Pitney Bowes
Business Insight

Centrus GeoStan Reference Manual

Software Release 24.00

April 2011

For Windows, UNIX, z/OS

PITNEY BOWES SOFTWARE, INC
DOCUMENTATION DEPARTMENT
4200 PARLIAMENT PL STE 600
LANHAM MD 20706-1844

A Reflection of Our Commitment to You... The Pitney Bowes Business Insight Mission Statement

"The first and foremost objective of Pitney Bowes Business Insight is the total, unreserved satisfaction of each Pitney Bowes Business Insight Client — satisfaction with Pitney Bowes Business Insight products and product support, with Pitney Bowes Business Insight people and with all other facets of the Client relationship. Every other Corporate objective is subordinate to and is addressed by this one."

Table of Contents

Before You Begin

Chapter 1. Getting started

Chapter 2. Using GeoStan

CHAPTER 3. DEFINING C FUNCTIONS

APPENDIX D. STATUS CODES

APPENDIX E. CASS CERTIFICATION

APPENDIX F. USPS LINK PRODUCTS

APPENDIX G. USER DEFINED DATA FILES

Before You Begin

This chapter discusses the purpose and use of this guide, how its conventions are presented, and how to obtain assistance from Pitney Bowes Business Insight (PBBI).

Purpose of this guide

Welcome to GeoStan. This guide provides information on using the GeoStan library, along with the sample applications geotest and geocoder. Besides providing information on how to use GeoStan functions, this guide also provides information on several USPS standards and policies.

If you need more help

If you are unable to resolve a problem, a PBBI Technical Support Representative can help guide you to a solution. When you call PBBI Technical Support, please have the following information ready:

- A description of the task you were performing
- The resulting reports (specifically, the Execution Log and Parameter Record Listing).

Reporting complete details to Technical Support will help you and the technical support representative quickly resolve the problem.

PBBI technical support representatives work closely with you so that:

- Your questions on using GeoStan are answered quickly
- Any problems you may encounter while using GeoStan are resolved.

The **Technical Support Hotline** number for customers in the United States and Canada is **800-367-6950**.

The Web site

You can also find out about PBBI products and services through the PBBI Web site at http://www.g1.com. Registered users can obtain electronic copies of product documentation, join online discussion forums, download software and databases, find out about our training classes, or sign up for the PBBI List Services. From the home page, click Support and log in to the Customer Support area.

NOTE: Registered users are those who have a maintenance agreement with PBBI. Contact Technical Support to obtain your user ID and password.

eService

eService, available through the PBBI Web site, provides supported PBBI customers with direct access to Service Requests and the PBBI Solutions Knowledge Base. You can search for your own Service Requests, search the PBBI Knowledge Base, and access a variety of other Technical Support services. You must be a registered user, with an ID and password, to access eService.

Education

PBBI offers comprehensive training courses on all of its products, allowing you to maximize the usability of your software.

PBBI offers access to its courses in three ways: public seminars, eLearning, and on-site training. You can choose the option that best fits your needs.

- For information about public seminars or eLearning classes, visit our Web site at www.g1.com/Education**.**
- For information about on-site training, contact the PBBI Education Manager at (301) 918-6392.

Related publications

The following are additional publications that may assist you with installing, configuring, and using GeoStan.

- *Release Notes for the Centrus Data Product Suite*
- *Release Notes for the Centrus z/OS Product Suite*

To obtain additional user guides

To obtain electronic copies of our product manuals, go to the PBBI Web site at www.g1.com and click on Customer Login in the upper right corner. Log in with your User ID and password. On the left side of the window under “Technical Services” click on Documentation to download copies (.pdf files) of available user guides.

NOTE: To view.pdf files, you must have Adobe Acrobat Reader version 5.0 (or later) installed on your PC. If you do not have Acrobat Reader, you can download it for free from http://www.adobe.com.

Your comments are welcome

We appreciate and welcome your comments concerning this guide! If you have suggestions, please let us know. For your convenience, a Documentation Comment Form is available online at http://www.g1.com/Support/Forms/. Please print the form and send it to:

PITNEY BOWES SOFTWARE INC.
CENTRUS DIVISION
4200 PARLIAMENT PL STE 600
LANHAM MD 20706-1844

Your product enhancement suggestions are also appreciated and encouraged. With your help and suggestions, we can continue to provide quality software. For your convenience, a System Enhancement Form can be submitted online at http://www.g1.com/support/forms.

Thank you.

CHAPTER 1

Getting started

This chapter provides the following information on GeoStan.:

- About GeoStan
- Supported languages
- Installation requirements
- Installing GeoStan
- Configuring GeoStan
- Initializing GeoStan

About GeoStan

This manual details the use and functionality of Centrus GeoStan. These functions may be in the form of a library, DLL, or copybook depending on the platform you are developing on. The functions allow the developer to incorporate national address standardization, as well as address and ZIP centroid geocoding into virtually any application.

The process of standardizing and geocoding an address is very straightforward using these functions and gives the developer a large amount of freedom in implementing the user interface.

A standardized address contains the street address, city, state, and complete ZIP+4, corrected to USPS standards. A geocoded address contains the address as found in the geocode database, as well as items such as the Latitude, Longitude, and Census Block. A detailed match code is also returned for each process.

GeoStan uses data from two primary sources to perform its functions: a street network database, acquired from one of several premium vendors or the U.S. Census Bureau, and the U.S. Postal Service ZIP+4 Directory files. To fully understand the capabilities of GeoStan and the data elements that GeoStan returns, it is important to understand these underlying file types. PBBI suggests you familiarize yourself with the content of these source files.

For information about TIGER files, contact the U.S. Census Bureau and request the TIGER Technical Documentation. Information regarding street network data from premium vendors is available at the vendor’s Web site. For information about the ZIP+4 Directory file, contact the USPS National Customer Support Center at 1-800-238-3150 and request the Address Information System Products Technical Guide.

Supported languages

GeoStan includes the following API libraries, click the linked item next to each one for more information:

- C - Chapter 3, "Defining C functions".
- COBOL - Chapter 4, "Defining COBOL procedures".
- Java - Chapter 5, "Java quick reference".
- .Net - Chapter 6, ".Net quick reference".

Installation requirements

You can install GeoStan on platforms with thread-safe and non thread-safe versions as listed in the following table.

Windows OS	UNIX 32-bit and 64-bit OS	z/OS OS
Windows 32-bit Windows 64-bit	IBM AIX 32-bit IBM AIX 64-bit	IBM z/OS
	HP-UX 32-bit HP-UX 64-bit	
	Sun Solaris 32-bit Sun Solaris 64-bit	
	Solaris Forte 32- bit	
	SuSE Linux	
	Red Hat Enterprise 32-bit Red Hat Enterprise 64-bit	

NOTE: To see a detailed list of the officially supported OS versions, see the PBBI Centrus Products Supported Platforms document available at http://www.g1.com/support.

PBBI distributes GeoStan as Windows DLL files and as UNIX C libraries. The following table lists the filenames for the Windows DLL and UNIX libraries.

Platform	Filename
UNIX Thread-Safe	`libgeostanMT.a` - Object library. `libgeostanMT.so` - Shared object library (DLL).
UNIX Non Thread-Safe	`libgeostan.a` - Object library. `libgeostan.so` - Shared object library (DLL).
Windows 32-bit Thread-Safe Windows 64-bit Thread-Safe	`GeoStn32MT.dll` `GeoStn64MT.dll`
Windows 32-bit Non Thread-Safe Windows 64-bit Non Thread-Safe	`GeoStn32.dll` `GeoStn64.dll`
NOTE: See the *Release Notes for GeoStan OS/390* for a complete listing of OS/390 filenames.	

The following table lists the APIs supported on each operating system.

	C	COBOL	Java	.Net
Windows	✓		✓	✓
UNIX	✓		✓	
z/OS	✓	✓		

Installing GeoStan

You can find complete instructions on installing GeoStan in the Centrus Product Suite Installation Guide for your platform.

To install GeoStan, install the software prior to installing the data. If you have purchased additional data products, such as DPV or the point-level data option, you may have more than one data set to install. Your data may reside on tapes, CDs, or DVDs depending on your operating system and your license agreement with PBBI. See the related release notes for specific installation instructions for your license.

After you have installed the software and data, install Adobe Acrobat Reader. To view and print the product manuals, you must have Acrobat Reader installed on your machine. Visit www.adobe.com to download the appropriate reader.

Configuring GeoStan

The following sections provide information on configuring GeoStan for different platforms and include the following topics:

- Understanding the symbolic constants of GeoStan
- Using GeoStan on Windows
- Using GeoStan on UNIX
- Using GeoStan on z/OS

Understanding the symbolic constants of GeoStan

The C header file, `geostan.h`, contains all symbolic constants used or returned by the GeoStan functions. Pitney Bowes Business Insight recommends you review this file before using any GeoStan functions. `geostan.h` is the proper include file for all C environments.

For each return value, GeoStan defines the maximum length of the value. Symbolic constants use the naming convention `<<NAME>>_LENGTH`. For example the enum `GS_ADDRLINE` returns the input or output address line. The maximum length of an address line is 61. Therefore, the symbolic constant is `GS_ADDRLINE_LENGTH` and is defined as 61.

Using GeoStan on Windows

When using GeoStan on Windows, be sure to define the `_WINDOWS` symbol. To successfully use the 32-bit version of GeoStan in Windows NT and XP, your project must define one or both of the following: `__NT__` and `_WIN32`. You are not warned with compile or link errors if you fail to define these symbols; however, your application will not function properly, and can result in insufficient stack space. Microsoft compilers typically define these symbols for you, but Pitney Bowes Business Insight strongly recommends that you define them in your project for completeness.

Using GeoStan on UNIX

Although PBBI builds GeoStan using the latest technology available, some UNIX developers may encounter difficulties caused by compiler - specific problems or system dependencies. This section explores some peculiarities of the compiler used in the UNIX environment.

NOTE: When using GeoStan in UNIX, be sure to define the symbol `__UNIX`. This applies to all UNIX platforms that GeoStan supports.

Object dependencies and standard link libraries

Because UNIX kernels typically have few dependencies and do not require all the standard development environment libraries and files, some standard link libraries (such as the C++ class library) are not included with the “cc” system configuration compiler. GeoStan requires certain standard libraries that are available on every UNIX development platform, but which the application may not find unless you install the appropriate system option.

NOTE: Some systems may include these libraries as part of a development option, or with an additional package such as the optimizing compiler. Contact your OS vendor for more information about the standard libraries.

To determine what system libraries are needed, run the ″ldd″ command (http://linuxmanpages.com/man1/ldd.1.php).

For example ldd libgeostan.so.22.01returns the following on RH Linux:

```
linux-gate.so.1 => (0xffffe000)
libdl.so.2 => /lib/libdl.so.2 (0xf7d1d000)
libstdc++.so.6 => /usr/lib/libstdc++.so.6 (0xf7c53000)
libm.so.6 => /lib/tls/libm.so.6 (0xf7c30000)
libgcc_s.so.1 => /lib/libgcc_s.so.1 (0xf7c28000)
libc.so.6 => /lib/tls/libc.so.6 (0xf7aff000)
/lib/ld-linux.so.2 (0x56555000)
```

The standard libraries used by GeoStan are:

C++ class library	`libC.so`, `libC.sl`, `libC.a`, etc.
Math library	`libm.a`, `libm.so`, `libm.sl`, etc.
C library	`libc.a`, `libc.so`, `libc.sl`, etc.

See the sample make files included as part of the installation for the libraries needed for your system.

Installing libraries and shared libraries

The UNIX system can recognize development libraries when placed in the system environment variables library path. On most systems, the path for this variable is `/lib:/usr/lib`. Some systems may include additional library directories needed for specific applications or configurations. When a library file has a name such as `libname.a`, `libname.so`, or `libname.sl(HPUX only)`, the UNIX system recognizes the library by default as long as it is in the library path.

The GeoStan library is distributed with the names `libgeostan.so` or `libgeostan.sl(HPUX only)`. To guarantee your system can locate the needed GeoStan libraries, you should include the full path to your centrus/geostan directory in your systems library search path. For example:

AIX	`LIBPATH=/usr/centrus/geostan:$LIBPATH ; export LIBPATH`
Solaris and Linux	`LD_LIBRARY_PATH=/usr/centrus/geostan:$LD_LIBRARY_PATH ; export LD_LIBRARY_PATH`
HP-UX	`SHLIB_PATH=/usr/centrus/geostan:$SHLIB_PATH ; export SHLIB_PATH`

Using GeoStan on z/OS

When using GeoStan in z/OS, refer to the installation notes provided with the tapes and use the following data definitions (DD names):

DD Name	File Type	Y – Required for GEOCODER C – Required for CASS N – Optional	Usage
INFILE	Input	Y	GEOCODER processing
FORMAT	Input	Y	GEOCODER processing
OUTFILE	Output	Y	Written by GEOCODER
AUDIT	Input	N	Written by GEOCODER for audit reports
GEORPT	Input	N	Accuracy/speed report written by GEOCODER
CASSRPT	Input	C	CASS certification report
CASSTMP	Input	C	CASS template
MVSMSG	Input	N	MVS error messages
CITYDIR	Input	Y	City lookup file ctyst.dir
CBSACDIR	Input	N	CBSA lookup file cbsac.dir
PARSDIR	Input	Y	Parsing tables file parse.dir
GSDFILE[a]	Input	Y	First GSD file loaded by GeoStan. If you are using multiple GSD files, you must use this DD Name for the first file.
GSDFIL*xx*[a]	Input	Y	Additional GSD files loaded by GeoStan. The first file must use GSDFILE for correct processing. Additional files must be in sequential order (01, 02, 03..). If the files are not in sequential order, GeoStan will not read the files after the first blank GSDFIL*xx* file.
Z9FILE	Input	N	ZIP + 4 centroid file us.z9

DD Name	File Type	Y – Required for GEOCODER C – Required for CASS N – Optional	Usage
GSLFILE	Input	C	eLOT and Z4Change data file us.gsl
GSUFILE	Input	C	ZIP9 index file gsu
EWSFILE	Input	C	Valid, unexpired EWS file ews.txt; see Appendix E, "CASS certification."
ZIPMOVE	Input	C	ZIPMove data file us.gsz
CPNTEAPN	Input	N	Centrus APN
CPNT01PN	Input	N	Centrus APN #2
CPNT02PN	Input	N	Centrus APN #3
CPNTSELV	Input	N	Centrus Elevation
GSDFIL01	Input	N	Centrus Points
LLKUDB	Input	C	CASS processing
LLKSUD[b]	Input	C	CASS processing
REVALIAS	Input	C	CASS processing
LACFPRPT[b]	Input	N	LACS^{Link} processing
SEEDLOG[b, c]	Input	N	LACS^{Link} and DPV processing
DPVHDB	Input	C	CASS processing
DPVSUD[c]	Input	C	CASS processing
DPVFPRPT	Input	N	DPV processing
CEEDUMP	Output	N	Data set for error messages generated by Language Environment Dump Services. Must be a sequential data set and must be allocated to SYSOUT, a terminal, or a unit record device; or the data set must have the attributes RECFM=VBA, LRECL=125, and BLKSIZE=882.
SYSPRINT	Output	N	Data set for listings and diagnostics from the user program. LRECL=137, RECFM=VBA, and BLKSIZE=882.
SYSTERM	Output	N	Alternate output for some C++ programs. Not required unless SYSPRINT is missing.
SYSERR	Output	N	Alternate output for some C++ programs. Not required unless SYSPRINT and SYSTERM are missing.
CITYDIR	Output	N	CICS platform specific.
CITYCCI	Output	N	CICS platform specific.

DD Name	File Type	Y – Required for GEOCODER C – Required for CASS N – Optional	Usage
CITYZPI	Output	N	CICS platform specific.
CITYCTI	Output	N	CICS platform specific.
CITYCNI	Output	N	CICS platform specific.
AUXFILE	Output	N	Auxiliary file processing.
NSTDFILE	Output	N	Non-standard report file.

[a] Needed for the GeoStan Point-Level Data Option.

[b] Needed for LACS/Link processing.

[c] Needed for DPV processing.

Initializing GeoStan

All GeoStan functions use a common internal data structure you initialize with a call to the initialization function. This function returns an ID, which GeoStan passes to every other GeoStan function. Upon completion of the geocoding session, a call to the termination function frees the memory allocated by the initialization function and closes all files used by GeoStan.

Each GeoStan function returns either a status code or a pointer. There is a symbolic constant for every status code returned by a function. If the function returns a pointer, there is usually only one possible error indicated by the return of NULL. See the individual language chapters for possible status codes for each function.

NOTE: GeoStan includes additional initialization functions for DPV and LACS^{Link} processing. You must call these functions if you have installed the DPV and LACS^{Link} data for GeoStan to process your input using these options. That is, unless, you are using the property driven API then you only make one initialization call. See Appendix F, “USPS Link products” for more information on DPV and LACS^{Link}.

CHAPTER 2

Using GeoStan

This chapter provides the following information on the concepts used by GeoStan:

- Getting started
- Finding an address
- Using GeoStan match modes
- Address matching
- Using enhanced search options
- Using geocoding features
- Using USPS matching enhancements
- Elements available via GeoStan
- Using custom supplemental data sets
- Using reverse geocoding
- Using reverse APN matching

Getting started

Detailed below is an introduction to running GeoStan. See the following topics for more information:

- Get and set functions
- Basic use of GeoStan
- Optimization issues

Get and set functions

GeoStan provides a variety of functions to load and retrieve data from the internal data structures of GeoStan. The Set functions require a “Switch” and a “Value” argument. The “Switch” argument is a symbolic constant that identifies the data element to be stored. The “Value” argument is the actual data string to be stored. The Get functions require the same “Switch” arguments used by the Set function, a pointer to a buffer to be filled with the returned data, and an argument specifying the maximum length of data that GeoStan should return.

Basic use of GeoStan

The GeoStan functions all use a common internal data structure that is initialized with a call to GsInitWithProps. This function returns a GsId that is then passed to every other GeoStan function. Upon completion of the geocoding session, a call to GsTerm frees the memory allocated by GsInitWithProps and closes all files used by GeoStan.

Each GeoStan function returns either a status code or a pointer. There is a symbolic constant for every status code returned by a function. If the function returns a pointer, there is usually only one possible error that would be indicated by the return of NULL. Status codes detail all possible status codes for each function.

NOTE: Calling GsInitWithProps with the appropriate DPV initialization properties set initializes DPV if you have installed the DPV data. Calling GsInitWithProps with the appropriate LACS^{Link} properties set initializes LACSLink if you have installed the LACSLink data. We recommend that you initialize GeoStan with DPV and/or LACS^{Link} at the same time with a single GsInitWithProps call. Otherwise, you can initialize either features after GeoStan is initialized by calling GsInitWithProps. DPV, Suite^{Link}, and LACS^{Link} are optional when processing records in CASS mode. However, you must use DPV, Suite^{Link}, and LACS^{Link} data for CASS certification.

Basic matching

Once GeoStan initializes, there are many options available through the GeoStan functions to allow diverse implementations of matching strategies. A basic approach is outlined below:

1 Construct initialization properties.

2 Initialize GeoStan using GsInitWithProps.

3 Construct find properties.

4 Load all components of an address using GsDataSet.

5 Ask GeoStan to find the address by calling GsFindWithProps.

6 Retrieve the match information by calling GsDataGet.

7 Clear the Gs data buffer by calling GsClear.

8 Repeat steps 2 through 5 for each address to be geocoded.

9 Close GeoStan using GsTerm.

This approach does not include any “advanced” features such as query or match candidate resolution. In reality, however, the burden of development is mostly in creating the user interface; query or match candidate logic can be implemented quickly and easily.

Samples

Refer to the sample code provided in the installation directory as follows:

Windows

```
\Geolib\Ms_c
```

UNIX

```
<install directory>/samples
```

where <install directory> is the user-specified directory to which the tar file was extracted.

Optimization issues

PBBI designed and developed the GeoStan data format and functions to allow you to directly access from CD or installation tape. However this design is most effective when you process records in State plus City order, or in ZIP Code order.

Finding an address

GeoStan processing modes and options are used to determine how address searches are performed and to specify the placement of the geocodes returned.

In some cases, GeoStan cannot determine which match candidate is the best match. In these instances, GeoStan returns a status indicating multiple candidates for a match for you to determine which is the appropriate candidate.

For more information on matching addresses, see the following topics:

- Entering an address
- Understanding the typical address
- GeoStan also recognizes the elements on individual input lines when matching in multiline mode.
- Search area designation
- Hyphenated addresses

- Parsing city names

Entering an address

For optimal performance and match rates, input an address free of misspellings and with all known address components. If you are using GeoStan for mailable address standardization, to obtain the highest match rate, verify that your input follows USPS standards.

GeoStan accepts many input fields that make up an address. Ensure that your input address contains a street address line and a last line, or a single line with both address and last line elements. These inputs help ensure that GeoStan can accurately identify a location area in which to search for a match candidate based on the city, state, and ZIP Code.

GeoStan also accepts a street address line with individual city, state, and ZIP Code fields instead of a last line. Use this type of input configuration if you are confident that you have a complete address list; free of misspellings and incomplete addresses.

If you are using GeoStan for address standardization, GeoStan requires addresses to have at least a street name, and either a city and state or a ZIP Code to obtain a match.

If you are using GeoStan to obtain geocoding information, GeoStan requires that you enter a ZIP Code and ZIP + 4 to receive geocoding information.

Understanding the typical address

The typical address consists of the following elements:

Element	Description
Firm Name	*Optional.* You can configure GeoStan to match the input firm name, or business name, rather than an address.
Street Address Line	Contains the street address. GeoStan recognizes the following types of street addresses: • PO Box, such as PO Box 100 • Rural Route or Highway Contract address • General Delivery (stated in the address line) • Street, such as 4750 WALNUT ST • Highrise that contains unit information, such as 4750 WALNUT ST STE 200 • Building name where it matches the firm name in the ZIP+4 database. See enhanced search options. These types of street addresses can have all or some of the following elements: • House Number • Predirectional, such as N, NE, and S • Street Name • Street Type, such as Ave, Blvd and Pkwy • Postdirectional, such as N, NE, and S • Unit Type, such as Apt, Ste, and Bldg • Unit Number • Private Mail Box
Last Line	Contains the location of the address. GeoStan can recognize the following last line elements: • City • State • ZIP Code • ZIP + 4 Code

GeoStan also recognizes the elements on individual input lines when matching in multiline mode.

Understanding how GeoStan processes addresses

GeoStan processes addresses in the following order:

1 Parses the address elements.

When you load data in GeoStan, GeoStan parses the data into single elements. Parsing occurs on data in the order in which you load the data. Even if a valid address is missing an element, GeoStan can find a match. In addition, some elements, such as pre-directionals, may not be critical elements of some addresses. By comparing an address as input against all known addresses in a search area, GeoStan can usually determine if any of these elements are missing or incorrect.

2 Determines the search area.

GeoStan uses the last line elements of an address to determine a search area. You can specify if you want the search area to be based on a finance area or on an area defined by the city, state, and ZIP Code. If the city and state are not in the ZIP Code, GeoStan performs separate searches for the ZIP Code and city.

NOTE: A finance area is a collection of ZIP Codes within a contiguous geographic region.

3 Finds possible matches within the search area.

Once GeoStan has determined the search area, it tries to match the elements from the street address line to the records in the standardized data files and completes the following:

- Checks input address ranges for missing or misplaced hyphens, and alpha-numeric ranges for proper sequence.
- Searches for any misspellings and standard abbreviations. For example, GeoStan can recognize *Mane* for *Main* and *KC* for *Kansas City*.
- Searches for any street name alias matches to the USPS and Spatial data (TIGER, GDT, User Dictionaries, Navteq and Points). For example, GeoStan recognizes that in Boulder, “CO Highway 36” is also known as “28th Street”.

NOTE: The USPS does not consider matches to data that they did not create and these are not considered valid addresses for postal delivery. Therefore, GeoStan does not match to any spatial alias data, User Dictionaries, or auxiliary files, when processing in CASS mode

- Searches for addresses that contain a house number and unit number as the same element. For example, GeoStan recognizes the input 4750-200 Walnut Street and performs recombination to output 4750 WALNUT ST STE 200
- Searches for any USPS recognized firm names for additional match verification.
- Searches for street intersection matches. GeoStan recognizes `and, &, &&, at,` AT, and `@` in the input address for an intersection match.

NOTE: The USPS does not consider intersections valid addresses for postal delivery. Therefore, GeoStan does not match intersections when processing in CASS mode

4 Scores each possible match against the parsed input.

NOTE: If you need to develop your own scoring system, use the Find functions. See Appendix B, “Extracting data from GSD files” for more information

GeoStan compares each element in the input address to the corresponding element in the match candidates, and assigns a confidence level. GeoStan weighs the confidence level for all of the elements within a match candidate, and assigns a final score.

NOTE: GeoStan uses a penalty scoring system. If an element does not exactly match an element in the match candidate, GeoStan adds a penalty to the score of the match candidate. Therefore, scores with lower numbers are better matches.

5 Determines the match.

GeoStan prioritizes each match candidate based on the assigned confidence score and returns as a match the candidate that has the lowest score.

The match mode you choose determines the range that GeoStan allows for a match. GeoStan only returns a match if the score of the target address falls within the range designated by the selected match mode.

In some cases, more than one match candidate may have the lowest score. In this instance, GeoStan cannot always determine on its own which record is correct, and returns a status indicating a match candidate. You can retrieve the list of possible matches, present them to the user, and allow the user to select one of the choices. Alternatively, you can resolve the matches internally in your application.

NOTE: If you have installed DPV, GeoStan automatically attempts to resolve a match candidate using DPV. For more information see Appendix F, "USPS Link products."

Along with a standardized address, GeoStan also returns the following:

NOTE: For more information on match, location, and result codes, see Appendix D, "Status codes."

GeocodeLongitude and latitude for the address

Match codeInformation about the match of the input address to the reference data

Location codePrecision level of the geocode

Result codeGeoStan returns a result code for every record it attempts to match.

Street ParityThe side of the street on which the match resides.

Range ParityThe parity of the house number in the range: even, odd, or both.
NOTE: GeoStan does not return parity when processing in relaxed mode.

Search area designation

To assist in finding a match when the input address contains limited or inaccurate city or ZIP Code information, GS_FIND_SEARCH_AREA may be set using one of the following options:

- 0 - GS_FIND_SEARCH_AREA_CITY - Searches the specified city.
- 1 - GS_FIND_SEARCH_AREA_FINANCE - Searches the entire Finance Area for possible streets.

  NOTE: This option has no effect when performing a ZIP centroid match.
- 2 - GS_FIND_SEARCH_AREA_EXPANDED - This value effectively has two options that can be set:
  - Allows the setting of the radius in miles (up to 99) around which your record lies. The default radius setting is 25 miles.
  - Allows for limiting the search to the state. The default setting is True.

  When the value of GS_FIND_SEARCH AREA is set to GS_FIND_SEARCH_AREA_EXPANDED, you can set two other properties that tell GeoStan how far to expand the search. To set the radius, use the property, GS_FIND_EXPANDED_SEARCH_RADIUS and specify the number of feet to search around the area defined by the Last-line lookup information. To limit the search to the state, even if the radius would extend past state borders, set the property GS_FIND_EXPND_SRCH_LIM_TO_STATE to True.

These expanded search options allow for searching within a city, its finance areas (groups of ZIP Codes as defined by the USPS), or by specifying the radius searched in miles.

Hyphenated addresses

GeoStan handles hyphens in address ranges. Address ranges are checked for missing or misplaced hyphens. Alpha-numeric ranges are checked for proper sequence.

If a house number is incorrectly entered with a hyphen, the number is first concatenated. If no match is resolved, GeoStan tries the section of the house number following the hyphen as a unit number. See Matching address ranges for special processing of hyphens.

Parsing city names

GeoStan correctly parses out the separate elements of City, State, and ZIP Code from the data entered under GS_LASTLINE. When the city element is preparsed using GS_CITY, GeoStan parsing is bypassed. For example, “Mt Pilot” entered into GS_CITY would not be parsed to look for “Mount Pilot.” Best results are obtained keeping all three elements under GS_LASTLINE.

Using GeoStan match modes

Match modes determine the leniency used to make a match between your input and the Centrus data. Select a match mode based on the quality of your input and your desired output. For example, if you have an input database that is prone to errors, you may want to select the relaxed match mode.

The following match modes are available:

Mode	Description
Exact	Requires a very tight match. This restrictive mode generates the fewest match candidates, which decreases the processing time. When using this mode, ensure that your input is very clean; free of misspellings and incomplete addresses.
Close	*Default.* Requires a close match and generates a moderate number of match candidates.
CASS	Imposes additional rules to ensure compliance with the USPS CASS regulations. Use this mode to standardize your input for mailing. This mode generates a large number of match candidates. CASS mode deviates from the other modes in its processing. This mode does not perform intersection, building name, or spatial alias (TIGER and GDT street name alias) or matches to User Dictionaries or auxiliary files. It does not match to candidates from data sources that do not have USPS equivalent records. This mode recognizes and parses two unit numbers on the same address line, for example a building and unit number.
Relaxed	Allows a loose match and generates the most match candidates, which increases the processing time and results in more multiple matches. Use this mode if you are not confident that your input is clean; free of misspellings and incomplete addresses. This is the only mode that does not respect the street parity when making an address match.
Custom	Allows applications to specify individual "must match" field matching rules for address number, addressline, city, zipcode, state.

Address matching

GeoStan provides support for several variations in the format of an input address. This section contains information on the following topics:

- Multi-line matching
- Two-line matching
- Matching dual addresses
- Single line address matching
- City name matching
- Address elements scoring
- Match rate and performance

- Analyzing a match

Multi-line matching

Multi-line matching allows you to pass any six lines of address information to GeoStan to extract and standardize the address. To specify multi-line matching, use the following:

C	Set the *`switch`* parameter in **`GsDataSet`** to any of the `GS_LINE`*`X`* enums.
COBOL	Set the *`GSOPTIONS`* argument in **`GSDATSET`** to any of the `GS-LINE`*`X`* variables.
JAVA	Set the *`element`* argument in **`setData`** to any of the `LINE`*`X`* fields of **`GeoStanBase`**.

NOTE: Do not mix the two-line address input enums with the multi-line address input enums. GeoStan executes a two-line search if you use the two-line address input enums and ignores the multi-line enums.

After processing GeoStan outputs address data in the normal address output fields. You can retrieve extraneous information from the lines that *did not* contain valid address data by using the appropriate multi-line enums for your language. GeoStan returns lines that contain valid address data as blank values (empty strings).

The following example uses the C language.

Input values		Output values	
John Doe	(GS_LINE1)	John Doe	(GS_LINE1)
123 Main St	(GS_LINE2)	*blank*	(GS_LINE2)
Deliver around back	(GS_LINE3)	Deliver around back	(GS_LINE3)
New York, NY	(GS_LINE4)	*blank*	(GS_LINE4)
		blank	(GS_LINE5)
		blank	(GS_LINE6)

GeoStan takes the valid address components from `GS_LINE2` and `GS_LINE4` and outputs the invalid values in `GS_LINE1` and `GS_LINE3`.

NOTE: GeoStan does not perform dual address matching on multi-line addresses.

Preferences for multi-line addresses

If GeoStan finds a match, it returns the matched address as the first line (for example, `GS_ADDRLINE`) and returns the extraneous address information in the appropriate multi-line address line (for example, `GS_LINE1`).

If you do not specify a preference for a P.O. Box or a street address, GeoStan uses the value in the bottom-most address as the input address. For example, if the address is:

John Doe
123 Main St
P.O. Box 24
New York, NY

GeoStan returns P.O. Box 24.

For more information on setting these preferences see Specifying a preference for street name or P.O. Box.

Two-line matching

Two-line address matching allows you to pass two different address lines to GeoStan. GeoStan scans these two lines and extracts and standardizes a two-line address.

To specify two-line matching, use the following:

C	Use the `GS_ADDRLINE` and `GS_ADDR2` enums in the *`fswitch`* parameter of **`GsDataSet`**.
COBOL	Use the `GS-ADDRLINE` and `GS-ADDR2` variables in the *`GSOPTIONS`* parameter of **`GSDATSET`**
JAVA	Use the `ADDRLINE` and `ADDR2` fields of **`GeoStanBase`** in the *`element`* argument in **`setData`**.

NOTE: Do not mix the two-line address input enums with the multi-line address input enums. GeoStan executes a two-line search if you use the two-line address input enums and ignores the multi-line enums.

If a unit designator is present on one address line and a valid address appears on the other line, GeoStan appends the unit information to the address.

Preferences for two-line addresses

If GeoStan finds a match, it returns the matched address as the first line (for example, `GS_ADDRLINE`) and returns the extraneous address line as the second address (for example, `GS_ADDR2`).

If you do not specify a preference for a P.O. Box or a street address, GeoStan uses the value in the first line of the address as the preferred input. For more information on setting these preferences see Specifying a preference for street name or P.O. Box.

Matching dual addresses

GeoStan can process input that contains two addresses on the same address line. For example, GeoStan can process the following input address:

3138 HWY 371 PO BOX 120
PRESCOTT AR 71857

GeoStan does not recognize dual addresses where the two addresses are both street addresses. For example, GeoStan does NOT recognize *135 Main St 4750 Walnut St Ste 200*. GeoStan does recognize dual addresses where the two addresses are the same type of address but are not street addresses. For example, GeoStan does recognize *PO BOX 12 PO BOX 2000*.

After GeoStan parses the dual address, it searches for a match. GeoStan determines which address has preference for a match based on the processing mode. In CASS mode, GeoStan ignores the prefer PO Box and prefer street options, and attempts to find a match based on the following order: PO Box, Street, Rural Route, and General Delivery. In Relaxed mode, GeoStan recognizes the prefer PO Box and prefer street options.

NOTE: GeoStan does not perform dual address processing in Exact, Custom, and Close modes. GeoStan does not perform dual address processing on multi-line addresses.

Single line address matching

Single line address matching allows you to pass a single line of address information where the street address and lastline (city, state, and ZIP Code) are contained on one text line instead of the traditional two-line address. For example, GeoStan can process the following input address:

4750 WALNUT ST BOULDER
CO 80301

GeoStan does not require an address component to execute geocoding. The minimum information required in the lastline component for geocoding is a ZIP Code or a valid city and state combination. If an address component is included in the input, it must precede the lastline component.

NOTE: GeoStan does not support single line addresses in the CASS or Exact match modes.

To specify single line matching, use the following:

C	Use the `GS_ADDRLINE` enum in the *`fswitch`* parameter of **`GsDataSet`**.
COBOL	Use the `GS-ADDRLINE` variable in the *`GSOPTIONS`* parameter of **`GSDATSET`**
JAVA	Use the `ADDRLINE` fields of **`GeoStanBase`** in the *`element`* argument in **`setData`**.

City name matching

Extended city name matching allows GeoStan to recognize a large number of city abbreviations.

Address elements scoring

As GeoStan encounters possible matches, a list of those matches is kept internally with an accuracy rating. Each element in the input address is compared to the corresponding element in the target address and assigned a confidence level. The confidence levels are weighted and the sum yields a final score. The match with the lowest overall score (assuming the rating is unique and the match meets other criteria) is returned as the proper match. A match with a rating of 0 exactly matches all scored elements.

The following score tables are available for each match mode:

- GS_MODE_EXACT requires an exact name match. Generates the fewest number of possibles to search.
- GS_MODE_CLOSE requires a very close name match. Generates a moderate number of possibles to search (default).
- GS_MODE_RELAX requires a close name match. Generates the largest number of possibles to search.
- GS_MODE_CASS requires a close name match. Generates a moderate number of possibles to search. This setting imposes additional rules to ensure compliance with the USPS regulations for CASS software. For example, this mode disables intersection matching, and matching to the User Dictionary matches for standardization.
- GS_MODE_CUSTOM allows applications to specify individual "must match" field matching rules for address number, addressline, city, ZIP Code, state.

These tables have been tested on tens of millions of addresses. Should you need to modify a score table for a specialized application, call PBBI for additional details on this complex subject. However, you are likely better off using the Find First and Find Next functionality and developing your own scoring system.

Match rate and performance

Computational time varies according to the way the application loads the address record into GeoStan. The following table gives some representative values that might help you assess the processing time and success rate for three different methods of loading addresses with GsDataSet. Assume a scenario in which a job takes 100 seconds in the best of conditions, and further assume that 100 percent matching is possible.

