpolationCode);

In VB: myGeocoder.getPrecisionCode resultCode, InterpolationCode

In Java: myGeocoder.getPrecisionCode(resultCode, InterpolationCode);

Remarks

Return type: Integer.

Please see *Appendix 2* for a more detailed explanation of the precision codes.

Argument:
resultCode: Inputs the resultCode of the result;
InterpolationCode: Inputs the interpolation code;

Example

myGeocoder.getPrecisionCode(resultCode, InterpolationCode);

Conclusion

DMTI Spatial's geocoder, GeoPinpoint Suite, is able to attach geographic coordinates to records in a database by matching the data from certain fields in the target database against an existing geo-reference database. The geo-reference database is made up of digital street geometry, address ranges, postal coordinates, and other point-location coordinates, which are updated regularly by DMTI Spatial to ensure the greatest possible accuracy. After the data is geocoded, it can be transferred into a geographic information system such as AutoCAD Map, MapInfo, ArcInfo, ArcView or other system that supports spatial data.

GeoPinpoint Suite is able to geocode address data, intersection data, and points of interest data as long as the data is stored in an Oracle 8i/9i, Access database (*.mdb), dBase or FoxPro file. GeoPinpoint Suite offers a great deal of flexibility in data entry because it handles French-style addressing as efficiently as English-style addressing and it is able to geocode unparsed or parsed addresses. In addition, GeoPinpoint Suite gives the user many options to improve geocoding accuracy, such as the option to refine by postal code, and to obtain higher matching rates, such as the ability to "relax" on street type, street direction, or street prefix.

The staff at DMTI Spatial is committed to helping all of its clients realize the potential of spatial solutions. This document has been written to provide all users with the information needed to effectively use GeoPinpoint Suite. It is noted, however, that each user has unique circumstances that may pose challenges to geocoding in addition to those covered in this document. DMTI Spatial welcomes any further questions users may have and any comments they would like to make regarding this, or any other, product or service provided by DMTI Spatial.

DMTI Spatial can be contacted at:

DMTI Spatial Inc.
625 Cochrane Drive, 3rd Floor
Markham, Ontario
L3R 9R9   Canada

Telephone:	905-948-2000
Toll Free:	1-877-477-DMTI (3684)
Fax:	905-948-9404
URL:	www.dmtispatial.com
Email:	info@dmtispatial.com

Appendix 1: Interpretation of Result Code (Rcode)

When GeoPinpoint Suite geocodes, it generates a 9-digit result code, which helps the user to understand how the record was processed and the degree to which the record was successfully geocoded. A 9-digit result code, such as 111101001, is written to the Result Code (i.e.: Rcode) field or alternatively, to a user-specified field in the target database table. Each digit represents one component of the geocoding process as defined in *Chart 1*.

For example, 111101001 represents

1	1	1	1	0	1	0	0	1
STATUS	**TYPE**	**STREET**	**POSTAL CODE / PPN**	**MUNI NAME**	**RELAX**	**PROV**	**REFINED**	**PARSER**
OK	SEGMENT ADDRESS	ORIGINAL	CANMAP HI	NOT FOUND	ON STREET TYPE	NOT FOUND	NOT FOUND	OK

Appendix 1: Interpretation of Result Code (Rcode) *(cont'd)*

Chart 1: Individual digit definitions of 9-digit result codes.

Format:		**D**	**D**	**D**	**D**	**D**	**D**	**D**	**D**	**D**
STATUS										
OK		1								
FAIL		2								
TYPE										
NOT FOUND / NOT APPLICABLE			0							
BY SEGMENT ADDRESS			1							
BY INTERSECTION			2							
BY POINT OF INTEREST			3							
BY POSTAL CODE			4							
BY MUNICIPAL CENTROID			5							
BY FSA CENTROID			6							
BY POPULATED PLACE NAME			7							
BY CLOSEST ADDRESS			8							
BY SEGMENT			9							
STREET										
NOT FOUND / NOT APPLICABLE				0						
ORIGINAL				1						
ZERO ADDRESSING				2						
STREET ALIAS (FORMER NAME)				3						
IN-BETWEEN FUNCTION / SUBSTITUTED				4						
LOOK FOR SEGMENT INFO				5						
SEARCH BY SOUNDEX				6						
SEARCH BY SCRUBBER				7						
POSTAL CODE OR	**POPULATED PLACE NAME / POINT OF INTEREST[1]**									
NOT FOUND / NOT APPLICABLE	NOT FOUND / NOT APPLICABLE				0					
CANMAP HI	NTDB				1					
CANMAP LO	CANMAP HI				2					
LDU Centroid	CANMAP LO				3					
FSA Centroid	POSTAL CODE BLOCK FACE				4					
PPN Centroid	POSTAL CODE EA CENTROID				5					
	MUNICIPAL CENTROID				6					
	CANADIAN GEOGRAPHIC NAMES DATABASE				7					
SEGMENT										
SEGMENT CENTROID					8					
TO SEGMENTS (LOWEST ADDRESS)					9					

(table continued)

[1] The result code for POI is similar to PPN except that the value of 7 (CANADIAN GEOGRAPHIC NAMES DATABASE) is not used.

Appendix 1: Interpretation of Result Code (Rcode) *(cont'd)*

MUNICIPALITY NAME										
NOT FOUND / NOT APPLICABLE						0				
ORIGINAL						1				
FROM POSTAL CODE						2				
BY FSA BOUNDARY						4				
SUBSTITUTED						5				
FROM MUNICIPAL ALIAS LIST						6				
		D	**D**	**D**	**D**	**D**	**D**	**D**	**D**	**D**
RELAX										
NOT FOUND / NOT APPLICABLE							0			
ON STREET TYPE							1			
ON STREET DIRECTION							2			
ON STREET PREFIX							3			
ON STREET TYPE & STREET DIRECTION							4			
ON STREET TYPE & STREET PREFIX							5			
ON STREET DIRECTION & STREET PREFIX							6			
ON STREET TYPE & STREET DIRECTION & STREET PREFIX							7			
PROVINCE										
NOT FOUND / NOT APPLICABLE								0		
ORIGINAL								1		
SUBSTITUTED								2		
CUSTOM LOOKUP								3		
REFINED										
NOT FOUND / NOT APPLICABLE									0	
REFINED BY POSTAL CODE									1	
REFINED BY FSA									2	
REFINED BY RURAL POSTAL CODE (MUNI / FSA)									3	
REFINED BY RURAL POSTAL CODE (MUNI)									4	
PARSER										
NOT FOUND / NOT APPLICABLE										0
OK										1
FAIL										2