Address	Field names	Time	Rate
Single line	GS_ADDRLINE, GS_LASTLINE	100 sec.	100%
Two-line	GS_ADDRLINE, GS_ADDR2, GS_LASTLINE	120 sec.	100%
Multiple-line	GS_LINE1 through GS_LINE6	300 sec.	90-95%

Analyzing a match

GsDataGet can return a status code that defines the match. The match ratings are described in Status codes. This code defines exactly what elements of an input address were modified, or the reason a match was not made. The return value for the find method yields a high-level status match process

Address matching sample code

This program is written to the GeoStan C property driven API specification.

```
#define GEOSTAN_PROPERTIES

/* Geostan include files */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "geostan.h"

int main (int argc, char **argv)
{
   GsId gs;
   int iNumMatched = 0;
   int iNumFailed = 0;
   intl count;
   char buffer[GS_MAX_STR_LEN];
   char buffer2[GS_MAX_STR_LEN];
   char address[GS_ADDRLINE_LENGTH],
        lastline[GS_LASTLINE_LENGTH],
        longitude[GS_LON_LENGTH],
        latitude[GS_LAT_LENGTH],
        matchcode[GS_MATCH_CODE_LENGTH],
        loccode[GS_LOC_CODE_LENGTH];

   /* Add addresses to this N-row, 2-column array to cause them to
    * be processed */
   char* addresses[][2] =
      {{"PO BOX 880066",              "SAN FRANCISCO CA 94188"},
       {"4750 WALNUT ST STE 200",     "BOULDER CO"},
       {"275 SKYLANE DR",             "ERIE CO 80516"},
       {"1110 W 7th St",              "Loveland CO"},
       {"2604 GRANDVIEW",             "PLANO TX 75075"}};
   int i;
   int numAddresses = sizeof(addresses) / sizeof(*addresses);

   /* property lists for initialization and finding */
   qbool bVal;
   long lVal;
   PropList initProps;
   PropList statusProps;
   PropList findProps;
   GsPropListCreate( &initProps, GS_INIT_PROP_LIST_TYPE );
   GsPropListCreate( &statusProps, GS_STATUS_PROP_LIST_TYPE );
   GsPropListCreate( &findProps, GS_FIND_PROP_LIST_TYPE );

   /* This version strictly uses properties only */
   /* First the GeoStan properties */
   GsPropSetStr(  &initProps, GS_INIT_LICFILENAME,
      "S:\\Geostan\\Java\\test\\data\\licenses\\all.lic");
   GsPropSetStr(  &initProps, GS_INIT_DATAPATH,
      "\\\\cog1nas1\\current\\DVDGDT" );
   GsPropSetStr(  &initProps, GS_INIT_Z4FILE,
      "\\\\cog1nas1\\current\\DVDGDT\\us.z9" );
  GsPropSetLong( &initProps, GS_INIT_PASSWORD, 87654312 );
  GsPropSetLong( &initProps, GS_INIT_GSVERSION, GS_GEOSTAN_VERSION );
   GsPropSetBool( &initProps, GS_INIT_OPTIONS_ADDR_CODE, TRUE );
   GsPropSetBool( &initProps, GS_INIT_OPTIONS_Z9_CODE, TRUE );
   GsPropSetBool( &initProps, GS_INIT_OPTIONS_SPATIAL_QUERY, TRUE );
   GsPropSetBool( &initProps, GS_INIT_OPTIONS_APN_CODE, TRUE );
   GsPropSetBool( &initProps, GS_INIT_OPTIONS_ELEVATION_CODE, TRUE );
  GsPropSetLong( &initProps, GS_INIT_CACHESIZE, 0 );

   /* DPV properties */
   GsPropSetBool( &initProps, GS_INIT_DPV, TRUE );
   GsPropSetLong( &initProps, GS_INIT_DPV_DATA_ACCESS,
      DPV_DATA_FULL_FILEIO );
   GsPropSetStr(  &initProps, GS_INIT_DPV_DIRECTORY,
      "\\\\cog1nas1\\Current\\LinkDPV" );
   GsPropSetStr(  &initProps, GS_INIT_DPV_SECURITYKEY,
      "C51F-02Z7-7906-Q9BG" );

   /* LACS properties */
  GsPropSetBool( &initProps, GS_INIT_LACSLINK, TRUE );
   GsPropSetStr(  &initProps, GS_INIT_LACSLINK_DIRECTORY,
      "\\\\cog1nas1\\Current\\LinkLACSLink" );
   GsPropSetStr(  &initProps, GS_INIT_LACSLINK_SECURITY_KEY,
      "740F-05L2-A791-51D7" );

   /* exercise the clearing of the lists */
```

```
GsPropListGetCount( &initProps, &count );
if ( count != 22 ) {
   printf( "Unexpected number of properties in initlist\n");
   exit( 1 );
}

/* initialize the GeoStan library */
gs = GsInitWithProps( &initProps, &statusProps );
GsPropListWrite( &initProps, "stdout", NULL, 0 );
GsPropListWrite( &statusProps, "stdout", NULL, 0 );

/* Demonstrate that the old GsInitStruct status bitfield is accessible. */
GsPropGetLong( &statusProps, GS_STATUS_STATUS_BITFIELD, &lVal );
if ( lVal & GS_FILE_LICENSE )
   printf("Loaded GeoStan license file.\n");
if ( lVal & GS_FILE_CITY_DIR )
   printf("Loaded GeoStan Ctyst.dir file.\n");

/* test GeoStan status */
if ( gs == 0 ) {
   printf("GeoStan did not initialize.\n");
   exit( 1 );
}

/* Now get properties from output status property list */
GsPropGetBool( &statusProps, GS_STATUS_FILE_LICENSE, &bVal );
if ( !bVal ) {
   printf("GeoStan license invalid.\n");
   exit( 1 );
}

GsPropGetBool( &statusProps, GS_STATUS_FILE_CITY_DIR, &bVal );
if ( !bVal ) {
   printf("GeoStan citystate file could not be loaded.\n");
   exit( 1 );
}

GsPropGetBool( &statusProps, GS_STATUS_FILE_EXPIRED, &bVal );
if ( bVal ) {
   printf("One or more GeoStan files have expired.\n");
   exit( 1 );
}

GsPropGetLong( &statusProps, GS_STATUS_FILE_MEMORY_USED, &lVal);
printf("File Memory Used so far <%lu> megabytes.\n", lVal );

GsPropGetBool( &statusProps, GS_STATUS_DPV_FILE_SECURITY, &bVal );
if ( !bVal ) {
   printf("DPV security key failed to verify.\n");
   exit( 1 );
}

GsPropGetBool( &statusProps, GS_STATUS_DPV_FILE_ALL, &bVal );
if ( !bVal ) {
   printf("DPV data failed to initialize.\n");
   exit( 1 );
}
```

```
GsPropGetBool( &statusProps, GS_STATUS_LACSLINK_FILE_SECUR, &bVal );
if ( !bVal ) {
   printf("LacLink security key failed to verify.\n");
   exit( 1 );
}

GsPropGetBool( &statusProps, GS_STATUS_LACSLINK_FILE_ALL, &bVal );
if ( !bVal ) {
   printf("LacsLink data failed to initialize.\n");
   exit( 1 );
}

if (gs == 0) {

   /* GeoStan did not initialize */
   fprintf(stderr, "GsInit failed\n" );
   GsPropListWrite( &statusProps, "c:\\temp\\out.txt", NULL, 0 );
   while (GsErrorHas(gs)) {
      GsErrorGetEx (gs, buffer, buffer2);
      printf ("%s\n%s\n", buffer, buffer2);
   }

} else {

   /* Set find flags using properties */
   GsPropSetLong( &findProps, GS_FIND_MATCH_MODE, GS_MODE_CASS );
   GsPropSetLong( &findProps, GS_FIND_STREET_OFFSET, 50 );
   GsPropSetLong( &findProps, GS_FIND_CENTERLINE_OFFSET, 0 );
   GsPropSetStr ( &findProps, GS_FIND_CLIENT_CRS, "NAD83" );
  GsPropSetBool( &findProps, GS_FIND_ADDRCODE, TRUE );
  GsPropSetBool( &findProps, GS_FIND_Z9_CODE, TRUE );
 GsPropSetBool( &findProps, GS_FIND_FINANCE_SEARCH, TRUE );
   GsPropSetBool( &findProps, GS_FIND_MIXED_CASE, TRUE );
   GsPropSetBool( &findProps, GS_FIND_Z_CODE, TRUE );
   GsPropSetBool( &findProps, GS_FIND_Z5_CODE, FALSE );
   GsPropListWrite( &findProps, "stdout", NULL, 0 );

   /* Look up the input addresses, using the same find options
    * each time */
   for ( i = 0; i < numAddresses; i++ ) {

      GsClear(gs);
      GsDataSet(gs, GS_ADDRLINE, addresses[i][0]);
      GsDataSet(gs, GS_LASTLINE, addresses[i][1]);

      /* this version uses the new style find */
      switch(GsFindWithProps( gs, &findProps )) {
         /* Get standardized address and geocode information */
         case GS_SUCCESS:

            /* Output the results */
            iNumMatched++;
            GsDataGet(gs, GS_OUTPUT, GS_ADDRLINE, address,
               sizeof(address));
            GsDataGet(gs, GS_OUTPUT, GS_LASTLINE, lastline,
               sizeof(lastline));
            GsDataGet(gs, GS_OUTPUT, GS_MATCH_CODE, matchcode,
               sizeof(matchcode));
               GsDataGet(gs, GS_OUTPUT, GS_LOC_CODE, loccode,
                  sizeof(loccode));
               GsDataGet(gs, GS_OUTPUT, GS_LON, longitude,
                  sizeof(longitude));
               GsDataGet(gs, GS_OUTPUT, GS_LAT, latitude, sizeof(latitude));
               printf("\n***********************************************\n");
              printf("match code: %s\nlocation code: %s\naddress: %s\nlastline: 
%s\nlongitude: %lf\nlatitude: %lf",
                  matchcode, loccode, address, lastline,
                  (double)(atof(longitude)/1000000.0),
                  (double)(atof(latitude)/1000000.0));
               printf(
                  "\n***********************************************\n\n");
               fflush(stdout);
               break;
            case GS_ERROR:/* Each error can be handled */
            case GS_ADDRESS_NOT_FOUND:
            case GS_ADDRESS_NOT_RESOLVED:
            case GS_LASTLINE_NOT_FOUND:
            default: /* or just the default */
               iNumFailed++;
               fprintf(stderr, "Error geocoding address.\n");
         }
      }

      /* Terminate GeoStan */
      GsTerm (gs);
   }

   /* Regardless of success or failure of GeoStan initialization, clean up */
   /* exercise the clearing of the lists */
   GsPropListReset( &initProps );
   GsPropListGetCount( &initProps, &count );
   if ( count != 1 ) {
      printf( "Unexpected number of properties in initlist\n");
      exit( 1 );
   }

   /* delete the lists */
   GsPropListDestroy( &initProps );
   GsPropListDestroy( &statusProps );
   GsPropListDestroy( &findProps );

   /* summary report */
   printf( "\nSummary of matches: \n");
   printf( "   Total addresses: %d\n", numAddresses );
   printf( "   Matched:         %d\n", iNumMatched );
   printf( "   Failed:          %d\n\n", iNumFailed );
   return 0;
}
```

Using enhanced search options

This section contains information on additional address concepts used by GeoStan, and includes the following topics:

- Specifying database search order
- Specifying a preference for street name or P.O. Box
- Using building name and firm matching
- Using correct last line
- Preferring a ZIP Code over a city
- Matching address ranges
- Understanding missing and wrong first letter
- Permitting relaxed address number matching
- Forcing an exact match on specific address elements

Specifying database search order

GeoStan is able to process addresses using multiple databases at the same time. This allows you to find the best possible match from a variety of data sources and types of data (point data as well as street segment data). GeoStan processes these multiple data sources using a default search order. When GeoStan matches an address exactly, it stops searching rather than continuing on to search in additional databases. This saves on processing time. When an exact match is not found, GeoStan continues searching all of the available data sources for candidate address matches. The candidates are then scored and the highest scoring match from all of the data sources is returned as the match. If multiple candidates receive an identical score, a multi-match is returned instead. The default search order for GeoStan is:

- Auxiliary files
- User Dictionaries
- Point GSD files
- Street GSD files

You may specifically set the database search order using the GS_FIND_DB_ORDER property. This property allows you to specifically set the order in which User Dictionary and GSD databases are searched. This property is designed for situations where you feel that the data in particular data sources is superior or should be preferred over other available sources.

NOTE: The GS_FIND_DB_ORDER can not be used to change the priority of auxiliary files. Auxiliary files always have the highest search priority.

GeoStan supports the creation and use of User Dictionaries based on your own source data. A User Dictionary can be used independently or as a supplement to the supplied GSD. For more information, see User Dictionary.

You can geocode using:

- A single User Dictionary or multiple User Dictionaries.
- Standard GSD.

- A combination of GSD and User Dictionaries

Specifying a preference for street name or P.O. Box

When using multi-line or two-line addresses, you can specify which input address GeoStan uses for matching: a P.O. Box or a street address. Use the following to specify your address preference:

C	Use **GS_PREFER_STREET** or **GS_PREFER_POBOX** in the find properties of **GsFindWithProps**.
COBOL	Use the **GS_PREFER_STREET** or **GS_PREFER_POBOX** in the find properties of **GSFINDWP.**

If GeoStan cannot match to the preferred address, it tries to match to the alternative address.

The following example uses the C Language.

If you set the GS_FIND_PREFER_POBOX find property used with GsFindWithProps, and the input address is

123 Main St	(GS_ADDRLINE)
P.O. Box 24	(GS_ADDR2)

GeoStan first attempts to match to P.O. Box 24. If GeoStan cannot find a match, it then attempts to match to 123 Main St.

NOTE: GeoStan ignores the address preference if processing in CASS mode.

If you do not specify a preference for a P.O. Box or street address, GeoStan attempts to match to the first address line it receives as input.

Using building name and firm matching

GeoStan can enhance standard address matching by matching to building and business names.

By default, GeoStan is able to match building names with unit numbers in the address line, the Chrysler building as an example:

Firm:
Address: 5001 Chrysler Bldg
Last Line: New York New York 10174

The returned information is the address of the Chrysler building. GeoStan returns a standardized address in place of the building name:

Firm:
Address: 405 Lexington Ave RM 5001
Last Line: New York, NY 10174-5002

Entering the White House, as an example, into the address line, the address for the White House returns in the address field:

Firm:
Address: White House
Last Line: Washington DC 20500

GeoStan returns the following address:

Firm:
Address: 1600 Pennsylvania Ave NW
Last Line: Washington DC 20500-0004

The ability to search by building name entered in the address line is controlled by setting the find property, GS_FIND_BUILDING_SEARCH, to True.

Entering a firm name in the Firm name field returns the address for the input firm in the address field:

Firm: White House
Address:
Last Line: Washington DC 20500

GeoStan returns the following address:

Firm: White House
Address: 1600 Pennsylvania Ave NW
Last Line: Washington DC 20500-0004

NOTE: GS_FIND_BUILDING_SEARCH is not available in Exact mode.

By modifying GS_FIND_ALTERNATE_LOOKUP, you can specify whether GeoStan searches for the following:

- GS_STREET_LOOKUP_ONLY (default) - Matches to the address line.

- GS_PREFER_STREET_LOOKUP - Matches to the address line, if a match is not made, then GeoStan matches to the Firm name line.
- GS_PREFER_FIRM_LOOKUP - Matches to the Firm name line, if a match is not made, then GeoStan matches to address line.

NOTE: Neither building nor firm name searches are available when processing in CASS mode.

Using correct last line

GS_FIND_CORRECT_LASTLINE, when set to True, corrects elements of the output last line, providing a good ZIP Code or close match on the soundex even if the address would not match or was non-existent.

The feature works when GS_FIND_ADDRCODE is True and the address does not match a candidate or when GS_FIND_Z_CODE is True and only last line information is input.

For example when GS_FIND_ADDRCODE = True

Address: 0 MAIN
LastLine: BOLDER CA 80301

Returns:
MATCH_CODE=E622
LASTLINE=BOULDER, CO 80301
CITY=BOULDER
STATE=CO
ZIP=80301

For example, GS_FIND_Z_CODE = True

Address:
LastLine: BOLDER CA 80301

Returns:
MATCH_CODE=Z6
LASTLINE=BOULDER, CO 80301
CITY=BOULDER
STATE=CO
ZIP=80301

The following elements are corrected:

- City correction - The city correction is based on input ZIP unless a match to city and state exists in which case both search areas are retained. The state input must be correct or spelled out correctly when no ZIP is input, location code, and coordinates based on input ZIP.

  - Input city is incorrect

    HAUDENVILLE MA 01039
    Returns LASTLINE=HAYDENVILLE, MA 01039
    LAT= 42396500   LON= -72689100

- State correction - State is abbreviated when spelled out correctly or corrected when a zip is present. There are some variations of state input which are recognized, ILL, ILLI, CAL, but not MASS. GeoStan does not consideration the abbreviation of the variation a change so ILL to IL is not identified as a change in the match code. In addition the output of the ZIP for a single ZIP city is not considered a change.
  - Input city exists

    Bronx NT, 10451
    Returns LASTLINE= BRONX, NY 10451

    Bronx NT
    Returns LASTLINE= BRONX NT
    No zip for correction

  - input city does not exist - preferred city for ZIP Code returned

    60515
    Returns LASTLINE=DOWNERS GROVE, IL 60515
    MATCH_CODE=E622

    ILLINOIS 60515   (or ILL 60515   or IL 60515   or ILLI 60515)
    Returns LASTLINE=DOWNERS GROVE, IL 60515
    MATCH_CODE=E222

- ZIP correction - ZIP is corrected only when a valid city/state is identified and has only one ZIP.
  - Exists on input

    HAUDENVILLE MA 01039
    Returns LASTLINE=HAYDENVILLE, MA 01039

  - incorrect on input - ZIP correction is not performed, both search areas are retained.

    HAUDENVILLE MA 01030
    Returns LASTLINE=HAYDENVILLE, MA 01030
    City and ZIP do not correspond

  - DOES NOT EXIST ON INPUT

    DOWNRS GROVE, IL
    Returns LASTLINE=DOWNERS GROVE, IL
    City with multiple zips

    LILSE IL
    Returns LASTLINE=LISLE, IL 60532
    City with a single zip

    DOWNERS GROVE LL

Returns LASTLINE=DOWNERS GROVE LL,
No ZIP for correction

DOWNRS GROVE, LL
Returns LASTLINE=DOWNRS GROVE, LL
No ZIP for correction

LILSE ILLINOIS
Returns LASTLINE= LISLE, IL 60532
Correct spelled out state

LISLE ILLINOS
Returns LASTLINE= LISLE ILLINOS
Incorrect spelled out state, no ZIP for correction

NOTE: For information on returned match codes see Correct last line match codes.

Preferring a ZIP Code over a city

The GS_FIND_PREFER_ZIP_OVER_CITY property allows a user to prefer candidates that match to input ZIP over candidates that match to input city. GeoStan creates multiple search areas when input city and ZIP do not correspond and this feature helps establish how the candidates should be scored.

NOTE: GeoStan ignores the ZIP over city preference if processing in CASS mode.

When there is more than one candidate in the input ZIP, some attempt is made to alleviate a multiple candidates for a match, or, where all the candidates get the same last line score. If a candidate also matches the city and/or preferred city, that candidate gets a better score. Matching to just preferred city is a lesser score than matching both.

Input Address: 24 GLEN HAVEN RD
Input Last Line: NEW HAVEN CT 06513

Found:
24 GLEN HAVEN RD
NEW HAVEN, CT 06513-1105

Possible candidates:

```
                                          score       pref.last line city
2 98    GLEN HAVEN RD  06513-1105 S   0.8100000  NEW HAVEN    * best match
24 98   GLEN HAVEN RD  06513-1248 S   2.2500000  EAST HAVEN
16 66   GLEN RD        06511-2825 S  46.3925000  NEW HAVEN
2 86    GLEN PKWY      06517-1415 S  52.1525000  HAMDEN
2 28    GLEN RD        06516-6509 S  52.1525000  WEST HAVEN
2 98    GLENHAM RD     06518-2517 S  75.0100000  HAMDEN
2 72    GLEN VIEW TER  06515-1519 S  97.0900000  NEW HAVEN
```

When there is more than one candidate, candidates matching the input ZIP score better.

Input Address: 301 BRYANT ST
Input Last Line: SAN FRANCISCO CA 94301

Found:
301 BRYANT ST
PALO ALTO, CA 94301-1408

Possible candidates:

```
                                                score     pref.last line city
301 301  BRYANT ST  94301-1408 S   3.2400000   PALO ALTO    * ZIP preferred match
301 305  BRYANT CT  94301-1401 S  28.2400000  PALO ALTO
300 306  BRYANT CT  94301-00ND T 35.6600000  PALO ALTO
301 301  BRYANT ST  94107-4167 H  39.6900000  SAN FRANCISCO * default match
301 319  BRYANT ST  94107-1406 S  39.6900000   SAN FRANCISCO
```

When there is more than one candidate, candidates that match the ZIP search area score better. The ZIP search area is the finance area for the input ZIP.

This example with SEARCH AREA set to FINANCE. With SEARCH AREA set to CITY the match is made to EAST AURORA 14052 as there is no candidate in 14166 the input ZIP.

Input Address: 100 MAIN ST
Input Last Line: EAST AURORA NY 14166

Found:
100 MAIN ST
DUNKIRK, NY 14048-1844

Possible candidates:

```
                                          score          pref.last line city
100 198  MAIN ST   14048-1844  S  3.2400000  DUNKIRK * same finance as input ZIP 14166
100 168  MAIN ST  14052-1633 S  39.6900000  EAST AURORA
```

This example with GS_FIND_SEARCH_AREA set to 0.

Input Address: 4200 arapahoe
Input Last Line: denver co 80301

Found:
4200 ARAPAHOE AVE
BOULDER, CO 80303-1164

Possible candidates:

```
                                                 score            pref.last line city
4200 4210 ARAPAHOE AVE  80303-1164 S   38.7400000   BOULDER *
                                                                * same city as input zip 80301
4200 4210  ARAPAHOE RD   80303-1164 S   40.7000000   BOULDER (A06)
```

```
4200 4298 E ARAPAHOE PL  80122-00ND T  62.0900000   LITTLETON
4200 4498 E ARAPAHOE RD  80122-00ND T  62.0900000  LITTLETON
4181 4499 E ARAPAHOE RD  80122-00ND T  68.3400000  LITTLETON
```

Matching address ranges

Some business locations are identified by address ranges. For example, a shopping plaza could be addressed as 10-12 Front St. This is how business mail is typically addressed to such a business location. These address ranges can be geocoded to the interpolated mid-point of the range.

Address ranges are different from hyphenated (dashed) addresses that occur in some metropolitan areas. For example, a hyphenated address in Queens County (New York City) could be 243-20 147 Ave. This represents a single residence (rather than an address range) and is geocoded as a single address. If a hyphenated address similar to this example returns as an exact match, then there is no attempt to address range match.

Address range matching is disabled by default and is an optional mode. To enable address range matching, use GS_FIND_ADDRESS_RANGE. Address range matching is not available in Exact or CASS™ modes, since an address range is not an actual, mailable USPS® address. The following fields are not returned by address range geocoding:

- ZIP+4® (in multiple segment cases)
- Delivery Point
- Check Digit
- Carrier Route
- Record Type
- Multi-Unit
- Default flag

Address Range matching capabilities and guidelines

Address Range matching works within the following guidelines:

- There must be two numbers separated by a hyphen.
- The first number must be lower than the second number.
- Both numbers must be of the same parity (odd or even) unless the address range itself has mixed odd and even addresses.
- Numbers can be on the same street segment or can be on two different segments. The segments do not have to be contiguous.
- If both numbers are on the same street segment, the geocoded point is interpolated to the approximate mid-point of the range.

- If the numbers are on two different segments, the geocoded point is based on the last valid house number of the first segment. The ZIP Code and FIPS Code are based on the first segment.
- In all cases, odd/even parity is evaluated to place the point on the correct side of the street.

A close match to a single address number is preferred over a ranged address match. GeoStan attempts a close match on the recombined address number before making a ranged match, as seen in the following example:

Input: 4750-4760 Walnut St, Boulder, CO
Output: 4750-4760 Walnut St, Boulder, CO

The Address Range match is to a single street segment, with the geocode being placed on the mid-point of the range.

Input: 47-50 Walnut St, Boulder, CO
Output: 4750 Walnut St, Boulder, CO

In the example below, the second number is not larger than the first, so GeoStan treats this as a unit number rather than a ranged address:

Input: 4750-200 Walnut St, Boulder, CO
Output: 4750 Walnut St STE 200, Boulder, CO

Understanding missing and wrong first letter

The missing and wrong first letter feature enables GeoStan to look for the correct first letter of a street address if the first letter is missing or incorrect. GeoStan searches through the alphabet looking for possible, correct first letters to complete the street address.

The feature is disabled by default and cannot be enabled in EXACT mode. To enable this feature, modify GS_FIND_FIRST_LETTER_EXPANDED.

Below are some examples of wrong, missing first letter, and duplicate first letter input addresses and the corresponding GeoStan output:

The example includes an incorrect first letter:

Input: 4750 nalnut boulder co 80301
Output: 4750 Walnut St Boulder CO 80301-2532

This example excludes a first letter:

Input: 4750 alnut boulder co 80301
Output: 4750 Walnut St Boulder CO 80301-2532

This example includes an extra first letter:

Input: 4750 wwalnut boulder co 80301
Output: 4750 Walnut St Boulder CO 80301-2532

Permitting relaxed address number matching

When GeoStan matches an input address, its default behavior is to match to the address number. This default behavior corresponds to GS_FIND_MUST_MATCH_ADDRNUM set to True. GeoStan must match to an address number.

If GS_FIND_MUST_MATCH_ADDRNUM is set to False (in GS_MODE_CUSTOM only), then GeoStan no longer must match the address number, therefore permitting relaxed address number matching. By permitting relaxed address number matching, an inexact match can be found. If the input address number is missing, no matches are returned unless GS_FIND_STREET_CENTROID (Understanding street locator geocoding) is also enabled.

If the input address number is not within a house number range from the street, GeoStan selects the nearest range on the street which has the same parity (even or odd) as the input address number. GeoStan returns one or more of the closest matches inside this range that preserves street parity. This requires GeoStan to change the house number. The new house number is equal to one of the range's endpoints, possibly plus or minus one to preserve street parity.

NOTE: Even when GS_FIND_MUST_MATCH_ADDRNUM is set to False and an inexact match on the house number is found, GeoStan still returns an error code and the inexact house number has to be retrieved as one would retrieve a multi-match.

When GS_FIND_MUST_MATCH_ADDRNUM is set to False and no exact matching house number is found, a match code of either E029 (no matching range, single street segment found), or E030 (no matching range, multiple street segment) is returned by GsDataGet and GeoStan does not change the house number on the output address.

In order to access the inexact address number candidates, the GsMultipleGet function must be called. If there are inexact house number candidates returned by GsMultipleGet, the corresponding match codes begin with the letter 'H' indicating that the house number was not matched. Additionally, even when one or more exact candidates are found, inexact matches to the house number are still on the list of possible candidates, and these can be differentiated from the others by their Hxx match codes. For more information on match codes, see "Match codes" on page 435.

Forcing an exact match on specific address elements

Listed below are the properties to specify a desired exact match:

- GS_FIND_MUST_MATCH_MAINADDR - Matches to an input street name exactly. This match is dependent on source data.
- GS_FIND_MUST_MATCH_STATE - Matches to an input state exactly.
- GS_FIND_MUST_MATCH_ZIPCODE - Matches to an input ZIP 5 exactly.
- GS_FIND_MUST_MATCH_CITY - Matches to an input city exactly.

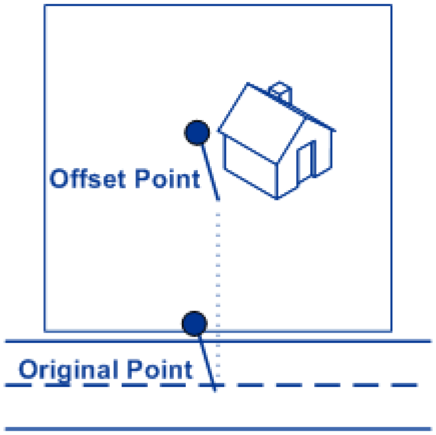
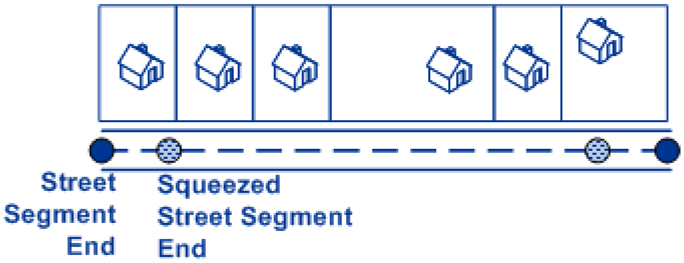
Using geocoding features

This section contain information on geocoding concepts used by GeoStan, and includes the following topics:

- Using geocode placement options
- Using point-level matching
- Using centerline matching
- Understanding address point interpolation
- Understanding street-level matching
- Understanding street locator geocoding
- Understanding ZIP Code centroid matching
- Matching to a geographic centroid

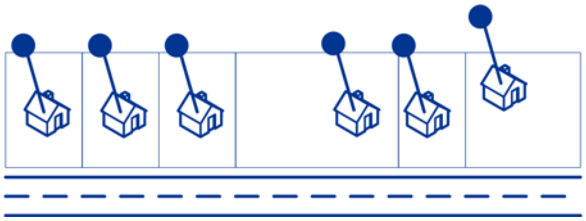
Using geocode placement options

You can set the following options that affect how GeoStan calculates the geocode for an interpolated street match.

Option	Search Types
Offset Ensures the point does not reside in the middle of a street. Moves the point perpendicular to the portion of the street segment in which it lands by the value you specify. The default is 50 feet for the best visual representation for mapping packages.  	- Address - Reverse Geocoding
Centerline Offset Similar to Offset, but used only with centerline matching. Moves the point from the street centerline toward the parcel point the value you specify. This is useful for routing applications. The default is 0 feet to return the street centerline geocode.	- Address
Squeeze Ensures the point does not reside in an intersection or too close to the end of a street. Using a squeeze distance moves both street end points closer to the center of the segment by the value you indicate.  	- Address - Reverse Geocoding

Using point-level matching

Point-level matching locates the center of the actual building footprint or parcel. This is the most accurate type of geocode and is used in industries such as internet mapping, flood hazard determination, property and casualty insurance, telecommunications, and utilities.



If you are licensed for the point-level data option, you do not need to execute any additional initialization or setup after you have installed the point-level data. GeoStan automatically processes your address lists through the point-level data.

When processing address lists, GeoStan first searches for a match in the point-level data. If it cannot find an exact match in the point-level data, GeoStan continues searching for a better match in the street network data. GeoStan returns the best match found, with preference given to matches from the point-level dataset.

Using centerline matching

Centerline matching is used with point-level matching to tie a point-level geocode with its parent street segment. This functionality is useful for routing applications.

This provides you with additional data about the parent street segment that is not retrievable using only the point-level match. The output information also includes the bearing and distance from the point data geocode to the centerline match.

NOTE: Centerline matching requires a license for point-level matching. For more information on point-level matching, see Using point-level matching.

To use centerline matching:

- Make sure you have loaded point-level data.

If you have loaded point-level data, GeoStan automatically processes your address using the point-level data.

- Use the GS_FIND_CENTRLN_PROJ_OF_POINT find property so segment candidates are located along with the point-level match.

NOTE: If GeoStan cannot find a match within the point-level data, GeoStan returns data from the centerline of the segment without any offset.

- Use the GS_FIND_CENTERLINE_OFFSET find property to return a geocode for the centerline, offset from the center of the street, like the curb or sidewalk. You cannot set the point farther back from the street then the associated point-level geocode.
- Optionally, use the bearing enum to receive the compass direction of the point data match to the centerline match. Knowing the bearing is useful for routing applications.

Understanding address point interpolation

Address point interpolation uses a patented process that improves upon regular street segment interpolation by inserting point data into the interpolation process. To utilize this feature, set GS_FIND_ADDRPOINT_INTERP to True.

NOTE: See Understanding street-level matching for more information on street segment interpolation.

When an address point User Dictionary or a point GSD is present, more precise address geometry is used for interpolation than what is available by the use of street segments alone.

GeoStan first attempts to find a match using the loaded point data, in priority order, for more information on priority order please see Specifying database search order. If an exact point match is found in the point data, then searching ceases and the point match is returned. If an exact point match was not found, GeoStan attempts to find high and low boundary address points to use for address point interpolation.

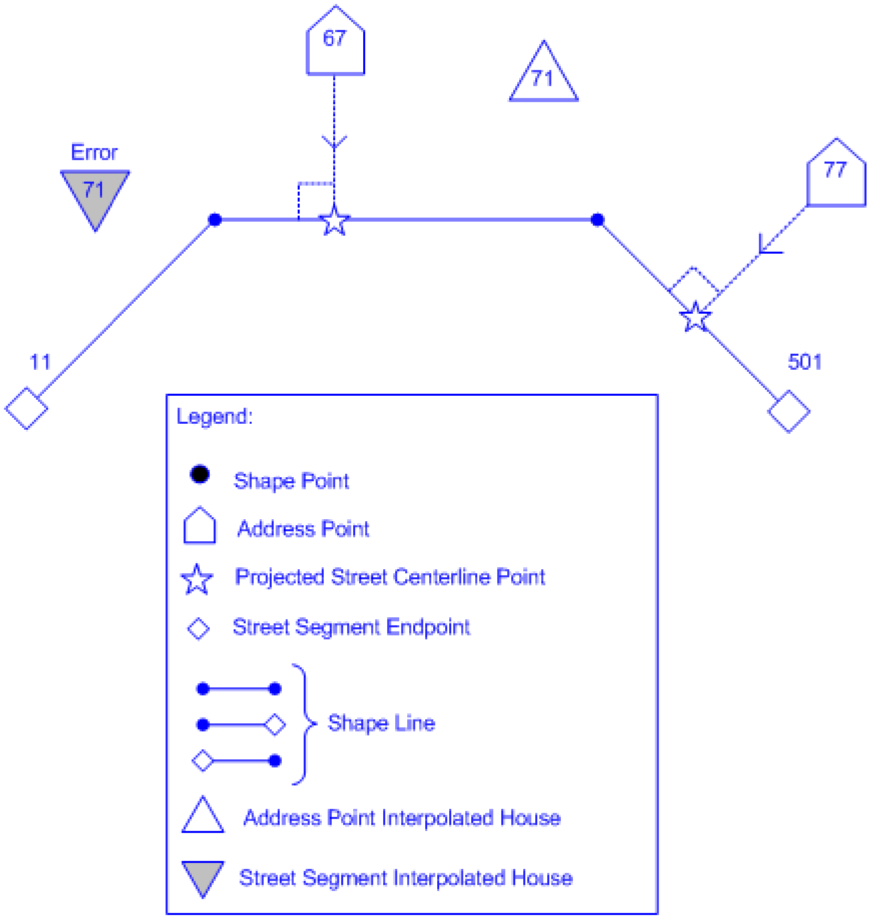
NOTE: This feature does not work with point addresses in auxiliary files.

To illustrate the use of this feature, see the example on the following page. In the example, if the input house number is 71, the point GSD contains address points for 67 and 77.

The street segment ranges from 11 to 501. The street segment contains shape lines describing the actual layout of the street.

GeoStan attempts to map the points for addresses 67 and 77 onto the closest shape line. After finding a point on the centerline of the street, GeoStan then performs the interpolation for the input house number 71 with the new street centerline points of 67 and 77.

Without this feature, GeoStan performs an interpolation with the street segment end points of 11 and 501. This creates a far less accurate result (labeled in the diagram) than using the centerline points of the closest surrounding high and low address points.


67
71
Error
71
77
11
501
Legend:
● Shape Point
Address Point
Projected Street Centerline Point
Street Segment Endpoint
Shape Line
Address Point Interpolated House
Street Segment Interpolated House

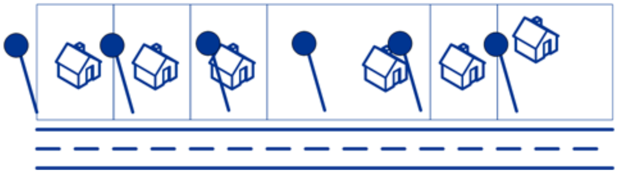
In the occasional scenario, if there is a situation where the boundaries found have the same parity but are on opposite sides of the street. To determine on which side of the street the address should be, address point interpolation uses information from the matched street segment.

NOTE: Please see Using User Dictionaries with address point interpolation.

Understanding street-level matching

Street matching identifies the approximate location of an address on a street segment. In street matching, the location is determined by calculating the approximate location of a house number based on the range of numbers in the location’s street, a process referred to as interpolation. For example, if the address is on a street segment with a range of addresses from 50 to 99, then it is assumed that the house number 75 would be in the middle of the street segment. This method assumes that the addresses are evenly spaced along the street segment. As a result, it is not as exact as point matching since addresses may not be evenly distributed along a street segment.

For example, the following diagram shows the results of street-level matching along a segment with unevenly-spaced buildings. The first three buildings are fairly accurately geocoded because they are evenly spaced. The fourth building, however, resides on a slightly larger parcel than the others along this street. Since street-level matching assumes that the buildings are evenly spaced, the result is that fourth, fifth, and sixth houses are not as precise as the first three. If you were to use point-level geocoding, or address point interpolation, the results would be more accurate.



Understanding street locator geocoding

Street locator geocoding is an optional feature, enabled using GS_FIND_STREET_CENTROID. When enabled, if an input street address cannot be found using the street number and name, GeoStan then searches the input ZIP Code or city/state for the closest match. If GeoStan is able to locate the street, it returns a geocode along the matched street segment rather than the geocode for the entered ZIP Code or ZIP + 4.

If street locator is enabled, and the input address is 5000 Walnut Street, Boulder, CO 80301, and there is no 5000 Walnut Street, GeoStan searches for the closest match to that address within the input ZIP Code. If there is no input ZIP Code, GeoStan searches for the closest match to the input address within Boulder, CO.

If the input address is Walnut Street, Boulder, CO 80301, since there is no street number, GeoStan then searches for that street within the input ZIP Code. As with the previous example, if there is no input ZIP Code, then GeoStan searches within Boulder, CO for the closest match.

When using street locator geocoding, if no exact matching house number is found, a match code of either E029 (no matching range, single street segment found), or E030 (no matching range, multiple street segment) returns. The GeoStan Find call is returned by GsDataGet. GS_ADDRESS_NOT_FOUND or GS_ADDRESS_NOT_RESOLVED respectively for the match codes E029 and E030. For example, if you enter Main St and there are both an E Main St and a W Main St within the input ZIP Code then an E030 returns and the location code returned is reflective of the input ZIP Code. The location code returned begins with a 'C' when matched to a single street segment, indicated by E029.

GeoStan does not change the street name on the output address. In order to access the street name that corresponds to the geocode which was returned, the GsMultipleGet function must be called. Street range information for the matched street or streets is returned by GsMultipleGet, and the match codes returned indicate the changes made to the street name to make the match (as with regular match codes). For more information regarding the match and location codes associated with this feature, see "Status codes" on page 435 and "Street centroid location codes" on page 445.

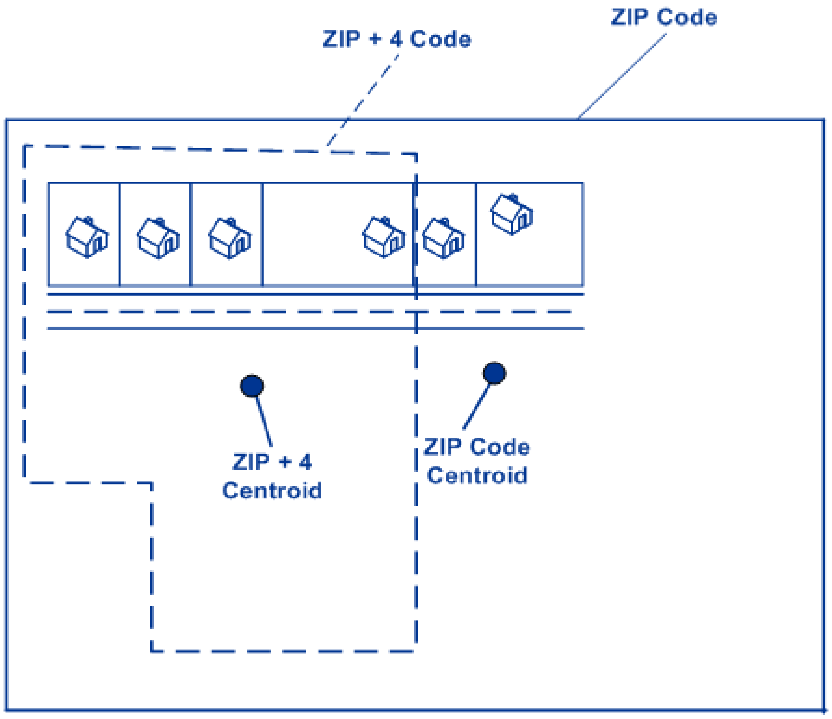
NOTE: This option is not available in CASS mode.

Understanding ZIP Code centroid matching

ZIP Code centroid matching is a center point of an area defined by either a ZIP Code or a ZIP + 4, and is the least accurate type of geocode. A ZIP Centroid is the center of a ZIP Code; a ZIP + 4 centroid is the center of a ZIP + 4. Since a ZIP + 4 represents a smaller area than a ZIP Code, a ZIP + 4 centroid is more accurate than a ZIP Code centroid.

The following diagram illustrates centroid matching. In the following example:

All six houses would have the same geocode in this example because they all reside in the same ZIP + 4 code.



Matching to a geographic centroid

If the input address includes a valid combination of city, county, and state (but no further address information), you can still geocode to the city, county, or state centroid. Geographic centroid geocoding is less precise than street or postal geocoding, but may be suitable for certain applications. Geographic centroid geocoding can be accomplished using

GsFindGeographicFirst and GsFindGeographicNext, if GeoStan cannot match a record to the level or precision you originally requested (such as street level).

For geographic geocoding, GeoStan returns the most precise geographic centroid that it can, based on the user input. It is recommended that the Last-line lookup functions of GeoStan be used to standardize and parse the address elements prior to using GsFindGeographicFirst and GsFindGeographicNext as these functions only provide some basic fuzzy matching on the input values and do not do the kind of comprehensive matching and parsing achieved through the Last-line lookup functions.

The following table shows some examples of address input and the best possible geographic centroid candidate. For more information on the location codes listed, see Geographic centroid location codes.

Input address	Geocoded to (Location code)
Valid City Valid County Valid State Troy, Rensselaer, NY	City (GM). City, County, and State are all valid.
Invalid City Valid County Valid State San Diego, Albany, NY	County (GC) There is no city of San Diego in NY. However, there is an Albany County in NY, so County is the best possible match.
Invalid City Invalid County Valid State Phoenix, Maricopa, NY	State (GS). There is no city of Phoenix in NY, nor is there a Maricopa County in NY, so State is the best possible match.
Valid City Invalid County Valid State Albany, Saratoga, NY	City (GM). There is a city of Albany in NY. There is a Saratoga County in NY, but Albany is not in that county. However, the city match is still made.
Valid Major City Chicago	City (GM). There are approximately 300 U.S. cities recognized as a major city and can be geocoded to the city centroid with no other information provided.
Valid City No County Valid State Albany, NY	Seven matched candidates. The city of Albany, NY is a close match (GM). Five instances of cities in New York that "sound like" Albany (such as Albion) are non-close GM matches. The state centroid (GS) is also a non-close match.
Valid City Valid County No State Iberia, New Iberia	No matched candidates. Since the state (LA) is missing, only a city match can be attempted. Since Iberia is not recognized as one of the Major U.S. Cities, no match is possible.

Input address	Geocoded to (Location code)
Invalid City No County Valid State St. Louis, NY	Three matched candidates. The state centroid (GS) is a close match and two city centroids (GM) are non-close "sound like" matches in New York State. If the input contains a state, all matches must be within that state.
No City Valid County Valid State Otsego, NY	Two matched candidates. The county centroid (GC) is a close match and the state centroid (GM) is a non-close match (GM).
No City Valid County No State Otsego	No matched candidates. A county name alone is not enough for a match.
Valid City No County No State Saco	No matched candidates. A city name alone is not enough for a match, except if it is one of the recognized Major U.S. Cities.

Using USPS matching enhancements

- Understanding Delivery Point Validation
- Understanding LACS^{Link}
- Understanding Suite^{Link}
- Understanding Early Warning System (EWS)
- USPS Line of Travel codes

Understanding Delivery Point Validation

Delivery Point Validation (DPV™) is a United States Postal Service (USPS®) technology that validates the accuracy of address information down to the individual mailing address. By using DPV to validate addresses, you can reduce undeliverable-as-addressed (UAA) mail, thereby reducing postage costs and other business costs associated with inaccurate address information.

Without DPV, the address validation process only verifies that an individual address is within a range of valid addresses for the given street. For example, the USPS data indicates that the range of addresses on Maple Ln. is 500 to 1000. You attempt to validate an address of 610 Maple Ln. Without DPV, this address would appear to be valid because it is in the range of 500 to 1000. However, in reality the address 610 Maple Ln. does not exist: the house numbers in this section of the street are 608, 609, 613, and 616. With DPV processing, you would be alerted to the fact that 610 Maple Ln. does not exist and you could take action to correct the address.

DPV also provides unique address attributes to help produce more targeted mailing lists. For example, DPV can indicate if a location is vacant and can identify commercial mail receiving agencies (CMRAs) and private mail boxes (PMBs).

Although DPV can validate the accuracy of an existing address, you cannot use DPV to create address lists. For example, you can validate that 123 Elm Street Apartment 6 exists, but you cannot ask if there is an Apartment 7 at the same street address. To prevent the generation of address lists, the DPV database contains false positive records. False positive records are artificially generated invalid addresses which reside in the DPV database. A match to a false positive address disables any further queries to the DPV database, as required by the USPS.

If you have licensed the DPV processing option, you must install the DPV data and set DPV initialization properties for GeoStan to process your addresses through DPV. If you initialize DPV, GeoStan automatically attempts to resolve any multiple candidates for a match. For more information on DPV, see Appendix F, "USPS Link products."

DPV is required for CASS certification.

Understanding LACS^{Link}

The USPS Locatable Address Conversion System (LACS) corrects addresses that have changed as a result of a rural route address converting to street-style address, a PO Box renumbering, or a street-style address changing. The following are examples of LACS^{Link} conversions:

- Rural Route Converted to Street-Style Address:

  Old Address:
  RR 3 Box 45

New Address:
1292 North Ridgeland Dr

- Street Renamed and Renumbered:

  Old Address:
  23 Main St

  New Address:
  45 West First Ave

- PO Box Renumbered:

  Old Address:
  PO Box 453

  New Address:
  PO Box 10435

LACS$^{\text{Link}}$ is required for CASS certification.

If you have licensed the LACS$^{\text{Link}}$ processing option, you must install the LACS$^{\text{Link}}$ data and set the LACS$^{\text{Link}}$ initialization properties for GeoStan to process your address through LACS$^{\text{Link}}$. For more information on LACS$^{\text{Link}}$, see Appendix F, "USPS Link products."

Understanding Suite$^{\text{Link}}$

The purpose of Suite$^{\text{Link™}}$ is to improve business addressing by adding known secondary (suite) numbers to allow delivery sequencing where it would otherwise not be possible. Suite$^{\text{Link}}$ uses the input business name, street number location, and 9 digit ZIP+4 to return a unit type (i.e. "STE") and unit number for that business.

As an example, when entering the following address with Suite$^{\text{Link}}$ enabled in CASS mode...

UT Animal Research
910 Madison Ave
Memphis TN 38103

GeoStan returns the following:

UT Animal Research
910 Madison Ave STE 823
Memphis TN 38103

Or

UT Animal Research
910 Madison Ave #823
Memphis TN 38103

If you have licensed the Suite^{Link} processing option, you must install the Suite^{Link} data and set the Suite^{Link} initialization properties for GeoStan to process your address through Suite^{Link}. For more information on Suite^{Link}, see Appendix F, "USPS Link products."

For more information on Suite^{Link} enums and functions, see the following sections:

- "Common enums for storing and retrieving data" on page 68 for C and page 248 for COBOL.
- GsFindWithProps properties

Suite^{Link} is required for CASS certification.

Understanding Early Warning System (EWS)

The Early Warning System (EWS) provides up-to-date address information for new and recently changed addresses that have not yet been updated in the monthly USPS data updates. EWS prevents address records from miscoding due to a delay in postal data reaching the USPS address matching files.

The older the U.S. Postal Databases, the higher potential you have for miscoding addresses. When a valid address is miscoded because the address it matches to in the U.S. Postal Database is inexact, it will result in a broken address.

EWS data consists of records containing partial address information limited to the zip code, street name, predirectional, postdirectional, and a suffix. For an address record to be EWS-eligible, it must be an address not present on the most recent monthly production U.S. Postal Database.

The USPS refreshes the EWS file on a weekly basis. You can download the EWS file from the USPS Web site at http://ribbs.usps.gov/files/CASS/.

NOTE: In order to use the USPS-supplied EWS file, GeoStan requires that the EWS file be named ews.txt and be located in the GeoStan data search path. The date of the EWS file must be the same month or newer than your GeoStan data files.

USPS Line of Travel codes

GeoStan can return the enhanced USPS Line of Travel (eLOT) codes, which are based upon the ZIP + 4 and Carrier Route of an address and signify the approximate order that the postal carrier delivers the mail.

NOTE: Using eLOT codes may help you qualify for greater bulk mail discounts.

To assign eLOT Codes, you must input mailable addresses that GeoStan can standardize; ZIP and ZIP + 4 Codes are not sufficient.

To access the eLOT codes, which include a sequence number and code, the following enums exist for **`GsDataGet`** on output:

`GS_LOT_NUM`	4-digit eLOT number (5 including the NULL).
`GS_LOT_CODE`	Single character string (2 including the NULL) indicating the direction of carrier travel. • A – Ascending • D – Descending

The `Us.gsl` file contains the eLOT information and GeoStan automatically initializes the file if it is in the path. See "GsFileStatus" on page 153 for information on verifying the loaded files.

The license file controls access to the `Us.gsl` file. If the license file does not have eLOT enabled, you cannot return eLOT values. To upgrade your license to use the eLOT data, contact your Pitney Bowes Business Insight sales representative.

Elements available via GeoStan

A complete list of data available through GeoStan can be found in, "Common enums for storing and retrieving data." For information on Line-of-Travel codes, see USPS Line of Travel codes. For more information on multiline or free form addresses, see Using enhanced search options.

Using custom supplemental data sets

- Understanding User Dictionaries
- Understanding auxiliary file matching

Understanding User Dictionaries

A User Dictionary is a table of streets and address ranges that you use as a source for geocoding. If you have newer or more precise data than what is available in GSD files, creating a dictionary with this data can help you obtain more accurate geocoding results. For example, if you have address point data you can create a User Dictionary that enables you to take advantage of the GeoStan address point interpolation capabilities.

A User Dictionary can be used by itself to geocode records, or can be used in combination with the supplied GSD (GS_STATUS_FILE_UD).

You can also specify the search priority for matching against GSD records and User Dictionary records (GS_FIND_DB_ORDER). This option allows greater control over which candidates are returned; in particular, it enables selection of User Dictionary candidates over GSD candidates.

For more information see User Dictionary.

NOTE: The USPS does not consider matches to data that they did not create and these are not considered valid addresses for postal delivery. Therefore, GeoStan does not match to User Dictionaries when processing in CASS mode

Understanding auxiliary file matching

Although PBBI provides robust data for you to match your input address against, in some cases you may want to match your address against speciality data. GeoStan provides you with the ability to create auxiliary files for these instances. For more information on auxiliary file matching, see Appendix G, "Auxiliary file layout."

NOTE: GeoStan only matches an input address to a record in the auxiliary file if there is an exact match. The USPS does not consider matches to data that they did not create and these are not considered valid addresses for postal delivery. Therefore, GeoStan does not match to auxiliary files when processing in CASS mode

Using reverse geocoding

Reverse geocoding is an optional, licensed feature, which takes input geocodes and retunrs possible addresses, segment IDs, and block IDs.

For example, entering the following information in geotest using the nearest address processing option

Longitude:	-105239771
Latitude:	40018912
Search Distance:	150 feet

results in the following output

```
4750 WALNUT ST
BOULDER, CO 80301-2538
MatchCode = NS0
LocCode = AS0
Lon = -105239773
Lat = 40018911
Distances: Search = 150 Offset = 50 Squeeze = 50 Nearest = 50.0 Pct Geocode = 94.0
SegID = 472881795
PtID = GDT
Block = 080130122032066
County Name = BOULDER COUNTY
DPBC = 50
```

Although many of the GeoStan outputs apply to reverse geocoding searches, several outputs are unavailable such as LACS$^{\text{Link}}$ conversion and unit numbers. GeoStan returns these outputs as blank. GeoStan also has outputs specific to reverse geocoding searches.

If multiple street segments are equally close to the input location, GeoStan returns a status indicating multiple candidates for a match so you can determine which segment is the appropriate match.

NOTE: The information GeoStan returns contains an approximate address based on the geocode you input. This approximate address may not exist or may not accept mail delivery.

See the following sections for more information on reversing a geocode match:

- Preparing for processing geocodes with GeoStan
- Understanding how GeoStan processes reverse geocodes
- Reverse geocode matching sample code

NOTE: Reverse Geocoding is not available on z/OS.

Preparing for processing geocodes with GeoStan

Before processing geocodes, follow the steps below to ensure proper geocoding:

1 Verify your input list

Although your input list can contain geocode information as well as address information, GeoStan tries to match to either the geocode or the address information; not both. GeoStan returns an error if you try to match to both the geocode and the address information.

Your input list should contain longitude and latitude points in millionths of decimal degrees.

2 Pass the geocode in GeoStan using the longitude and latitude enums with the data set function.

3 Load your GSX files

A reverse geocode search requires GSX files. On Windows, you can install these files when you install the Centrus data. You can create the files using the `batchind` program if you are on a non-Windows platform or did not install files during your data installation. For more information on GSX files, see the *Centrus Utilities Manual* on the software installation media or at www.g1.com/support.

4 Set your processing options.

When processing a geocode through GeoStan you can set the following processing options using the data set and find functions:

Option	Description
Offset Distance	Sets the offset distance from the street segment.
Squeeze Distance	Set the distance to offset the first and last address-level geocodes from the street ends.
Search Distance	Sets the radius in which GeoStan searches for a match to the input geocode.
Searchable Address Types	Sets the type of match candidates that you allow GeoStan to match against. This can include intersections, addresses interpolated on street segments, street segments with no number range, and point data locations.

Understanding how GeoStan processes reverse geocodes

GeoStan processes geocodes in the following order:

1 GeoStan defines a rectangle based on your input geocode and search distance.

2 GeoStan computes the distance between each street segment and the input location.

3 If one segment is closest, GeoStan finds the offset and interpolated percentage and the street side. It then computes an approximate house number based on this information.

If there is more than one segment that is equally close to the input location, a multi-match occurs. GeoStan returns the information for all of the equally close segments so that you can determine which segment is applicable.

4 GeoStan returns the address information, including the segment range, the approximate house number, and the parity of the range along with other standard address information.

NOTE: Although many of the standard address matching outputs apply to the reverse geocoding option, several outputs are unavailable (such as LACS^{Link} conversion and unit numbers). GeoStan returns these outputs as blank. GeoStan also has outputs specific to reverse geocode processing, such as specific match codes and the distance from the input location to the matched segment.

Reverse geocode matching sample code

The following is a code sample for matching an address using the C functions of GeoStan.

```
char matchcode[GS_MATCH_CODE_LENGTH];
char outaddr[GS_ADDRLINE_LENGTH];
char lastline[GS_LASTLINE_LENGTH];
char pct[GS_PERCENT_GEOCODE_LENGTH];
char nearestdist[GS_NEAREST_DIST_LENGTH];
GsClear(gs);
GsDataSet(gs,GS_LON,"-105239771");
GsDataSet(gs,GS_LAT,"40018912");
GsDataSet(gs,GS_OFFSET_DIST,"5");
GsDataSet(gs,GS_SQUEEZE_DIST, “50”);
if(GsFind(gs,GS_NEAREST_ADDRESS)==GS_SUCCESS)
{
   GsDataGet(gs,GS_OUTPUT,GS_MATCH_CODE,matchcode,sizeof(matchcode));
   GsDataGet(gs,GS_OUTPUT,GS_ADDRLINE,outaddr,sizeof(outaddr));
   GsDataGet(gs,GS_OUTPUT,GS_LASTLINE,lastline,sizeof(lastline));
   GsDataGet(gs,GS_OUTPUT,GS_PERCENT_GEOCODE,pct,sizeof(pct));
   GsDataGet(gs,GS_OUTPUT,GS_NEAREST_DIST,nearestdist,
   sizeof(nearestdist));
}
```

Using reverse APN matching

Reverse APN matching is an optional processing feature. You enter FIPS and Assessor’s Parcel Number (APN) codes to receive information about the corresponding parcel. GeoStan must exactly match to your input to locate a match candidate.

To perform a reverse APN search, follow the steps below:

1 License and install the Centrus Points and Centrus APN data files.

2 Include the APN code flag when initializing GeoStan

3 Include the APN ID, county FIPS, and state FIPS fields with the data set function.

4 Include the APN search option when using the find function.

Chapter 3

Defining C functions

This chapter lists the public functions of the C API in alphabetical order. The following sections provide a quick reference to the data types and functions; organized by usage.

Data types and enum functions

Data types
Provides information on the data types used by the C functions.
Common enums for storing and retrieving data
Provides information on the enums used for GsHandleGet, GsDataSet, GsDataGet, and GsMultipleGet.
GsFindWithProps properties
Lists the GsFindWithProps find properties.
GsInitWithProps input properties
Lists the GsInitWithProps initialization properties.
GsInitWithProps output status properties
Lists GsInitWithProps status properties.

Initialization & termination functions

GsFileStatus
Returns if the files successfully opened, and the date of the USPS data used in generating the primary GSD file.
GsFileStatusEx
Returns information about the current GeoStan data set and license.
GsGetLibVersion
Returns the current version of the GeoStan library.
GsInitWithProps
Initializes GeoStan using properties.
GsTerm
Terminates GeoStan.

Last-line lookup

GsCityDataGet
Retrieves data for the found city record.
GsCityFindFirst
Finds the first city matching the partial name or valid ZIP Code.
GsCityFindNext
Finds the next matching city.

GsFindFirstState

Finds the first state matching the name, abbreviation, or valid ZIP Code.

Address lookup

GsDataGet

Returns data for all address and matched elements from GeoStan.

GsDataSet

Passes data for all address elements to GeoStan.

GsFindWithProps

Finds a match with user settable match preferences (properties).

GsGetCoordsEx

Retrieves coordinates for the street segment found via GsFindWithProps.

GsMultipleGet

Returns the address elements for the match candidate specified

GsMultipleGetHandle

Returns the range handle for the match candidate specified.

GsNumMultiple

Returns the number of match candidates found.

GsSetSelection

Allows you to select a match from a set of match candidates.

Data querying

GsHandleGet

Retrieves data for objects found via GsFindFirst and GsFindNext.

GsHandleGetCoordsEx

Retrieves the coordinates for a street segment object found via GsFindFirst and GsFindNext.

GsGetDBMetadata

Populates the supplied buffer with the information about the item being queried for the specified database.

GsGetNumDB

Returns the number of loaded databases.

Error and message handling

GsErrorGet

Retrieves current error information.

GsErrorGetEx

Retrieves informational messages, error messages, and fatal warnings for the current GeoStan thread.

GsErrorHas

Indicates if any errors occurred or any informational messages are available in the current thread.

Data passing functions

GsFindFirst___

Finds the first street, segment, or range object that meets the search criteria.

GsFindNext___

Finds the next street, segment, or range object that meets the search criteria.

GsSetSelectionRange

Allows GeoStan to use a record found outside of GsFindWithProps as a match.

GsFindGeographicFirst

Finds the first geographic centroid match for a city, county, and or state from the set of possible matches found.

GsFindGeographicFirstEx

Finds the first geographic centroid match for a city, county, and or state from the set of possible matches found.

GsFindGeographicNext

Finds the next geographic centroid match for a city, county, and or state from the set of possible matches found.

GsFindGeographicNextEx

Finds the next geographic centroid match for a city, county, and or state from the set of possible matches found.

Utilities

GsClear

Clears the data buffer in the internal data structures.

GsSoundex

Generates a soundex key for use in GsFindFirst.

GsTestRange

Tests if a house number falls within a range.

GsVerifyAuxiliaryRecord

Validates an auxiliary file record.

GsVerifyGSDDataPresent

Validates the presence of data for specified states in the GSD data file.

GsZ4Changed

Determines if the USPS changed a ZIP + 4 address definition since GeoStan last processed the record.

CASS report functions

GsFormatCASSHeader

Writes a CASS 3553 report to the header buffer argument.

GsWriteExtendCASSReport

Writes a USPS CASS report based on the specified template.

DPV report functions

GsDpvGetCompleteStats

Obtains the complete DPV statistics since the application initialized DPV.

GsDpvGetFalsePosDetail

Retrieves the detail record for a DPV false-positive report.

GsDpvGetFalsePosHeaderStats
Retrieves DPV statistics for the header record for a DPV false-positive report.
GsFormatDpvFalsePosDetail
Formats a DPV false-positive detail record from GsDpvGetFalsePosHeaderStats.
GsFormatDpvFalsePosHeader
Formats a DPV false-positive header record with data from GsDpvGetFalsePosDetail.

LACS^{Link} report functions

GsFormatLACSFalsePosDetail
Formats a LACS^{Link} false-positive detail record from GsLACSGetFalsePosDetail.
GsFormatLACSFalsePosHeader
Formats a LACS^{Link} false-positive header record with data from GsLACSGetFalsePosHeaderStats.
GsLACSClearStats
Clears the LACS^{Link} statistics.
GsLACSGetCompleteStats
Obtains the complete LACS^{Link} statistics since the application initialized LACS^{Link}.

GsLACSGetFalsePosDetail
Retrieves the detail record for a LACS^{Link} false-positive report.
GsLACSGetFalsePosHeaderStats
Retrieves LACS^{Link} statistics for the header record for a LACS^{Link} false-positive report.

Spatial query functions

GsFindFirstSegmentByMbr
Finds the first segment handle within the specified MBR.
GsFindNextSegmentByMbr
Finds the first segment handle within the specified MBR.
GsGetMbr
Returns an MBR for the specified geographic element.
GsPrepareIndexFinance
Prepares all .gsx files needed to cover the finance area.
GsPrepareIndexMbr
Prepares all .gsx files needed to cover the minimum bounding rectangle (MBR).

Property setting functions

GsPropSetAsStr
Sets a property where input name and value are both strings.
GsPropSetBool
Sets a boolean property.
GsPropSetDouble
Sets a double property.
GsPropSetLong
Sets an integer property.

GsPropSetStr
Sets a string property.

Property retrieving functions

GsPropGetBool
Retrieves a boolean property.
GsPropGetDouble
Retrieves a double property.
GsPropGetInfo
Retrieves information about a property.
GsPropGetLong
Retrieves an integer property.
GsPropGetStr
Retrieves a string property.
GsPropGetStrLen
Retrieves the length of the string value of a property.

Property list management

GsPropListCreate
Creates and initializes a property list for GeoStan initialization.
GsPropListClear
Clears (empties) a property list.
GsPropListDestroy
Destroys a property list.
GsPropFind
Finds a property in a property list.
GsPropFirst
Sets property iterator to first property.
GsPropGetNext
Iterates to the next sequential property in the property list.
GsPropListGetCount
Retrieves the number of properties in a property list.
GsPropListGetType
Retrieves the type of the property list.
GsPropListRead
Reads the property list from a file or character string.
GsPropListReset
Resets the property list to its default state.
GsPropRemove
Removes a property from the property list.
GsPropListWrite
Writes the property list to a file or character string.

Data types

The following table contains the data types used by GeoStan.

The asterisk symbol after the type indicates a pointer or reference to a variable. For example, `intl *p` indicates that `p` is a pointer to a long integer. All `short` data types are 16-bit; `long` data types are 32-bit.

Data Type	Description
`intlu`	32-bit unsigned long integer
`intl`	32-bit long integer
`intsu`	16-bit unsigned short integer
`ints`	16-bit short integer
`char *`	pointer to null terminated string
`gs_const_str`	pointer to constant string
`pstr`	pointer to null terminated string
`intlu *`	pointer to unsigned long integer
`intl *`	pointer to long integer
`pintl`	pointer to long integer
`ints *`	pointer to short integer
`GsId`	32-bit unsigned long integer
`GsEnum`	16-bit unsigned short integer
`GsFunStat`	32-bit unsigned long integer

Common enums for storing and retrieving data

This chapter contains the valid enums used with **`GsHandleGet`**, **`GsDataSet`**, **`GsDataGet`**, and **`GsMultipleGet`**.

The following describes the contents of each table column.:

Enum	Lists the defined constant used to access this element.
GsHandleGet	• Street A bullet indicates this enum is valid when using a Street handle with **`GsHandleGet`**. Includes only Street level information. • Segment A bullet indicates this is valid when using a Segment handle with **`GsHandleGet`**. Includes Street and Segment information. • Range A bullet indicates this enum is valid when using a Range handle with **`GsHandleGet`**. Includes Street, Segment, and Range information.
GsDataSet	A bullet indicates this enum is valid when using **`GsDataSet.`**
GsDataGet	• Input A bullet indicates this is valid when using input flag with this function. • Output Indicates this enum is valid when using the output flag with this function. Can be: – *A* – Valid for non-intersection matches only – *I* – Valid for intersection matches only – *L* - Valid for street locator matches only – *P* - Centrus point data match. – R - Reverse geocoding only. – *Bullet* – Valid for all types of match
GsMultipleGet	Indicates the valid match types. • *A* – Valid for non-intersection matches. • *I* – Valid for intersection matches only. • *L* - Valid for street locator and relaxed address matches only. • *P* - Centrus point data match. • *Bullet* – Valid for all types of matches.
Length	A number indicating the largest string GeoStan returns for this enum. This length includes a NULL-termination character. Symbolic constants use the naming convention `<<NAME>>_LENGTH`. For example the enum `GS_ADDRLINE` returns the input or output address line. The maximum length of an address line is 61. Therefore, the symbolic constant is `GS_ADDRLINE_LENGTH` and is defined as 61.
Description	A brief explanation of the information returned. If you use an invalid enum, GeoStan returns an Invalid Argument error.

	Length[a]	GsHandleGet			GsDataSet	GsDataGet			GsMultipleGet
		Street	Segment	Range		Input	Output	Center line	
GS_ADDRLINE First address line or address line for single line addresses. For single line addresses, must include lastline information and can also include address information.	61				●	●	●		●
GS_ADDRLINE_SHORT Shortest possible address line that can be constructed from available short street name and the other address line components.	61						●		
GS_ADDR2 Second address line.	61				●	●	●		
GS_ALIAS Use this to specify alias info instead of normal.		●	●	●					●
GS_ALT_FLAG Base/Alternate record • *B* – Base • A – Alternate	2			●			●		A
GS_APN_ID Assessor's Parcel Number (APN)	46				●		A,P		A,P
GS_AUX_USERDATA User data from the auxiliary file. Blank if no auxiliary file.	301						●		●

	Length[a]	GsHandleGet			GsDataSet	GsDataGet			GsMultipleGet
		Street	Segment	Range		Input	Output	Center line	
GS_BEARING Compass direction, in decimal degrees, from the point data match to the centerline match. Measured clockwise from 0 degrees north.	6							●	
GS_BLOCK 15-digit census block ID/census FIPS code, using the syntax sscccttttttgbbb where: • ss – 2-digit State FIPS Code • ccc – 3-digit County FIPS Code • tttttt – 6-digit Census Tract FIPS Code • g – Single-digit Block Group FIPS Code, and • bbb – Block FIPS Code If the address is on the left side of the street, GS_BLOCK will contain the equivalent of GS_BLOCK_LEFT. If the address is on the right, GS_BLOCK will contain the equivalent of GS_BLOCK_RIGHT. NOTE: GeoStan does not return a period for the Centrus Tract FIPS Code, this may deviate from the industry standard.	16						A		A
GS_BLOCK_LEFT Census block ID from the left side of the street.	16		●	●			A	●	●
GS_BLOCK_LEFT2 GS_BLOCK_LEFT for the second segment in the intersection.	16								I
GS_BLOCK_RIGHT Census block ID from the right side of the street.	16		●	●			A	●	●

	Length[a]	GsHandleGet			GsDataSet	GsDataGet			GsMultipleGet
		Street	Segment	Range		Input	Output	Center line	
GS_BLOCK_RIGHT2 GS_BLOCK_RIGHT for the second segment in the intersection.	16								I
GS_BLOCK_SFX Current block suffix for Census 2000 geography.	2		●	●			●		A
GS_BLOCK_SFX_LEFT Current left block suffix for Census 2000 geography.	2		●	●			●	●	●
GS_BLOCK_SFX_LEFT2 GS_BLOCK_SFX_LEFT for the second segment in the intersection.	2								I
GS_BLOCK_SFX_RIGHT Current right block suffix for Census 2000 geography. *Blank* if the matched record is from point-level data.	2		●	●			●	●	●
GS_BLOCK_SFX_RIGHT2 GS_BLOCK_SFX_RIGHT for the second segment in the intersection.	2								I
GS_CART Carrier route.	5			●			A		A
GS_CBSA_DIVISION_NAME CBSA division name.	128						●		●
GS_CBSA_DIVISION_NAME2 GS_CBSA_DIVISION_NAME for the second segment in the intersection.	128								I

	Length[a]	GsHandleGet			GsDataSet	GsDataGet			GsMultipleGet
		Street	Segment	Range		Input	Output	Center line	
GS_CBSA_DIVISION_NUMBER CBSA division number.	6						●		●
GS_CBSA_DIVISION_NUMBER2 GS_CBSA_DIVISION_NUMBER for the second segment in the intersection.	6								I
GS_CBSA_NAME CBSA name.	128						●		●
GS_CBSA_NAME2 GS_CBSA_NAME for the second segment in the intersection.	128								I
GS_CBSA_NUMBER CBSA number.	6						●		●
GS_CBSA_NUMBER2 GS_CBSA_NUMBER for the second segment in the intersection.	6								I
GS_CHECKDIGIT[b] Check digit.	2						●		
GS_CITY Abbreviated city name from the last line of the input or output address; the value from GS_NAME_CITY or GS_PREF_CITY.	29			●	●	●	●		●

	Length[a]	GsHandleGet			GsDataSet	GsDataGet			GsMultipleGet
		Street	Segment	Range		Input	Output	Center line	
GS_CITY_SHORT The output city name that appears in GS_LASTLINE_SHORT. This value is determined by logic similar to GS_CITY. Whenever possible, this city name is 13 characters or less. This output city name is determined by CASS rules. This can be either City State Name, City State Name Abbreviation, or Preferred Last Line City State Name.	29						●		●
GS_CLOSE_MATCH Indicates if the address was a Close match. • T - Match candidate is a Close match to the input address. • F - Match candidate is not a Close match to the input address.							●		●
GS_CMSA CMSA name.	81						●		●
GS_CMSA_NUMBER CMSA number.	5						●		●
GS_COUNTY County FIPS code. (For **GsHandleGet** and **GsMultipleGet** this is the USPS county for the Range record. For **GsDataGet** this is the county from the segment record or the USPS county if matched to a USPS only range.)	6			●			●		●
GS_COUNTY2 GS_COUNTY for the second segment in the intersection.	6								I

	Length[a]	GsHandleGet			GsDataSet	GsDataGet			GsMultipleGet
		Street	Segment	Range		Input	Output	Center line	
GS_COUNTY_FIPS County FIPS code	4					●			
GS_COUNTY_NAME County name.	128			●			●		●
GS_COUNTY_NAME2 GS_COUNTY_NAME for the second segment in the intersection.	128								I
GS_CSA_NAME CSA name.	128						●		●
GS_CSA_NAME2 GS_CSA_NAME for the second segment in the intersection.	128								I
GS_CSA_NUMBER CSA number.	4						●		●
GS_CSA_NUMBER2 GS_CSA_NUMBER for the second segment in the intersection.	4								I
GS_CTYST_KEY USPS city state key (an alphanumeric value that uniquely identifies a locale in the USPS city state product).	7			●			●		●

	Length[a]	**GsHandleGet**			**GsDataSet**	**GsDataGet**			**GsMultipleGet**
		Street	Segment	Range		Input	Output	Center line	
GS_DATATYPE Type of data used to make the match. • *0* – USPS data • *1* – TIGER data • *2* – Tele Atlas data • *3* – Sanborn point-level data • *4* – Deprecated • *6* – NAVTEQ data • *7* – Tele Atlas point-level data • *8* – Centrus point data • *9* – Auxiliary file data • 10 – User Dictionary	3		●				●	●	●
GS_DATATYPE2 GS_DATATYPE for the second segment in the intersection.	3								I
GS_DFLT[b] Indicates the return status of **GS_HI_RISE_DFLT** and **GS_R_RTE_DFLT** • *Y* – Either **GS_HI_RISE_DFLT** or **GS_R_RTE_DFLT** returned Y. • *Blank* – Both **GS_HI_RISE_DFLT** and **GS_R_RTE_DFLT** returned N or *Blank*.	2						●		A
GS_DPBC[b] Delivery Point Bar Code.	3						●		

	Length[a]	GsHandleGet			GsDataSet	GsDataGet			GsMultipleGet
		Street	Segment	Range		Input	Output	Center line	
GS_DPV_CONFIRM Indicates if a match occurred for DPV data. • *N* – Nothing confirmed • *Y* – Everything confirmed (ZIP + 4, primary, and secondary) • *S* – ZIP + 4 and primary (house number) confirmed • *D* – ZIP + 4 and primary (house number) confirmed and a default match (GS_HI_RISE_DFLT = Y), secondary did not confirm. • *Blank* – non-matched input address to USPS ZIP + 4 data, or DPV data not loaded	2						●		
GS_DPV_CMRA DPV CMRA indicator. • *Y* – Address found in CMRA table • *N* – Address not found in CMRA table • *Blank* – DPV not loaded	2						●		
GS_DPV_FALSE_POS DPV false-positive indicator. • *Y* – False-positive match found • *Blank* – False-positive match not found	2						●		
GS_DPV_FOOTNOTE1 Information about matched DPV records. • *AA* – ZIP + 4 matched • *A1* – Failure to match a ZIP + 4 • *Blank* – Address not presented to hash table or DPV data not loaded	3						●		

	Length[a]	GsHandleGet			GsDataSet	GsDataGet			GsMultipleGet
		Street	Segment	Range		Input	Output	Center line	
`GS_DPV_FOOTNOTE2` Information about matched DPV records. • *BB* – All DPV categories matched • *CC* – DPV matched primary/house number, where the secondary/unit number did not match (present but invalid) • *M1* – Missing primary/house number • *M3* – Invalid primary/house number • *N1* – DPV matched primary/house number, with a missing secondary number • *P1* – Missing PS, RR, or HC Box number • *P3* – Invalid PS, RR or HC Box number • *F1* – All military addresses • *G1* – All general delivery addresses • *U1* – All unique ZIP Code addresses • *Blank* – Address not presented to hash table or DPV data not loaded NOTE: A unique ZIP Code is a ZIP Code assigned to a company, agency, or entity with sufficient mail volume to receive its own ZIP Code.	3						●		
`GS_DPV_FOOTNOTE3` Information about matched DPV records. • *R1* – Matched to CMRA but PMB designator not present. • *RR* – Matched to CMRA and PMB designator present (PMB 123 or # 123). • *Blank* – Address not presented to hash table or DPV data not loaded	3						●		

	Length[a]	GsHandleGet			GsDataSet	GsDataGet			GsMultipleGet
		Street	Segment	Range		Input	Output	Center line	
GS_DPV_FOOTNOTE4 Reserved by USPS for future use.	3						●		
GS_DPV_FOOTNOTE5 Reserved by USPS for future use.	3						●		
GS_DPV_FOOTNOTE6 Reserved by USPS for future use.	3						●		
GS_DPV_NO_STAT • Y - The address is valid for CDS pre-processing. • N - The address is not valid for CDS pre-processing. • Blank - DPV is not loaded or DPV did not confirm.	2								
GS_DPV_SHUTDOWN • Y - Address was found in false-positive table. • N - Address was not found in false-positive table. • Blank - Address was not presented to hash table or DPV data not loaded.	2						●		
GS_DPV_VACANT • Y - The address is vacant. • N - The address is not vacant. • Blank - DPV is not loaded or DPV did not confirm (so vacancy is irrelevant).	3						●		
GS_EWS_MATCH Indicates if GeoStan made an EWS match. • *Y* – Match denied because it matched to EWS data. • *Blank* – Input record did not match to EWS data.	2						●		

	Length[a]	GsHandleGet			GsDataSet	GsDataGet			GsMultipleGet
		Street	Segment	Range		Input	Output	Center line	
`GS_FIRM_NAME` Name of firm from the USPS data or the input firm name.	41			●	●	●	A		A
`GS_GOVT_FLAG` Government building indicator. • *A* – City government building • *B* – Federal government building • *C* – State government building • *D* – Firm-only • *E* – City government building and firm only • *F* – Federal government building and firm only • *G* – State government building and firm only A, B, C, E, F, and G are valid for alternate records only (GS_ALT_FLAG=A). D is valid for both base and alternate records.	2			●			●		A
`GS_HI_RISE_DFLT` Indicates if GeoStan matched to a highrise. • *N* – Matched to an exact highrise record or a street record • *Y* – Did not match to an exact record. Matched to the USPS default highrise record or a street record. Check the input address for accuracy and completeness. • *Blank* – Does not apply to the input address (for example, PO Boxes and General Delivery addresses) or did not find a match.	2						●		A
`GS_HIRANGE` House number at high end of range.	12			●			A		A

	Length[a]	GsHandleGet			GsDataSet	GsDataGet			GsMultipleGet
		Street	Segment	Range		Input	Output	Center line	
`GS_HIUNIT` High unit number.	12			●			A		A
`GS_HIZIP4` High ZIP + 4 for this range.	5			●					A
`GS_HOUSE_NUMBER` House number of input or output address.	12					●	A		●
`GS_HOUSE_NUMBER2` Second house number for a ranged house number match. If GS_FIND_ADDRESS_RANGE is enabled, the second number in the ranged address.	12					●	A		●
`GS_INTERSECTION` Indicates if the address is an intersection. • *T* – Input address is an intersection • *F* – Input address is a street address	2					●	●		●

	Length[a]	GsHandleGet			GsDataSet	GsDataGet			GsMultipleGet
		Street	Segment	Range		Input	Output	Center line	
`GS_IS_ALIAS` The first character is: • *N* – Normal street match • *A* – Alias match (including buildings, aliases, firms, etc.) The next 2 characters are: • *01* – Basic index, normal address match • *02* – USPS street name alias index • *03* – USPS building index • *05* – Statewide intersection alias (when using the `Usw.gsi, Use.gsi,` or `US.gsi` file). • *06* – Spatial data street name alias (when using ent, the `Us_pw.gsi`, `Usw.gsi`, `Us_pe.gsi`, `Use.gsi`, `Us_ps.gsi`, `Usp.gsi`, `Us_psw.gsi`, or `Us_pse.gsi` file is required.) • *07* – Alternate index (when using `Zip9.gsu`, `Zip9e.gsu`, and `Zip9w.gsu`) • *08* – LACS^{Link} • *G09* – Auxiliary file match. **NOTE:** The GIS file is not required if you use **`GsHandleGet`**.	4	●	●	●			A	●	●
`GS_LACS_FLAG` Indicates if GeoStan marked the address for conversion. • *L* – Address marked for LACS conversion. • *Blank* – Address not marked for LACS conversion.	2			●			●		A

	Length[a]	GsHandleGet			GsDataSet	GsDataGet			GsMultipleGet
		Street	Segment	Range		Input	Output	Center line	
GS_LACSLINK_IND LACS$^{\text{Link}}$ indicator. • *Y* – Matched LACS$^{\text{Link}}$ record • *N* – LACS$^{\text{Link}}$ match NOT found • *F* – False-positive LACS$^{\text{Link}}$ record • *S* – Secondary information (unit number) removed to make a LACS$^{\text{Link}}$ match • *Blank* – Not processed through LACS$^{\text{Link}}$	2						●		
GS_LACSLINK_SHUTDOWN • *Y* – False-positive occurred and LACSLink library shutdown. • *N* – LACSLink library has not shutdown or not loaded.	2						●		
GS_LACSLINK_RETCODE LACS$^{\text{Link}}$ return code. • *A* – Matched LACS$^{\text{Link}}$ record • *00* – LACS$^{\text{Link}}$ match NOT found • *09* – Matched to highrise default, but no LACS$^{\text{Link}}$ conversion • *14* – Found LACS$^{\text{Link}}$ match, but no LACS$^{\text{Link}}$ conversion • *92* – Secondary information (unit number) removed to make a LACS$^{\text{Link}}$ match • *Blank* – Not processed through LACS$^{\text{Link}}$	3						●		
GS_LASTLINE[c] Complete last line (ZIP + 4 not available with **GsMultipleGet** or intersection matches).	61			●	●	●	●		●