Appendix 1: Interpretation of Result Code (Rcode) *(cont'd)*

Result Code Digit Definitions

STATUS (1st digit → e.g.: **1**11101001)

Code	Value	Definition
1	OK	Record was geocoded
2	FAIL	Record was not geocoded

TYPE (2nd digit → e.g.: 1**1**1101001)

Code	Value	Definition
0	NOT FOUND / NOT APPLICABLE	The result codes generated by GeoPinpoint Suite do not currently distinguish between "Not Found" or "Not Applicable" and a zero is used for both cases. Depending on the data input there are two possible interpretations: 1) "Not found", implies that an address component was entered, such as a municipality, but that the municipality was not found in the geo-reference database. 2) "Not Applicable", implies that an address component was not entered. For example, if no municipality value is selected in the *Input Specifications* section of the dialog, then all records without this value will display a zero for the fifth digit of the result code.
1	BY SEGMENT ADDRESS	Record was geocoded using the address geocoder
2	BY INTERSECTION	Record was geocoded to intersection using the address geocoder (see Intersection Delimiter)
3	BY POINT OF INTEREST	Record was geocoded to Point of Interest using the POI geocoder
4	BY POSTAL CODE	Record was geocoded to postal code using the Postal Code geocoder
5	BY MUNICIPAL CENTROID	Record was geocoded to municipal centroid using the Boundary geocoder
6	BY FSA CENTROID	Record was geocoded to FSA centroid using the Boundary geocoder
7	BY POPULATED PLACE NAME	Record was geocoded to PPN point using the Boundary geocoder
8	BY CLOSEST ADDRESS	Records was geocoded to the closest address using the address geocoder
9	BY SEGMENT	Records was geocoded to the address segment centroid using the Segment geocoder

Appendix 1: Interpretation of Result Code (Rcode) *(cont'd)*

STREET (3rd digit → e.g.: 11**1**101001)

Code	Value	Definition
0	NOT FOUND / NOT APPLICABLE	The result codes generated by GeoPinpoint Suite do not currently distinguish between "Not Found" or "Not Applicable" and a zero is used for both cases. Depending on the data input there are two possible interpretations: 1) "Not found", implies that an address component was entered, such as a municipality, but that the municipality was not found in the geo-reference database. 2) "Not Applicable", implies that an address component was not entered. For example, if no municipality value is selected in the *Input Specifications* section of the dialog, then all records without this value would display a zero for the fifth digit of the result code.
1	ORIGINAL	Record was geocoded successfully because input street name matched to the geo-reference database.
2	ZERO ADDRESSING	Record was geocoded to a street whose segments are all zero addressed (i.e.: address range is 0 to 0)
3	STREET ALIAS (INCLUDES FORMER NAME)	When user selects the option 'Geocode to Street Alias' the record will geocode to street aliases (refer to page 87).
4	IN-BETWEEN FUNCTION / SUBSTITUTED	When user selects the option 'Geocode to In-Between Addresses' the record will geocode to a segment that is located inbetween two known CanMap address segments. If the 'In-Between' function is not selected, this result code will reflect GPP changing the street data where a % or $ symbol is found.
5	LOWEST ADDRESS	Record was geocoded using the Segment geocoder
6	SEARCH BY SOUNDEX	Record was geocoded using the assistance of the GPP Suite Soundex function
7	SEARCH BY SCRUBBER	Record was geocoded using the assistance of the GPP Suite Scrubber function

Appendix 1: Interpretation of Result Code (Rcode) *(cont'd)*

POSTAL CODE (4th digit → e.g.: 111**1**01001)

Code	Value	Definition
0	NOT FOUND / NOT APPLICABLE	The result codes generated by GeoPinpoint Suite do not currently distinguish between "Not Found" or "Not Applicable" and a zero is used for both cases. Depending on the data input there are two possible interpretations: 1) "Not found", implies that an address component was entered, such as a municipality, but that the municipality was not found in the geo-reference database. 2) "Not Applicable", implies that an address component was not entered. For example, if no municipality value is selected in the *Input Specifications* section of the dialog, then all records without this value would display a zero for the fifth digit of the result code.
1	CANMAP HI	Record was geocoded to a block-face* representative point from CanMap streets (High precision). High precision indicates that the postal code has been geocoded to CanMap Streetfiles, which contain the address (midpoint of the address range in CPC data) of the postal code.
2	CANMAP LO	Record was geocoded to a block-face* representative point from CanMap streets (Lower precision). Low precision indicates that the postal code has been geocoded to CanMap "closest address" with the tolerance set to 10.
3	LDU Centroid	Record was geocoded to postalcode representative point which is from the Platinum Postal Code$^{\text{OM*}}$ Suite (PPCS) Local Delivery Unit (LDU) centroid.
4	FSA Centroid	Record was geocoded to postalcode representative point which is from the Platinum Postal Code$^{\text{OM*}}$ Suite (PPCS) Forward Sortation Area (FSA) centroid.
5	PPN Centroid	Record was geocoded to Populated Placename (PPN) centroid.

**A block face is one side of a street between two consecutive features intersecting that street. The points are set back a perpendicular distance of either 10, 5, or 1 meter(s) from the street centre line to ensure that all points have unique coordinates, and are located in the correct block and on the correct side of the street*[1]*.*

[1] Statistics Canada: Postal Code Conversion File January 2003 Postal Codes Reference Guide.

Appendix 1: Interpretation of Result Code (Rcode) *(cont'd)*

POPULATED PLACE NAME (PPN) / POINT OF INTEREST (POI) (4th digit → e.g.: 111**1**01001)

Code	Value	Definition
0	NOT FOUND / NOT APPLICABLE	The result codes generated by GeoPinpoint Suite do not currently distinguish between "Not Found" or "Not Applicable" and a zero is used for both cases. Depending on the data input there are two possible interpretations: 1) "Not found", implies that an address component was entered, such as a municipality, but that the municipality was not found in the geo-reference database. 2) "Not Applicable", implies that an address component was not entered. For example, if no municipality value is selected in the *Input Specifications* section of the dialog, then all records without this value would display a zero for the fifth digit of the result code.
1	NTDB	PPN: Record was geocoded to a PPN point, which represents the centroid of a National Topographic Database (NTDB) feature. POI: Record was geocoded to a POI point, which represents the centroid of 1:50 000 NTDB feature or placed via Orthorectified photo
2	CANMAP HI	PPN: Record was geocoded to PPN point, which represents the coordinate of a CanMap Major intersection, City Hall or Enhanced Point of Interest (EPOI) (i.e.: manually placed and verified). POI: Record was geocoded to a POI point, which represents a block-face* representative point from CanMap streets - High precision.

**A block face is one side of a street between two consecutive features intersecting that street. The points are set back a perpendicular distance of either 10, 5, or 1 meter(s) from the street centre line to ensure that all points have unique coordinates, and are located in the correct block and on the correct side of the street.*[1]

[1] Statistics Canada: Postal Code Conversion File January 2003 Postal Codes Reference Guide.

Appendix 1: Interpretation of Result Code (Rcode) *(cont'd)*

3	CANMAP LO	PPN: Record was geocoded to PPN, which is attached to a coordinate of a CanMap nearest intersection (i.e.: manually placed and verified). POI: Record was geocoded to POI point, which represents a block-face* representative point from CanMap streets - Lower precision.
5	POSTAL CODE EA CENTROID	PPN: Record was geocoded to PPN point, which represents the coordinate of a postal code EA** centroid POI: Record was geocoded to POI point, which represents a postal Code – EA** Centroid / FSA Centroid
6	MUNICIPAL CENTROID	PPN: Record was geocoded to PPN point, which represents the coordinate of a Statistics Canada municipal centroid or a designated places centroid POI: Record was geocoded to POI point, which represents a municipal centroid
7	CANADIAN GEOGRAPHIC NAMES DATABASE	PPN: Record was geocoded to PPN point, which represents a coordinate from the Canadian Geographic Names Database POI: N/A

***An enumeration area (EA) is the geographic area canvassed by one census representative. An EA is composed of one or more adjacent blocks.*[1]

SEGMENT (4th digit → e.g.: 111**1**01001)

Code	Value	Definition
8	SEGMENT CENTROID	Record was geocoded using the Segment geocoder | To Segment centroid function. The input address number locates the appropriate segment and geocodes to the segment centroid.
9	TO SEGMENTS (LOWEST ADDRESS)	Record was geocoded using the Segment geocoder | To Street Segments. Input address records without address numbers are geocoded to the first segment of the street and then to the lowest address.

[1] Statistics Canada document: Postal Code Conversion File January 2003 Postal Codes Reference Guide.

Appendix 1: Interpretation of Result Code (Rcode) *(cont'd)*

MUNICIPALITY NAME (5th digit → e.g.: 1111**0**1001)

Code	Value	Definition
0	NOT FOUND / NOT APPLICABLE	The result codes generated by GeoPinpoint Suite do not currently distinguish between "Not Found" or "Not Applicable" and a zero is used for both cases. Depending on the data input there are two possible interpretations: 1) "Not found", implies that an address component was entered, such as a municipality, but that the municipality was not found in the geo-reference database. 2) "Not Applicable", implies that an address component was not entered. For example, if no municipality value is selected in the *Input Specifications* section of the dialog, then all records without this value would display a zero for the fifth digit of the result code.
1	ORIGINAL	When GeoPinpoint Suite matches an input municipality using the Statistics Canada municipality list, the fifth digit of the result code will become a 1.
2	FROM POSTAL CODE	Municipality information contained in record was matched using the function 'Lookup Municipality via Postal Code'.
4	BY FSA BOUNDARY	Municipality information contained in record was matched by using FSA boundary.
5	SUBSTITUTED	User information may contain invalid characters such a % or $. If GPP Suite encounters such invalid characters it will strip them out in the standardization process in order to geocode.
6	FROM MUNICIPAL ALIAS LIST	DMTI Spatial has created a list of municipal aliases that allows the user to geocode databases that contain place names that do not have individual Census Subdivision boundaries as defined by Statistics Canada. If the input municipality does not match then GeoPinpoint Suite will check it against an alias list. If the record geocodes, then the fifth digit of the result code, will be a 6.

Appendix 1: Interpretation of Result Code (Rcode) *(cont'd)*

RELAX (6th digit → e.g.: 111101001)

Code	Value	Definition
0	NOT FOUND / NOT APPLICABLE	No relax functions were selected
1	ON STREET TYPE	Function 'Relax Matching on Street Type' was selected
2	ON STREET DIRECTION	Function 'Relax Matching on Street Direction' was selected
3	ON STREET PREFIX	Function 'Relax Matching on Street Prefix' was selected
4	ON STREET TYPE & STREET DIRECTION	Function 'Relax Matching on Street Type' was selected AND Function 'Relax Matching on Street Direction' was selected
5	ON STREET TYPE & STREET PREFIX	Function 'Relax Matching on Street Type' was selected AND Function 'Relax Matching on Street Prefix' was selected
6	ON STREET DIRECTION & STREET PREFIX	Function 'Relax Matching on Street Direction' was selected AND Function 'Relax Matching on Street Prefix' was selected
7	ON STREET TYPE & STREET DIRECTION & STREET PREFIX	Function 'Relax Matching on Street Type' was selected AND Function 'Relax Matching on Street Direction' was selected AND Function 'Relax Matching on Street Prefix' was selected

Appendix 1: Interpretation of Result Code (Rcode) *(cont'd)*

PROVINCE (7th digit → e.g.: 111101001)

Code	Value	Definition
0	NOT FOUND / NOT APPLICABLE	The result codes generated by GeoPinpoint Suite do not currently distinguish between "Not Found" or "Not Applicable" and a zero is used for both cases. Depending on the data input there are two possible interpretations: 1) "Not found", implies that an address component was entered, such as a municipality, but that the municipality was not found in the geo-reference database. 2) "Not Applicable", implies that an address component was not entered. For example, if no municipality value is selected in the *Input Specifications* section of the dialog, then all records without this value would display a zero for the fifth digit of the result code.
1	ORIGINAL	Record was geocoded as GPP Suite found the provincial value in the geo-reference database
2	SUBSTITUTED	Provincial information in record was standardized and then used to make a match in the Georef
3	CUSTOM LOOKUP	This is for customers who have customized geo-reference databases.