	Length[a]	GsHandleGet			GsDataSet	GsDataGet			GsMultipleGet
		Street	Segment	Range		Input	Output	Center line	
GS_LASTLINE_SHORT This is the address "lastline". It contains the values of GS_CITY_SHORT, GS_STATE, and GS_ZIP10. Whenever possible, this field is 29 characters or less: • 13-character city name • 2 (comma and space) • 2-character state abbreviation • 2 spaces • 10-digit ZIP code	61						●		
GS_LAT Latitude of located point (in millionths of degrees).	11				●		●	●	●
GS_LINE1 GS_LINE2 GS_LINE3 GS_LINE4 GS_LINE5 GS_LINE6 Used for multiple line entry. GeoStan does not process these as output. PMB_DESIGNATOR and PMB_NUMBER are not returned in this mode.	104					●			
GS_LOC_CODE Location code. For descriptions of location codes, see the *Status Codes* appendix.	5						●		●
GS_LON Longitude of located point (in millionths of degrees).	12				●		●	●	●

	Length[a]	GsHandleGet			GsDataSet	GsDataGet			GsMultipleGet
		Street	Segment	Range		Input	Output	Center line	
GS_LORANGE House number at low end of range.	12			●			A		A
GS_LOT_CODE[b] • *A* – Ascending • *D* – Descending. eLot ascending and descending value. Only available for addresses GeoStan can standardize.	2						●		
GS_LOT_NUM[b] 4-digit eLOT number. Requires an input address GeoStan can standardize.	5						●		
GS_LOUNIT Low unit number.	12			●			A		A
GS_LOZIP4 Low ZIP + 4 for this range.	5			●					A
GS_MAIL_STOP Returns address information appearing after mail stop designator words: MSC, MS, MAILSTOP, MAIL STOP, ATTN, ATTENTION.	61						●		
GS_MATCH_CODE Match code. For descriptions of match codes, see the *Status Codes* appendix.	5						●		●
GS_MATCHED_DB Index of database for matched record.							●		●

	Length[a]	GsHandleGet			GsDataSet	GsDataGet			GsMultipleGet
		Street	Segment	Range		Input	Output	Center line	
GS_MCD_NAME Minor Civil Division name from the auxiliary file. Blank if no auxiliary file match.	41						●		A
GS_MCD_NUMBER Minor Civil Division number from the auxiliary file. Blank if no auxiliary file match.	6						●		A
GS_METRO_FLAG Indicates if GS_CBSA_NAME • *Y* – Contains "Metropolitan Statistical Area" • *N* – Is non-blank but does not contain "Metropolitan Statistical Area" • *Blank* – Is blank (the county does not contain a CBSA).	2						●		●
GS_METRO_FLAG2 GS_METRO_FLAG for the second segment in the intersection.	2								I
GS_MM_RESULT_CODE Returns a MapMarker style result code.	12						●		●
GS_NAME Street name	41	●	●	●		●	●	●	●
GS_NAME2 Second street name in an intersection match.	41					I	I		●
GS_NAME_CITY City name for the matched address from the City State record.	29						●		●

	Length[a]	GsHandleGet			GsDataSet	GsDataGet			GsMultipleGet
		Street	Segment	Range		Input	Output	Center line	
GS_NAME_SHORT The short street name field contains the short street name used to construct the short address line. All attempts are made to abbreviate this name according to the process specified by the USPS in the 30 Character Abbreviation - Cycle M Flow Chart. If an abbreviated address cannot be constructed that is 30 characters or less, this field then contains the same street name value as the GS_NAME field enum return.	41						●		
GS_PARCEN_ELEVATION Elevation of the geocode at the parcel centroid.	7						A,P		A,P
GS_PERCENT_GEOCODE Percentage along the street segment to the interpolated match. The range is 0.0 - 100.0. The range is always 0.0 for point data.	5						R		A
GS_PMB_DESIGNATOR PMB designator (always "PMB"). **NOTE:** Not output when using `GS_LINE` to set the address.	5						●		
GS_PMB_NUMBER PMB number. **NOTE:** Not output when using `GS_LINE` to set the address.	9						●		

	Length[a]	GsHandleGet			GsDataSet	GsDataGet			GsMultipleGet
		Street	Segment	Range		Input	Output	Center line	
GS_POINT_ID Unique point ID of the matched record when matched to point-level data. *Blank* if the matched record is not from point-level data. NOTE: This variable is also available for other point datasets, i.e.TeleAtlas.	11		A,P				A,P		A,P
GS_POSTDIR Postfix direction. Can be blank, N, S, E, W, NE, NW, SW, or SE.	3	●	●	●		●	●	●	●
GS_POSTDIR2 Cross street postfix direction.	3					I	I		I
GS_POSTDIR_SHORT Postdir from the GS_ADDRLINE_SHORT.	3						●		
GS_PREDIR Prefix direction. Can be blank, N, S, E, W, NE, NW, SW, or SE.	3	●	●	●		●	●	●	●
GS_PREDIR2 Cross street prefix direction.	3					I	I		I
GS_PREDIR_SHORT Predir from the GS_ADDRLINE_SHORT.	3						●		●
GS_PREF_CITY Preferred city name for the output ZIP Code of the matched address.	29						●		●

	Length[a]	GsHandleGet			GsDataSet	GsDataGet			GsMultipleGet
		Street	Segment	Range		Input	Output	Center line	
GS_QCITY State , city , or finance numbers.	10	●	●	●			●	●	●
GS_R_RTE_DFLT Match indication for rural routes. • *N* – Matched to an exact rural route record. • *Y* – Did not find an exact record. Matched to the USPS default rural route record. Check the input address for accuracy and completeness. • *Blank* – Does not apply to the input address (for example, street addresses, P.O. Boxes, and General Delivery addresses) or no match found.	2						●		A
GS_RANGE_PARITY Indicates the parity of the house number in the range. • *E* – Even • *O* – Odd • *B* – Both	41			●					A

	Length[a]	GsHandleGet			GsDataSet	GsDataGet			GsMultipleGet
		Street	Segment	Range		Input	Output	Center line	
GS_REC_TYPE[b] Range record type. • A – Auxiliary file • *F* – Firm • *G* – General delivery • *H* – Highrise • *P* – PO Box • *R* – Rural route/highway contract • *S* – Street • *T* – TIGER file • *U* - User Dictionary	2			●			A		A
GS_RESOLVED_LINE indicates which line in a 2-line address GeoStan used to resolve the address. Relates to GS_ADDRLINE and GS_ADDR2 with **GsDataSet**.	5						●		
GS_ROAD_CLASS Road class code. • *0* – Minor road, main data file • *1* – Major road, main data file • *10* – Minor road, supplemental file • *11* – Major road, supplemental data file	3		●	●			A	●	●
GS_ROAD_CLASS2 GS_ROAD_CLASS for the second segment in the intersection.	3								I

	Length[a]	GsHandleGet			GsDataSet	GsDataGet			GsMultipleGet
		Street	Segment	Range		Input	Output	Center line	
`GS_SEG_HIRANGE` High house number in segment.	12		●	●				●	●
`GS_SEG_HIRANGE2` `GS_SEG_HIRANGE` for the second segment in the intersection.	12								I
`GS_SEG_LORANGE` Low house number in segment.	12		●	●				●	●
`GS_SEG_LORANGE2` `GS_SEG_LORANGE` for the second segment in the intersection.	12								I
`GS_SEGMENT_DIRECTION` • *F* – Numbers are forward • *R* – Numbers are reversed	2		●	●				●	●
`GS_SEGMENT_DIRECTION2` `GS_SEGMENT_DIRECTION` for the second segment in the intersection.	2								I
`GS_SEGMENT_ID` Segment ID (TLID) or unique ID from premium data vendors.	11		●	●			A	●	●
`GS_SEGMENT_ID2` `GS_SEGMENT_ID` for the second segment in the intersection.	11								I

	Length[a]	GsHandleGet			GsDataSet	GsDataGet			GsMultipleGet
		Street	Segment	Range		Input	Output	Center line	
GS_SEGMENT_PARITY Odd numbers in the segment are on: • *L* – Left side of the street • *R* – Right side of the street • *B* – Both sides of the street • *U* – Unknown	2		●	●				●	●
GS_SEGMENT_PARITY2 GS_SEGMENT_PARITY for the second segment in the intersection.	2								I
GS_STATE State abbreviation.	3	●	●	●	●	●	●		●
GS_STATE_FIPS State FIPS code.	3				●				
GS_STREET_SIDE The matched address is on the following side of the street: • *L* – Left side of the street • *R* – Right side of the street • *B* – Both sides of the street • *U* – Unknown side of the street This is relative to the segment end points and the segment direction (GS_SEGMENT_DIRECTION). NOTE: This enum does not reflect the value in the "Side of Street" field of any particular data vendor.	2						●		A

	Length[a]	GsHandleGet			GsDataSet	GsDataGet			GsMultipleGet
		Street	Segment	Range		Input	Output	Center line	
GS_SUITELINK_RET_CODE • A - Suite$^{\text{Link}}$ record match. • 00 - No Suite$^{\text{Link}}$ match. • Blank - This address was not processed through Suite$^{\text{Link}}$.	4						A		
GS_TYPE Street type (also called street suffix).	5	●	●	●		●	●	●	●
GS_TYPE2 Cross street type.	5					I	I		I
GS_TYPE_SHORT Postdir from the GS_ADDRLINE_SHORT.	6						●		
GS_UNIT_NUMBER Unit number.	12					A	A		A
GS_UNIT_NUMBER2 Second unit number parsed from the address line. Available in CASS and relaxed modes only.	12					A	A		
GS_UNIT_PARITY Indicates the parity of the unit number range (**GS_LOUNIT** and **GS_HIUNIT**). • *E* – Even • *O* – Odd • *B* – Both	2			●					A
GS_UNIT_TYPE Unit type (APT, STE, etc.).	5			●		A	A		A

	Length[a]	GsHandleGet			GsDataSet	GsDataGet			GsMultipleGet
		Street	Segment	Range		Input	Output	Center line	
GS_UNIT_TYPE2 Second unit type parsed from the address line. Available in CASS and relaxed modes only.	5					A	A		
GS_URB_NAME Urbanization name for Puerto Rico.	31				●	●	●		A
GS_ZIP 5-digit ZIP Code.	6			●	●	A	A		●
GS_ZIP4[b] 4-digit ZIP Code extension.	5				●	A	A		A
GS_ZIP9[c] 9-digit ZIP Code (ZIP + 4).	10				●	A	A		A
GS_ZIP10[c] 9-digit ZIP Code (ZIP + 4) with dash separator.	11				●	A	A		A
GS_ZIP_CARRTSORT Indicates the type of cart sort allowed. • *A* – Automation cart allowed, optional cart merging allowed. • *B* – Automation cart allowed, optional cart merging not allowed. • *C* – Automation cart not allowed, optional cart merging allowed. • *D* – Automation cart not allowed, optional cart merging not allowed.	2			●			A		●

	Length[a]	GsHandleGet			GsDataSet	GsDataGet			GsMultipleGet
		Street	Segment	Range		Input	Output	Center line	
GS_ZIP_CITYDELV Indicates if city-delivery is available. • Y – Post office has city-delivery carrier routes • N – Post office does not have city-delivery	2			●			A		●
GS_ZIP_CLASS ZIP Classification Code: • Blank – Standard ZIP Code • M – Military ZIP Code • P – ZIP Code has P.O. Boxes only • U – Unique ZIP Code. (A unique ZIP Code is a ZIP Code assigned to a company, agency, or entity with sufficient mail volume to receive its own ZIP Code.)	2			●			●		●

	Length[a]		GsHandleGet		GsDataSet		GsDataGet		GsMultipleGet
		Street	Segment	Range		Input	Output	Center line	
GS_ZIP_FACILITY Returns USPS City State Name Facility Code. • *A* – Airport Mail Facility (AMF) • *B* – Branch • *C* – Community Post Office (CPO) • *D* – Area Distribution Center (ADC) • *E* – Sectional Center Facility (SCF) • *F* – Delivery Distribution Center (DDC) • *G* – General Mail Facility (GMF) • *K* – Bulk Mail Center (BMC) • *M* – Money Order Unit • *N* – Non-Postal Community Name, Former Postal Facility, or Place Name • *P* – Post Office • *S* – Station • *U* – Urbanization	2			●			●		●
GS_ZIP_UNIQUE • *Y* – unique ZIP Code. (ZIP Code assigned to a company, agency, or entity with sufficient mail volume to receive its own ZIP Code.) • *Blank* – Not applicable.	2			●			●		●

a. The length column is not applicable.

b. Blank if running in CASS mode and you have not initialized DPV or the output address does not DPV confirm.

c. Last 4 digits are blank if running in CASS mode and you have not initialized DPV or the output address does not DPV confirm. For GsMultipleGet(), GeoStan returns ZIP+4 but only for Address matches.

GsFindWithProps properties

NOTE: Find Boolean Type properties have a default value of False when the property is not in the find property list, except for GS_FIND_MUST_MATCH_ADDRNUM.

Conflicting API functions

Detailed here are the categories and categories that conflict in the GsFindWithProps properties listed below.

GeoStan applications control the behavior of GsFindWithProps by setting any of the find properties. You can find the complete list of these find properties in table GsFindWithProps properties. These properties naturally fall into several categories. Rules exist that control which categories of properties can intermix. The tables below show which property categories conflict. Those categories that are not shown to be in conflict can be used together.

GeoStan detects conflicting properties and returns appropriate error messages from the several GsPropSet* functions. This prevents illegal, inconsistent, conflicting, or confusing property combinations from occurring. By guaranteeing fully consistent property lists, this allows the GeoStan engine code to have a clear path to perform the matching logic.

By clearly defining the categories into which the properties fall, and which categories of properties are for use together, this aids in familiarizing you in the semantics and capabilities of the properties. The categories are described below:

Category	Description
Any	No restrictions on the use of this property with other properties. In other words, a property in this category can be used in combination with other properties from a similar property type (init, status, or find).
AnyForward	Find properties that are usable with any use of GsFindWithProps which is doing "forward geocoding", that is, standardizing and geocoding an input address.
GeoStanReverse	Find properties that are usable with any use of GsFindWithProps which is doing a reverse geocoding search.
GeoStanAPN	Find properties that are usable with any use of GsFindWithProps which is doing an APN search.

Category	Description
GeoStanFindFirstStreet	Find properties that are usable with any use of GsFindWithProps which is doing an FindFirstStreet/Segment/Range search.
Custom	Find properties that are usable with any use of GsFindWithProps which is using custom find properties.

The properties with the categories listed below in the Category column cannot be used with the properties with categories listed in the Conflicting Category column.

Category	Conflicting category
Any	None.
AnyForward	GeoStanReverse GeoStanAPN GeoStanFindFirstStreet
GeoStanReverse	GeoStanAPN GeoStanFindFirstStreet Custom AnyForward
GeoStanAPN	GeoStanFindFirstStreet Custom AnyForward GeoStanReverse
GeoStanFindFirstStreet	Custom AnyForward GeoStanReverse GeoStanAPN
Custom	GeoStanReverse GeoStanAPN GeoStanFindFirstStreet

GeoStan does not allow an application to add a property from the category listed in the Category column, if the property list already contains a MatchMode property type listed in the MatchMode value column below:

Category	MatchMode value
Custom	Relax Close Exact
GeoStanReverse GeoStanAPN GeoStanFindFirstStreet	Custom

GeoStan does not allow an application to change the MatchMode value to those listed in the MatchMode value column, if the property list already contains at least on property from any of the categories listed in the Category column below:

MatchMode value	Property
Relax Close Exact Custom	Custom
Custom	AnyForward GeoStanReverse GeoStanAPN GeoStanFindFirstStreet
CASS	BuildingSearch PreferStreet PreferPOBox AddressRange StreetCentroid Any MustMatch* property

When the MatchMode is already set to CASS, none of the properties listed in the Property column can be set. Similarly, if any of the properties in the Property column are already in the property list, the MatchMode cannot be set to CASS.

This early property conflict detection feature prevents the conflicts described in this section from occurring.

Properties formerly set with GsFind:

GS_FIND_ADDRCODE

Type	Boolean
Description	Attempts to standardize and find an address geocode. Set this switch if you want the address parsed and standardized. If this switch is not set, GeoStan uses only the input ZIP and ZIP + 4 address elements.
Category	AnyForward
Usage with other properties in other categories.	Compatible with: AnyForward and Any. Conflicts with: GeoStanReverse, GeoStanAPN, and GeoStanFindFirstStreet.
Usable match modes	Relax, Close, Exact, and CASS.
Values	True or False
Default value	False
Relation to deprecated feature	Overrides GS_ADDR_CODE.

GS_FIND_WIDE_SEARCH

Type	Boolean
Description	Considers all records matching the first letter of the street name, rather than the Soundex key on the street name, which allows a wider search. This option has no effect when performing a ZIP centroid match.
Category	Any
Usage with other properties in other categories.	Compatible with: AnyForward, Any, Custom, GeoStanReverse, GeoStanAPN, and GeoStanFindFirstStreet. Conflicts with: none.
Usable match modes	Relax, Close, and Exact.
Values	True or False
Default value	False
Relation to deprecated feature	Overrides GS_WIDE_SEARCH.

GS_FIND_SEARCH_AREA

Type	Long
Description	Assists in finding a match when the input address contains limited or inaccurate city or ZIP Code information.
Category	Any
Usage with other properties in other categories.	Compatible with: Any, AnyForward, GeoStanReverse, GeoStanAPN, and GeoStanFindFirstStreet. Conflicts with: None.
Usable match modes	Relax, Close, and Exact.
Values	0 - GS_FIND_SEARCH_AREA_CITY - Searches the specified city. 1 - GS_FIND_SEARCH_AREA_FINANCE - Searches the entire Finance Area for possible streets. This option has no effect when performing a ZIP centroid match. 2 - GS_FIND_SEARCH_AREA_EXPANDED - This value effectively has two options that can be set: Allows the setting of the radius in miles (up to 99) around which your record lies. The default radius setting is 25 miles. Allows for limiting the search to the state. The default setting is True. For more information see Search area designation.
Default value	0
Relation to deprecated feature	Overrides GS_FINANCE_SEARCH and GS_FIND_EXPANDED_SEARCH.

GS_FIND_BUILDING_SEARCH

Type	Boolean
Description	Controls the ability to search by building name entered in the address line. Enables matching to building names even when no unit numbers are present.
Category	AnyForward
Usage with other properties in other categories.	Compatible with: AnyForward and Any. Conflicts with: GeoStanReverse, GeoStanAPN, and GeoStanFindFirstStreet.
Usable match modes	Relax, Close, and CASS.
Values	True or False

Default value	False
Relation to deprecated feature	Overrides GS_BUILDING_SEARCH.

GS_FIND_CENTRLN_PROJ_OF_POINT

Type	Boolean
Description	Enables GeoStan to keep multiple candidate records when matching with point-level data for use with centerline matching. Not valid when using the reverse geocoding options.
Category	AnyForward
Usage with other properties in other categories.	Compatible with: AnyForwardand Any. Conflicts with: GeoStanReverse, GeoStanAPN, and GeoStanFindFirstStreet.
Usable match modes	Relax, Close, Exact, and CASS.
Values	True or False
Default value	False
Relation to deprecated feature	Overrides GS_ALWAYS_FIND_CANDIDATES.

GS_FIND_Z9_CODE

Type	Boolean
Description	Attempts to find a ZIP + 4 centroid match only. Not valid when using the reverse geocoding options.
Category	AnyForward
Usage with other properties in other categories.	Compatible with: AnyForwardand Any. Conflicts with: GeoStanReverse, GeoStanAPN, and GeoStanFindFirstStreet.
Usable match modes	Relax, Close, Exact, and CASS.
Values	True or False
Default value	False
Relation to deprecated feature	Overrides GS_Z9_CODE.

GS_FIND_Z7_CODE

Type	Boolean
Description	Attempts to find a ZIP+2 centroid match only (no ZIP + 4 or ZIP). Not valid when using the reverse geocoding options.
Category	AnyForward
Usage with other properties in other categories.	Compatible with: AnyForward and Any. Conflicts with: GeoStanReverse, GeoStanAPN, and GeoStanFindFirstStreet.
Usable match modes	Relax, Close, Exact, and CASS.
Values	True or False
Default value	False
Relation to deprecated feature	Overrides GS_Z7_CODE.

GS_FIND_Z5_CODE

Type	Boolean
Description	Attempts to find a ZIP centroid match (no ZIP + 4 or ZIP+2). Not valid when using the reverse geocoding options.
Category	AnyForward
Usage with other properties in other categories.	Compatible with: AnyForward and Any. Conflicts with: GeoStanReverse, GeoStanAPN, and GeoStanFindFirstStreet.
Usable match modes	Relax, Close, Exact, and CASS.
Values	True or False
Default value	False
Relation to deprecated feature	Overrides GS_Z5_CODE.

GS_FIND_Z_CODE

Type	Boolean
Description	Attempts to find any ZIP centroid match. Not valid when using the reverse geocoding options.
Category	AnyForward

Usage with other properties in other categories.	Compatible with: AnyForward and Any. Conflicts with: GeoStanReverse, GeoStanAPN, and GeoStanFindFirstStreet.
Usable match modes	Relax, Close, Exact, and CASS.
Values	True or False
Default value	False
Relation to deprecated feature	Overrides GS_Z_CODE.

GS_FIND_PREFER_POBOX

Type	Boolean
Description	Sets the preference to P.O. Box addresses instead of street addresses for multi-line input addresses. See "Specifying a preference for street name or P.O. Box" on page 36 for more information. Ignored if processing in CASS mode. If both GS_FIND_PREFER_POBOX and GS_FIND_PREFERSTREET are set to True, then they are ignored and the default, GS_FIND_PREFER_STREET is used.
Category	AnyForward
Usage with other properties in other categories.	Compatible with: AnyForward and Any. Conflicts with: GeoStanReverse, GeoStanAPN, and GeoStanFindFirstStreet.
Usable match modes	Relax, Close, and Exact.
Values	True or False
Default value	False
Relation to deprecated feature	Overrides GS_PREFER_POBOX.

GS_FIND_PREFER_STREET

Type	Boolean
Description	Sets the preference to street addresses instead of P.O. Box addresses for multi-line input addresses. Ignored if processing in CASS mode. If both GS_FIND_PREFER_POBOX and GS_FIND_PREFERSTREET are set to True, then they are ignored and the default, GS_FIND_PREFER_STREET is used.
Category	AnyForward

Usage with other properties in other categories.	Compatible with: AnyForward and Any. Conflicts with: GeoStanReverse, GeoStanAPN, GeoStanFindFirstStreet and Custom.
Usable match modes	Relax, Close, and Exact.
Values	True or False
Default value	False
Relation to deprecated feature	Overrides GS_PREFER_STREET.

GS_FIND_NEAREST_ADDRESS

Type	Boolean
Description	Specifies that GeoStan can match to addresses interpolated on street segments or to point data locations. You can use GS_FIND_NEAREST_ADDRESS and GS_FIND_NEAREST_INTERSECTION together to specify reverse geocoding to both addresses and intersections. For reverse geocoding, you need to set the reverse geocoding processing find properties: GS_FIND_NEAREST_ADDRESS, GS_FIND_NEAREST_INTERSECTION, and/or GS_FIND_NEAREST_UNRANGED. The address processing and multi-line address processing options are not valid for reverse geocoding.
Category	GeoStanReverse
Usage with other properties in other categories.	Compatible with: Reverse and Any. Conflicts with: GeoStanAPN, GeoStanFindFirstStreet, Custom, AnyForward
Usable match modes	Relax, Close, Exact, and CASS.
Values	True or False
Default value	False
Relation to deprecated feature	Overrides GS_NEAREST_ADDRESS.

GS_FIND_NEAREST_INTERSECTION

Type	Boolean
Description	Specifies that GeoStan can match to intersections. You can use GS_FIND_NEAREST_INTERSECTION and GS_FIND_NEAREST_ADDRESS together to specify reverse geocoding to both addresses and intersections. For reverse geocoding, you need to set the reverse geocoding processing find properties:GS_FIND_NEAREST_INTERSECTION, GS_FIND_NEAREST_ADDRESS, and/or GS_FIND_NEAREST_UNRANGED. The address processing and multi-line address processing options are not valid for reverse geocoding.
Category	GeoStanReverse
Usage with other properties in other categories.	Compatible with: Reverse and Any. Conflicts with: GeoStanAPN, GeoStanFindFirstStreet, Custom, AnyForward
Usable match modes	Relax, Close, Exact, and CASS.
Values	True or False
Default value	False
Relation to deprecated feature	Overrides GS_NEAREST_INTERSECTION.

GS_FIND_NEAREST_UNRANGED

Type	Boolean
Description	Specifies that GeoStan can match a street segment with no number ranges. Enabled with GS_FIND_NEAREST_ADDRESS. Ignored for point data and intersection matches.
Category	GeoStanReverse
Usage with other properties in other categories.	Compatible with: Reverse and Any. Conflicts with: GeoStanAPN, GeoStanFindFirstStreet, Custom, AnyForward
Usable match modes	Relax, Close, Exact, and CASS.
Values	True or False
Default value	False
Relation to deprecated feature	Overrides GS_NEAREST_UNRANGED.

GS_FIND_APN_SEARCH

Type	Boolean
Description	Specifies that GeoStan match an input FIPS and APN code to receive information on the corresponding parcel
Category	GeoStanAPN
Usage with other properties in other categories.	Compatible with: GeoStanAPN and Any. Conflicts with: GeoStanFindFirstStreet, AnyForward, and GeoStanReverse.
Usable match modes	Relax, Close, and CASS.
Values	True or False
Default value	False
Relation to deprecated feature	Overrides GS_APN_SEARCH.

Properties formerly set with GsDataSet:

GS_FIND_SEARCH_DIST

Type	Long
Description	Radius, in feet, that GeoStan searches for a reverse geocode match. The range is 0 - 5280 feet.
Category	GeoStanReverse
Usage with other properties in other categories.	Compatible with: GeoStanReverse and Any. Conflicts with: GeoStanAPN, GeoStanFindFirstStreet, and AnyForward
Usable match modes	Relax, Close, and CASS.
Values	Any positive integer, which represents number of feet
Default value	150 feet.
Relation to deprecated feature	Overrides enum value, GS_SEARCH_DIST.

GS_FIND_STREET_OFFSET

Type	Long
Description	Left and right offset distance from the street segments. The range is 0 - 999 feet.
Category	Any
Usage with other properties in other categories.	Compatible with: AnyForward, Any, Custom, GeoStanReverse, GeoStanAPN, and GeoStanFindFirstStreet.. Conflicts with: None.
Usable match modes	Custom.
Values	Any positive integer, which represents number of feet
Default value	50
Relation to deprecated feature	Overrides enum value, GS_OFFSET_DIST.

GS_FIND_CENTERLINE_OFFSET

Type	Long
Description	Offset distance from the street center for a centerline match.
Category	AnyForward
Usage with other properties in other categories.	Compatible with: AnyForward and Any. Conflicts with: GeoStanReverse, GeoStanAPN, and GeoStanFindFirstStreet.
Usable match modes	Relax, Close, Exact, and CASS.
Values	Any positive integer, which represents number of feet
Default value	0 feet.
Relation to deprecated feature	Overrides enum value, GS_CENTERLINE_OFFSET.

GS_FIND_CORNER_OFFSET

Type	Long
Description	Distance, in feet, to offset address-level geocodes from the street end points. If the corner offset distance is more than half the segment length, GeoStan sets the distance to the midpoint of the segment.
Category	Any

Usage with other properties in other categories.	Compatible with: AnyForward, Any, Custom, GeoStanReverse, GeoStanAPN, and GeoStanFindFirstStreet. Conflicts with: None.
Usable match modes	Custom.
Values	
Default value	50
Relation to deprecated feature	Overrides enum value, GS_SQUEEZE_DIST.

Properties formerly set with their own individual API functions:

GS_FIND_MATCH_MODE

Type	Long
Description	Controls the closeness-of-match rules used for matching in GsFindWithProps. This property affects how GsDataSet performs, specifically how the input address is parsed. For this reason, only alter this find property before calling GsDataSet to input the address and lastline data. If you alter this property after calling GsDataSet, the results of the subsequent find will be unpredictable.
Category	AnyForward
Usage with other properties in other categories.	Compatible with: AnyForward and Any. Conflicts with: GeoStanReverse, GeoStanAPN, and GeoStanFindFirstStreet.
Usable match modes	Relax, Close, Exact, and CASS.
Values	GS_MODE_EXACT: Requires an exact name match. Generates the fewest number of possibles to search. GS_MODE_CLOSE: Requires a very close name match. Generates a moderate number of possibles to search GS_MODE_RELAX: Requires a close name match. Generates the largest number of possibles to search. GS_MODE_CASS: Requires a close name match. Generates a modest number of possibles to search. This setting imposes additional rules to ensure compliance with the USPS regulations for CASS software. For example, this mode disables intersection matching, and matching to the User Dictionary matches for standardization GS_MODE_CUSTOM: Allows applications to specify individual "must match" field matching rules for address number, address line, city, ZIP Code, state.

Default value	GS_MODE_CLOSE
Relation to deprecated feature	Overrides function GsSetMatchMode().

GS_FIND_CLIENT_CRS

Type	String
Description	
Category	Any
Usage with other properties in other categories.	Compatible with: AnyForward, Any, Custom, GeoStanReverse, GeoStanAPN, and GeoStanFindFirstStreet. Conflicts with: None.
Usable match modes	Custom.
Values	NAD27 and NAD83.
Default value	NAD83.
Relation to deprecated feature	Overrides functions GsSetDatum and GsGetDatum.

GS_FIND_MIXED_CASE

Type	Boolean
Description	Returns results in mixed case.
Category	Any
Usage with other properties in other categories.	Compatible with: AnyForward, Any, Custom, GeoStanReverse, GeoStanAPN, and GeoStanFindFirstStreet. Conflicts with: none.
Usable match modes	Custom.
Values	True or False
Default value	False
Relation to deprecated feature	Overrides function GsSetMixedCase.

GS_FIND_STREET_CENTROID

Type	Boolean
Description	Turns on street locator geocoding.

Category	AnyForward
Usage with other properties in other categories.	Compatible with: AnyForward, Any. Conflicts with: GeoStanReverse, GeoStanAPN, and GeoStanFindFirstStreet..
Usable match modes	Relax, Close, and Exact.
Values	True or False
Default value	False
Relation to deprecated feature	Overrides function GsSetStreetCentroid.

Custom properties:

GS_FIND_MUST_MATCH_CITY

Type	Boolean
Description	Candidates must match city.
Category	Custom
Usage with other properties in other categories.	• Compatible with: AnyForward and Any. • Conflicts with: GeoStanReverse, GeoStanAPN, and GeoStanFindFirstStreet.
Usable match modes	Custom and CASS.
Values	True or False
Default value	False
Relation to deprecated feature	None

GS_FIND_MUST_MATCH_ADDRNUM

Type	Boolean
Description	Candidates must match house number exactly.
Category	Custom
Usage with other properties in other categories.	• Compatible with: AnyForward and Any. • Conflicts with: GeoStanReverse, GeoStanAPN, and GeoStanFindFirstStreet.

Usable match modes	Custom and CASS.
Values	True or False
Default value	True
Relation to deprecated feature	None

GS_FIND_MUST_MATCH_MAINADDR

Type	Boolean
Description	Candidates must match main address exactly.
Category	Custom
Usage with other properties in other categories.	• Compatible with: AnyForward and Any. • Conflicts with: GeoStanReverse, GeoStanAPN, and GeoStanFindFirstStreet.
Usable match modes	Custom and CASS.
Values	True or False
Default value	False
Relation to deprecated feature	None

GS_FIND_MUST_MATCH_STATE

Type	Boolean
Description	Candidates must match state.
Category	Custom
Usage with other properties in other categories.	• Compatible with: AnyForward and Any. • Conflicts with: GeoStanReverse, GeoStanAPN, and GeoStanFindFirstStreet.
Usable match modes	Custom and CASS.
Values	True or False
Default value	False
Relation to deprecated feature	None

GS_FIND_MUST_MATCH_ZIPCODE

Type	Boolean
Description	Candidates must match ZIP Code.
Category	Custom
Usage with other properties in other categories.	• Compatible with: AnyForward and Any. • Conflicts with: GeoStanReverse, GeoStanAPN, and GeoStanFindFirstStreet.
Usable match modes	Custom and CASS.
Values	True or False
Default value	False
Relation to deprecated feature	None

Properties exclusive to GsFindFirstStreet

GS_FIND_ZIP_SEARCH

Type	Boolean
Description	GeoStan searches the ZIP Code specified by pLocale. When this option is set with GsFindFirstStreet, GeoStan performs a finance search.
Category	GeoStanFindFirstStreet
Usage with other properties in other categories.	• Compatible with: GeoStanFindFirstStreet and Any. • Conflicts with:Custom, AnyForward, GeoStanReverse, and GeoStanAPN.
Usable match modes	Relax, Close, and Exact.
Values	True or False
Default value	
Relation to deprecated feature	Overrides enum GS_ZIP_SEARCH.

GS_FIND_CITY_SEARCH

Type	Boolean
Description	GeoStan searches the city and state specified by `pLocale`.

Category	GeoStanFindFirstStreet
Usage with other properties in other categories.	• Compatible with: GeoStanFindFirstStreet and Any. • Conflicts with:Custom, AnyForward, GeoStanReverse, and GeoStanAPN.
Usable match modes	Relax, Close, and Exact.
Values	True or False
Default value	
Relation to deprecated feature	Overrides enum GS_CITY_SEARCH.

GS_FIND_SDX_SEARCH

Type	Boolean
Description	GeoStan searches by Soundex. *pName* is a pointer to a numeric soundex key returned by **GsSoundex**. If you do not specify this option, GeoStan assumes that *pName* is a street name.
Category	GeoStanFindFirstStreet
Usage with other properties in other categories.	• Compatible with: GeoStanFindFirstStreet and Any. • Conflicts with:Custom, AnyForward, GeoStanReverse, and GeoStanAPN.
Usable match modes	Relax, Close, and Exact.
Values	True or False
Default value	
Relation to deprecated feature	Overrides enum GS_SDX_SEARCH.

GS_FIND_EXTRA_SEARCH

Type	Boolean
Description	GeoStan searches the primary street files, `Use.gsd` and `Usw.gsd`, and the supplemental street files, `Ustw.gsd` and `Uste.gsd`, for streets without house numbers. If you specify this option, `pNumber` must be blank. If you specify *pNumber*, GeoStan searches only the primary street files, `Usw.gsd` and `Use.gsd`, which contain addressed segments. Assign the directory for this file to the `pGeoPaths` member of `GsInitStruct` when calling **`GsInitWithProps`**. NOTE: MVS customers receive one primary street file, `Us.gsd`, and one supplemental street file, `Ust.gsd`.
Category	GeoStanFindFirstStreet

Usage with other properties in other categories.	• Compatible with: GeoStanFind, GeoStanFindFirstStreet, and Any. • Conflicts with:Custom, AnyForward, GeoStanReverse, and GeoStanAPN.
Usable match modes	Relax, Close, and Exact.
Values	True or False
Default value	
Relation to deprecated feature	Overrides enum GS_EXTRA_SEARCH.

New GeoStan Find properties

GS_FIND_DB_ORDER

Type	String
Description	List of database index values [starting at 0, separated by semi-colons] indicating which to search and in what order. See also the related functions GsGetNumDB() and GsGetDBMetadata(), and GS_MATCHED_DB. If this property is not set, the behavior is to search the DBs in the following order: – Aux files – UDs of all types (street and point) – Point files (both Centrus points and TANA points) – GSD (TANA/GDT and Navteq) – GSD (TIGER) – GSD (USPS)
Category	Any
Usage with other properties in other categories.	• Compatible with: AnyForward, Any, GeoStanReverse, GeoStanAPN, and GeoStanFindFirstStreet. • Conflicts with: None.
Usable match modes	Custom.
Values	Empty string.
Default value	“”
Relation to deprecated feature	None

GS_FIND_EXPANDED_SEARCH_RADIUS

Type	Long
Description	Distance, in feet, to use for the expanded search radius.
Category	Any
Usage with other properties in other categories.	• Compatible with: AnyForward, Any, Custom, GeoStanReverse, GeoStanAPN, and GeoStanFindFirstStreet. • Conflicts with: None.
Usable match modes	Relax, Close, Exact, and Custom.
Values	Any positive number.
Default value	132000
Relation to deprecated feature	None

GS_FIND_EXPND_SRCH_LIM_TO_STATE

Type	Boolean
Description	Do not cross state boundaries when doing an expanded search.
Category	Any
Usage with other properties in other categories.	• Compatible with: AnyForward, Any, Custom, GeoStanReverse, GeoStanAPN, and GeoStanFindFirstStreet. • Conflicts with: None.
Usable match modes	Relax, Close, Exact, and Custom.
Values	True or False
Default value	True
Relation to deprecated feature	None

GS_FIND_DPV

Type	Boolean
Description	Turn on DPV mode.
Category	AnyForward

Usage with other properties in other categories.	- Compatible with: AnyForward, and Any. - Conflicts with: GeoStanReverse, GeoStanAPN, and GeoStanFindFirstStreet.
Usable match modes	Custom and CASS.
Values	True or False
Default value	True
Relation to deprecated feature	None

GS_FIND_ALTERNATE_LOOKUP

Type	Long
Description	Determines whether the preferred lookup is to look for the streets first or the firms first.
Category	AnyForward
Usage with other properties in other categories.	- Compatible with: AnyForward and Any. - Conflicts with: GeoStanReverse, GeoStanAPN, and GeoStanFindFirstStreet.
Usable match modes	Relax, Close, Exact, and CASS.
Values	- 1 - GS_PREFER_STREET_LOOKUP - Matches to the address line, if a match is not made, then GeoStan matches to the Firm name line. - 2 - GS_PREFER_FIRM_LOOKUP - Matches to the Firm name line, if a match is not made, then GeoStan matches to address line. - 3 - GS_STREET_LOOKUP_ONLY (default) - Matches to the address line.
Default value	3
Relation to deprecated feature	None

GS_FIND_CORRECT_LASTLINE

Type	Boolean
Description	Corrects the output last line.
Category	AnyForward
Usage with other properties in other categories.	- Compatible with: AnyForward and Any. - Conflicts with: GeoStanReverse, GeoStanAPN, and GeoStanFindFirstStreet.