REFINED (8th digit → e.g.: 111101001)

Code	Value	Definition
0	NOT FOUND / NOT APPLICABLE	'Refine By Postal Code' option was not selected
1	REFINED BY POSTAL CODE	Value occurs when 'Refine By Postal Code' option is selected
2	REFINED BY FSA	Address was geocoded using both the municipality and the FSA value. Municipality value result code is found in the 5th digit of the result code.
3	REFINED BY RURAL POSTAL CODE (MUNI / FSA)	Rural pcode to PPN function was used which indicates that a rural FSA was identified and the record geocoded to PPN (or Muni). The PPN (or Muni) value was matched using a municipality **and** FSA value to select the correct PPN (or Muni) value.
4	REFINED BY RURAL POSTAL CODE (MUNI)	Rural pcode to PPN function was used which indicates that a rural FSA was identified and the record geocoded to PPN. The PPN (or Muni) value was matched using a municipality value to select a PPN value. NOTE: The correct (intended) PPN (or Muni) may not have been selected because a FSA value was not provided to help decide between duplicate values.

Appendix 1: Interpretation of Result Code (Rcode) *(cont'd)*

PARSER (9th digit → e.g.: 111101001)

Code	Value	Definition
0	NOT FOUND / NOT APPLICABLE	Value is possible when geocoding parsed data which does not require the parser
1	OK	Parser parsed record successfully
2	FAIL	Parser was unable to parse record successfully

Appendix 2: Interpretation of Precision Code (Prescode)

Each time GeoPinpoint Suite successfully geocodes a record, a precision code is written to the Precision Code (i.e.: Prescode) field (or alternatively, other user-specified field) in the target database table. This code is an indicator of the spatial precision of the geocoded point. The precision code is generated based on the result code and an interpolation code.

Interpretation of Precision Code

An interpolation code is an indicator for the address point status that describes whether or not an address was geocoded to a specific point location that exists in the geo-reference database or to an interpolated address. GeoPinpoint Suite will not return these interpolation codes of 0, 20 or 21 to the user but will reference them through certain specific precision codes (e.g.: Precision codes: 10, 15, 30).

Interpolation Code 0 **-** indicates that the address point is interpolated and its existence in the real world is not guaranteed;

Interpolation Code 20 - indicates that the address point is the start or end point of an original street segment, and it most likely exists in the real world;

Interpolation Code 21 **-** indicates the address point is both the starting and the end point of the corresponding street segment (i.e. there is only one address point on the street segment), and the location of this point has been relocated to the centre of the segment. This address point also most likely exists in the real world.

Precision Code

Precision code values calculated based on the geocoding method are shown in *Chart 2*.

Chart 2: *Definitions of precision codes.*

	Precision Code
Not Geocoded (Result Code is greater or equal to 200000000)	0
Geocoded By Street Name (using Muni **and** FSA boundaries):	5
Geocoded by Street Name (using Muni **or** FSA boundaries)	
With interpolation code 0:	10
With interpolation code 20:	15
With interpolation code 21:	30
From Address Point Data	35
Geocoded by Closest Address (Muni **or** FSA):	40
Geocoded by Closest Address (Muni **and** FSA):	45
Geocoded by Street Alias (Includes Former Name) (using Muni **and** FSA boundaries):	55
Geocoded by Street Alias (Includes Former Name) (using Muni **or** FSA boundaries):	

Appendix 2: Interpretation of Precision Code (Prescode) *(cont'd)*

With interpolation code 0:	60
With interpolation code 20:	65
With interpolation code 21:	70
From Address Point Data	75
Geocoded by Point of Interest (POI)	
Centroid of 1:50 000 NTDB feature or placed via Orthorectified photo	81
Block-face representative point from CanMap streets - High precision	82
Block-face representative point from CanMap streets - Low precision	83
Postal Code - Block-face representative point	84
Postal Code - EA Centroid / FSA Centroid	85
Municipal Centroid	86
Geocoded by Intersection	90
Geocoded by Postal Code	
CanMap Street High Precision	100
CanMap Street Low Precision	200
LDU Centroid	300
FSA Centroid	400
PPN Centroid	500
Geocoded to Segment Centroid	900
Geocoded to Street Segment	950
Geocoded to In-Between Function	955
Geocoded to zero address	960
Populated Place Name (PPN) Centroid	1050
Municipal Centroid	1100
FSA Centroid	2100

Appendix 3: Points of Interest Layers

GeoPinpoint Suite currently has the capability to geocode to the Points of Interest (POI) layers produced by DMTI Spatial, if the appropriate option is selected in the GeoPinpoint Suite dialog prior to commencing the geocoding operation. These layers and their descriptions are outlined in *Table 4*.

If geocoding to points of interest is desired, then the *Use Un-Parsed Address Field* option must be selected, and the names of the points of interest must be entered in the address field of the target database table.

GeoPinpoint Suite is able to geocode unparsed addresses and points of interest (and also intersection data) at the same time, as long as the same specified address field is used to store all the data. Parsed address data cannot be processed at the same time as points of interest (or intersection) data.

Note: When geocoding data by POI Name on the Input Specifications tab:
Un-parsed data: Select the POI Name for the combo box Un-parsed address
Parsed data: Select the POI Name for the combo box Street Name.

GeoPinpoint Suite can geocode to a point of interest using the functions listed under the Define Geocoding Path. If an attempt to geocode to a point of interest is unsuccessful, the first line of action should be to double check that the name of the point of interest in the target database table is complete and accurately spelled.

There are several points of interest layers that DMTI Spatial maintains that are not included in the GeoPinpoint Suite geo-reference database and as such, are not available for geocoding. The reason for these exclusions lies in the lack of standardized naming for these points.