Usable match modes	Relax, Close, Exact, and CASS.
Values	True or False
Default value	
Relation to deprecated feature	None

GS_FIND_LACSLINK

Type	Boolean
Description	Turns on LACS$^{\text{Link}}$ mode.
Category	AnyForward
Usage with other properties in other categories.	• Compatible with: AnyForward and Any. • Conflicts with: GeoStanReverse, GeoStanAPN, and GeoStanFindFirstStreet.
Usable match modes	Relax, Close, Exact, and CASS.
Values	True or False
Default value	True
Relation to deprecated feature	None

GS_FIND_PREFER_ZIP_OVER_CITY

Type	Boolean
Description	Prefer candidates matching input ZIP over matches to input city.
Category	AnyForward
Usage with other properties in other categories.	• Compatible with: AnyForward and Any. • Conflicts with: GeoStanReverse, GeoStanAPN, and GeoStanFindFirstStreet.
Usable match modes	Relax, Close, Exact, and CASS.
Values	True or False
Default value	True
Relation to deprecated feature	None

GS_FIND_RET_INTERSECTION_NUM

Type	Boolean
Description	Returns intersection numbers.
Category	AnyForward
Usage with other properties in other categories.	• Compatible with: AnyForward and Any. • Conflicts with: GeoStanReverse, GeoStanAPN, and GeoStanFindFirstStreet.
Usable match modes	Relax, Close, Exact, and CASS.
Values	True or False
Default value	False
Relation to deprecated feature	None

GS_FIND_POINT_ZIP_MATCH

Type	Boolean
Description	Enables point ZIP matching.
Category	AnyForward
Usage with other properties in other categories.	• Compatible with: AnyForward andAny. • Conflicts with: GeoStanReverse, GeoStanAPN, and GeoStanFindFirstStreet.
Usable match modes	Relax, Close, Exact, and CASS.
Values	True or False
Default value	False
Relation to deprecated feature	None

GS_FIND_ADDRPOINT_INTERP

Type	Boolean
Description	Enables address point interpolation.
Category	AnyForward

Usage with other properties in other categories.	• Compatible with: AnyForward and Any. • Conflicts with: GeoStanReverse, GeoStanAPN, and GeoStanFindFirstStreet.
Usable match modes	Relax, Close, Exact, and CASS.
Values	True or False
Default value	False
Relation to deprecated feature	None

GS_FIND_FIRST_LETTER_EXPANDED

Type	Boolean
Description	Enables extra processing for bad first letter (missing, wrong, etc.).
Category	AnyForward
Usage with other properties in other categories.	• Compatible with: AnyForward and Any. • Conflicts with: GeoStanReverse, GeoStanAPN, and GeoStanFindFirstStreet.
Usable match modes	Relax, Close, Exact, and CASS.
Values	True or False
Default value	False
Relation to deprecated feature	None

GS_FIND_ADDRESS_RANGE

Type	Boolean
Description	Enables Map Marker address range geocoding compatibility.
Category	AnyForward
Usage with other properties in other categories.	• Compatible with: AnyForward and Any. • Conflicts with: GeoStanReverse, GeoStanAPN, and GeoStanFindFirstStreet.
Usable match modes	Relax, Close, Exact, and CASS.
Values	True or False

Default value	False
Relation to deprecated feature	None

GS_FIND_SUITELINK

Type	Boolean
Description	Enables Suite$^{\text{Link}}$ mode.
Category	AnyForward
Usage with other properties in other categories.	• Compatible with: AnyForward, and Any. • Conflicts with: GeoStanReverse, GeoStanAPN, and GeoStanFindFirstStreet.
Usable match modes	Custom and CASS.
Values	True or False
Default value	True
Relation to deprecated feature	None

GsInitWithProps input properties

GS_INIT_LICFILENAME

Type	String
Description	License file name.
Values	
Default value	None
Relation to deprecated feature	GsSetLicense and GsInit_r

GS_INIT_DATAPATH

Type	String
Description	List of paths to search for necessary files.

Values	
Default value	None
Relation to deprecated feature	GsInit_r

GS_INIT_Z4FILE

Type	String
Description	Name of the Zip +4 directory file, us.z9.
Values	
Default value	None
Relation to deprecated feature	GsInit_r

GS_INIT_PASSWORD

Type	Long
Description	Long License password.
Values	
Default value	None
Relation to deprecated feature	GsSetLicense and GsInit_r

GS_INIT_GSVERSION

Type	Long
Description	GeoStan version.
Values	
Default value	None
Relation to deprecated feature	GsInit_r

GS_INIT_OPTIONS_ADDR_CODE

Type	Boolean
Description	Open files needed for address standardization and geocoding.
Values	True or False.
Default value	False
Relation to deprecated feature	Overrides GS_FILE_ADDR_CODE and GsInit_r.

GS_INIT_OPTIONS_Z9_CODE

Type	Boolean
Description	Open files needed for Zip centroid geocoding.
Values	True or False.
Default value	False
Relation to deprecated feature	Overrides GS_FILE_Z9_CODE and GsInit_r.

GS_INIT_OPTIONS_SPATIAL_QUERY

Type	Boolean
Description	Open spatial query files.
Values	True or False.
Default value	False
Relation to deprecated feature	Overrides `GS_FILE_SPATIAL_QUERY`.

GS_INIT_OPTIONS_APN_CODE

Type	Boolean
Description	Open files needed for APN data.
Values	True or False.

Default value	False
Relation to deprecated feature	Overrides `GS_FILE_APN_CODE.`

GS_INIT_OPTIONS_ELEVATION_CODE

Type	Boolean
Description	Open files needed for elevation data.
Values	True or False.
Default value	False
Relation to deprecated feature	Overrides `GS_FILE_ELEVATION_CODE.`

GS_INIT_RELDATE

Type	String
Description	String of the latest data to use in initialization.
Values	
Default value	None
Relation to deprecated feature	GsInit_r

GS_INIT_CACHESIZE

Type	Long
Description	Relative cache size used by GeoStan. Controls the amount of memory that GeoStan allocates to store temporary street data during address processing. A smaller cache may slow the performance of GeoStan. A cache size of 2 gives best performance, but uses more memory.
Values	0,1, or 2.
Default value	1
Relation to deprecated feature	GsInit_r

GS_INIT_DPV

Type	Boolean
Description	Initialize DPV.
Values	True or False.
Default value	False
Relation to deprecated feature	GsInitDpv_r

GS_INIT_LACSLINK

Type	Boolean
Description	Initialize LACS$^{\text{Link}}$.
Values	True or False.
Default value	False
Relation to deprecated feature	GsInitLACSLink_r

GS_INIT_FILE_MEMORY_LIMIT

Type	Long
Description	You can use GS_INIT_FILE_MEMORY_LIMIT to control the amount of memory used for memory mapped files. The allowed range is from 0 to 4294 (megabytes). If you input 0, GeoStan uses a small cache and reads from the disk. If you use the maximum amount, GeoStan maps as many files as possible until the limit is exhausted. This property can only be used once per GeoStan executable process. If you are using this property, you can use it once per process. When calling GsInitWithProps more than once, you do not need to used the GS_INIT_FILE_MEMORY_LIMIT property again, and if you do, subsequent usages of it will be ignored. Mapped files are the gsd, gsu, gsi, cbsac.dir, finmbr.dat, and us.gsl files. If you are using DPV and LACSLink, you must establish the memory limit before initializing them. Although the DPV and LACSLink files are not memory mapped files, these libraries load large files that can cause GsInitWithProps to fail if not enough memory is reserved. To determine the amount of virtual bytes used by your process, set the file memory limit to 0 and run your process while monitoring memory.
Values	0 - 4294 (megabytes)

Default value	
Relation to deprecated feature	GsSetFileMemoryLimit

GS_INIT_DPV_DATA_ACCESS

Type	Long
Description	Indicates the type of files to load and how to access the files. The following table contains the possible values.
Values	• DPV_DATA_FULL_FILEIO - Files are not loaded into memory. Table access is through file I/O. • DPV_DATA_FULL_MEMORY - Files are loaded into memory. Use this option to gain performance improvement by reducing repetitive file I/O. • DPV_DATA_SPLIT_FILEIO - Use if your file is sorted by ZIP Code and you have limited memory. Uses a split data format that separates the DPV data file into multiple smaller files, based on the first 2 digits of the ZIP Code. If you sort your mailing file by ZIP Code, you can use this value to bring the relevant portion of the DPV file into memory. This process uses 32 MB of storage, but reduces the number of I/O requests that normally occurs when you use the full DPV data file • DPV_DATA_FLAT_FILEIO - DPV flat file accessed via file I/O.
Default value	`DPV_DATA_FULL_FILEIO`
Relation to deprecated feature	Overrides DpvInitStruct.

GS_INIT_DPV_DIRECTORY

Type	String
Description	String that specifies the directory containing DPV data.
Values	
Default value	None
Relation to deprecated feature	Overrides DpvInitStruct

GS_INIT_DPV_SECURITYKEY

Type	String
Description	Security key for product authorization.

Values	
Default value	None
Relation to deprecated feature	Overrides DpvInitStruct.

GS_INIT_LACSLINK_DIRECTORY

Type	String
Description	Directory of Lack^{Link} data.
Values	
Default value	None
Relation to deprecated feature	Overrides LACS^{Link} Init struct.

GS_INIT_LACSLINK_SECURITY_KEY

Type	String
Description	Security key.
Values	
Default value	None
Relation to deprecated feature	Overrides LACS^{Link} Init struct.

GS_INIT_GEOSTAN_ID

Type	Long
Description	GeoStan ID.
Values	If NULL, this means initialize GeoStan.
Default value	None
Relation to deprecated feature	GsInit_r

GS_INIT_SUITELINK

Type	Boolean
Description	Initializes Suite$^{\text{Link}}$.
Values	True or False.
Default value	False
Relation to deprecated feature	None

GS_INIT_SUITELINK_DIRECTORY

Type	String
Description	Directory containing Suite$^{\text{Link}}$ data.
Values	
Default value	None
Relation to deprecated feature	None

GsInitWithProps output status properties

GS_STATUS_FILE_GEO_DIR

Type	Boolean
Description	The GeoStan directory file loaded successfully (*.gsd).
Values	True or False.
Default value	False
Relation to deprecated feature	Overrides `GS_FILE_GEO_DIR`.

GS_STATUS_FILE_STATE_XLAT

Type	Boolean
Description	The state lookup file loaded successfully. Obsolete.

Values	True or False.
Default value	False
Relation to deprecated feature	None

GS_STATUS_FILE_CITY_DIR

Type	Boolean
Description	The city lookup file loaded successfully (ctyst.dir).
Values	True or False.
Default value	False
Relation to deprecated feature	Overrides `GS_FILE_CITY_DIR`.

GS_STATUS_FILE_ZCF_DIR

Type	Boolean
Description	The ZIP Code file loaded successfully. Obsolete.
Values	True or False.
Default value	False
Relation to deprecated feature	None.

GS_STATUS_FILE_PARSE_TABLES

Type	Boolean
Description	The parsing tables (parse.dir) loaded successfully.
Values	True or False.
Default value	False
Relation to deprecated feature	Overrides `GS_FILE_PARSE_TABLES`.

GS_STATUS_FILE_POSTTYPE_XLAT

Type	Boolean
Description	The street type lookup file loaded successfully. Obsolete.
Values	True or False.
Default value	False
Relation to deprecated feature	None.

GS_STATUS_FILE_ZIP4CENT_DIR

Type	Boolean
Description	The ZIP + 4 centroid file loaded successfully (us.z9).
Values	True or False.
Default value	False
Relation to deprecated feature	Overrides `GS_FILE_ZIP4CENT_DIR`.

GS_STATUS_FILE_CBSA_DIR

Type	Boolean
Description	The CBSA file loaded successfully (cbsac.dir).
Values	True or False.
Default value	False
Relation to deprecated feature	Overrides `GS_FILE_CBSA_DIR`.

GS_STATUS_FILE_ALIAS

Type	Boolean
Description	The street name alias file. Obsolete.
Values	True or False.

Default value	False
Relation to deprecated feature	None

GS_STATUS_FILE_LICENSE

Type	Boolean
Description	The license file (*.lic).
Values	True or False.
Default value	False
Relation to deprecated feature	Overrides `GS_FILE_LICENSE`.

GS_STATUS_FILE_EXPIRED

Type	Boolean
Description	One or more GSD files have expired (output only).
Values	True or False.
Default value	False
Relation to deprecated feature	Overrides `GS_FILE_EXPIRED`.

GS_STATUS_FILE_SPATIAL_QUERY

Type	Boolean
Description	Spatial query file loaded successfully (finmbr.dat).
Values	True or False.
Default value	False
Relation to deprecated feature	Overrides `GS_FILE_SPATIAL_QUERY`.

GS_STATUS_FILE_ZIPMOVE

Type	Boolean
Description	ZIPMove file loaded successfully (us.gsz).
Values	True or False.
Default value	False
Relation to deprecated feature	Overrides `GS_FILE_ZIPMOVE`.

GS_STATUS_FILE_LOT

Type	Boolean
Description	ELOT/Z4Change file loaded successfully (us.gsl).
Values	True or False.
Default value	False
Relation to deprecated feature	Overrides `GS_FILE_LOT`.

GS_STATUS_FILE_EPI

Type	Boolean
Description	Extended Performance Index file loaded successfully. Obsolete.
Values	True or False.
Default value	False
Relation to deprecated feature	None

GS_STATUS_FILE_ERROR_INFO

Type	Boolean
Description	Error information is available from GeoStan via GsErrorGetEx.
Values	True or False.

Default value	False
Relation to deprecated feature	GsErrorHas

GS_STATUS_FILE_ZIP9_IDX

Type	Boolean
Description	ZIP9 index file loaded successfully (*.gsu).
Values	True or False.
Default value	False
Relation to deprecated feature	Overrides `GS_FILE_ZIP9_IDX`.

GS_STATUS_FILE_EWS

Type	Boolean
Description	EWS file loaded successfully (EWS.txt).
Values	True or False.
Default value	False
Relation to deprecated feature	Overrides `GS_FILE_EWS`.

GS_STATUS_FILE_AUXILIARY

Type	Boolean
Description	At least one auxiliary files loaded (*.gax).
Values	True or False.
Default value	False
Relation to deprecated feature	Overrides `GS_FILE_AUXILIARY`.

GS_STATUS_FILE_APN

Type	Boolean
Description	APN file loaded successfully (*.apn).
Values	True or False.
Default value	False
Relation to deprecated feature	Overrides `GS_FILE_APN`.

GS_STATUS_FILE_ELEVATION

Type	Boolean
Description	Elevation file loaded successfully (*.elv).
Values	True or False.
Default value	False
Relation to deprecated feature	Overrides `GS_FILE_ELEVATION`.

GS_STATUS_FILE_MEM_LIM_EXCDD

Type	Boolean
Description	Requested file memory limit exceeded.
Values	True or False.
Default value	False
Relation to deprecated feature	Overrides `GS_FILE_MEM_LIM_EXCDD`.

GS_STATUS_FILE_MEM_SYS_EXCDD

Type	Boolean
Description	System resources exceeded, preventing memory mapping.
Values	True or False.

Default value	False
Relation to deprecated feature	Overrides `GS_FILE_MEM_SYS_EXCDD`.

GS_STATUS_DPV_FILE_SECURITY

Type	Boolean
Description	DPV security key verified successfully.
Values	True or False.
Default value	False
Relation to deprecated feature	Overrides GsInitDpv_r.

GS_STATUS_DPV_FILE_ALL

Type	Boolean
Description	DPV data loaded successfully.
Values	True or False.
Default value	False
Relation to deprecated feature	GsInitDpv_r

GS_STATUS_STATUS_BITFIELD

Type	Long
Description	GsInitStruct status bitfield for backwards compatibility.
Values	
Default value	
Relation to deprecated feature	GsInit_r

GS_STATUS_FILE_MEMORY_USED

Type	Long
Description	Amount of memory used by file memory limits.
Values	
Default value	
Relation to deprecated feature	

GS_STATUS_FILE_UD

Type	Boolean
Description	At least one User Dictionary loaded successfully.
Values	True or False.
Default value	False
Relation to deprecated feature	None

GS_STATUS_FILE_REVERSE_ALIAS

Type	Boolean
Description	Indexes file (usps.gdi) loaded successfully.
Values	True or False.
Default value	False
Relation to deprecated feature	None

GS_STATUS_LACSLINK_FILE_SECUR

Type	Boolean
Description	LACS$^{\text{Link}}$ security key verified successfully.
Values	True or False.

Default value	False
Relation to deprecated feature	None

GS_STATUS_LACSLINK_FILE_ALL

Type	Boolean
Description	LACS$^{\text{Link}}$ data loaded successfully..
Values	True or False.
Default value	False
Relation to deprecated feature	None

GS_STATUS_SUITELINK_FILE_ALL

Type	Boolean
Description	Suite$^{\text{Link}}$ data loaded successfully.
Values	True or False.
Default value	False
Relation to deprecated feature	None

GsCityDataGet

WIN | UNIX | z/OS

Retrieves data located with GsCityFindFirst or GsCityFindNext.

Syntax

```
GsFunStat GsCityDataGet ( GsId gs, intlu rec, GsEnum which,
pstr pbuffer, int bufSize );
```

Arguments

gs ID returned by *GsInitWithProps* for the current instance of GeoStan. *Input.*

rec City record number, as returned by **GsCityFindFirst** or **GsCityFindNext**. *Input*.

which Symbolic constant for the data item to retrieve. *Input*.

The following table lists the valid GsEnums for most languages and the returned data sizes (including NULL characters).

GsEnum	Size	Description
GS_CITY_CARRTSORT	(2)	• Y – Carrier Route Sort • N – No Carrier Route Sort
GS_CITY_CITYDELV	(2)	• Y – Post Office has city-delivery carrier routes • N – Post Office does not have city-delivery carrier routes
GS_CITY_CLASS	(2)	ZIP Classification Code.
GS_CITY_CTYSTKEY	(7)	Returns the 6-character USPS City State Key, which uniquely identifies a locale in the city/state file.
GS_CITY_FACILITY	(2)	Returns the USPS City State Name Facility Code: • A – Airport Mail Facility (AMF) • B – Branch • C – Community Post Office (CPO) • D – Area Distribution Center (ADC) • E – Sectional Center Facility (SCF) • F – Delivery Distribution Center (DDC) • G – General Mail Facility (GMF) • K – Bulk Mail Center (BMC) • M – Money Order Unit • N – Non-Postal Community Name, Former Postal Facility, or Place Name • P – Post Office • S – Station • U – Urbanization
GS_CITY_IS_ABBREV	(2)	• Y - If the current record is the 13-character USPS city name. • N - If the full city name.
GS_CITY_MAILIND	(2)	• Y – Can use City State Name as last line on a mail piece • N – Cannot use City State Name as last line on a mail piece
GS_CITY_NAME	(30)	City name. May be an alternative name.
GS_CITY_PREFNAME	(30)	Returns the USPS preferred city name.

GsEnum	Size	Description
GS_CITY_QCITY	(10)	City number; using the format ssffffccc, where s=state number, f=finance number, and c=city number.
GS_CITY_UNIQUE	(2)	• Y – Unique ZIP Code (ZIP assigned to a single organization.) • Blank – Not applicable
GS_CITY_STATE	(3)	State abbreviation
GS_CITY_ZIP	(6)	ZIP Code

pbuffer Location to store the returned data. *Output*.

bufSize Maximum size of data to return in the buffer. If *bufSize* is shorter than the data returned by GeoStan, GeoStan truncates the data and does not generate an error. *Input*.

Return Values

GS_SUCCESS

GS_ERROR

Prerequisites

GsCityFindFirst or **GsCityFindNext**

Notes

The **GsCityDataGet** function retrieves data about the city located with **GsCityFindFirst** or **GsCityFindNext**. GeoStan derives city information from the USPS City/State file.

Example

See GsCityFindFirst for an example.

GsCityFindFirst

WIN	UNIX	z/OS

FINDS THE FIRST CITY MATCHING THE PARTIAL NAME OR VALID ZIP CODE.

Syntax

```
intlu GsCityFindFirst( GsId gs, gs_const_str State,
gs_const_str cityPattern );
```

Arguments

gs	ID returned by *GsInitWithProps* for the current instance of GeoStan. *Input.*
State	Various state abbreviations and spellings, or blank for a ZIP Code search. For example, for New Hampshire, it accepts: N HAMP, N HAMPSHIRE, NEW HAMPSHIRE, NEWHAMPSHIRE, NH, and NHAMPSHIRE. *Input*.
cityPattern	City to search for (may be just a partial string), or a 3- or 5-digit ZIP Code. *Input*.

Return Values

Record number of the city located, or zero if GeoStan did not find any cities.

Prerequisites

`GsClear`

Notes

This function retrieves the record number of the first city in a state that matches the city or ZIP Code search string. For example if the entered state string is `CA` and the city string is `S`, GeoStan finds the first city that begins with `S` in California. If the entered city string is `803` and the state string is null, GeoStan returns the first city in that sectional center. GeoStan does not return cities in any predefined order.

Before each find function, call **`GsClear`** to reset the internal buffers. If you do not reset the buffers, you may receive incorrect results with information from a previous find.

Example

```
/* This example prints all city names in the 803 sectional center. */

intlu CityRecNum;
char CityName[30];

GsClear(gs);
CityRecNum = GsCityFindFirst( gs, ““, “803” );
while (CityRecNum != 0 )
{
GsCityDataGet( gs, CityRecNum, GS_CITY_NAME,CityName,
sizeof(CityName) );
printf( “%s\n”, CityName );
GsClear(gs);
CityRecNum = GsCityFindNext( gs );
}
```

GsCityFindNext

WIN | **UNIX** | z/**OS**

FINDS THE NEXT CITY MATCHING THE PARTIAL NAME OR VALID ZIP CODE.

Syntax

```
intlu GsCityFindNext ( GsId gs );
```

Arguments

gs	ID returned by *GsInitWithProps* for the current instance of GeoStan. *Input.*

Return Values

Record number of the located city, or zero if GeoStan did not find any cities.

Prerequisites

GsCityFindFirst and **GsClear**

Notes

Call this function after **GsCityFindFirst**.

Before each find function, call **GsClear** to reset the internal buffers. If you do not reset the buffers, you may receive incorrect results with information from a previous find.

Example

See **GsCityFindFirst**.

GsClear

NT | **UNIX** | z/**OS**

CLEARS THE DATA BUFFER IN THE INTERNAL DATA STRUCTURES.

Syntax

```
GsFunStat GsClear( GsId gs );
```

Arguments

gs ID returned by *GsInitWithProps* for the current instance of GeoStan. *Input.*

Return Values

`GS_SUCCESS`

`GS_ERROR`

Notes

Before each find function, call **`GsClear`** to reset the internal buffers to null. If you do not reset the buffers, you may receive incorrect results with information from a previous find. This call clears only address element and location information about the previously processed address and does not affect distance or centroid match type.

GsDataGet

WIN	UNIX	z/OS

RETURNS DATA FOR ALL ADDRESS AND MATCHED ELEMENTS FROM GEOSTAN.

Syntax

```
GsFunStat GsDataGet( GsId gs, GsEnum fInOut, GsEnum fSwitch,
pstr pbuffer, intsu bufLen );
```

Arguments

gs ID returned by *GsInitWithProps* for the current instance of GeoStan. *Input.*

fInOut Flag indicating whether to retrieve original or processed data. *Input.*

To retrieve parsed address components, set *fInOut* to `GS_INPUT`. To retrieve matched address information from the parsed input address and the GeoStan database, set *fInOut* to `GS_OUTPUT`.

If the input address does not match, setting *fInOut* to `GS_OUTPUT` returns the address exactly as entered, except for the lastline information, which returns only the parsed lastline components.

The parsed lastline components correspond to the following enums:

GS_LASTLINE	GS_ZIP4
GS_CITY	GS_ZIP9
GS_STATE	GS_ZIP10
GS_ZIP	

If there is extra data on the input lastline (`GS_LASTLINE`), this data is not retrievable. For example, in the lastline “BOULDER CO 80301 US OF A”, “US OF A” is not retrievable from any **`GsDataGet()`** call.

To retrieve centerline return values, setting *`fInOut`* to `GS_CENTERLINE retunrs a` collection of ranged segments for streets that contain addresses. Individual addresses may be interpolated from the low-high ranges of the segments..

`fSwitch` Symbolic constant for the data item to retrieve. *Input*.

`pbuffer` Location to store the returned data. *Output*.

`bufLen` Maximum size of data for GeoStan to return. `geostan.h` lists as constants the recommended buffer sizes for each item. These sizes include the null terminator and are the maximum lengths required to get the full output string. You can allocate a buffer that is smaller or larger than these values. However, if *`bufLen`* is shorter than the returned data, GeoStan truncates the data and does not generate an error. *Input*.

Return Values

GS_SUCCESS

GS_ERROR

Prerequisites

GsFindWithProps

Notes

This function retrieves data elements from internal GeoStan buffers for either the original (input) or matched (output) data elements. To retrieve original data, set *`fInOut`* to `GS_INPUT`. To retrieve matched data, set *`fInOut`* to `GS_OUTPUT`.

If your GeoStan license has metering enabled, using **`GsDataGet`** to return a geocode increments the counter unless the match code indicates an error or a 5-digit ZIP geocode.

GsDataSet

WIN	UNIX	Z/OS

PASSES DATA FOR ALL ADDRESS ELEMENTS TO GEOSTAN.

Syntax

```
GsFunStat GsDataSet( GsId gs, GsEnum fSwitch, gs_const_str pBuffer );
```

Arguments

gs	ID returned by *GsInitWithProps* for the current instance of GeoStan. *Input*.
fSwitch	Symbolic constant for the data item to load. See "Common enums for storing and retrieving data" on page 68 for valid enums. *Input*.
pBuffer	String pointer containing the data to load. *Input*.

Return Values

GS_SUCCESS

GS_ERROR

Prerequisites

GsInitWithProps

Notes

This function loads data elements into internal GeoStan buffers. When loading address information as a complete address or last line, GeoStan parses the data into fields. For example, if you enter a last line of "Boulder, CO 80301-1234", GeoStan parses the data and sets the city, state, ZIP, and ZIP + 4 fields. You can retrieve parsed input data using **GsDataGet** with the GS_INPUT argument.

If you are passing both an address line and a last line or ZIP Code, enter the last line or ZIP Code *first* to ensure the greatest accuracy in address standardization.

If you are passing both the address information and the last line information as one input line, enter the address information *first*.

Using the appropriate GsEnums, you can pass single line addresses, two-line addresses, or multiline addresses of up to six lines. For more information, see "Extracting data from GSD files" on page 407.

Do not call alter the GS_FIND_MATCH_MODE property value in your Find Properties after you have called GsDataSet, because the previously established match mode affects how the input address is parsed. If you need to change the match mode in mid-process, you must re-enter the data for the current address with GsDataSet.

GsDpvGetCompleteStats

WIN	UNIX	Z/OS

OBTAINS THE COMPLETE DPV STATISTICS SINCE THE APPLICATION INITIALIZED DPV.

Syntax

```
GsFunStat GsDpvGetCompleteStats( GsId gs, GsDPVCompleteStats* pdata, int
structSize );
```

Arguments

`gs` ID returned by *GsInitWithProps* for the current instance of GeoStan. *Input.*

`*pdata` Structure contains the following DPV statistics since the application initialized DPV.

`intlu nCMRAInError` Number of records with an unconfirmed CMRA. *Output.*

`intlu nCMRAValid` Number of records with a confirmed CMRA. *Output.*

`intlu nDPVNoStatFound` Not Applicable for this release.

`intlu nDPVInError` Number of records not confirmed by DPV. *Output.*

`intlu nDPVSeedHits` Number of distinct DPV false-positive matches. *Output.*

`intlu nDPVTypeDValid` Number of records with a confirmed primary house number, but the secondary unit number is missing. *Output.*

`intlu nDPVTypeSValid` Number of records with a confirmed primary house number, but the secondary unit number is not confirmed. *Output.*

`intlu nDPVTypeYValid` Number of records fully confirmed by DPV. *Output.*

`intlu nDPVZipsOnFile` Number of distinct ZIP Codes in file. *Output.*

`intlu nFirmCMRAPresented` Number of USPS Firm records with PMB presented. *Output.*

`intlu nFirmCMRAValid` Number of USPS Firm records CMRA confirmed with or without PMB. *Output.*

`intlu` *`nFirmPrimeFail`* Number of USPS Firm records without a confirmed primary house number. *Output.*

`intlu` *`nFirmSecondaryFail`* Number of USPS Firm records without a confirmed secondary unit number. *Output.*

`intlu` *`nFirmValid`* Number of confirmed USPS Firm records. *Output.*

`intlu` *`nFootnoteAA`* Number of records with DPV footnote value AA. *Output.*

`intlu` *`nFootnoteA1`* Number of records with DPV footnote value A1. *Output.*

`intlu` *`nFootnoteBB`* Number of records with DPV footnote value BB. *Output.*

`intlu` *`nFootnoteCC`* Number of records with DPV footnote value CC. *Output.*

`intlu` *`nFootnoteF1`* Number of records with DPV footnote value F1. *Output.*

`intlu` *`nFootnoteG`*1 Number of records with DPV footnote value G1. *Output.*

`intlu` *`nFootnoteM1`* Number of records with DPV footnote value M1. *Output.*

`intlu` *`nFootnoteM3`* Number of records with DPV footnote value M3. *Output.*

`intlu` *`nFootnoteN1`* Number of records with DPV footnote value N1. Output.

`intlu` *`nFootnoteP1`* Number of records with DPV footnote value P1. *Output.*

`intlu` *`nFootnoteP3`* Number of records with DPV footnote value P3. *Output.*

`intlu` *`nFootnoteRR`* Number of records with DPV footnote value RR. *Output.*

`intlu` *`nFootnoteR1`* Number of records with DPV footnote value R1. *Output.*

`intlu` *`nFootnoteU1`* Number of records with DPV footnote value U1. *Output.*

`intlu` *`nGenDelValid`* Number of confirmed USPS General Delivery records. *Output.*

`intlu` *`nHRCMRAPresented`* Number of USPS Highrise records with PMB presented. *Output.*

`intlu` *`nHRCMRAValid`* Number of USPS Highrise records with confirmed CMRA records with or without PMB. *Output.*

`intlu` *`nHRPrimeFail`* Number of USPS Highrise records without a confirmed primary house number. *Output.*

`intlu` *`nHRSecondaryFail`* Number of USPS Highrise records without a confirmed secondary unit number. *Output.*

intlu nHRValid Number of confirmed USPS Highrise records. *Output.*

intlu nPOBoxFail Number of USPS PO Box records without a confirmed primary box number. *Output.*

intlu nPOBoxValid Number of confirmed USPS PO Box records. *Output.*

intlu nRRHCCMRAPresented Number of USPS Rural Route records with PMB presented. *Output.*

intlu nRRHCCMRAValid Number of USPS Rural Route records CMRA confirmed with or without PMB. *Output.*

intlu nRRHCPrimeFail Number of USPS Rural Route records without a confirmed primary house number. *Output.*

intlu nRRHCValid Number of confirmed USPS Rural Route records. *Output*

intlu nStCMRAPresented Number of USPS Street records with PMB presented. *Output.*

intlu nStCMRAValid Number of USPS Street records with CMRA confirmed with or without PMB. *Output.*

intlu nStreetPrimeFail Number of USPS Street records without a confirmed primary house number. *Output.*

intlu nStreetSecondaryFail Number of USPS Street records without a confirmed secondary unit number. *Output.*

intlu nStreetValid Number of confirmed DPV records that matched to a USPS street record (`GS_REC_TYPE=S`). *Output.*

intlu nTotalDPVProcessed Number of records processed through DPV. *Output.*

intlu nTotalDPVwithZip4 Number of ZIP + 4 records processed through DPV. *Output.*

structSize Size of the **`GsDpvCompleteStats`** data structure. *Input.*

Return Values

`GS_SUCCESS`

`GS_ERROR` Call `GsErrorGetEx` for more information.

`GS_WARNING` Call `GsErrorGetEx` for more information.

Prerequisites

```
GsInitDpv_r
```

GsDpvGetFalsePosDetail

WIN	UNIX	Z/OS

RETRIEVES THE DETAIL RECORD FOR A DPV FALSE-POSITIVE REPORT.

Syntax

```
GsFunStat GsDpvGetFalsePosDetail( GsId gs,
GsFalsePosDetailData* pdata, int structSize );
```

Arguments

gs ID returned by *GsInitWithProps* for the current instance of GeoStan. *Input*.

pdata* This structure retrieves the DPV detail record for the false-positive address match using the data passed in **GsFalsePosDetailData. The data members are details provided by GeoStan for the false-positive report. This structure contains the following:

char AddressPrimaryNumber House number. *Output.*

*char AddressSecondaryAbbreviation*Unit type (APT, SUITE, LOT). *Output.*

char AddressSecondaryNumber Unit number. *Output.*

char Filler Reserved for future implementation. *Output.*

char MatchedPlus4 ZIP Code extension. *Output.*

char MatchedZIPCODE ZIP Code. *Output.*

char StreetName Name of the street. *Output.*

char StreetPostDirectional Street name postdirectional (N, S, E, W). *Output.*

char StreetpreDirectional Street name predirectional (N, S, E, W). *Output.*

char StreetSuffixAbbreviation Street type (AVE, ST, RD). *Output.*

structSize Size of the *GsFalsePosDetailData* data structure. *Input*.

Return Values

`GS_SUCCESS`	
`GS_ERROR`	Call `GsErrorGetEx` for more information.
`GS_WARNING`	Call `GsErrorGetEx` for more information.

Prerequisites

`GsInitDpv_r`

Notes

Only valid if the current/last record is a DPV false-positive match:
`GS_DPV_FALSE_POS = “Y”`

GsDpvGetFalsePosHeaderStats

WIN	UNIX	z/OS

Retrieves DPV statistics for the header record for a DVP false-positive report.

Syntax

```
GsFunStat GsDpvGetFalsePosHeaderStats( GsId gs,
GsFalsePosHeaderData *pdata, int structSize );
```

Arguments

gs ID returned by *GsInitWithProps* for the current instance of GeoStan. *Input.*

pdata* This structure retrieves DPV statistics for the header record for false-positive address matches using the data passed in **`GsFalsePosHeaderData`. The output values are statistics for the false-positive reports. The input values include information to correctly complete the false-positive report for the USPS. This structure contains the following:

char MailersAddressLine Address of the mailer. *Input.*

char MailersCityName City of the mailer. *Input.*

char MailerCompanyName Name of the mailer. *Input.*

char MailersStateName State of the mailer. *Input.*

char Mailers9DigitZip ZIP Code of the mailer. *Input.*

intlu nNumberZipCodeOnFile Number of distinct ZIP Codes processed through DPV. *Output*.

intlu nTotalRecordsMatched Number of records matched with DPV data. *Output*.

intlu nTotalRecordsProcessed Number of records processed through DPV. *Output.*

intlu nTotalRecordsZIP4Matched Number of records that have matched with ZIP + 4. *Output*.

intlu NumberFalsePositives Number of found DPV false-positive matches. *Output*.

structSize Size of the **GsFalsePosHeaderData** data structure. *Input.*

Return Values

GS_SUCCESS	
GS_ERROR	Call GsErrorGetEx for more information.
GS_WARNING	Call GsErrorGetEx for more information.

Prerequisites

GsInitDpv_r

GsErrorGet

WIN UNIX z/OS

RETRIEVES CURRENT ERROR INFORMATION.

Syntax

```
GsFunStat GsErrorGet( GsId gs, pstr pMessage, pstr pDetail );
```

Arguments

gs	ID returned by *GsInitWithProps* for the current instance of GeoStan. *Input*.
pMessage	Basic explanation for the error; up to 256 bytes in length. *Output*.
pDetail	Particulars of an error, such as filename; up to 256 bytes in length. *Output*.

NOTE: Both *`pMessage`* and *`pDetail`* can return up to 256 bytes of data. Always set buffers for these arguments to 256 bytes or larger.

Return Values

Error number of the most recent GeoStan error.

`-1`	No error.
`0 through 99`	Indicates the actual DOS error values.
`100`	Unclassified error.
`101`	Unknown error.
`102`	Invalid file signature.
`103`	Table overflow.
`104`	Insufficient memory.
`105`	File not found.
`106`	Invalid argument to a function.
`107`	File is out of date.

Alternates

`GsErrorGetEx`

GsErrorGetEx

WIN	UNIX	z/OS

Retrieves GeoStan informational, error, and fatal warning information for the current thread.

Syntax

`void` **`GsErrorGetEx`** `(GsID` *`id`*`, pstr` *`error_message`*`, pstr` *`details`*`)`

Arguments

`id` `GsId returned by` **`GsInitWithProps`** if initialization completed, or set to NULL if initialization failed and **`GsInitWithProps`** returned NULL.

`error_message` GeoStan error code and descriptive text. The `error_message` has the following format: `SGTppeeeDESCRIPTION`, where:

- `SGT` is a constant value.
- `pp` is the product number (01 for GeoStan).
- `eee` is the three character error code.
- `DESCRIPTION` is a short text message naming the error.

`details` Descriptive message, such as the file name associated with the error.

Return values

The next GeoStan error detected in the current thread. Upon return, error messages contain a brief description and additional text, such as the name of the file or directory associated with the error.

NOTE: Not all errors are fatal. For example, if GeoStan finds an invalid data file, it reports the error but continues executing.

Prerequisites

`GsInitWithProps`
`GsErrorHas`

Example

See GsInit_r for example code.

GsErrorHas

WIN	**UNIX**	z/**OS**

INDICATES IF ANY ERRORS OCCURRED OR ANY INFORMATIONAL MESSAGES ARE AVAILABLE IN THE CURRENT THREAD.

Syntax

`GsFunStat` **`GsErrorHas`** `(GsId` *`id`*`)`

Arguments

`id` `GsId returned by` **`GsInitWithProps`** if initialization completed, or set to NULL if initialization failed and **`GsInitWithProps`** returned NULL. *Input.*

Return Values

True GeoStan generated errors or informational messages.

GSDFalse	GeoStan did not generate any errors or informational messages

Prerequisites

`GsInitWithProps`

Example

See GsErrorGetEx for example code.

GsFileStatus

WIN | UNIX | Z/OS

RETURNS IF THE FILES SUCCESSFULLY OPENED, AND THE DATE OF THE USPS DATA USED IN GENERATING THE PRIMARY GSD FILE.

Syntax

```
intsu GsFileStatus( GsId gs, gs_const_str stateCode, intsu *date );
```

Arguments

gs ID returned by *GsInitWithProps* for the current instance of GeoStan. *Input.*

stateCode State FIPS or abbreviation (this argument is now ignored). *Input.*

**date* Publish date of the USPS data used in the current GeoStan release. *Output.*

Return Values

The return value is a series of bit-flags. The following `GS_ENUMs` are used to test these flags:

GS_FILE_EXISTS	Opened a primary GSD file (us.gsd, use.gsd, usw.gsd).
GS_SUPP_EXISTS	Opened a supplemental GSD file (ust.gsd, uste.gsd, uswt.gsd).
GS_STATEWIDE_EXISTS	Initialized a state-wide intersection file* (us.gsi, use.gsi, usw.gsi).[a]
GS_GSL_EXISTS	Opened eLOT and Z4Change data[b] (us.gsl).
GS_EWS_MATCH_EXISTS	Loaded a valid, unexpired EWS file (ews.txt).

GS_ZIPMOVE_EXISTS	Opened ZIPMove dataYou must have an additional license to use this file (us.gsz).
GS_ZIP9_IDX_EXISTS	Opened ZIP9 index file (ZIP9.gsu, ZIP9e.gsu, ZIP9w.gsu).
GS_AUXILIARY_EXISTS	Opened an auxiliary file (.gax).
GS_ELV_EXISTS	Opened an elevation file (.elv).
GS_APN_EXISTS	Opened an APN file (.apn).

a. You must have an additional license to use this file

b. eLOT data requires an additional license. However, the Z4Change data is always enabled.

NOTE: Additional file status information is available through **GSErrorGetEx**.