Note: *Layers not available: Car Pool Lots, Weigh Stations, Toll Booths, Transit Stops, and Retail Postal Outlets.*

Table 4: *Points of Interest layers available for geocoding.*

LAYER NAME	ABBREVIATION	DESCRIPTION
Education	edu	The Education layer includes Elementary, High Schools, Colleges, Cégeps and Universities across Canada.
Health Care	hcr	Health Care facilities include Hospitals, Long-Term Care Centers, Nursing Stations, Outpatient Clinics and Community Health Centers across Canada.
Car Rental	car	This layer includes the following Car Rental Agencies; Avis, Budget, Discount, Enterprise, Hertz, National and Thrifty.
Accommodation	acc	The Accommodation layer includes hotels across Canada.
Border Crossings and Customs Offices	bor	Border Crossings and Customs Offices include independent offices, subordinate offices, service sites, and warehouses. The information was obtained from: Canada Customs and Revenue Agency (www.ccra-adrc.gc.ca)

Appendix 3: Points of Interest Layers *(cont'd)*

Golf Courses	glf	There are approximately 2000 golf courses in Canada. This layer contains both Private and Public golf courses as well as their locations, phone numbers and number of holes.
Police Stations	pol	Police Stations contain approximately 2000 records and include Police forces such as OPP, QPP, municipal police forces, RCMP and the Military Police.
Tourist POIs	tou	Tourist Information contains approximately 5000 records and includes categories such as Art Galleries, Attractions, Fishing Resorts, Historic Sites, Science Centre Tourist Information and Zoos.
Financial Institutions	fin	The Financial Institutions layer includes financial institutions belonging to one of the following three groups: (1) Banks, (2) Credit Unions and Caisses Populaires, (3) Trust Companies, Loan Companies, and Other Deposit-Taking Institutions.
Gasoline Service Stations	gas	The Gasoline Service Stations layer is comprised of six different gas companies: Husky, Irving, PetroCanada, Pioneer, Shell, and Sunoco. Pioneer and Sunoco gas stations are provided for Ontario only.
Ski Centers	ski	The Ski Centers layer is comprised of over 200 skiing facilities across Canada.
Shopping Centers	shp	The shopping centers layer is comprised of shopping centres/malls for Canada.
Cinemas	cin	The cinema layer contains cinemas across Canada.
Fire Stations	fre	The Fire Stations layer includes fire stations in the provinces of Ontario, Quebec, British Columbia, Alberta, and Saskatchewan.
Building Names	bld	The Building Names layer includes building names and building types.

In addition to the listed POI layers, the user can also geocode to a variation of DMTI Spatial's Aerodrome (AER) file that contains airports, waterdromes, heliports and airfields. Due to the names that appear in the standard naming field, DMTI Spatial has enhanced the Aerodrome file to be utilized by GeoPinpoint Suite, by adding in aliases and four digit airport codes (where source data was available). For instance, when attempting to geocode to aerodromes identified in *Table 5*, the user may enter the data seen under Name 1, Name 2 or Name 3.

Table 5: *Aerodrome sample data.*

NAME 1	NAME 2	NAME 3
Toronto/Lester B. Pearson Intl	Lester B Pearson International Airport	CYYZ
Medicine Hat	Medicine Hat Airport	CYXH
Alma	Terrain d'aviation d'Alma	CYTF
Vancouver Intl	Vancouver International Airport	CYVR
Yellowknife	Yellowknife Airport	CYZF

Appendix 4: Valid Street Type and Street Direction Data

The street types that are valid entries for geocoding parsed data with GeoPinpoint Suite are listed in *Table 6*. The Street Type column contains the full name of each street type. Both French and English types are listed. The column labeled "CanMap® Street Type" lists the CanMap® abbreviations for each type. The CanMap® street type corresponds to the standard abbreviations used by Canada Post. The language column distinguishes between street types in French (F) and street types in English (E).

Table 7 lists the valid entries for street direction for parsed data. The first column names the direction in full, and the second column denotes an appropriate abbreviation. Any of the full names or abbreviations may be used to obtain successful matches.

If a street type that is not listed in *Table 6* occurs in a target database table, and parsed address geocoding is desired, then that street type should be included in the field containing the street name, and the field containing street types should be left blank for that record.

Table 6: *Valid Street Types and Abbreviations.*

Street Type	CanMap Street Type	Language
Abbey	ABBEY	E
Acres	ACRES	E
Allée	ALLÉE	F
Alley	ALLEY	E
Autoroute	AUT	F
Avenue	AV	F
Avenue	AVE	E
Bay	BAY	E
Beach	BEACH	E
Bend	BEND	E
Boulevard	BLVD	E
Boulevard	BOUL	F
By-Pass	BYPASS	E
Byway	BYWAY	E
Campus	CAMPUS	E
Cape	CAPE	E
Carré	CAR	F
Carrefour	CARREF	F
Centre	C	F
Centre	CTR	E
Cercle	CERCLE	F
Chase	CHASE	E
Chemin	CH	F
Circle	CIR	E
Circuit	CIRCT	E
Close	CLOSE	E
Common	COMMON	E
Concession	CONC	E
Corners	CRNRS	E
Côte	CÔTE	F
Cour	COUR	F
Cours	COURS	F
Court	CRT	E
Cove	COVE	E
Crescent	CRES	E
Croissant	CROIS	F
Crossing	CROSS	E
Cul-de-sac	CDS	E
Dale	DALE	E
Dell	DELL	E
Diversion	DIVERS	E
Downs	DOWNS	E
Drive	DR	E
Échangeur	ÉCH	F
End	END	E
Esplanade	ESPL	E
Estates	ESTATE	E
Expressway	EXPY	E
Extension	EXTEN	E
Farm	FARM	E
Field	FIELD	E
Forest	FOREST	E
Freeway	FWY	E
Front	FRONT	E
Gardens	GDNS	E
Gate	GATE	E
Glade	GLADE	E
Glen	GLEN	E
Green	GREEN	E
Grounds	GRNDS	E
Grove	GROVE	E
Harbour	HARBR	E

(table continued)

Appendix 4: Valid Street Type and Street Direction Data *(cont'd)*

Street Type	CanMap Street Type	Language
Heath	HEATH	E
Heights	HTS	E
Highlands	HGHLDS	E
Highway	HWY	E
Hill	HILL	E
Hollow	HOLLOW	E
Île	ÎLE	F
Impasse	IMP	F
Inlet	INLET	E
Island	ISLAND	E
Key	KEY	E
Knoll	KNOLL	E
Landing	LANDNG	E
Lane	LANE	E
Limits	LMTS	E
Line	LINE	E
Link	LINK	E
Lookout	LKOUT	E
Loop	LOOP	E
Mall	MALL	E
Manor	MANOR	E
Maze	MAZE	E
Meadow	MEADOW	E
Mews	MEWS	E
Montée	MONTÉE	F
Moor	MOOR	E
Mount	MOUNT	E
Mountain	MTN	E
Orchard	ORCH	E
Parade	PARADE	E
Parc	PARC	F
Park	PK	E
Parkway	PKY	E
Passage	PASS	E
Path	PATH	E
Pathway	PTWAY	E
Pines	PINES	E
Place	PL	E
Place	PLACE	F
Plateau	PLAT	E
Plaza	PLAZA	E
Point	PT	E