Prerequisites

GsInitWithProps

Notes

The date argument indicates the creation date the primary GSD file. The date argument and the **GS_INIT_RELDATE** initialization property value can be evaluated as follows:

Year	(date/384) + 1990
Month	((date % 384) / 32) + 1
Day	date % 32

Example

```
/* This example tests for the primary GSD file and eLOT files. */

intsu fileStatus, date;

fileStatus = GsFileStatus(gs, "", &date);
if(!(fileStatus & GS_GSL_EXISTS))
printf("eLOT file not initialized. \n");
if(!(fileStatus & GS_FILE_EXISTS))
printf("Primary GSD file not initialized.\n");
```

GsFileStatusEx

WIN	UNIX	z/OS

RETURNS INFORMATION ABOUT THE CURRENT GEOSTAN DATA SET AND LICENSE.

Syntax

```
intlu GsFileStatusEx( GsId gs, int mode, pstr buffer, int bufSize)
```

Arguments

gs ID returned by *GsInitWithProps* for the current instance of GeoStan. *Input.*

mode Specific information to return. The following table includes the types of information available. *Input.*

See the *Return Values* section for information on the data types and datum returned.

Mode	Data Returned
GS_STATUS_DATATYPE_NUM	Returns a numeric constant representing the data type.
GS_STATUS_DATATYPE_STR	Returns **GS_SUCCESS** or **GS_ERROR.** GeoStan places this retrieved information in the buffer.
GS_STATUS_DATUM_NUM	Returns a numeric constant representing the datum.
GS_STATUS_DATUM_STR	This is the NAD used natively by the data. It does not reflect the datum currently in use by GeoStan. You can use the Find property, GS_FIND_CLIENT_CRS, to set the returned NAD. Returns **GS_SUCCESS** or **GS_ERROR**. The retrieved information is placed in the buffer.
GS_STATUS_DAYS_REMAINING	• For unmetered licenses and metered unlimited licenses, it returns **DAYS_UNLIMITED** *or* the number of days remaining before the expiration of the license. • For metered limited licenses, it returns the days remaining before license expiration.
GS_STATUS_FILE_CHKSUM_NUM	Returns a calculated value (an integer) used to check data integrity. Set the **buffer** and **bufsize** parameters to 0; they are unused.
GS_STATUS_GEO_PRECISION	Returns the number of decimal places of longitude/latitude stored in the GSD file. Default is **4**. Point-level data is **6**.

GS_STATUS_GEO_RECORD_TOTAL	• For unmetered licenses, it returns 0. • For metered licenses, it returns the total number of records geocoded.
GS_STATUS_RECORDS_REMAINING	• For unmetered licenses and metered licenses it returns RECORDS_UNLIMITED. • For metered limited licenses, it returns the number of records remaining on the license. Use GS_STATUS_DAYS_REMAINING, GS_STATUS_RECORDS_REMAINING, or GS_STATUS_GEO_RECORD_TOTAL to request information on the status of the GeoStan license.

buffer Location to store the returned data. *Output*.

bufSize Maximum size of data that GeoStan returns. If *bufSize* is shorter than the data returned by GeoStan, GeoStan truncates the data and does not generate an error. *Input*.

Return Values

This function returns a variety of information about the current GeoStan data set.

GS_STATUS_DATATYPE_*X* Lists the data vendor

GS_STATUS_DATUM_*X* Returns the datum used to identify geographic coordinates.

GS_STATUS_RECORDS_REMAINING, GS_STATUS_DAYS_REMAINING,
GS_STATUS_GEO_RECORD_TOTAL Return license usage information.

You can use the preceding arguments to determine the status of your GeoStan license, as shown in the **GsInitWithProps** code example.

The following table shows the return values for GS_STATUS_DATATYPE_NUM.

GS_STATUS_DATATYPE_NUM	Data Type
0	USPS
1	TIGER
2	TANA
3	Sanborn point-level
4	Deprecated
6	NAVTEQ
7	TANA point
8	Centrus point

GS_STATUS_DATATYPE_NUM	Data Type
9	Auxiliary file
10	User Dictionary

The following table shows the return values for GS_STATUS_DATATYPE_STR.

GS_STATUS_DATATYPE_STR	Data Type
USPS	USPS
TIGER	TIGER
TANA	TANA
SANBORN POINT DATA	Sanborn point-level
NAVTEQ	NAVTEQ
TANA POINT DATA	TANA point-level
AUXILIARY	Auxiliary file
CENTRUS POINT DATA	Centrus point

The following table shows the return values for GS_STATUS_DATUM_STR.

GS_STATUS_DATUM_STR	Data Type
0	NAD27
1	NAD83 (WGS84 for GDT data)

Prerequisites

GsInitWithProps

GsFindFirst___

WIN	UNIX	z/OS

FINDS THE FIRST STREET, SEGMENT, OR RANGE OBJECT THAT MEETS THE SEARCH CRITERIA.

Syntax

```
GsFunStat GsFindFirstRange( GsId gs, GsRangeHandle *pRange );
GsFunStat GsFindFirstSegment( GsId gs, GsSegmentHandle *pSegment );
GsFunStat GsFindFirstStreet( GsId gs, GsStreetHandle *pStreet,
GsEnum option, gs_const_str pLocale, gs_const_str pName, gs_const_str pNumber
);
```

Arguments

gs ID returned by *GsInitWithProps* for the current instance of GeoStan. *Input.*

**pRange* Pointer to a segment/range handle. Returns a valid handle if GeoStan finds a range. *Input*, *Output*.

**pSegment* Pointer to a street/segment handle. Returns a valid handle If GeoStan finds a segment. *Input*, *Output*.

**pStreet* Pointer to a street handle. Returns a valid handle if GeoStan finds a street. *Output.*

option Determines the type of search performed by GeoStan. The following table lists the type of searches available. *Input*.

GS_ZIP_SEARCH	GeoStan searches the ZIP Code specified by pLocale. When this option is set with **GsFindFirstStreet**, GeoStan performs a finance search.
GS_CITY_SEARCH	GeoStan searches the city and state specified by pLocale.
GS_EXTRA_SEARCH	GeoStan searches the primary street files, Use.gsd and Usw.gsd, and the supplemental street files, Ustw.gsd and Uste.gsd, for streets without house numbers. If you specify this option, pNumber must be blank. If you specify *pNumber*, GeoStan searches only the primary street files, Usw.gsd and Use.gsd, which contain addressed segments. Assign the directory for this file to the GS_INIT_DATAPATH initialization property of the property list that you pass when calling **GsInitWithProps**. MVS customers receive one primary street file, Us.gsd, and one supplemental street file, Ust.gsd.
GS_SDX_SEARCH	GeoStan searches by Soundex. *pName* is a pointer to a numeric soundex key returned by **GsSoundex**. If you do not specify this option, GeoStan assumes that *pName* is a street name.

You must specify either **GS_ZIP_SEARCH** or **GS_CITY_SEARCH**.

pLocale Sets the search area. If *option* is set to GS_ZIP_SEARCH, then *pLocale* is a valid ZIP Code. If *option* is set to GS_CITY_SEARCH, then *pLocale* is a valid city and state. GeoStan performs a city search by searching all finance areas that cover that city. This may result in found objects outside the actual city.*Input*.

pName Street name, or partial street name, for which to search. If *option* is set to GS_SDX_SEARCH, then *pName* is a pointer to a numeric soundex key. *Input*.

This limits the search to street names that begin with the name string. If *pName* is "APPLE," then GeoStan returns only streets beginning with Apple, such as Apple or Appleton. If *pName* is not specified, GeoStan finds all streets specified by *pLocale*.

pNumber House number that must be within a range. If *option* is set to GS_EXTRA_SEARCH, then *pNumber* must be blank. *Input*.

This returns only those ranges that contain that house number. This can be an alphanumeric house number, such as 12N123W.

Return Values

GS_ERROR	Error occurred. You can retrieve the error using **GsErrorGet**.
GS_LASTLINE_NOT_FOUND	GeoStan could not find *pLocale*.
GS_NOT_FOUND	GeoStan did not find a match.
GS_SUCCESS	GeoStan found a match. You can retrieve the data using **GsHandleGet**.

Prerequisites

GsInitWithProps and **GsClear**

Notes

You must call **GsFindFirstStreet** before **GsFindFirstSegment** and **GsFindFirstSegment** before **GsFindFirstRange**. **GsFindFirstStreet** also sets the area and criteria for subsequent segment and range searches.

If you are using **GsFindFirstStreet** to find a street that has a number as a component of the street name, such as “US HWY 41” or “I-95,” enter just the number, as the text is not part of the index for such streets.

GsFindFirst___ and **GsFindNext___** functions work together to allow you to access to the entire address directory database.

See Appendix B, “Extracting data from GSD files” for more information on the Find First functions and segment and range searches.

NOTE: GsFindFirstStreet does not respect parity.

Example

```
/* This example searches ZIP Code 80301 for streets beginning with "A" and no
* specific house number. */
char tempstr[60];
GsStreetHandle hStreet;
GsSegmentHandle hSegment;
GsRangeHandle hRange;

/* Use GsFindFirstStreet to get the first street handle that matches the
criteria. */
int iStat = GsFindFirstStreet( gs, &hStreet, GS_ZIP_SEARCH, "80301", "A", "");
while( iStat == GS_SUCCESS )
{
```

```
/* We have a valid street handle, which we convert to a range handle so that
we can use GsHandleGet to retrieve information to show the user. Even though
we converted to a range handle, it really is still just a street handle, so
we can only access those elements that are valid at the street level. See
Appendix A for a complete list of valid elements for each handle type.*/

hSegment.hStreet = hStreet;
hRange.hSegment = hSegment;
GsHandleGet( gs, GS_PREDIR, &hRange, tempstr,
sizeof(tempstr) );
printf( "Street: %s ", tempstr );
GsHandleGet( gs, GS_NAME, &hRange, tempstr, sizeof(tempstr) );
printf( "%s ", tempstr );
GsHandleGet( gs, GS_TYPE, &hRange, tempstr, sizeof(tempstr) );
printf( "%s ", tempstr );
GsHandleGet( gs, GS_POSTDIR, &hRange, tempstr,
sizeof(tempstr) );
printf( "%s\n", tempstr );
/* find the first valid segment for the current street*/
iStat = GsFindFirstSegment( gs, &hSegment );
while( iStat == GS_SUCCESS )
{
/* We have a valid segment, convert to a range handle to get information to
display.*/
hRange.hSegment = hSegment;
printf( " Segment: " );
GsHandleGet( gs, GS_BLOCK_LEFT, &hRange, tempstr, sizeof(tempstr) );
printf( "%s ", tempstr );
GsHandleGet( gs, GS_BLOCK_RIGHT, &hRange, tempstr, sizeof(tempstr) );
printf( "%s ", tempstr );
GsHandleGet( gs, GS_SEGMENT_PARITY, &hRange, tempstr, sizeof(tempstr) );
printf( "%s ", tempstr );
GsHandleGet( gs, GS_SEGMENT_DIRECTION, &hRange, tempstr, sizeof(tempstr) );
printf( "%s \n", tempstr );

/* Get the first valid range for the current segment. */
iStat = GsFindFirstRange( gs, &hRange );
while( iStat == GS_SUCCESS )
{
GsHandleGet( gs, GS_LORANGE, &hRange, tempstr,
sizeof(tempstr) );
printf( " Range: %s ", tempstr );
GsHandleGet( gs, GS_HIRANGE, &hRange, tempstr,
sizeof(tempstr) );
printf( "%s ", tempstr );
GsHandleGet( gs, GS_BLOCK_LEFT, &hRange, tempstr, sizeof(tempstr) );
printf( "%s ", tempstr );
GsHandleGet( gs, GS_PREDIR, &hRange, tempstr,
sizeof(tempstr) );
printf( "%s ", tempstr );
GsHandleGet( gs, GS_NAME, &hRange, tempstr, sizeof(tempstr) );
printf( "%s ", tempstr );
GsHandleGet( gs, GS_TYPE, &hRange, tempstr, sizeof(tempstr) );
printf( "%s ", tempstr );
GsHandleGet( gs, GS_POSTDIR, &hRange, tempstr,
sizeof(tempstr) );
printf( "%s ", tempstr );
GsHandleGet( gs, GS_ZIP, &hRange, tempstr, sizeof(tempstr) );
printf( "%s ", tempstr );
GsHandleGet( gs, GS_LOUNIT, &hRange, tempstr,
sizeof(tempstr) );
printf( "%s ", tempstr );
GsHandleGet( gs, GS_HIUNIT, &hRange, tempstr,
sizeof(tempstr) );
printf( "%s\n", tempstr );
/* Get the next valid range for this segment. */
iStat = GsFindNextRange( gs, &hRange );
}
if ( iStat == GS_ERROR ) break;
/* get the next valid segment for this street*/
iStat = GsFindNextSegment( gs, &hSegment );
}
if ( iStat == GS_ERROR ) break;
/* Get the next street that meets the initial criteria. */
iStat = GsFindNextStreet( gs, &hStreet );
}
```

GsFindFirstSegmentByMbr

WIN UNIX

Finds the first segment handle within the specified MBR.

Syntax

```
GsFunStat GsFindFirstSegmentByMbr( GsId gs, GsSegmentHandle *pSegment, long
lon1, long lat1, long lon2, long lat2 );
```

Arguments

gs ID returned by *GsInitWithProps* for the current instance of GeoStan. *Input.*

pSegment Pointer to a street/segment handle. Returns a valid handle if GeoStan finds a segment. *Input, Output.*

lon1 Minimum longitude. One of four values that make up the MBR. *Input.*

lat1 Minimum latitude. One of four values that make up the MBR. *Input.*

lon2 Maximum longitude. One of four values that make up the MBR. *Input.*

lat2 Maximum latitude. One of four values that make up the MBR. *Input.*

Return Values

`GS_ERROR` Error occurred. You can retrieve the error using **`GsErrorGet`**.

`GS_LASTLINE_NOT_FOUND`
GeoStan could not find *`pSegment`*.

`GS_NOT_FOUND` GeoStan did not find a match.

`GS_SUCCESS` GeoStan found a match. If returned, *`pSegment`* is a valid segment handle that you can use with **`GsHandleGet`**, **`GsHandleGetCoords`**, or **`GsGetMbr`**.

Prerequisites

`GsInitWithProps` `and` **`GsClear`**

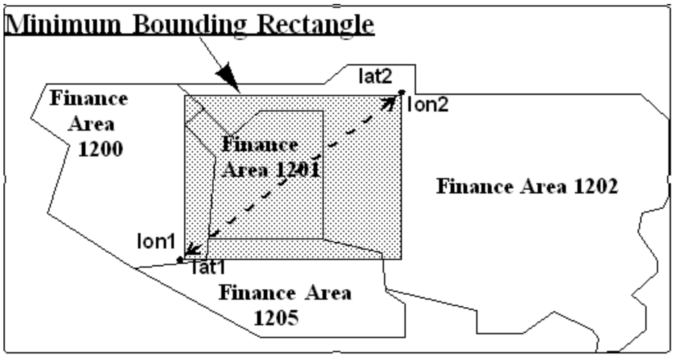
Notes

GeoStan handles all spatial query functions in ten-thousandths of degrees—not millionths of degrees as with **`GsDataGet`**.

Before each find function, call **`GsClear`** to reset the internal buffers. If you do not reset the buffers, you may receive incorrect results with information from a previous find.

MBR Representation by Lat/Lon

GeoStan defines the minimum bounding rectangle by the four arguments given as *lon1, lon2, lat1* and *lat2*, defined from the two pairs of intersecting points given as lat1/lon1 and lat2/lon2. A line between those points creates the diagonal of the rectangle as shown in the following figure.



Example

```
#include <stdio.h>
#include <string.h>

#define GEOSTAN_PROPERTIES
#include "geostan.h"

const char *gsxOutPath = "c:\\temp";
```

```
/* This is a random MBR. */
const long minLon = -939460;
const long minLat = 438910;
const long maxLon = -936299;
const long maxLat = 440973;

/* This matches the spatialQueryProgress typedef in geostan.h. */
intl GS_STDCALL progress( void * param, intl mode, intl value )
{
   static long totalSteps;
   switch ( mode )
   {
      case 0:
         printf( "%sStarting...\n", param );
         totalSteps = value;
         break;
      case 1:
         printf( "%s%%%.2f complete\r", param, (double)value/
         (double)totalSteps * 100 );
         break;
      case 2:
         printf( "\n%sDone\n", param );
         break;
   }
   /* In an interactive progress function, such as a dialog box, the user
   could choose to stop this operation. The correct return value would
   then be GS_PROGRESS_CANCEL. */
   return GS_PROGRESS_CONTINUE;
}

int main( int argc, char **argv )
{
   GsId gs;
   PropList initProps;
   PropList statusProps;
   GsFunStat funStat;
   GsSegmentHandle hSegment;

   /* Initialization */
   /* create an initialization property list with geostan information */
   GsPropListCreate(&initProps, GS_INIT_PROP_LIST_TYPE);
   /* GeoStan version */
   GsPropSetLong(&initProps, GS_INIT_GSVERSION, GS_GEOSTAN_VERSION);
   /* license filename and path */
   GsPropSetStr(&initProps, GS_INIT_LICFILENAME, "c:\\lic\\geostan.lic");
   /* license file password */
   GsPropSetLong(&initProps, GS_INIT_PASSWORD, 43218765);
   /* path to the data files, geostan library, and GSX files*/
   GsPropSetStr(&initProps, GS_INIT_DATAPATH,
"c:\\geostan\\datasets;c:\\geostan;c:\\temp");
   /* path and file name of the zip code geocoding data file*/
   GsPropSetStr(&initProps, GS_INIT_Z4FILE, "c:\\geostan\\datasets\\us.z9");
   /* GeoStan options */
   GsPropSetBool(&initProps, GS_INIT_OPTIONS_ADDR_CODE, TRUE);
   GsPropSetBool(&initProps, GS_INIT_OPTIONS_SPATIAL_QUERY, TRUE);
   GsPropSetLong(&initProps, GS_INIT_CACHESIZE, 2);

   /* create a status property list */
   GsPropListCreate(&statusProps, GS_STATUS_PROP_LIST_TYPE);

   gs = GsInitWithProps(&initProps, &statusProps);
   if ( !gs )
   {
      printf( "Failed to initialize GeoStan library.\n" );
      return 1;
   }
  funStat = GsPrepareIndexMbr( gs, minLon, minLat, maxLon, maxLat, gsxOutPath,
"MBR index preparation: ", progress );
   if ( funStat == GS_ERROR )
   {
      printf( "GsPrepareIndexMbr returned GS_ERROR.\n" );
      return 1;
   }
   funStat = GsFindFirstSegmentByMbr( gs, &hSegment, minLon, minLat, maxLon,
maxLat );
   while ( funStat == GS_SUCCESS )
   {
      /* Insert code here to operate on each segment handle retrieved. */
      funStat = GsFindNextSegmentByMbr( gs, &hSegment );
   }
   if ( funStat == GS_ERROR )
   {
      printf( "GsFindFirstSegmentByMbr or GsFindNextSegmentByMbr returned
GS_ERROR.\n" );
      return 1;
   }
   GsTerm(gs);
   GsPropListDestroy(&initProps);
   GsPropListDestroy(&statusProps);
   return 0;
```

GsFindFirstState

WIN	UNIX	z/OS

FINDS THE FIRST STATE MATCHING THE NAME, ABBREVIATION, OR VALID ZIP CODE.

Syntax

```
GsFunStat GsFindFirstState ( GsId gs, gs_const_str statePattern,
pstr stateFound, int bufSize );
```

Arguments

gs ID returned by *GsInitWithProps* for the current instance of GeoStan. *Input.*

statePattern Search string for the state. *Input.*

Can be any of the following items.

- A five-digit ZIP Code.

- The first three digits of a ZIP Code known as the Sectional Center Facility (SCF).
- Various state abbreviations and spellings. For example, for New Hampshire, it accepts: N HAMP, N HAMPSHIRE, NEW HAMPSHIRE, NEWHAMPSHIRE, NH, and NHAMPSHIRE.

stateFound	Proper state abbreviation for the located state. *Output*.
bufSize	Maximum length of data returned in *stateFound*, including NULL. *Input*.

Return Values

GS_ERROR

GS_SUCCESS

GS_NOT_FOUND

Prerequisites

GsInitWithProps and **GsClear**

Notes

This function returns the proper abbreviation for the requested state. You can request states by ZIP Code, full state name, or an alternate abbreviation

Before each find function, call **GsClear** to reset the internal buffers. If you do not reset the buffers, you may receive incorrect results with information from a previous find.

Example

```
/* This example prints the official state abbreviation for ZIP Code 80301. */

char StateAbbr[3];
GsFunStat iStat;

iStat = GsFindFirstState(gs, “80301”, StateAbbr, sizeof(StateAbbr));
if (iStat == GS_SUCCESS)
printf(“%s\n”, StateAbbr);
```

GsFindGeographicFirst

WIN	UNIX	Z/OS

FINDS THE FIRST CITY, COUNTY, AND OR STATE CENTROID MATCH FROM THE SET OF POSSIBLE MATCHES FOUND.

Syntax

```
GsFunStat GsFindGeographicFirst( GsId gs, GsGeoStruct *gstruct );
```

Arguments

gs	ID returned by *GsInitWithProps* for the current instance of GeoStan. *Input.*
**gstruct*	Pointer to the GsGeoStruct containing the input and output fields for this API. *Input, Output.*
inCity	City for which to search. *Input.*
inCounty	County for which to search. *Input.*
inState	State for which to search. *Input.*
outCity	Output city. *Output.*
outCountry	Output county. *Output.*
outState	Output state. *Output.*
outLat	Returned latitude of the geographic centroid. *Output.*
outLong	Returned longitude of the geographic centroid. *Output.*
outRank	Returned geographic rank of the city for city centroid. *Output.*
outResultCode	Result code equivalent (G1 - city centroid, G2 - country centroid, G3 - state centroid). *Output.*
outLocCode	Location code equivalent (GM - city, GC - county, GS - state). *Output.*
outClose	True indicates a close match. *Output.*

Return Values

GS_SUCCESS

GS_ERROR

GS_NOT_FOUND

Prerequisites

GsInitWithProps

Notes

It is recommended that the user first use the Last-line lookup functions to standardize the city, county and state names. This function only performs minimal fuzzy matching on the input city and county names. The location code returned by this function is to provide users with a location code equivalent and is not retrievable using GsDataGet. It is merely provided to offer a consistent label for the type of address match that is returned and will only consist of one of the three Geographic location codes (GM – City, GC – County and GS – State).

Example

To use this API, you will use the following structure defined in geostan.h.

```
/* The Geographic query structure. */
typedef struct GsGeoStruct {
  char inCity[GS_CITY_CCS_LENGTH];                      /* input city */
  char inCounty[GS_COUNTY_CCS_LENGTH];                /* input county */
  char inState[GS_STATE_CCS_LENGTH];                   /* input state */
  char outCity[GS_CITY_CCS_LENGTH];                    /* output city */
  char outCounty[GS_COUNTY_CCS_LENGTH];               /* output county */
  char outState[GS_STATE_CCS_LENGTH];                 /* output state */
  char outLat[GS_LAT_LENGTH];                      /* output latitude */
  char outLong[GS_LON_LENGTH];                    /* output longitude */
  char outRank[GS_RANK_LENGTH];             /* output geographic rank */
  char outResultCode[GS_MM_RESULT_CODE_LENGTH ];  /* output result code */
  char outLocCode[GS_LOC_CODE_LENGTH];       /* output location code */
  char outClose;                         /* output close match flag */
}GsGeoStruct;
```

An example of using the Geographic Geocoding API is as follows. This assumes you have a valid GsId named gs from a prior Geostan Initialization.

```
void GeographicGeocoding(GsId gs)

{
    GsGeoStruct p;
    int ret;

    printf("Enter City : ");
    gets(p.inCity );

    printf("Enter County : ");
    gets(p.inCounty);

    printf("Enter State : ");
    gets(p.inState);

    ret = GsFindGeographicFirst(gs,&p);

    if(ret != 0)
       printf("No record found");
    else
    {
       if(strcmp(p.outResultCode,"G3")==0)
```

```
           printf("%s \t %s \t %s  \t %s \t %s \n",
             p.outCity,p.outState,p.outResultCode,p.outLat,p.outLong);
if(strcmp(p.outResultCode,"G2")==0)
       printf("%s \t %s \t %s  \t %s \t %s \n",
              p.outCounty,p.outState,p.outResultCode,p.outLat,p.outLong);
    if(strcmp(p.outResultCode,"G1")==0)
        printf("%s \t %s \t %s \t %s \n",
              p.outState,p.outResultCode,p.outLat,p.outLong);
    printf("\n");
    while(ret==0)
    {
     ret=GsFindGeographicNext(gs, &p );
     if(ret == 0)
     {

         if(strcmp(p.outResultCode,"G3")==0)
         printf("%s \t %s \t %s  \t %s \t %s \n",
                p.outCity,p.outState,p.outResultCode,p.outLat,p.outLong);
         if(strcmp(p.outResultCode,"G2")==0)
         printf("%s \t %s \t %s  \t %s \t %s \n",
              p.outCounty,p.outState,p.outResultCode,p.outLat,p.outLong);
         if(strcmp(p.outResultCode,"G1")==0)
         printf("%s \t %s \t %s \t %s \n",
               p.outState,p.outResultCode,p.outLat,p.outLong);
         printf("\n");
         }
       }
   }
}
```

GsFindGeographicFirstEx

WIN | UNIX | z/OS

FINDS THE FIRST CITY, COUNTY, AND OR STATE CENTROID MATCH FROM THE SET OF POSSIBLE MATCHES FOUND.

Syntax

`GsFunStat` **`GsFindGeographicFirstEx`**`( GsId` *`gs`*`, GsGeoStructEx*` *`pGeoStruct`* `);`

Arguments

`gs` ID returned by *GsInitWithProps* for the current instance of GeoStan. *Input.*

`*pGeoStruct` Pointer to the GsGeoStruct containing the input and output fields for this API. *Input, Output.*

`inCity` City for which to search. *Input.*

`inCounty` County for which to search. *Input.*

inState State for which to search. *Input*.

outCity Output city. *Output*.

outCountry Output county. *Output*.

outState Output state. *Output*.

outLat Returned latitude of the geographic centroid. *Output*.

outLong Returned longitude of the geographic centroid. *Output*.

outRank Returned geographic rank of the city for city centroid. *Output*.

outResultCode Result code equivalent (G1 - city centroid, G2 - country centroid, G3 - state centroid). *Output*.

outLocCode Location code equivalent (GM - city, GC - county, GS - state). *Output*.

inGeoLibVer GeoStan version. *Input*.

outClose True indicates a close match. *Output*.

outFipsCode FIPS Code. *Output*.

Return Values

GS_SUCCESS

GS_ERROR

GS_NOT_FOUND

Prerequisites

GsInitWithProps

Notes

It is recommended that the user first use the Last-line lookup functions to standardize the city, county and state names. This function only performs minimal fuzzy matching on the input city and county names. The location code returned by this function is to provide users with a location code equivalent and is not retrievable using GsDataGet. It is merely provided to offer a consistent label for the type of address match that is returned and will only consist of one of the three Geographic location codes (GM – City, GC – County and GS – State).

Example

To use this API, you will use the following structure defined in geostan.h.

```
/* The Geographic query structure. */
typedef struct GsGeoStructEx{
   char inCity[GS_CITY_CCS_LENGTH];                /* input city */
   char inCounty[GS_COUNTY_CCS_LENGTH];            /* input county */
   char inState[GS_STATE_CCS_LENGTH];              /* input state */
   char outCity[GS_CITY_CCS_LENGTH];               /* output city */
   char outCounty[GS_COUNTY_CCS_LENGTH];           /* output county */
   char outState[GS_STATE_CCS_LENGTH];             /* output state */
   char outLat[GS_LAT_LENGTH];                     /* output latitude */
   char outLong[GS_LON_LENGTH];                    /* output longitude */
   char outRank[GS_RANK_LENGTH];                 /* output geographic rank */
   char outResultCode[GS_MM_RESULT_CODE_LENGTH ];  /* output result code */
   char outLocCode[GS_LOC_CODE_LENGTH];           /* output location code */
   char outClose;                               /* output close match flag */
   long inGeoLibVer;                               /* geostan version */
   char outFipsCode[GS_FIPS_CCS_LENGTH];           /* output fips code */
} GsGeoStructEx;
```

An example of using the Geographic Geocoding API is as follows. This assumes you have a valid GsId named gs from a prior Geostan Initialization.

```
void GeographicGeocoding(GsId gs)

{
    GsGeoStruct p;
    int ret;

    printf("Enter City : ");
    gets(p.inCity );

    printf("Enter County : ");
    gets(p.inCounty);

    printf("Enter State : ");
    gets(p.inState);

    ret = GsFindGeographicFirst(gs,&p);

    if(ret != 0)
       printf("No record found");
    else
    {
       if(strcmp(p.outResultCode,"G3")==0)
           printf("%s \t %s \t %s  \t %s \t %s \n",
             p.outCity,p.outState,p.outResultCode,p.outLat,p.outLong);
if(strcmp(p.outResultCode,"G2")==0)
       printf("%s \t %s \t %s  \t %s \t %s \n",
              p.outCounty,p.outState,p.outResultCode,p.outLat,p.outLong);
    if(strcmp(p.outResultCode,"G1")==0)
        printf("%s \t %s \t %s \t %s \n",
              p.outState,p.outResultCode,p.outLat,p.outLong);
    printf("\n");
    while(ret==0)
    {
     ret=GsFindGeographicNext(gs, &p );
     if(ret == 0)
     {

         if(strcmp(p.outResultCode,"G3")==0)
         printf("%s \t %s \t %s  \t %s \t %s \n",
                p.outCity,p.outState,p.outResultCode,p.outLat,p.outLong);
         if(strcmp(p.outResultCode,"G2")==0)
         printf("%s \t %s \t %s  \t %s \t %s \n",
              p.outCounty,p.outState,p.outResultCode,p.outLat,p.outLong);
         if(strcmp(p.outResultCode,"G1")==0)
         printf("%s \t %s \t %s \t %s \n",
               p.outState,p.outResultCode,p.outLat,p.outLong);
         printf("\n");
         }
       }
   }
}
```

WIN | UNIX | z/OS

GsFindGeographicNext

FINDS THE NEXT CITY, COUNTY, AND OR STATE CENTROID MATCH FROM THE SET OF POSSIBLE MATCHES FOUND.

Syntax

```
GsFunStat GsFindGeographicNext( GsId gs, GsGeoStruct *gstruct );
```

Arguments

gs ID returned by *GsInitWithProps* for the current instance of GeoStan. *Input.*

**gstruct* Pointer to the GsGeoStruct containing the input and output fields for this API. *Input, Output.*

outCity Output city. *Output.*

outCountry Output county. *Output.*

outState Output state. *Output.*

outLat Returned latitude of the geographic centroid. *Output.*

outLong Returned longitude of the geographic centroid. *Output.*

outRank Returned geographic rank of the city for city centroid. *Output.*

outResultCode Result code equivalent (G1 - city centroid, G2 - country centroid, G3 - state centroid). *Output*.

outLocCode Location code equivalent (GM - city, GC - county, GS - state). *Output*.

outClose True indicates a close match. *Output*.

Return Values

`GS_SUCCESS`

`GS_ERROR`

`GS_NOT_FOUND`

Prerequisites

`GsInitWithProps`

Notes

It is recommended that the user first use the Last-line lookup functions to standardize the city, county and state names. This function only performs minimal fuzzy matching on the input city and county names. The location code returned by this function is to provide users with a location code equivalent and is not retrievable using GsDataGet. It is merely provided to offer a consistent label for the type of address match that is returned and will only consist of one of the three Geographic location codes (GM – City, GC – County and GS – State).

Example

See the example for GsFindGeographicFirst.

GsFindGeographicNextEx

WIN	UNIX	z/OS

FINDS THE NEXT CITY, COUNTY, AND OR STATE CENTROID MATCH FROM THE SET OF POSSIBLE MATCHES FOUND.

Syntax

```
GsFunStat GsFindGeographicNextEx( GsId gs, GsGeoStructEx* pGeoStruct );
```

Arguments

`gs` ID returned by *GsInitWithProps* for the current instance of GeoStan. *Input.*

`*pGeoStruct` Pointer to the GsGeoStruct containing the input and output fields for this API. *Input, Output.*

`inCity` City for which to search. *Input.*

`inCounty` County for which to search. *Input.*

`inState` State for which to search. *Input.*

`outCity` Output city. *Output.*

`outCountry` Output county. *Output.*

`outState` Output state. *Output.*

`outLat` Returned latitude of the geographic centroid. *Output.*

`outLong` Returned longitude of the geographic centroid. *Output.*

`outRank` Returned geographic rank of the city for city centroid. *Output.*

`outResultCode` Result code equivalent (G1 - city centroid, G2 - country centroid, G3 - state centroid). *Output.*

`outLocCode` Location code equivalent (GM - city, GC - county, GS - state). *Output.*

`inGeoLibVer` GeoStan version. *Input.*

`outClose` True indicates a close match. *Output.*

`outFipsCode` FIPS Code. *Output.*

Return Values

GS_SUCCESS

GS_ERROR

GS_NOT_FOUND

Prerequisites

GsInitWithProps

Notes

It is recommended that the user first use the Last-line lookup functions to standardize the city, county and state names. This function only performs minimal fuzzy matching on the input city and county names. The location code returned by this function is to provide users with a location code equivalent and is not retrievable using GsDataGet. It is merely provided to offer a consistent label for the type of address match that is returned and will only consist of one of the three Geographic location codes (GM – City, GC – County and GS – State).

Example

See the example for GsFindGeographicFirstEx.

GsFindNext___

WIN	UNIX	z/OS

FINDS THE NEXT STREET, SEGMENT, OR RANGE OBJECT THAT MEETS THE SEARCH CRITERIA.

Syntax

```
GsFunStat GsFindNextRange( GsId gs, GsRangeHandle *pRange );
GsFunStat GsFindNextSegment( GsId gs, GsSegmentHandle *pSegment );
GsFunStat GsFindNextStreet( GsId gs, GsStreetHandle *pStreet );
```

Arguments

gs	ID returned by *GsInitWithProps* for the current instance of GeoStan. *Input.*
**pRange*	Pointer to a segment/range handle. Returns a valid handle to the next range object, if there is one. *Input, Output.*
**pSegment*	Pointer to a street/segment handle. Returns a valid handle to the next segment object, if there is one. *Input, Output.*
**pStreet*	Pointer to a street handle. Returns a valid handle to the next street object, if there is one. *Input, Output.*

Return Values

GS_SUCCESS

GS_NOT_FOUND

GS_ERROR

Prerequisites

`GsFindFirst_` and `GsClear`

Notes

This function continues to find objects matching the criteria specified in `GsFindFirst__`.

If GeoStan finds a matching segment, you can retrieve the data using **`GsHandleGet`**.

Before each find function, call **`GsClear`** to reset the internal buffers. If you do not reset the buffers, you may receive incorrect results with information from a previous find.

Refer to "Extracting data from GSD files" on page 407 for details on using all of the **`GsFindFirst___`** and **`GsFindNext___`** functions.

NOTE: GsFindNextStreet does not respect parity.

Example

See the code example in Appendix B, "Extracting data from GSD files."

GsFindNextSegmentByMbr

WIN	UNIX

FINDS THE NEXT SEGMENT HANDLE WITHIN THE SPECIFIED MBR.

Syntax

```
GsFunStat GsFindNextSegmentByMbr( GsId gs, GsSegmentHandle *pSegment );
```

Arguments

gs ID returned by *GsInitWithProps* for the current instance of GeoStan. *Input.*

pSegment Pointer to a segment handle. Returns a valid handle to the next segment object, if there is one. *Output.*

Return Values

`GS_ERROR` Error occurred. You can retrieve the error using **`GsErrorGet`**.

`GS_LASTLINE_NOT_FOUND` GeoStan could not find *`pLocale`*.

`GS_NOT_FOUND` GeoStan did not find a match.

GS_SUCCESS	GeoStan found a match. If returned, *pSegment* is a valid segment handle that you can use with **GsHandleGet**, **GsHandleGetCoords**, or **GsGetMbr**.

Prerequisites

GsFindFirstSegmentByMBR and **GsClear**

Notes

GeoStan handles all geocodes in the spatial query functions in ten-thousandths of degrees—not millionths like **GsDataGet**.

Before each find function, call **GsClear** to reset the internal buffers. If you do not reset the buffers, you may receive incorrect results with information from a previous find.

Example

See GsFindFirstSegmentByMbr for a code example.

MBR Representation by Lat/Lon

See GsFindFirstSegmentByMbr for an MBR representation by latitude and longitude.

GsFindWithProps

WIN	UNIX	z/OS

FINDS A MATCH WITH USER SETTABLE MATCH PREFERENCES (PROPERTIES).

Syntax

GsFunStat **GsFindWithProps** (*GsId gs*, *PropList** pProps);

Arguments

GsId gs	GeoStan handle. *Input.*
*PropList**	Pointer to property list structure. *Input.*

Return Values

GS_ADDRESS_NOT_RESOLVED	GeoStan cannot resolve a match candidate.

GS_ADDRESS_NOT_FOUND GeoStan did not find an address match or if you have a metered license and the GeoStan record count is depleted.

For metered licenses, **GsFind** decrements the GeoStan record counter in the license each time it returns a non-error match code. If the counter reaches zero or below, **GsFind** returns GS_ADDRESS_NOT_FOUND and sets the match code to E015. If you have an unmetered license, no decrementing occurs, and processing is unlimited.

GS_ERROR Low-level errors; use **GsErrorGet** to retrieve the error information.

GS_LASTLINE_NOT_FOUNDGeoStan did not find a match for city/state or ZIP Code.

GS_SUCCESS GeoStan found a match.

Prerequisites

GsPropListCreate
GsPropSet*

Notes

The application owns the property list, but GeoStan is the active user.

Do not destroy the property list while GeoStan is still using that property list.

See table GsFindWithProps properties.

Example

```
GsFunStat retval;
PropList findProps;
GsPropListCreate( &findProps, GS_FIND_PROP_LIST_TYPE  );
GsPropSetLong( &findProps, GS_FIND_MATCH_MODE, GS_MODE_CASS );
GsPropSetLong( &findProps, GS_FIND_STREET_OFFSET, 50 );
GsPropSetLong( &findProps, GS_FIND_CENTERLINE_OFFSET, 0 );
GsPropSetStr ( &findProps, GS_FIND_STREET_OFFSET_UNITS, "F" );
GsPropSetStr ( &findProps, GS_FIND_CLIENT_CRS, "NAD83" );
GsPropSetBool( &findProps, GS_FIND_ADDRCODE, TRUE );
GsPropSetBool( &findProps, GS_FIND_Z_CODE, FALSE );
GsPropSetBool( &findProps, GS_FIND_FINANCE_SEARCH, TRUE );
GsPropSetBool( &findProps, GS_FIND_MIXED_CASE, TRUE );
   switch( GsFindWithProps( gs, &findProps )) {
      case GS_SUCCESS:
          GsDataGet(gs, GS_OUTPUT, GS_ADDRLINE, address, sizeof(address));
          break;
      case GS_ERROR:
      case GS_ADDRESS_NOT_FOUND:
      case GS_ADDRESS_NOT_RESOLVED:
      case GS_LASTLINE_NOT_FOUND:
      default:
         fprintf(stderr, "Error geocoding address.\n");
   }
```

GsFormatCASSHeader

WIN | UNIX | z/OS

WRITES A CASS 3553 REPORT TO THE HEADER BUFFER ARGUMENT.

Syntax

```
GsFunStat GsFormatCASSHeader( GsId gs, GsExtendCASSData *pData,
int dataSize, pstr headerBuffer, int bufSize);
```

Arguments

gs ID returned by *GsInitWithProps* for the current instance of GeoStan. *Input.*

pData* Writes the CASS information to the header buffer using the data passed in **GsExtendCASSData. This structure contains the following (lengths are in brackets):

char CASS_Z4CompanyName[40] Name of the company requesting certification. *Input.*

char CASS_Z4Config[4] CASS Z4Change configuration. *Input.*

char CASS_Z4SoftwareName[30] Name of the CASS Z4Change software. *Input.*

char DPV_FileFormat DPV date file format: F - Full, S - Split, and L - Flat. *Input.*

char certificationDate[24] Date of certification. *Input.*

intlu DPVDatabaseDate DPV database date. *Input.*

intlu EWS_Denial Number of matches denied due to a matching address found in `Ews.txt`. *Input.*

char listName[20] Name of the list. *Input.*

char listProcessorName[25] Name of the list processor. *Input.*

intlu LOTDatabaseDate eLOT database date. *Input.*

char LOT_DPCCompanyName[40] Name of the company requesting eLOT & DPC Utility certification. *Input.*

char LOT_DPCConfig[4] eLOT & DPC Utility configuration. *Input.*

char LOT_DPCSoftwareName[30] Name of the eLOT & DPC Software. *Input.*

char LOTVersion[12] eLOT software version. *Input.*

`char version[12]` Software version. *Input.*

`intlu nCarrt` Number of Carrier Routes coded records. *Input.*

`intlu nDpbc` Number of delivery point coded records. *Input.*

`intlu nHighRiseDefault` Number of records that matched to the default highrise record. *Input.*

`intlu nHighRiseExact` Number of records that matched to an exact highrise record. *Input.*

`intlu nLACS` Number of LACS^{Link} converted addresses, if LACS^{Link} is loaded. If LACS^{Link} is not loaded, the number of records that match to a LACS ZIP + 4 record. *Input.*

`intlu nLot` Number of eLOT coded records. *Input.*

`intlu nRecs` Number of processed records. *Input.*

`intlu nRuralRouteDefault` Number of records matched to a default rural route record. *Input.*

`intlu nRuralRouteExact` Number of records matched to an exact rural record. *Input.*

`intlu nSuiteLink` Number of records processed through Suite^{Link}. *Input.*

`intlu nTotalDPV` Deprecated. Number of DPV confirmed addresses. Input.

`intlu nZIP` Number of 5-digit ZIP Coded records. *Input.*

`intlu nZIP4` Number of ZIP + 4 coded records (number of DPV confirmed records). *Input.*

`intlu nZ4Changed` Number of records skipped (not reprocessed) because the ZIP + 4 did not change. *Input.*

`char pSearchPath[256]` Location of GeoStan data. *Input.*

`intlu structVersion` Set equal to `GS_GEOSTAN_VERSION`. *Input.*

`char templateName[256]` Full path and name of `cass3553.frm` template file. *Input.*

`intlu zip4DatabaseDate` ZIP + 4 database date. *Input.*

`char z4ChangeVersion[12]` Z4Change software version. *Input.*

`dataSize` Size of the CASS report data structure. *Input.*

`headerBuffer` Buffer containing the CASS header line from the Stage file. *Input, Output.*

`bufSize` Length of the header buffer. *Input.*

Return Values

GS_SUCCESS	
GS_ERROR	Call GsErrorGetEx for more information.
GS_WARNING	Call GsErrorGetEx for more information.

Prerequisites

GsInitWithProps

Notes

When you specify a version number for either version, z4ChangeVersion or LOTVersion, GeoStan updates the corresponding fields in the header buffer similar to GsWriteExtendCASSReport. For example, entering a version number for z4ChangeVersion prompts GeoStan to fill the following fields in cass3553.frm:

- Section "B. List", Item "2b": Date List Processed Z4Change
- Section "B. List", Item "3b": Date of Database Product Used Z4Change
- Section "C. Output", Item "1b": Total Coded Z4Change Processed

To develop CASS certified applications in GeoStan, you must have the correct license agreement with Pitney Bowes Business Insight. You must also obtain CASS certification from the USPS for your application. Using GeoStan does not make an application CASS certified. For information on becoming CASS certified, refer to Appendix E, "CASS certification."

This data is defined in geostan.h in the data structure **GsExtendCASSData**.

Example

```
/* assign outCASSHeader flag to process->writeCASSHeader */
/* open address input file */
/* if writeCASSHeader flag is set to 1 read and store headerline first */
/* otherwise ignore it */
if( process->writeCASSHeader == 1 )
{
process->headerBuffer = (pstr)malloc( process->inRecLen + 1 );

/* get the headerline */
fgets( process->headerBuffer, process->inRecLen,
process->inputFile);

/* write the header line as is to the output file as a spaceholder */
fwrite(process->headerBuffer, process->inRecLen, 1, process
->outputFile);
      fputs( "\n", process->outputFile );
}
.
```

```
.
.
/* Geocode each address line and write its output record to the output file */
.
.
.
/* if writeCASSHeader flag is set to 1 -> write the CASS header to the output
file... */
if( process->writeCASSHeader == 1 )
{
/* copy the data from CASSData structure into the correct positions in the
header as defined by USPS */
GsFormatCASSHeader( process->gs, &CASSData, sizeof( CASSData ), process-
>headerBuffer, process->inRecLen );

/* set the file pointer to the beginning of the file */
fseek( process->outputFile, 0L, SEEK_SET );

/* - write the formatted header line to the output file */
fwrite(process->headerBuffer, process->inRecLen, 1,
process->outputFile);
   }
```

GsFormatDpvFalsePosDetail

WIN	UNIX	z/OS

Formats a DPV false-positive detail record from GsDpvGetFalsePosDetail.

Syntax

```
GsFunStat GsFormatDpvFalsePosDetail( GsId gs,
GsDpvFalsePosDetailData *pData, int structSize, pstr detail,
int detailLength );
```

Arguments

gs ID returned by *GsInitWithProps* for the current instance of GeoStan. *Input.*

pData* Pointer to the DPV report data structure. This value is completed by **`GsDpvGetFalsePosDetail()`.

structSize Size of the **`GsDpvFalsePosDetail`** data structure. *Input.*

detail Buffer containing the DPV false-positive detail record after `GsFormatDpvFalsePosDetail()` successfully completes. When the GeoStan application writes the false-positive report, it writes this buffer to the last line of the file. *Output.*

detailLength Length of the detail buffer. *Input.*

Return Values

`GS_SUCCESS`	
`GS_ERROR`	Call **`GsErrorGetEx`** for more information.
`GS_WARNING`	Call **`GsErrorGetEx`** for more information.

Prerequisites

`GsDpvGetFalsePosDetail()`

GsFormatDpvFalsePosHeader

WIN	UNIX	z/OS

FORMATS A DPV FALSE-POSITIVE HEADER RECORD WITH DATA FROM GSDPVGETFALSEPOSHEADERSTATS.

Syntax

```
GsFunStat GsFormatDpvFalsePosHeader( GsId gs,
GsFalsePosHeaderData *pData, int structSize, pstr header,
int headerLength );
```

Arguments

gs	ID returned by *GsInitWithProps* for the current instance of GeoStan. *Input.*
**pData*	Pointer to the DPV report data structure. This value is completed by **`GsDpvGetFalsePosHeaderStats()`**. *Input.*
structSize	Size of the `GsFalsePosHeaderData` data structure. *Input.*
header	Buffer containing the DPV false-positive header after `GsFormatDpvFalsePosHeader()` successfully completes. When the GeoStan application writes the false-positive report, it writes this buffer to the first line of the file. *Output.*
headerLength	Length of the header buffer. *Input.*

Return Values

`GS_SUCCESS`	
`GS_ERROR`	Call `GsErrorGetEx` for more information.

`GS_WARNING` Call `GsErrorGetEx` for more information.

Prerequisites

```
GsInitWithProps() with appropriate DPV initialization properties set.
GsDpvGetFalsePosHeaderStat()
```

GsFormatLACSFalsePosDetail

WIN	UNIX	z/OS

Formats a LACS^{Link} false-positive detail record from GsLACSGetFalsePosDetail.

Syntax

```
GsFunStat GsFormatLACSFalsePosDetail( GsId gs,
GsFalsePosDetailData *pData, int structSize, pstr detail,
int detailLength );
```

Arguments

`gs` ID returned by *GsInitWithProps* for the current instance of GeoStan. *Input.*

`*pData` Pointer to the LACS^{Link} report data structure. This value is completed by **`GsLACSGetFalsePosDetail()`**. *Input.*

`structSize` Size of the LACS^{Link} report data structure. *Input.*

`detail` Buffer containing the LACS^{Link} false-positive detail record after `GsFormatLACSFalsePosDetail()` successfully completes. When the GeoStan application writes the false-positive report, it writes this buffer to the last line of the file. *Output.*

`detailLength` Length of the detail buffer. *Input.*

Return Values

`GS_SUCCESS`

`GS_ERROR` Call `GsErrorGetEx` for more information.

`GS_WARNING` Call `GsErrorGetEx` for more information.

Prerequisites

`GsLACSGetFalsePosDetail()`

GsFormatLACSFalsePosHeader

WIN	UNIX	z/OS

FORMATS A LACS^{Link} FALSE-POSITIVE HEADER RECORD WITH DATA FROM GSLACSGETFALSEPOSHEADERSTATS.

Syntax

```
GsFunStat GsFormatLACSFalsePosHeader( GsId gs,
GsFalsePosHeaderData * pData, int structSize, pstr header,
int headerLength );
```

Arguments

gs	ID returned by *GsInitWithProps* for the current instance of GeoStan. *Input.*
**pData*	Pointer to the LACS^{Link} report data structure. This value is completed by **`GsLACSGetFalsePosHeaderStats()`**. *Input.*
structSize	Size of `GsFalsePosHeaderData` data structure. *Input.*
header	Buffer containing the LACS^{Link} false-positive header after `GsFormatLACSFalsePosHeader()` successfully completes. When the GeoStan application writes the false-positive report, it writes this buffer to the first line of the file. *Output.*
headerLength	Length of the header buffer. *Input.*

Return Values

GS_SUCCESS	
GS_ERROR	Call GsErrorGetEx for more information.
GS_WARNING	Call GsErrorGetEx for more information.

Prerequisites

`GsLACSGetFalsePosHeaderStats()`

GsGetCoordsEx

WIN	UNIX	z/OS

RETRIEVES COORDINATES FOR THE STREET SEGMENT FOUND VIA GSFIND.

Syntax

```
ints GsGetCoordsEx( GsId gs, pintl pCoords, intsu maxPoints );
```

Arguments

`gs` ID returned by *GsInitWithProps* for the current instance of GeoStan. *Input.*

`pCoords` Array of coordinates, in `x,y` (longitude, latitude) order. *Output.*

`maxPoints` Maximum number of points to return; used to prevent writing past the end of the `pCoords` buffer. *Input.*

Return Values

Number of points assigned to buffer.

Prerequisites

`GsFindWithProps`

Notes

This function returns an array of coordinates for the current feature found via **`GsFind`**.

GeoStan can return a maximum of 64 coordinate pairs, each pair consisting of two long integers.

Pass in a value of 2 for the maxPoints parameter in order to return the end points of the segment matched.

GeoStan returns the coordinates to the full precision stored in the GSD file and scales coordinate pairs to integers. For example, GeoStan returns a point at (-98.3, 29.7) as (983000, 297000). This is a different scale from that expected by Spatial+ and similar GIS applications, which typically express coordinates in millionths of degrees. You may need to scale coordinates obtained with this function before using them as input to other software libraries or applications.

GsGetDBMetadata

WIN	UNIX	z/OS

POPULATES THE SUPPLIED BUFFER WITH THE INFORMATION ABOUT THE ITEM BEING

QUERIED FOR THE SPECIFIED DATABASE.

Syntax

```
ints GsGetDBMetadata( GsId gs, int WhichItem, int WhichDB, pstr buffer, int
bufSize );
```

Arguments

gs ID returned by *GsInitWithProps* for the current instance of GeoStan. *Input.*

WhichItem Specific information to return. The following table includes types of available information. *Input*.

NOTE: See the *Return Values* section for information on the data types and datum returned.

Mode	Data Returned
GS_STATUS_DATATYPE_NUM	Returns a numeric constant representing the data type.
GS_STATUS_DATATYPE_STR	Returns **GS_SUCCESS** or **GS_ERROR.** GeoStan places this retrieved information in the buffer.
GS_STATUS_DATUM_NUM	Returns a numeric constant representing the datum.
GS_STATUS_DATUM_STR	This is the NAD used natively by the data. It does not reflect the datum currently in use by GeoStan. See GS_FIND_CLIENT_CRS for further information on setting the NAD for geocoding. Returns GS_SUCCESS or GS_ERROR. The retrieved information is placed in the buffer.
GS_STATUS_GEO_PRECISION	Returns the number of decimal places of longitude/latitude stored in the GSD file. Default is **4**. Point-level data is **6**.
GS_STATUS_DB_COPYRIGHT	Retuns copyright information for User Dictionary.
GS_STATUS_DB_COUNTRY	Retuns country associated with User Dictionary.
GS_STATUS_DB_FILE_PATH	Retuns filesystem path to the User Dictionary.
GS_STATUS_DB_VERSION	Retuns version string for the User Dictionary.

WhichDB A database index, starting at 0 and less than the number of databases returned by GsGetNumDB. *Input.*

buffer Location to store the returned data. *Output.*

`bufSize`	Maximum size of data for GeoStan to return. Geostan.h lists as constraints the recommended buffer sizes for each item. These sizes include the null terminator and are the maximum lengths required to get the full output string. You can allocate a buffer that is smaller or larger than these values. However, if bufSize is shorter than the returned data, GeoStan truncates the data and does not generate an error. *Input*.

Return Values

`GS_ERROR`

`GS_SUCCESS`

Prerequisites

`GsInitWithProps`

GsGetLibVersion

WIN	UNIX	z/OS

RETURNS THE CURRENT VERSION OF THE GEOSTAN LIBRARY.

Syntax

```
GsFunStat GsGetLibVersion( );
```

Arguments

None.

Return Values

Low Byte	Major Version number.
High Byte	Minor Version number.

Prerequisites

`GsInitWithProps`

Notes

In general, the major version number changes whenever PBBI adds a new API features, or when the data structures in the GeoStan data files change.

GsGetMbr

WIN | UNIX

RETURNS AN MBR FOR THE SPECIFIED GEOGRAPHIC ELEMENT.

Syntax

```
GsFunStat GsGetMbr( GsId gs, GsEnum which, void *pData,
long *pMinLong, long *pMinLat, long *pMaxLong,
long *pMaxLat );
```

Arguments

gs ID returned by *GsInitWithProps* for the current instance of GeoStan. *Input.*

which The following table contains the possible values to which GeoStan assumes `pData` is pointing.

GS_MBR_SEGMENT	`GsSegmentHandle`.
GS_MBR_STREET	`GsStreetHandle`.
GS_MBR_CITY	pstr, containing digits in the format **SSFFFFCCC** where **S** is state, **F** is finance, and **C** is city.
GS_MBR_FINANCE	pstr, containing digits in the format **SSFFFF** where **S** is state, and **F** is finance

pData Points at value passed for *which*. *Input.*

pMinLong Minimum longitude. One of four values that make up the MBR. *Output.*

pMinLat Minimum latitude. One of four values that make up the MBR. *Output.*

pMaxLong Maximum longitude. One of four values that make up the MBR. *Output.*

pMaxLat Maximum latitude. One of four values that make up the MBR. *Output.*

Return Values

GS_ERROR For `GS_MBR_SEGMENT` and `GS_MBR_STREET`, if the specified geographical element has no spatial data, then all outputs have a value of zero. For `GS_MBR_CITY` and

`GS_MBR_FINANCE`, if GeoStan cannot find the specified geographical element all outputs have a value of 0 (zero).

`GS_SUCCESS` The four longs hold the requested MBR.

Prerequisites

`GsFindFirstStreet` `and` **`GsCityDataGet`**

Notes

GeoStan handles all geocodes in these functions in ten-thousandths of degrees—not millionths like **`GsDataGet`**.

To make a call with `GS_MBR_CITY`, you must first make a call to **`GsPrepareIndexFinance`**. To retrieve an MBR for `GS_MBR_CITY`, you must have already built the `.gsx` file for the finance number of the city. Use **`GsDataGet`** with `GS_QCITY` to get a city number or a finance number (finance number is the first 6 digits of the city number).

MBR Representation by Lat/Lon

See GsFindFirstSegmentByMbr for an MBR representation by latitude and longitude.

GsGetNumDB

WIN	UNIX	z/OS

RETURNS THE NUMBER OF LOADED DATABASES.

Syntax

```
intlu GsGetNumDB( GsId gs );
```

Arguments

`gs` ID returned by *GsInitWithProps* for the current instance of GeoStan. *Input.*

Return Values

`GS_ERROR`

`GS_SUCCESS`

Prerequisites

```
GsInitWithProps
```

GsHandleGet

WIN	UNIX	z/OS

RETRIEVES DATA FOR OBJECTS FOUND VIA GSFINDFIRST AND GSFINDNEXT.

Syntax

```
GsFunStat GsHandleGet( GsId gs, GsEnum fSwitch,
GsRangeHandle *pRange, pstr pBuffer, intsu bufLen );
```

Arguments

gs ID returned by *GsInitWithProps* for the current instance of GeoStan. *Input.*

NOTE: See "Common enums for storing and retrieving data" on page 68 for a list of valid enums and maximum sizes.

fSwitch Enum for the argument you wish to retrieve. *Input*.

**pRange* Pointer to the current range handle. *Input*.

pBuffer Location to store the returned data. *Output*.

bufLen Maximum size of the data that GeoStan returns. If *bufLen* is shorter than the data returned by GeoStan, GeoStan truncates the data and does not generate an error. *Input*.

Return Values

```
GS_SUCCESS

GS_ERROR
```

Prerequisites

```
GsFindFirst or GsFindNext
```

Notes

This function retrieves data from the geocode buffer for a given range handle. If you have a street or segment handle, you must convert the handle to a range handle before you can use this function.

See Appendix B, "Extracting data from GSD files" for more information on the Find First and Find Next searches. For information on extracting Alias information, refer to the description in `GsMultipleGet`.

GsHandleGetCoordsEx

WIN	UNIX	z/OS

RETRIEVES THE COORDINATES FOR A STREET SEGMENT OBJECT FOUND VIA GSFINDFIRST AND GSFINDNEXT.

Syntax

```
ints GsHandleGetCoordsEx( GsId gs, GsSegmentHandle *pHandle,
pintl pCoords, intsu maxPoints );
```

Arguments

gs	ID returned by *GsInitWithProps* for the current instance of GeoStan. *Input*.
**pHandle*	Handle of the segment object for the coordinates returned. *Input*.
pCoords	Array of coordinates, in `x,y` (longitude, latitude) order. *Output*.
maxPoints	Maximum number of points that **`GsGetCoordsEx`** returns; used to prevent writing past the end of *pCoords* buffer. *Input*.

Return Values

Number of points assigned to the buffer.

Prerequisites

`GsFindFirst` or **`GsFindNext`**

Notes

This function returns an array of coordinates for the segment to which *handle* points.

GeoStan can return a maximum of 64 coordinate pairs, each pair consisting of two long integers.

Pass in a value of 2 for the maxPoints parameter in order to return the end points of the segment matched.

Example

```
/*This example retrieves coordinates for a street segment found using
GsFindFirstSegment.*/

#define MAX_POINTS 50
.
.
.
ints NumCoords;
intl Coords[MAX_POINTS *2];

NumCoords = GsHandleGetCoordsEx(gs, &hSegment, Coords, MAX_POINTS);
for (int i = 0; i < NumCoords * 2; i += 2)
printf("Lon = %ld, Lat = %ld\n", Coords[i], Coords[i+1]);
```

GsInitWithProps

WIN	UNIX	z/OS

INITIALIZES GEOSTAN USING PROPERTIES.

Syntax

```
GsId GsInitWithProps (PropList* pInitProps, PropList* pStatusProps);
```

Arguments

*pInitProps** Pointer to property list structure of type GS_INIT_PROP_LIST_TYPE. *Input.*

pStatusProps Pointer to property list structure of type GS_STATUS_PROP_LIST_TYPE. *Output.*

Return Values

```
Returns the ID of the GeoStan instance that was initialized.
```

Prerequisites

GsPropListCreate and **GsPropSet***

Notes

Initializes GeoStan using the property list. The application owns the property lists. The status property list is guaranteed to contain properties for all defined status properties with their values properly set.

GsInitWithProps can be used to initialize GeoStan, DPV, and LACSLink with a single call (if the appropriate initialization properties are in the init list). When this function successfully completes, it populates the GS_INIT_GEOSTAN_ID property in the property list referred to by the pInitProps parameter with the actual GeoStan ID.

If you invoke this function with the GeoStan ID property pre-set to a valid GeoStan ID, it will not attempt to re-initialize GeoStan. This is how GsInitWithProps can be used to initialize DPV and/or LACSLink after GeoStan has already been initialized with a previous call to GsInitWithProps. If you intend to reuse the same initialization property list to initialize GeoStan several times you must reset the GS_INIT_GEOSTAN_ID property back to zero, or completely remove the property from the list. This informs GeoStan that you want a new GeoStan instance when you call GsInitWithProps.

These initialization properties are required for GsInitWithProps to function correctly:

- GS_INIT_LICFILENAME
- GS_INIT_DATAPATH
- GS_INIT_PASSWORD

These initialization properties are recommended, but not required:

- GS_INIT_Z4FILE (required for zip centroid matching)
- GS_INIT_OPTIONS_Z9_CODE (required for zip centroid matching)
- GS_INIT_CACHESIZE (to improve GeoStan performance)
- GS_INIT_OPTIONS_ADDR_CODE (required for address matching)
- GS_INIT_OPTIONS_SPATIAL_QUERY (required for reverse geocoding and the MBR-related functions)

All other available initialization properties are entirely optional. See tables GsInitWithProps input properties.

Example

```
GsFunStat retval;
PropList initProps;
PropList statusProps;
GsPropListCreate( &initProps, GS_INIT_PROP_LIST_TYPE);
GsPropListCreate( &statusProps, GS_STATUS_PROP_LIST_TYPE);
GsPropSetStr( &initProps, GS_INIT_LICFILENAME, "c:\\test\\Data\\my.lic");
GsPropSetStr(&initProps, GS_INIT_DATAPATH, "c:\\test\\Data" );
GsPropSetStr(&initProps, GS_INIT_Z4FILE, ""c:\\test\\Data\\us.z9" );
GsPropSetLong(&initProps, GS_INIT_PASSWORD, 12345678 );
GsPropSetLong(&initProps, GS_INIT_GSVERSION, GS_GEOSTAN_VERSION );
GsPropSetBool(&initProps, GS_INIT_OPTIONS_ADDR_CODE, TRUE );
GsPropSetBool(&initProps, GS_INIT_OPTIONS_Z9_CODE, TRUE );
GsPropSetBool(&initProps, GS_INIT_OPTIONS_SPATIAL_QUERY, TRUE );
GsPropSetBool(&initProps, GS_INIT_OPTIONS_APN_CODE, TRUE );
GsPropSetBool(&initProps, GS_INIT_OPTIONS_ELEVATION_CODE, TRUE );
GsPropSetLong(&initProps, GS_INIT_CACHESIZE, 0 );
GsPropSetBool(&initProps, GS_INIT_ENABLE_AIRPORT, TRUE );
GsPropSetBool(&initProps, GS_INIT_ENABLE_HIGHWAY_EXIT, TRUE );
```

```
GsPropSetLong(&initProps, GS_INIT_FILE_MEMORY_LIMIT, 4000 );
GsPropSetLong(&initProps, GS_INIT_GEOSTAN_ID, 0 );   // not required, but
demonstrates the usage.
gs = GsInitWithProps( &initProps, &statusProps );
```

GsLACSClearStats

WIN | UNIX | z/OS

Clears the LACS$^{\text{Link}}$ statistics.

Syntax

```
GsFunStat GsLACSClearStats( GsId gs, GsLACSCompleteStats *pStats, int
structSize );
```

Arguments

gs ID returned by *GsInitWithProps* for the current instance of GeoStan. *Input.*

structSize Size of the **GsLACSClearStats** data structure. *Input.*

Return Values

GS_SUCCESS

GS_ERROR Call GsErrorGetEx for information.

Prerequisites

GsInitWithProps() with appropriate LACSLink initialization properties set

GsLACSGetCompleteStats

WIN | UNIX | z/OS

Obtains the complete LACS$^{\text{Link}}$ statistics since the application initialized LACS$^{\text{Link}}$.

Syntax

```
GsFunStat GsLACSGetCompleteStats( GsId gs, GsLACSCompleteStats *pdata, int
structSize );
```

Arguments

gs ID returned by *GsInitWithProps* for the current instance of GeoStan. *Input*.

**pdata* This structure retrieves LACS^{Link} statistics since the application initialized LACS^{Link}. This structure contains the following:

intlu nTotalProcessed Total number of records processed through LACS^{Link}. *Output*.

intlu nTotalAMatches Total number of records matched through LACS^{Link}. *Output*.

intlu nTotal09Matches Total number of records converted through LACS^{Link}, but no new address provided. *Output*.

intlu nTotal00Matches Total number or records not matched through LACS^{Link}. *Output*.

intlu nTotal14Matches Total number of records converted through LACS^{Link}. *Output*.

intlu nTotal92Matches Total number of records matched through LACS^{Link} where GeoStan dropped the unit number. *Output*.

structSize Size of the **GsLACSCompleteStats** data structure. *Input*.

Return Values

GS_SUCCESS

GS_ERROR Call GsErrorGetEx for information.

GS_WARNING Call GsErrorGetEx for information.

Prerequisites

GsInitWithProps() with appropriate LACSLink initialization properties set.

GsLACSGetFalsePosDetail

WIN	UNIX	z/OS

RETRIEVES THE DETAIL RECORD FOR A LACS^{Link} FALSE-POSITIVE REPORT.

Syntax

```
GsFunStat GsLACSGetFalsePosDetail( GsId gs,
GsFalsePosDetailData *pdata, int structSize );
```

Arguments

gs ID returned by *GsInitWithProps* for the current instance of GeoStan. *Input*.

pdata* This structure retrieves the LACS^{Link} detail record for false-positive address match using the data passed in **GsFalsePosDetailData. The data members are details provided by GeoStan for the false-positive report. This structure contains the following:

char AddressSecondaryAbbreviation Unit type (APT, SUITE, LOT). *Output*.

char AddressSecondaryNumber Unit number. *Output*.

char AddressPrimaryNumber House number. *Output*.

char Filler Reserved for future implementation. *Output*.

char MatchedPlus4 ZIP Code extension. *Output*.

char MatchedZIPCODE ZIP Code. *Output*.

char StreetName Name of the street. *Output*.

char StreetpreDirectional Street name predirectional (N, S, E, W). *Output*.

char StreetPostDirectional Street name postdirectional (N, S, E, W). *Output*.

char StreetSuffixAbbreviation Street type (AVE, ST, RD). *Output*.

structSize Size of the **GsFalsePosDetailData** data structure. *Input*.

Return Values

GS_SUCCESS

GS_ERROR Call GsErrorGetEx for more information.

GS_WARNING Call GsErrorGetEx for more information.

Prerequisites

```
GsInitWithProps() with appropriate LACSLink initialization properties set.
GS_LACSLINK_IND == “F”
```

GsLACSGetFalsePosHeaderStats

WIN	UNIX	z/OS

RETRIEVES LACS^{Link} STATISTICS FOR THE HEADER RECORD FOR A LACS^{Link} FALSE-POSITIVE REPORT.

Syntax

```
GsFunStat GsLACSGetFalsePosHeaderStats( GsId gs,
GsFalsePosHeaderData* pdata, int structSize );
```

Arguments

gs ID returned by *GsInitWithProps* for the current instance of GeoStan. *Input.*

pdata* This structure retrieves the LACS^{Link} header record for false-positive address matches using the data passed in **GsFalsePosHeaderData. The output data members are statistics for the false-positive report. The input data members include information to correctly complete the false-positive report for the USPS. This structure contains the following:

intlu nTotalRecordsMatched Number of records matching through LACS^{Link}. *Output.*

intlu nTotalRecordsProcessed Number of records processed through LACS^{Link}. *Output.*

char MailersAddressLine Address of the mailer. *Input.*

char MailersCityName City of the mailer. *Input.*

char MailerCompanyName Name of the mailer. *Input.*

char MailersStateName State of the mailer. *Input.*

char Mailers9DigitZip 9-digit ZIP Code of the mailer. *Input.*

structSize Size of the **GsFalsePosHeaderData** data structure. *Input.*

Return Values

GS_SUCCESS

GS_ERROR Call GsErrorGetEx for more information

GS_WARNING Call GsErrorGetEx for more information.

Prerequisites

```
GsInitWithProps() with appropriate LACSLink initialization properties set.
GS_LACSLINK_IND == “F”
```

GsMultipleGet

WIN	UNIX	z/OS

RETURNS THE ADDRESS ELEMENTS FOR THE MATCH CANDIDATE SPECIFIED.

Syntax

```
GsFunStat GsMultipleGet( GsId gs, GsEnum fSwitch, ints index,
pstr pBuffer, intsu bufLen );
```

Arguments

gs	ID returned by *GsInitWithProps* for the current instance of GeoStan. *Input.*

NOTE: For a list of valid `GsEnums` and their sizes, see “Common enums for storing and retrieving data” on page 68.

fSwitch	Enum for the argument you want to retrieve. *Input*.
index	Entry number (0-based) of the possible match. *Input*.
pBuffer	Location for the returned data. *Output*.
bufLen	Maximum size of data for GeoStan to return. If *bufLen* is shorter than the data returned by GeoStan, GeoStan truncates the data and does not generate an error. *Input*.

Return Values

```
GS_SUCCESS

GS_ERROR
```

Prerequisites

```
GsNumMultiple
```

Notes

This function retrieves data from the GeoStan buffer for match candidates. GeoStan indicates a match candidate as the `GS_ADDRESS_NOT_RESOLVED` return code for **`GsFindWithProps`**. It is important to first test for an intersection match, since the enums are different for retrieving intersection and non-intersection matches.

When using any street name enum (`GS_NAME, GS_PREDIR, GS_POSTDIR, GS_TYPE`), an additional modifier is available. You can use `GS_ALIAS` to request specific alias information, rather than preferred name information. For example in Boulder, CO, Wallstreet is an alias for Fourmile Canyon. 123 Wallstreet, Boulder CO 80301, matches to 123 Fourmile Canyon Dr.

`GsMultipleGet``( ..., GS_NAME,... )` returns "FOURMILE CANYON". **`GsMultipleGet`**`( ..., `*`GS_ALIAS`*` | `*`GS_NAME`*`,... )` returns "WALLSTREET".

If you use `GS_ALIAS` with an enum that does not return alias information (such as `GS_ZIP`), GeoStan returns the information in the normal format. If `GS_IS_ALIAS` returns `A07`, you can only get information based on the returned address, not the alias.

NOTE: GsMultipleGet() index zero output is different from the GsDataGet().

When using GsDataGet(), returned results may be altered for the best match based on all the candidates found. GsMultipleGet() however does not change the results but provides raw data for the returned candidates. Therefore, some enums may differ from GsMultipleGet () and GsDataGet(). For example, GS_ADDRLINE, only contains information available from the candidate data records. If there is no unit low-high on the candidate, no unit information is added to the GS_ADDRLINE.

The following enums are different if the GsDataGet() spatial information is inferred against another candidate in the list:

GS_LAT	GS_LON
GS_LOC_CODE	

These enums differ when CASS logic decides the city/zip/zip+4 on the GsDataGet() lastline versus pure candidate information used for GsMultipleGet():

GS_CITY	GS_CITY_SHORT
GS_LASTLINE	GS_MM_RESULT_CODE
GS_MATCH_CODE	GS_NAME_CITY
GS_URB_NAME	GS_ZIP
GS_ZIP_CARRTSORT	GS_ZIP_CITYDELV
GS_ZIP_CLASS	GS_ZIP_FACILITY
GS_ZIP_UNIQUE	GS_ZIP4
GS_ZIP9	GS_ZIP10

These enums differ when information from the input carries over on the GsDataGet() versus pure candidate information used for GsMultipleGet():

GS_ADDRLINE	GS_FIRM_NAME
GS_HOUSE_NUMBER	GS_UNIT_NUMBER
GS_UNIT_TYPE	

Example

```
/* This example prints information about possible matches. It assumes that
GsFindWithProps returned GS_ADDRESS_NOT_RESOLVED. */

char buffer[60];
int bInter;

printf( "Possibles:\n" );
/* Test for match candidate and set bInter to non-zero if true. */
GsMultipleGet( gs, GS_INTERSECTION, 0, buffer,
sizeof(buffer) );
bInter = *buffer == 'T';
/* Get the number of multiples and print them as we loop through each one. */
int n = GsNumMultiple( gs );
for ( int i = 0; i < n; ++i )
{
printf( "%d: ", i );
   /* If we don't have an intersection, print the ranges. */
if ( !bInter )
{
GsMultipleGet( gs, GS_LORANGE, i, buffer, sizeof(buffer) );
printf( "%s ", buffer );
GsMultipleGet( gs, GS_HIRANGE, i, buffer, sizeof(buffer) );
printf( "%s ", buffer );
}
/* Print the street name for both intersections and non intersections.*/
GsMultipleGet( gs, GS_PREDIR, i, buffer, sizeof(buffer) );
printf( "%s ", buffer );
GsMultipleGet( gs, GS_NAME, i, buffer, sizeof(buffer) );
printf( "%s ", buffer );
GsMultipleGet( gs, GS_TYPE, i, buffer, sizeof(buffer) );
printf( "%s ", buffer );
GsMultipleGet( gs, GS_POSTDIR, i, buffer, sizeof(buffer) );
printf( "%s ", buffer );
/* If an intersection match, print the second street name. */
if ( bInter )
{
printf( " AT " );
GsMultipleGet( gs, GS_PREDIR2, i, buffer, sizeof(buffer) );
printf( "%s ", buffer );
GsMultipleGet( gs, GS_NAME2, i, buffer, sizeof(buffer) );
printf( "%s ", buffer );
GsMultipleGet( gs, GS_TYPE2, i, buffer, sizeof(buffer) );
printf("%s", buffer );
GsMultipleGet( gs, GS_POSTDIR2, i, buffer, sizeof(buffer) );
printf( "%s ", buffer );
}
/* Print each possible on a new line. */
printf( "(%s)\n", buffer );
}
```

GsMultipleGetHandle

WIN	UNIX	z/OS

RETURNS THE RANGE HANDLE FOR THE MATCH CANDIDATE SPECIFIED.

Syntax

```
GsFunStat GsMultipleGetHandle( GsId gs, ints index,
GsRangeHandle *pHandle );
```

Arguments

gs	ID returned by *GsInitWithProps* for the current instance of GeoStan. *Input*.
index	Entry number (0-based) of the possible match. *Input*.
**pHandle*	Range handle for the possible match entry. *Output*.

Return Values

GS_SUCCESS

GS_ERROR

Prerequisites

GsNumMultiple

Notes

This function can extract additional information about a possible match than **GsMultipleGet**. It retrieves the range handle for the possible match indicated by the entry argument. Once you have the correct range handle, you can retrieve information by using **GsHandleGet**. For a list of elements returned by **GsMultipleGet** or **GsHandleGet**, see "Common enums for storing and retrieving data" on page 68.

GsNumMultiple

WIN	UNIX	z/OS

RETURNS THE NUMBER OF MATCH CANDIDATES FOUND.

Syntax

```
GsFunStat GsNumMultiple(GsId gs);
```

Arguments

gs	ID returned by *GsInitWithProps* for the current instance of GeoStan. *Input*.

Return Values

Positive value indicates the number of match candidates. `GS_ERROR` indicates an error.

Prerequisites

`GsFind`

Notes

This function returns the number of match candidates found. Use **`GsMultipleGet`** to retrieve the actual address elements for each possible match. A return code from **`GsFindWithProps`** indicates a possible match.

Example

See "GsMultipleGet" on page 3-198.

GsPrepareIndexFinance

WIN	UNIX

PREPARES GSX FILES NEEDED TO COVER THE FINANCE AREA.

Syntax

```
GsFunStat GsPrepareIndexFinance( GsId gs, long finance,
const char *outDir, void *progressData,
spatialQueryProgress progress );
```

Arguments

`gs` ID returned by *GsInitWithProps* for the current instance of GeoStan. *Input.*

`finance` Finance area to index. In the format `ssffff,` where `s`=state and `f`=finance. *Input.*

`outDir` Directory to write `.gsx` files. *Input.*

`progressData` Data to pass to *`progress`*. *Input.*

`progress` Callback function for progress meter. If `NULL` GeoStan does not call *`progress`*. *Input.*

Return Values

`GS_SUCCESS`

`GS_ERROR`

`GS_CANCEL` Progress function had a cancel request from the user.

`GS_NOT_FOUND` More than 6 digits were entered for the *`finance`* parameter.

Prerequisites

`GsCityDataGet`

Notes

Prepares the single `.gsx` file for the requested finance area (`<finance>.gsx`), unlike **`GsPrepareIndexMbr`** that prepares a `.gsx` file for each finance area included partially or wholly within the area bounded by the four input points. See **`GsPrepareIndexMbr`** for more information. This function does not rebuild `.gsx` files if they already exist and are valid.

Call before **`GsFindFirstSegmentByMbr`** if `.gsx` files do not exist.

To generate all of the .gsx files for the United States at one time, use the batchind.exe utility included with GeoStan. For more information, refer to the *Centrus Utilities Manual.*

For proper execution, initialize GeoStan with a call to GsInitWithProps and include at least the following properties set to TRUE:

GS_INIT_OPTIONS_ADDR_CODE

GS_INIT_OPTIONS_SPATIAL_QUERY

GS_INIT_OPTIONS_Z9_CODE

In addition, when using **`GsInitWithProps`** include the *`outDir`* directory in the search path. For example, `C:\data;C:\gsxout`, where `C:\data` contains the primary data files and `C:\gsxout` is the output directory for `.gsx` files. Include `finmbr.dat` in the **`GsInitWithProps`** search path when creating `.gsx` files.

See **`GsPrepareIndexMbr`** for more information on the progress function.

GsPrepareIndexMbr

WIN	UNIX

PREPARES GSX FILES NEEDED TO COVER THE MBR.

Syntax

```
GsFunStat GsPrepareIndexMbr( GsId gs, long lon1, long lat1, long lon2, long
lat2, const char *outDir, void *progressData, spatialQueryProgress progress );
```

Arguments

`gs`	ID returned by *GsInitWithProps* for the current instance of GeoStan. *Input.*
`lon1`	Minimum longitude. One of four values that make up the MBR. *Input*.
`lat1`	Minimum latitude. One of four values that make up the MBR. *Input*.
`lon2`	Maximum longitude. One of four values that make up the MBR. *Input*.
`lat2`	Maximum latitude. One of four values that make up the MBR. *Input*.
`outDir`	Directory to write `.gsx` files.
`progressData`	Data to pass to *`progress`*. *Input*.
`progress`	Callback function for progress meter. If `NULL` GeoStan does not call *`progress`*. *Input*.