Street Type	CanMap Street Type	Language
Pointe	POINTE	F
Port	PORT	E
Private	PVT	E
Promenade	PROM	E
Quai	QUAI	F
Quay	QUAY	E
Ramp	RAMP	E
Rang	RANG	F
Range	RG	E
Ridge	RIDGE	E
Rise	RISE	E
Road	RD	E
Rond-point	RDPT	F
Route	RTE	E
Row	ROW	E
Rue	RUE	F
Ruelle	RLE	F
Run	RUN	E
Sentier	SENT	F
Square	SQ	E
Street	ST	E
Subdivision	SUBDIV	E
Terrace	TERR	E
Terrasse	TSSE	F
Thicket	THICK	E
Towers	TOWERS	E
Townline	TLINE	E
Trail	TRAIL	E
Turnabout	TRNABT	E
Vale	VALE	E
Via	VIA	E
View	VIEW	E
Village	VILLGE	E
Villas	VILLAS	E
Vista	VISTA	E
Voie	VOIE	F
Walk	WALK	E
Way	WAY	E
Wharf	WHARF	E
Wood	WOOD	E
Wynd	WYND	E

Appendix 4: Valid Street Type and Street Direction Data *(cont'd)*

Table 7: *Valid Street Directions.*

Street Direction	**Abbreviation**
East	E
Est	E
Nord	N
NordEst	NE
NordOuest	NO
North	N
NorthEast	NE
NorthWest	NW
Ouest	O
South	S
SouthEast	SE
SouthWest	SW
Sud	S
SudEst	SE
SudOuest	SO
West	W

Appendix 5: Geocoding Sequence

Throughout this document, each option in GeoPinpoint Suite is discussed and an effort has been made to clarify the order and under which circumstances GeoPinpoint Suite will perform each operation. The following list is a summary of the sequence of events.

After pressing the *Start* button, GeoPinpoint Suite will first evaluate the geocoding targets selected on the Geocoding Path tree. If any of them are selected, the target database table will be geocoded using these methods. If none of these are selected, the default path will be used which currently is set to geocode to address point(s) based on the segment data model using municipal boundaries.

These are the geocoding sequence steps for GeoPinpoint Suite v5.x:

The geocoding sequence is determined by the priority in the geocoding path tree. For currently implemented geocoding targets, if all of the targets are selected, its order will be in the following list and if some targets are not selected, these steps will be simply skipped.

1. Geocoding to Address point based on street segment data model using municipality boundary;

   ***Function:* Address Geocoder | By Municipality and FSA | Segment Data Model**
   ***Function:* Address Geocoder | By Municipality | Segment Data Model**

2. Geocoding to Address point based on street segment data model using FSA boundary;

   ***Function:* Address Geocoder | By FSA | Segment Data Model**

   Items a through e (i, ii) provide a detailed explanation of how the address geocoder is structured (see Steps 1, 2 above):

   a. If geocoding at intersection is requested, check for intersection data (look for user defined delimiter) and if found, geocode by intersection; otherwise,

   ***Function:* Use Intersection Delimiter**

   b. Attempt to geocode to address by matching street number, street name, street type, street direction and municipality information (if municipality information is not found, lookup municipality via postal code if this option is selected); and if this operation fails,

   ***Function:* Lookup Municipality via Postal Code**

   c. Attempt to geocode using street alias (includes street former name) if this option is selected; and is this operation fails,

   ***Function:* Geocode to Street Alias**

   d. When geocoding to address - relax first on street type, then on street direction, then on street prefix if these options are selected. If at this stage a match has been found:

   ***Function:* Relax Matching on:**

   - Street Type
   - Street Direction
   - Street Prefix

   i) Attempt to geocode the address to the correct municipality if this option is selected; and if this operation fails,

Appendix 5: Geocoding Sequence *(cont'd)*

Function: **Address Geocoder | By Municipality and FSA | Segment Data Model**

Function: **Address Geocoder | By Municipality | Segment Data Model**

ii) Attempt to geocode the address to the correct FSA boundary if this option is selected; and if this operation fails,

Function: **Address Geocoder | By FSA | Segment Data Model**

e. When geocoding using street alias - relax first on street type, then on street direction, then on street prefix if these options are selected. If at this stage a match has been found:

Function: **Geocode to Street Alias**

Function: **Relax Matching on:**

- Street Type
- Street Direction
- Street Prefix

i) Attempt to geocode the address to the correct municipality if this option is selected; and if this operation fails,

Function: **Address Geocoder | By Municipality and FSA| Segment Data Model**

Function: **Address Geocoder | By Municipality | Segment Data Model**

ii) Attempt to geocode the address to the correct FSA boundary if this option is selected; and if this operation fails,

Function: **Address Geocoder | By FSA | Segment Data Model**

f. Attempt to geocode to closest address (within user defined tolerance) if this option is selected; and if this operation fails,

Function: **Closest Address Tolerance**

3. Geocoding to POI point by matching to whole POI name using municipality boundary;

Function: **POI Geocoder | Use Whole Name | To POI Point By Municipality and FSA**
Function: **POI Geocoder | Use Whole Name | To POI Point By Municipality**

4. Geocoding to POI point by matching to whole POI name using FSA boundary;

Function: **POI Geocoder | Use Whole Name | To POI Point By FSA**

5. Geocoding to POI point by matching to whole POI Alias name using municipality boundary;

Function: **POI Geocoder | Use Whole Alias | To POI Point By Municipality and FSA**
Function: **POI Geocoder | Use Whole Alias | To POI Point By Municipality**

6. Geocoding to POI point by matching to whole POI Alias name using FSA boundary;

Function: **POI Geocoder | Use Whole Alias | To POI Point By FSA**

Appendix 5: Geocoding Sequence *(cont'd)*

7. Geocoding to POI point by matching to partial POI name using municipality boundary;

    ***Function:* POI Geocoder | Use Partial Name | To POI Point By Municipality and FSA**
    ***Function:* POI Geocoder | Use Partial Name | To POI Point By Municipality**

8. Geocoding to POI point by matching to partial POI name using FSA boundary;

    ***Function:* POI Geocoder | Use Partial Name | To POI Point By FSA**

9. Geocoding to POI point by matching to partial POI Alias name using municipality boundary;

    ***Function:* POI Geocoder | Use Partial Alias | To POI Point By Municipality and FSA**
    ***Function:* POI Geocoder | Use Partial Alias | To POI Point By Municipality**

10. Geocoding to POI point by matching to partial POI Alias name using FSA boundary;

    ***Function:* POI Geocoder | Use Partial Alias | To POI Point By FSA**

11. Geocoding to POI point by matching to POI code using municipality boundary;

    ***Function:* POI Geocoder | Use POI Type | To POI Point By Municipality and FSA**
    ***Function:* POI Geocoder | Use POI Type | To POI Point By Municipality**

12. Geocoding to POI point by matching to POI code using FSA boundary;

    ***Function:* POI Geocoder | Use POI Type | To POI Point By FSA**

13. Geocoding to Postal code point using Postal code;

    ***Function:* Postal Code Geocoder | Use Postal Code | To Postal Code Point**

14. Geocoding to street segment centroid using input address by municipal boundary;

    ***Function:* Segment Geocoder | Use Address | To Segment Centroid | By Municipality and FSA**
    ***Function:* Segment Geocoder | Use Address | To Segment Centroid | By Municipality**

15. Geocoding to street segment centroid using input address by FSA boundary;

    ***Function:* Segment Geocoder | Use Address | To Segment Centroid | By FSA**

16. Geocoding to street segments using input address by municipal boundary;

    ***Function:* Segment Geocoder | Use Address | To Street Segments | By Municipality and FSA**
    ***Function:* Segment Geocoder | Use Address | To Street Segments | By Municipality**

17. Geocoding to street segments using input address by FSA boundary;

    ***Function:* Segment Geocoder | Use Address | To Street Segments | By FSA**

Appendix 5: Geocoding Sequence *(cont'd)*

18. Geocoding to FSA centroid using FSA as input;

    Function: **Boundary | Use FSA | To FSA Centroid**

19. Geocoding to PPN point using municipality as input;

    Function: **Boundary | Use Municipality | To PPN Points**

20. Geocoding to municipal centroid using municipality as input;

    Function: **Boundary | Use Municipality | To Municipal Centroid**

21. Refine address to postal code if this option is selected.

    Function: **Refine Address By Postal Code**

Note: POI Code refers to POI Type listed in the CanMap product.
Example: Lester B Pearson International Airport has the POI Code CYYZ (See *Appendix 4*)

Appendix 6: Generic Java Wrapper

The GeoPinpoint Suite Java version is composed of the standard API libraries and Java Wrapper classes.

Standard Library Files

Please note that from this version on, “GeoCoder.so” in UNIX will be renamed “libdmtiGeoCoder.so” in order for the Java Wrapper class to be able to load this library file.

Windows

addressGeocoder.dll,
dmtidb.dll,
GPPglb.dll,
GPPSoundex.dll,
pathLookup.dll,
CPCodeGeocoder.dll,
dmtiGeoCoder.dll,
GPPParser.dll,
GPPstat.dll,
Soundex_DMTI.dll
CPOIGeocoder.dll,
GPPCalc.dll,
GPPScrubber.dll,

Unix

libAddressGeocoder.so,
libdmtidb.so,
libGPPglb.so,
libGPPstat.so,
libCPCodeGeocoder.so,
libdmtiGeoCoder.so,
libGPPParser.so,
libPathLookUp.so
libCPOIGeocoder.so,
libGPPCalc.so,
libGPPSoundex.so,

Java Class Files

JdmtiGeoCoder.java

The compiling of this file will generate the following class files under package gppJava:

JDmtiGeoCoder$GeoCoderOutput.class
JDmtiGeoCoder$GeoCoderOutput2.class
JdmtiGeoCoder.class

Java Class Interface Explanation

All of the Java class methods are trying to match the same names of corresponding GeoPinpoint Suite C++ class public methods. Therefore, the meaning of each method in the Java class methods can be found by looking up the explanation corresponding C++ class method. However, the following three methods have been re-formatted:

GetOutput()
GetOutput2()
GetOutput3()

Appendix 6: Generic Java Wrapper (cont'd)

Instead of returning results through arguments in C++, these two methods have been changed to return a Java class that contains the following results:

```
public  native JDmtiGeoCoder.GeoCoderOutput getOutput();

public  native JDmtiGeoCoder.GeoCoderOutput2 getOutput2();

public  native JDmtiGeoCoder.GeoCoderOutput3 getOutput3();
```

The remaining methods are considered to be "one to one" mapping, compared to its C++ counterparts.

The following is a list of the Java class definitions:

```
public class JDmtiGeoCoder {

        /* Geocoder initialiation constants. */
        public static final int INTERACTIVE = 1;
        public static final int BATCH = 2;

        /* Geocoder processing constants. */
        public static final int PROCESS_NORMAL = 0;
        public static final int PROCESS_POSTALCODE = 1;
        public static final int PROCESS_MUNICIPAL = 2;
        public static final int PROCESS_FSA = 3;
        public static final int PROCESS_POI = 4;
        public static final int PROCESS_PPN = 5;

        /* Geocoder output class. */
        public static class GeocoderOutput {
                public int addressID;
                public String stdStreetName;
                public String stdMuniName;
                public double longitude;
                public double latitude;
                public int interpolationCode;
                public int stSegID;
                public String StreetWholeName;
                public String StNum;
                public String StPrefix;
                public String StName;
                public String StType;
                public String StDir;
                public String Suite;
                public String MuniName;
                public String Prov;
                public String PostalCode;
                public String RangeNumber;
                public int resultCode;
                public int returnCode;
        };
        /* Geocoder output2 class. */
        public static class GeoCoderOutput2 {
```

Appendix 6: Generic Java Wrapper (cont’d)

```
                public int addressID;
                public String stdStreetName;
                public String stdMuniName;
                public double longitude;
                public double latitude;
                public int interpolationCode;
                public int stSegID;
                public String outPreDir;
                public String outPreType;
                public String outStName;
                public String outSufType;
                public String outSufdir;
                public String outPOIname;
                public String inStNum;
                public String inStPrefix;
                public String inStName;
                public String inStType;
                public String inStDir;
                public String inSuite;
                public String inMuniName;
                public String inProv;
                public String inPostalCode;
                public String RangeNumber;
                public int resultCode;
                public int returnCode;
        };
        /* Geocoder output3 class. */
        public  static class GeoCoderOutput3 {
                public int addressID;
                public String stdStreetName;
                public String stdMuniName;
                public double longitude;
                public double latitude;
                public int interpolationCode;
                public int stSegID;
                public String outPreDir;
                public String outPreType;
                public String outStName;
                public String outSufType;
                public String outSufdir;
                public String outPOIname;
                public String inStNum;
                public String inStPrefix;
                public String inStName;
                public String inStType;
                public String inStDir;
                public String inSuite;
                public String inMuniName;
                public String inProv;
                public String inPostalCode;
                public String RangeNumber;
                public int resultCode;
```

Appendix 6: Generic Java Wrapper (cont’d)

```
                public int returnCode;
                public double fromDistance;
                public double toDistance;
                public String intersectedStartStreet;
                public String intersectedEndStreet;
        };

/* DMTI geocoder native library methods. */

        public  native int initialize(String geoRefPath,
                int processMode, int offset);

        public  native int setInput(String streetName,
                String muniName, String prov, String postalCode);

        public  native int setParsedInput(String streetNum,
                String streetPrefix, String streetName, String streetType,
                String streetDir, String suite, String muniName,
                String prov, String postalCode);

        public native int setGeocodingPath(string GeocodingPath);

        public native int setGeocodingChoiceOn(int msgCode);

        public native int setGeocodingChoiceOff(int msgCode);

        public native int ClearGeocodePath();

        public  native int setRelaxOnType();

        public  native int setRelaxOnTypeOff();

        public  native int setRelaxOnDir();

        public  native int setRelaxOnDirOff();

        public  native int setGeocodeByFSA(boolean flag);

        public  native int setGeocodeByMunicipality(boolean flag);

        public  native int setIntersectionDelimiter(String delimiter);

        public  native int setGeocodeByPOI();

        public  native int setGeocodeByPOIOff();

        public  native int setGeocodeByStFName();

        public  native int setGeocodeByStFNameOff();

        public  native int setGeocodeByStAlias();

        public  native int setGeocodeByStAliasOff();
```

Appendix 6: Generic Java Wrapper (cont’d)

```
public  native int setGeocodeByRegion();

public  native int setRefineByPostalCode();

public  native int setRefineByPostalCodeOff();

public  native int setMuniIDByPostalCode();

public  native int setMuniIDByPostalCodeOff();

public  native int setSearchSegment();

public  native int setSearchSegmentOff();

public  native int setOffset(int offset);

public  native int setInset(int inset);

public  native int setFallbackPostalCode();

public  native int setFallbackPostalCodeOff();

public  native int setFallbackMunicipality();

public  native int setFallbackMunicipalityOff();

public  native int setFallbackFSA();

public  native int setFallbackFSAOff();

public  native int setOptClosestHouseNumDifferenceLimit(int val);

public native int setOppositeSideHouseNumDifferenceLimit(int val);

public  native int setStripPrefixFromStName()

public  native int setStripPrefixFromStNameOff()

public  native int setGetLinearRefResult()

public  native int setGetLinearRefResultOff()

public  native int setSearchSoundex();

public  native int setSearchSoundexOff();

public  native int setSearchScrubber();

public  native int setSearchScrubberOff();

public  native int setSearchPPNMuniFSAForRuralPostalCode();

public  native int setSearchPPNMuniFSAForRuralPostalCodeOff();
```

Appendix 6: Generic Java Wrapper (cont’d)

```
    public native int setPostalCodePrecisionCode(char *PrecisionCode);

    public native int setPostalCodePrecisionCodeOff();

    public  native int setSearchOppositeStreet();

    public  native int setSearchOppositeStreetOff();

    public  native int geocoding(int processFlag);

    public  native JDmtiGeoCoder.GeoCoderOutput getOutput();

    public  native JDmtiGeoCoder.GeoCoderOutput2 getOutput2();

    public  native JDmtiGeoCoder.GeoCoderOutput3 getOutput3();

    public  native int setResultPosAtStart();

    public  native String explainResultCode(int resultCode);

    public  native String getVersion();

    public  native int getPrecisionCode(int resultCode,
        int interpolationCode);

    /* Load the geocoder library. */
    static {
        try {
            System.loadLibrary("dmtiGeoCoder");
        } catch (Error exc) {
            System.err.println("Failed to load library dmtiGeoCoder");
        }
    }
    /* The following member and methods are used for accessing C++ class methods */

  public  native void DeleteGeoCoder();
  private  int handle;
}
```

Appendix 6: Generic Java Wrapper (cont'd)

Impact on Programming

C++ on UNIX

Now you need to link to the new name of GeoPinpoint Suite shared lib file: libdmtiGeoCoder.so;

Issues in Java Programming

Java Wrapper class enables users to program in Java and call GeoPinpoint Suite directly. Please note that in order to keep the Java methods for GeoPinpoint Suite close to its counterparts in C++, we intentionally did not implement the garbage collection method. Therefore, users need to call the function "DeleteGeoCoder()" to release the GeoPinpoint Suite object.

Users also need to have both Java classes and API library files in order to get the Java version of GeoPinpoint Suite working.

The environment variable "LD_LIBRARY_PATH" may also need to be set up in order for Java virtual machine to load the related library files for the Java Wrapper class.

In some platforms, the stack size may need to be specified as an insufficient size may lead to crashing. We suggest a stack size of **512KB** or greater.

Appendix 7: ISO 19115:2003 Compliant Metadata

Metadata Notification

On May 15th, 2005, DMTI Spatial data products will incorporate metadata that are IS0 19115:2003 compliant.

This product now includes structured metadata files as provided in XML and HTM format. These metadata files reside with the graphic or database files to which they are associated. It is recommended that users review and customize the metadata as per their specific needs.

This latest addition to CanMap® and its related products is another enhancement that we believe will benefit our users and increase your overall product satisfaction.

About DMTI Spatial:

DMTI has been providing industry leading enterprise Location Intelligence solutions for more than a decade to Global 2000 companies and government agencies. DMTI's world-class Location Hub® platform uniquely identifies, validates and maintains a universe of location-based data. DMTI is the creator of market leading Mapping Solutions and maintains the gold standard for GIS location-based data in Canada.

Contacting DMTI Spatial :

Email: info@dmtispatial.com
Phone: (877) 477-3684
Fax: (905) 948-9404