Return Values

GS_SUCCESS	
GS_ERROR	
GS_CANCEL	Progress function received a cancel request from user.

Prerequisites

`GsInitWithProps`

MBR Representation by Lat/Lon

See GsFindFirstSegmentByMbr for an MBR representation by latitude and longitude.

Notes

Prepares a .gsx file for each Finance Area needed to cover the MBR specified by the four lat/lon parameters.

This function does not rebuild `.gsx` files if they already exist and are valid.

Call before **`GsFindFirstSegmentByMbr`**.

NOTE: To generate all of the .gsx files for the United States at one time, use the batchind.exe utility included with GeoStan. For more information, refer to the *Centrus Utilities Manual*.

For proper execution, initialize GeoStan with a call to GsInitWithProps and include at least the following properties set to TRUE:

GS_INIT_OPTIONS_ADDR_CODE

GS_INIT_OPTIONS_SPATIAL_QUERY

GS_INIT_OPTIONS_Z9_CODE

In addition, when using **`GsInitWithProps`** include the *`outDir`* directory in the search path. For example, `C:\data;C:\gsxout`, where `C:\data` contains the primary data files and `C:\gsxout` is the output directory for `.gsx` files. Include `finmbr.dat` in the **`GsInitWithProps`** search path when creating `.gsx` files.

Example

See "GsFindFirstSegmentByMbr" on page 161 for an example.

Progress Function

You can use the progress function as a progress counter and to cancel spatial queries interactively. The progress function is defined in `geostan.h` as follows:

```
intl (*spatialQueryProgress)(void * param, intl mode, intl value);
```

`param` Points at the *`progressData`* argument passed to the **`GsPrepareIndexMbr`** function.

mode One of three values in the following table. Value contents vary depending on mode.

0	Initialize progress meter. *value* is estimated total number of steps.	Return GS_PROGRESS_CONTINUE or GS_PROGRESS_CANCEL.
1	*value* contains current step number.	Return GS_PROGRESS_CONTINUE or GS_PROGRESS_CANCEL.
2	Terminate progress meter. *value* is unused.	Return value ignored

GsPropFind

WIN	UNIX	z/OS

FINDS A PROPERTY IN A PROPERTY LIST.

Syntax

```
GsFunStat GsPropFind (PropList* pProps, PropEnum pPropID);
```

Arguments

*PropList** Pointer to property list structure. *Input.*

PropEnum pPropID The ID of the property to find. *Input.*

Return Values

GS_SUCCESS

GS_ERROR

Prerequisites

GsPropListCreate

Notes

GS_SUCCESS returns if the property is found in the property list. GS_ERROR returns if the property is not found.

Example

```
GsFunStat retval;
PropList findProps;
retval = GsPropListCreate( &findProps, GS_FIND_PROP_LIST_TYPE );
retval = GsPropSetBool( &findProps, GS_FIND_MIXED_CASE, TRUE );
retval = GsPropFind( &findProps, GS_FIND_Z9_CODE);
retval = GsPropFind( &findProps, GS_FIND_MIXED_CASE);
```

GsPropFirst

WIN	UNIX	z/OS

SETS PROPERTY ITERATOR TO FIRST PROPERTY.

Syntax

```
GsFunStat GsPropFirst (Proplist* pProps);
```

Arguments

*PropList**	Pointer to property list structure. *Input.*

Return Values

GS_SUCCESS

GS_ERROR

Prerequisites

GsPropListCreate

Notes

Prior to iterating through the properties in a property list, this function sets the property list iterator to the beginning of the list.

Example

```
GsFunStat retval;
PropList findProps;
PropEnum propID;
PropValue propValue = {GS_UNDEF_PROP_TYPE, 0};
retval = GsPropListCreate( &findProps, GS_FIND_PROP_LIST_TYPE );
retval = GsPropSetBool( &findProps, GS_FIND_MIXED_CASE, TRUE );
retval = GsPropSetLong( &findProps, GS_FIND_MATCH_MODE, GS_MODE_CLOSE );
retval = GsPropSetLong( &findProps, GS_FIND_STREET_OFFSET, 50 );
   GsPropFirst( &findProps );
   while ( GsPropGetNext( &findProps, &propID, &propValue ) == GS_SUCCESS ) {
      switch (propValue.propType ) {
         case GS_BOOL_PROP_TYPE:
            ...
            break;

         case GS_LONG_PROP_TYPE:
            ...
            break;

         case GS_STRING_PROP_TYPE:
            ...
            break;

         case GS_DOUBLE_PROP_TYPE:
         case GS_UNDEF_PROP_TYPE:
         default:
            ...
            break;
      }
   }
```

GsPropGetBool

WIN	UNIX	z/OS

RETRIEVES A BOOLEAN PROPERTY.

Syntax

```
GsFunStat GsPropGetBool (PropList* pProps, PropEnum propID, qbool *pbValue);
```

Arguments

*PropList**	Pointer to property list structure. *Input.*
PropEnum propID	The property ID. *Input.*
**pbValue*	The boolean value of the property. *Output.*

Return Values

GS_SUCCESS

GS_ERROR

Prerequisites

GsPropListCreate

WIN	UNIX	z/OS

GsPropGetDouble

RETRIEVES A DOUBLE PROPERTY.

Syntax

```
GsFunStat GsPropGetDouble (PropList* pProps, PropEnum propID, double
*pdValue);
```

Arguments

*PropList**	Pointer to property list structure. *Input.*
PropEnum propID	The property ID. *Input.*
**pbValue*	The double value of the property. *Output.*

Return Values

GS_SUCCESS

GS_ERROR

Prerequisites

GsPropListCreate

WIN	UNIX	z/OS

GsPropGetInfo

RETRIEVES INFORMATION ABOUT A PROPERTY.

Syntax

```
GsFunStat GsPropGetInfo (PropEnum propID, int* pPropType, PropListType*
pPropListType, pstr pPropName, int* pPropCategory);
```

Arguments

PropEnum propID The property ID. *Input.*

int* pPropType *Output.*

PropListType* pPropListType *Output.*

pstr pPropName *Output.*

int* pPropCategory *Output.*

Return Values

GS_SUCCESS

GS_ERROR

Prerequisites

GsPropListCreate

GsPropGetLong

WIN	UNIX	z/OS

RETRIEVES AN INTEGER PROPERTY.

Syntax

```
GsFunStat GsPropGetLong (Proplist* pProps, PropEnum propID, intl *plValue);
```

Arguments

*PropList** Pointer to property list structure. *Input.*

PropEnum propID The property ID. *Input.*

**plValue* Long value. *Output.*

Return Values

GS_SUCCESS

GS_ERROR

Prerequisites

```
GsPropListCreate
```

Example

```
GsFunStat retval;
PropList findProps;
long lValue;
retval = GsPropListCreate( &findProps, GS_FIND_PROP_LIST_TYPE  );
retval = GsPropSetLong(  &findProps, GS_FIND_MATCH_MODE, GS_MODE_CLOSE );
retval = GsPropGetLong( &statusProps, GS_FIND_MATCH_MODE, &lValue );
```

GsPropGetNext

WIN	UNIX	z/OS

ITERATES TO THE NEXT SEQUENTIAL PROPERTY IN THE PROPERTY LIST.

Syntax

```
GsFunStat GsPropGetNext (Proplist* pProps, PropEnum* propID, PropValue
pPropValue);
```

Arguments

*PropList**	Pointer to property list structure. *Input.*
PropEnum propID*	The property ID. *Output.*
PropValue	Pointer to union of property values. *Output.*

Return Values

```
GS_ERROR

GS_SUCCESS
```

Prerequisites

```
GsPropListCreate
```

Notes

GeoStan only populates one of the PropValue union members. The populated union member depends on the property output type. If there is no “next” entry, GeoStan returns an error.

Example

See GsPropFirst example.

GsPropGetStr

WIN	UNIX	z/OS

RETRIEVES A STRING PROPERTY.

Syntax

`GsFunStat **GsPropGetStr** (*Proplist** pProps, *PropEnum propID*, *pstr pBuffer*, *intsu bufLen*);`

Arguments

*PropList**	Pointer to property list structure. *Input.*
PropEnum propID	The property ID. *Input.*
pstr pBuffer	String buffer. *Output.*
intsu bufLen	Length of a string buffer. *Input.*

Return Values

GS_ERROR

GS_SUCCESS

Prerequisites

GsPropListCreate

Example

```
GsFunStat retval;
PropList findProps;
char buffer[256];
```

```
const int bufLen = 256;
retval = GsPropListCreate( &findProps, GS_FIND_PROP_LIST_TYPE );
retval = GsPropSetStr( &findProps, GS_FIND_CORNER_OFFSET_UNITS, "F" );
retval = GsPropGetStr( &findProps,  GS_FIND_CORNER_OFFSET_UNITS, buffer,
bufLen );
```

GsPropGetStrLen

WIN	UNIX	z/OS

RETRIEVES THE LENGTH OF THE STRING VALUE OF A PROPERTY.

Syntax

```
GsFunStat GsPropGetStrLen (Proplist* pProps, PropEnum propID, intlu* bufLen);
```

Arguments

*PropList**	Pointer to property list structure. *Input.*
PropEnum propID	Property ID. *Input.*
intlu bufLen*	Required string buffer length. *Output.*

Return Values

GS_ERROR

GS_SUCCESS

Prerequisites

GsPropListCreate

GsPropListClear

WIN	UNIX	z/OS

CLEARS (EMPTIES) A PROPERTY LIST.

Syntax

```
GsFunStat GsPropListClear( PropList* pProps );
```

Arguments

*PropList**	Pointer to property list structure. *Input and Output.*

Return Values

GS_ERROR

GS_SUCCESS

Example

```
GsFunStat retval;
PropList findProps;
retval = GsPropListCreate( &findProps, GS_FIND_PROP_LIST_TYPE );
retval = GsPropSetBool( &findProps, GS_FIND_Z_CODE, TRUE );
retval = GsPropSetLong( &findProps,  GS_FIND_STREET_OFFSET, 50 );
retval = GsPropListClear( &findProps );
```

Notes

You may want to clear a property list for re-populating from the beginning with new properties.

GsPropListCreate

WIN | UNIX | z/OS

CREATES AND INITIALIZES A PROPERTY LIST TO CONTAIN GEOSTAN INITIALIZATION, STATUS, OR FIND PROPERTIES.

Syntax

```
GsFunStat GsPropListCreate (Proplist* pProps, PropListType type);
```

Arguments

*PropList**	Pointer to property list structure. *Input and Output.*
PropListType type	Type of property list. *Input.*

Return Values

GS_ERROR

GS_SUCCESS

Notes

This function creates/initializes a property list for use in GsInitWithProps or GsFindWithProps. Any property list created by this function needs to eventually be destroyed with GsPropListDestroy.

Example

```
GsFunStat retval;
PropList initProps;
PropList statusProps;
PropList findProps1;
PropList findProps2;

/* For initialization of GeoStan */
GsPropListCreate( &initProps, GS_INIT_PROP_LIST_TYPE);
GsPropListCreate( &statusProps, GS_STATUS_PROP_LIST_TYPE);

/* For standard property default preference */
retval = GsPropListCreate( &findProps1, GS_FIND_PROP_LIST_TYPE  );

/* For custom property default preference */
retval = GsPropListCreate( &findProps2, GS_FIND_PROP_LIST_TYPE  );
```

GsPropListDestroy

WIN	UNIX	z/OS

DESTROYS A PROPERTY LIST.

Syntax

```
GsFunStat GsPropListDestroy (Proplist* pProps);
```

Arguments

PropList	Pointer to property list structure. *Input and Output.*

Return Values

GS_ERROR

GS_SUCCESS

Prerequisites

GsPropListCreate

Notes

Do not destroy any property list that GeoStan is actively using. Any list created by GsPropListCreate must eventually be destroyed with GsPropListDestroy.

Example

```
GsFunStat retval;
PropList findProps;
retval = GsPropListCreate( &findProps, GS_FIND_PROP_LIST_TYPE  );
retval = GsPropListDestroy ( &findProps );
```

GsPropListGetCount

WIN	UNIX	z/OS

RETRIEVES THE NUMBER OF PROPERTIES IN A PROPERTY LIST.

Syntax

```
GsFunStat GsPropListGetCount (Proplist* pProps, intl* pCount);
```

Arguments

*PropList**	Pointer to property list structure. *Input.*
*intl**	Returned count. *Output.*

Return Values

```
GS_ERROR

GS_SUCCESS
```

Prerequisites

```
GsPropListCreate
```

Example

```
intl count;
GsFunStat retval;
PropList findProps;
retval = GsPropListCreate( &findProps, GS_FIND_PROP_LIST_TYPE  );
retval = GsPropListGetCount( &findProps, &count );
```

GsPropListGetType

WIN	UNIX	z/OS

RETRIEVES THE TYPE OF THE PROPERTY LIST.

Syntax

```
GsFunStat GsPropListGetType (Proplist* pProps, PropListType* pType);
```

Arguments

*PropList** Pointer to property list structure. *Input.*

The following table contains valid enums.

GS_UNDEF_PROP_LIST_TYPE	Undefined
GS_INIT_PROP_LIST_TYPE	Initialization property list
GS_FIND_PROP_LIST_TYPE	Find property list
GS_STATUS_PROP_LIST_TYPE	Status property list

*PropListType** The property list type. *Output.*

Return Values

GS_ERROR

GS_SUCCESS

Prerequisites

GsPropListCreate

Example

```
GsFunStat retval;
PropList findProps;
PropListType type;
retval = GsPropListCreate( &findProps, GS_FIND_PROP_LIST_TYPE );
retval = GsPropListGetType( &findProps, &type );
```

GsPropListRead

WIN	UNIX	z/OS

READS THE PROPERTY LIST FROM A FILE OR CHARACTER STRING.

Syntax

GsFunStat **GsPropListRead** (*Proplist** pProps, *gs_const_str infile*, *gs_const_str pBuffer*);

Arguments

PropList	Pointer to property list structure. *Output.*
gs_const_str infile	File containing the list properties, or NULL. *Input.*
gs_const_str pBuffer	String buffer containing the properties to read if infile is NULL. *Input.*

Return Values

GS_ERROR

GS_SUCCESS

Prerequisites

GsPropListCreate

Notes

If the infile argument value is “stdin”, then this function reads the properties from standard in. If the infile argument is NULL, then this function reads the properties from the pBuffer string.

Example

```
GsFunStat retval;
PropList findProps;
retval = GsPropListCreate( &findProps, GS_FIND_PROP_LIST_TYPE );
retval = GsPropListRead( &findProps, "testprop.txt", NULL );
```

GsPropListReset

WIN	UNIX	z/OS

RESETS THE PROPERTY LIST TO ITS DEFAULT STATE.

Syntax

```
GsFunStat GsPropListReset (PropList* pProps);
```

Arguments

*PropList**	Pointer to property list structure. Input and *Output*.

Return Values

```
GS_ERROR

GS_SUCCESS
```

Prerequisites

GsPropListCreate

Notes

Depending on the type of property list, this might mean "empty" (init and status properties) or refill it with some needed defaults (find properties).

Example

```
GsFunStat retval;
PropList findProps;
retval = GsPropListCreate( &findProps, GS_FIND_PROP_LIST_TYPE );
retval = GsPropSetBool( &findProps, GS_FIND_Z_CODE, TRUE );
retval = GsPropSetLong( &findProps,  GS_FIND_STREET_OFFSET, 50 );
retval = GsPropListReset( &findProps );
```

GsPropListWrite

WIN	UNIX	z/OS

WRITES THE PROPERTY LIST TO A FILE OR A CHARACTER STRING.

Syntax

```
GsFunStat GsPropListWrite (PropList* pProps, gs_cont_str outfile, pstr
pBuffer, intsu bufLen);
```

Arguments

*PropList** Pointer to property list structure. *Input.*

gs_const_str outfile File containing the list properties, or NULL. *Output.*

pstr pBuffer String buffer to write to outfile is NULL. *Output.*

intsu bufLen Length of pBuffer (writing should not exceed bufLen). *Input.*

Return Values

GS_ERROR

GS_SUCCESS

Prerequisites

GsPropListCreate

Notes

If the outfile argument value is "stdout", then this function writes the properties to standard out. If the outfiles argument is NULL, then this function writes the properties to the pBuffer string.

Example

```
GsFunStat retval;
PropList findProps;
retval = GsPropListCreate( &findProps, GS_FIND_PROP_LIST_TYPE );
retval = GsPropSetBool( &findProps, GS_FIND_Z_CODE, TRUE );
retval = GsPropSetLong( &findProps,  GS_FIND_STREET_OFFSET, 50 );
retval = GsPropListWrite( &writeProps, "testprop.txt", NULL, 0 );
```

GsPropRemove

WIN	UNIX	z/OS

REMOVES A PROPERTY FROM THE PROPERTY LIST.

Syntax

GsFunStat **GsPropRemove** (*PropList** pProps, *PropEnum propID*);

Arguments

*PropList**	Pointer to property list structure. *Input and Output.*
PropEnum propID	Property ID. *Input.*

Return Values

GS_ERROR

GS_SUCCESS

Prerequisites

GsPropListCreate

Example

```
GsFunStat retval;
PropList findProps;
retval = GsPropListCreate( &findProps, GS_FIND_PROP_LIST_TYPE );
retval = GsPropSetBool( &findProps, GS_FIND_Z_CODE, TRUE );
retval = GsPropRemove( &findProps, GS_FIND_Z_CODE);
```

GsPropSetAsStr

WIN	UNIX	z/OS

SETS A PROPERTY WHERE INPUT NAME AND VALUE ARE BOTH STRINGS.

NOTE: If you intend to use the same Find property settings for all your address records, try to construct your application to set the Find properties once before processing. It is unnecessary to reset the properties again to the same values for each address record. However, if required, you can change Find property values for individual searches. Performance may be negatively affected by resetting for individual record searches. Reset Find properties between address searches only if changing the Find property value is necessary.

Syntax

```
GsFunStat GsPropSetAsStr (PropList* pProps, gs_const_str pszPropName,
PropEnum propID, gs_const_str pszValue);
```

Arguments

*PropList**	Pointer to property list structure. *Input and Output.*

gs_const_str pszPropName The string name of the property to set. *Input.*

PropEnum propID Property ID. *Input.*

gs_const_str pszValue The string value of the property. *Input.*

Return Values

GS_ERROR

GS_SUCCESS

Prerequisites

GsPropListCreate

Notes

This function performs the conversion to correct property enum ID and property value type. If the input property name is a NULL pointer, the property ID is for use in identifying the property instead. If both property name and property ID are present, preference is given to the property name, ignoring the property ID.

Example

```
GsFunStat retval;
PropList initProps;
retval = GsPropListCreate( &initProps, GS_INIT_PROP_LIST_TYPE);
retval = GsPropSetAsStr( &initProps, "GS_INIT_PASSWORD", GS_UNDEFINED_PROPID,
"12345678" );
retval = GsPropSetAsStr( &initProps, NULL, GS_INIT_PASSWORD, "12345678" );
```

GsPropSetBool

WIN	UNIX	z/OS

SETS A BOOLEAN PROPERTY.

NOTE: If you intend to use the same Find property settings for all your address records, try to construct your application to set the Find properties once before processing. It is unnecessary to reset the properties again to the same values for each address record. However, if required, you can change Find property values for individual searches. Performance may be negatively affected by resetting for individual record searches. Reset Find properties between address searches only if changing the Find property value is necessary.

Syntax

```
GsFunStat GsPropSetBool (PropList* pProps, PropEnum propID, qbool value);
```

Arguments

*PropList**	Pointer to property list structure. *Input and Output.*
PropEnum propID	Property ID. *Input.*
qbool value	Boolean value. *Input.*

Return Values

GS_ERROR

GS_SUCCESS

Prerequisites

GsPropListCreate

Example

```
GsFunStat retval;
PropList findProps;
retval = GsPropListCreate( &findProps, GS_FIND_PROP_LIST_TYPE );
retval = GsPropSetBool( &findProps, GS_FIND_MIXED_CASE, TRUE );
```

GsPropSetDouble

WIN	UNIX	z/OS

SETS A DOUBLE PROPERTY.

Syntax

```
GsFunStat GsPropSetDouble (PropList* pProps, PropEnum propID, double value);
```

Arguments

*PropList**	Pointer to property list structure. *Input and Output.*
PropEnum propID	Property ID. *Input.*

double value	Double value. *Input.*

Return Values

`GS_ERROR`

`GS_SUCCESS`

Prerequisites

`GsPropListCreate`

GsPropSetLong

WIN	UNIX	z/OS

SETS AN INTEGER PROPERTY.

NOTE: If you intend to use the same Find property settings for all your address records, try to construct your application to set the Find properties once before processing. It is unnecessary to reset the properties again to the same values for each address record. However, if required, you can change Find property values for individual searches. Performance may be negatively affected by resetting for individual record searches. Reset Find properties between address searches only if changing the Find property value is necessary.

Syntax

```
GsFunStat GsPropSetLong (PropList* pProps, PropEnum propID, intl value);
```

Arguments

*PropList**	Pointer to property list structure. *Input and Output.*
PropEnum propID	Property ID. *Input.*
intl Value	Long value. *Input and Output.*

Return Values

`GS_ERROR`

`GS_SUCCESS`

Prerequisites

```
GsPropListCreate
```

Example

```
GsFunStat retval;
PropList findProps;
retval = GsPropListCreate( &findProps, GS_FIND_PROP_LIST_TYPE );
retval = GsPropSetLong(  &findProps, GS_FIND_MATCH_MODE, GS_MODE_CASS );
```

GsPropSetStr

WIN	UNIX	z/OS

SETS A STRING PROPERTY.

NOTE: If you intend to use the same Find property settings for all your address records, try to construct your application to set the Find properties once before processing. It is unnecessary to reset the properties again to the same values for each address record. However, if required, you can change Find property values for individual searches. Performance may be negatively affected by resetting for individual record searches. Reset Find properties between address searches only if changing the Find property value is necessary.

Syntax

```
GsFunStat GsPropSetStr (PropList* pProps, PropEnum propID, gs_const_str
value);
```

Arguments

*PropList**	Pointer to property list structure. *Input and Output.*
PropEnum propID	Property ID. *Input.*
gs_const_str value	String value. *Input.*

Return Values

```
GS_ERROR

GS_SUCCESS
```

Prerequisites

```
GsPropListCreate
```

Example

```
GsFunStat retval;
PropList findProps;
retval = GsPropListCreate( &findProps, GS_FIND_PROP_LIST_TYPE );
retval = GsPropSetStr( &findProps, GS_FIND_CORNER_OFFSET_UNITS, "F" );
```

GsSetSelection

WIN	UNIX	z/OS

ALLOWS YOU TO SELECT A MATCH FROM A SET OF MATCH CANDIDATES.

Syntax

```
GsFunStat GsSetSelection( GsId gs, ints index );
```

Arguments

gs	ID returned by *GsInitWithProps* for the current instance of GeoStan. *Input.*
index	Index number (0-based) of the possible match. *Input*.

Return Values

GS_SUCCESS	
GS_ERROR	Selection is out of range.

Prerequisites

GsMultipleGet

Notes

After **GsFindWithProps** returns GS_ADDRESS_NOT_RESOLVED, use **GsNumMulitple** and **GsMultipleGet** to determine which possible match is the correct match. Then use **GsSetSelection** to load that possible match into the data retrieval buffers and retrieve it using **GsDataGet**.

NOTE: GsSetSelection() and GsSetSelectionRange() functions should only be called once per address find (i.e. GsFindWithProps). Also, these functions should only be called if the find resulted in a match candidate (match code = E023).

Example

```
/* This example prints information about possible matches. It assumes that
GsFind returned GS_ADDRESS_NOT_RESOLVED. */

char buffer[60];
int bInter;
printf( "Possibles:\n" );
/* Test for match candidate and set bInter to non-zero if true. */
GsMultipleGet( gs, GS_INTERSECTION, 0, buffer,
sizeof(buffer) );
bInter = *buffer == 'T';
/* Get the number of multiples and print them as we loop through each one. */
int n = GsNumMultiple( gs );
for ( int i = 0; i < n; ++i )
{
printf( "%d: ", i );
/* If we don't have an intersection, print the ranges. */
if ( !bInter )
{
GsMultipleGet( gs, GS_LORANGE, i, buffer, sizeof(buffer) );
printf( "%s ", buffer );
GsMultipleGet( gs, GS_HIRANGE, i, buffer, sizeof(buffer) );
printf( "%s ", buffer );
}
/* Print the street name for both intersections and nonintersections. */
GsMultipleGet( gs, GS_PREDIR, i, buffer, sizeof(buffer) );
printf( "%s ", buffer );
GsMultipleGet( gs, GS_NAME, i, buffer, sizeof(buffer) );
printf( "%s ", buffer );
GsMultipleGet( gs, GS_TYPE, i, buffer, sizeof(buffer) );
printf( "%s ", buffer );
GsMultipleGet( gs, GS_POSTDIR, i, buffer, sizeof(buffer) );
printf( "%s ", buffer );
/* If an intersection match, print the second street name. */
if ( bInter )
{
printf( " AT " );
GsMultipleGet( gs, GS_PREDIR2, i, buffer, sizeof(buffer) );
printf( "%s ", buffer );
GsMultipleGet( gs, GS_NAME2, i, buffer, sizeof(buffer) );
printf( "%s ", buffer );
GsMultipleGet( gs, GS_TYPE2, i, buffer, sizeof(buffer) );
printf("%s", buffer );
GsMultipleGet( gs, GS_POSTDIR2, i, buffer, sizeof(buffer) );
printf( "%s ", buffer );
}
/* Print each possible on a new line. */
printf( "(%s)\n", buffer );
}


int index = 0;
GsFunStat funStat;
printf(" Please enter selection number.");
scanf("%i", &index);

funStat = GsSetSelection(gs, index);
```

```
if (funStat == GS_SUCCESS)
{
GsDataGet( gs, GS_OUTPUT, GS_FIRM_NAME, firm, sizeof(firm) );
GsDataGet( gs, GS_OUTPUT, GS_ADDRLINE, address, sizeof(address) );
GsDataGet( gs, GS_OUTPUT, GS_LASTLINE, lastline, sizeof(lastline) );
GsDataGet( gs, GS_OUTPUT, GS_LON, longitude, sizeof(longitude) );
GsDataGet( gs, GS_OUTPUT, GS_LAT, latitude, sizeof(latitude) );

printf( "%s\n"
"%s\n"
"%s\n"
"longitude: %lf\n"
"latitude: %lf\n", firm, address, lastline, /* ascii data */
(double)(atof(longitude)/1000000.0), /* normalize longitude */
(double)(atof(latitude)/1000000.0) ); /* normalize latitude */
}
else
printf("Address information not found.");
```

GsSetSelectionRange

WIN | UNIX | Z/OS

ALLOWS GEOSTAN TO USE A RECORD FOUND OUTSIDE OF GSFIND AS A MATCH.

Syntax

```
GsFunStat GsSetSelectionRange( GsId gs, GsRangeHandle *pRange,
GsEnum options );
```

Arguments

gs ID returned by *GsInitWithProps* for the current instance of GeoStan. *Input.*

**pRange* Range pointer of the object to return as a match. A range record indicates a range of addresses such as “2800 - 2900 Center Green Drive”. *Input.*

options **GsSetSelectionRange** options. The following table contains the valid enums. *Input.*

GS_ADDR_CODE	Attempts to standardize and find an address geocode. You must set this switch for address standardization. If this switch is not set, GeoStan uses only the input ZIP and ZIP + 4 address elements.
GS_WIDE_SEARCH	GeoStan considers all records matching the first letter of the street name, rather than the Soundex key on the street name. This results in a wider search.

GS_FINANCE_SEARCH	When you add GS_FINANCE_SEARCH, GeoStan searches the entire Finance Area for possible streets. These modifiers have no meaning when doing a ZIP centroid match.
GS_Z9_CODE	Attempts to find ZIP + 4 centroid match only (no ZIP+2 or ZIP).
GS_Z7_CODE	Attempts to find ZIP+2 centroid match only (no ZIP + 4 or ZIP).
GS_Z5_CODE	Attempts to find a ZIP centroid match (no ZIP + 4 or ZIP+2).
GS_Z_CODE	Attempts to find a match based on all possible ZIP centroids available.

If you specify GS_ADDR_CODE and a ZIP centroid option, GeoStan only returns a ZIP Code centroid match if an address geocode is not available.

You must use either GS_ADDR_CODE or GS_Z_CODE or both.(In general, you should specify both.) These option settings are additive.

If you enter a pre-parsed address, it must contain the USPS abbreviations for street type, predirectionals, and postdirectionals.

Return Values

GS_SUCCESS

GS_ERROR

Prerequisites

GsFindFirstRange

Notes

This function allows you to get a range handle that points to the address range to which you want to match. Use this command to indicate that **GsFindWithProps** found the match, when in fact the match is set to whichever range handle is passed through the **pRange* argument. You must specify the *options* enum so that subsequent calls to **GsDataGet** have the type of location information to return.

This function returns standardized results from a list generated by **GsFindFirst**/ **GsFindNext**. Use this function to perform a query that lets the user choose from a list of possible matches.

NOTE: GsSetSelection() and GsSetSelectionRange() functions should only be called once per address find (i.e. GsFindWithProps). Also, these functions should only be called if the find resulted in multiple candidates for a match (match code = E023).

GsSoundex

WIN | UNIX | z/OS

GENERATES A SOUNDEX KEY FOR USE IN GSFINDFIRST.

Syntax

```
intlu GsSoundex( pstr pName );
```

Arguments

pName	String to convert to a Soundex key. *Input*.

Return Values

Soundex key.

Notes

NOTE: A soundex match is a match based on the pronunciation of a field.

This function generates a soundex key for a street name. Use it in conjunction with **GsFindFirstStreet** when you want to perform a soundex search. The soundex key, when combined with a locale (state, and city), is the primary index into the GeoStan databases. Searching by soundex allows you to use your own scoring and matching mechanism for geocoding.

GeoStan uses a modified version of the standard soundex algorithm first published by Donald Knuth. The modifications employed within GeoStan include special treatment of certain prefixes such as *MAC*, *KN*, and *WR*; numeric street names; and an encoding scheme to pack the key into the smallest number of bits.

GsTerm

WIN | UNIX | z/OS

TERMINATES GEOSTAN.

Syntax

```
GsFunStat GsTerm( GsId gs );
```

Arguments

gs	ID returned by *GsInitWithProps* for the current instance of GeoStan. *Input*.

Return Values

`GS_SUCCESS`

`GS_ERROR`

Notes

You must call **`GsTerm`** at the conclusion of a session to close files and release other resources. It closes GeoStan and invalidates **`GsId`**.

You cannot call any function that accepts a GsId parameter except GsInit_r following a call to GsTerm. Also, you should only delete the initialization, find, and status properties (by calling GsPropListDestroy) that you have used with GeoStan after the call to GsTerm.

GsTestRange

WIN	UNIX	z/OS

DETERMINES WHETHER A HOUSE NUMBER FALLS WITHIN A RANGE.

Syntax

```
GsFunStat GsTestRange( GsId gs, gs_const_str number,
gs_const_str rangeLo, gs_const_str rangeHi );
```

Arguments

gs	ID returned by *GsInitWithProps* for the current instance of GeoStan. *Input*.
number	House number to test. *Input*.
rangeLo	Low house number in the range. *Input*.
rangeHi	High house number in the range. *Input*.

Return Values

Non-zero	Number is within the range.
0	Number is not within the range or GeoStan cannot make the comparison.

Prerequisites

`GsInitWithProps`

Notes

GsTestRange tests to see if the house number is within the range defined by *rangeHi* and *rangeLo*. It handles all valid house number patterns, such as:

123	123A	A123	1A23	A1B23
1A23B	1-23	AB		

This function uses USPS rules defined in Publication 28, "Postal Addressing Standards," to determine if a number is in range. For more information, refer to this publication on the USPS Web site.

Some patterns are not comparable with other patterns; for example, 123 is not comparable to a range of `1A01` to `1A99`.

NOTE: GsTestRange does not respect parity.

GsVerifyAuxiliaryRecord

WIN	UNIX	z/OS

VALIDATES AN AUXILIARY FILE RECORD.

Syntax

```
GsFunStat GsVerifyAuxiliaryRecord (GsId gs, const char *pRecord)
```

Arguments

gs	ID returned by *GsInitWithProps* for the current instance of GeoStan. *Input.*
**pRecord*	Pointer to a string buffer containing one auxiliary file record. *Input.*

Return Values

GS_SUCCESS	The record is valid and GeoStan will load the record. This includes comment records; records whose first character is a semicolon.
GS_WARNING	The record contains a non-fatal error and GeoStan will load the record. Because this record has a problem, an input address may not be able to match to the record in its current state, or the output data from a match to the record may not be valid.
GS_ERROR	The record contains a fatal error and GeoStan will NOT load the file.

Prerequisites

`GsInitWithProps`

Example

```
char buffer[800];
char msg1[GS_MAX_STR_LEN], msg2[GS_MAX_STR_LEN];
FILE * fp;
GsId gs;
GsFunStat rc;
int    recno = 0;

/* open the GAX file */
fp = fopen( GAXfilename, "r" );
if ( fp )
{
/* get the first record from the file */
fgets( buffer, sizeof(buffer), fp );
/* while not EOF, process each record */
while( !feof(fp) )
{
recno++;
/* verify the record */
rc = GsVerifyAuxiliaryRecord( gs, buffer );
if ( rc != GS_SUCCESS )
{
/* the record is not perfect, so get more information */
while ( GsErrorHas(gs) )
{
GsErrorGetEx( gs, msg1, msg2 );
printf( “%s %d: %s %s\n”, rc == GS_ERROR ? “FATAL ERROR” : “WARNING”, recno,
msg1, msg2 );
}
printf(“\n”);
}
/* get the next record from the file */
fgets( buffer, sizeof(buffer), fp );
   }
}
```

GsVerifyGSDDataPresent

WIN UNIX z/OS

Validates the presence of data for specified states in the GSD data file.

Syntax

GsFunStat **GsVerifyGSDDataPresent** (GsId *gs*, GsStateList *states*)

Arguments

gs ID returned by *GsInitWithProps* for the current instance of GeoStan. *Input.*

States Structure containing state abbreviations and/or FIPS codes. *Input.*

Return Values

GS_SUCCESS All states or FIPS codes specified in *states* are present in GSD data.

GS_GSD_DATA_MISSING

One or more states or FIPS codes specified in *states* are not present in GSD data.

GS_ERROR *states* structure is invalid.

Prerequisites

You must create a valid GsId with a call to **GsInitWithProps**.

Notes

This function is not available through the Visual Basic interface.

If GeoStan did not find the GSD primary data for all the specified states, this function creates an error in the error list retrievable using **GSErrorGetEx**. The error message code is **114** and the error message is "NO GSD DATA FOUND FOR STATE" and indicates which states GeoStan did not find.

The state abbreviations and corresponding state FIPS codes are as follows:

State	Abbrev.	FIPS Code[a]	State	Abbrev.	FIPS Code[a]
Alabama	AL	1	New Jersey	NJ	34
Alaska	AK	2	New Mexico	NM	35
Arizona	AZ	4	New York	NY	36
Arkansas	AR	5	North Carolina	NC	37
California	CA	6	North Dakota	ND	38
Colorado	CO	8	Ohio	OH	39
Connecticut	CT	9	Oklahoma	OK	40
Delaware	DE	10	Oregon	OR	41

State	Abbrev.	FIPS Code[a]	State	Abbrev.	FIPS Code[a]
District of Columbia	DC	11	Pennsylvania	PA	42
Florida	FL	12	Rhode Island	RI	44
Georgia	GA	13	South Carolina	SC	45
Hawaii	HI	15	South Dakota	SD	46
Idaho	ID	16	Tennessee	TN	47
Illinois	IL	17	Texas	TX	48
Indiana	IN	18	Utah	UT	49
Iowa	IA	19	Vermont	VT	50
Kansas	KS	20	Virginia	VA	51
Kentucky	KY	21	Washington	WA	53
Louisiana	LA	22	West Virginia	WV	54
Maine	ME	23	Wisconsin	WI	55
Maryland	MD	24	Wyoming	WY	56
Massachusetts	MA	25	American Samoa	AS	60
Michigan	MI	26	Guam	GU	66
Minnesota	MN	27	North Mariana Islands	MP	69
Mississippi	MS	28	Palau	PW	70
Missouri	MO	29	Puerto Rico	PR	72
Montana	MT	30	Virgin Islands	VI	78
Nebraska	NE	31			
Nevada	NV	32			
New Hampshire	NH	33			

[a] Do not specify Minor Islands (UM, 74) or you will receive an error. This is defined as a FIPS code, but the USPS does not generate data for this code.

The following pseudo FIPS codes support APO and FPO addresses:

Address	Abbrev.	FIPS Code
Armed Forces Europe	AE	57
Armed Forces Pacific	AP	58
Armed Forces Americas	AA	59

`GsVerifyGSDDataPresent` uses the `GsStateList` structure to pass in a list of state abbreviations or state FIPS codes to verify their presence in the GSD data. This structure contains the following:

NOTE: This structure is also defined in `geostan.h`.

```
typedef struct GsStateList
{
   char *state[100]; /* array of 2 char null terminated strings */
   int cnt;         /* entries in 'state' that are filled in */
} GsStateList;
```

Example

```
typedef struct GsStateList
{
char state[100][3]; /* array of 2 char null terminated strings */
int cnt;             /* entries in 'state' that are filled in */
}
GsStateList;

GsStateList states;
char error_buffer[256];
char details_buffer[256];

/* populate states structure with all states that are required to be in the
GSD data in order to create a CASS report */

/* note that test for CASS required GSD data is not a fatal error so the 'error'
flag is not set */

states.cnt = 0;
strcpy(states.state[states.cnt++], "AL");
strcpy(states.state[states.cnt++], "AK");
strcpy(states.state[states.cnt++], "AZ");
/* additional states can be copied into the array here */

   ....
strcpy(states.state[states.cnt++], "AA");

if (GsVerifyGSDDataPresent(process->gs, states) == GS_SUCCESS)
      process->hasCASSRequiredData = 1;
else
{
process->hasCASSRequiredData = 0;
while ( GsErrorHas(process->gs))
GsErrorGetEx (process->gs, error_buffer, details_buffer);
printf ( "%s: %s\n", error_buffer, details_buffer ); /* report the errors */
}
```

GsWriteExtendCASSReport

WIN | UNIX | Z/OS

WRITES A USPS CASS REPORT BASED ON A SPECIFIED TEMPLATE.

Syntax

```
GsFunStat GsWriteExtendCASSReport ( GsId gs, GsExtendCASSData *pData, int
dataSize, gs_const_str outputName);
```

Arguments

gs ID returned by *GsInitWithProps* for the current instance of GeoStan. *Input*.

pData* The structure allows you to specify the name of the template form. This structure is also defined in `geostan.h`. This function writes a CASS Report using the data passed in **`GsExtendCASSData`. *Input*.

This structure contains the following:

char CASS_Z4CompanyName[40] Name of the company requesting certification.

char CASS_Z4Config[3] CASS Z4Change Configuration.

char CASS_Z4SoftwareName[30] Name of the CASS Z4Change software.

char DPV_FileFormat DPV date file format: F - Full, S - Split, and L - Flat. *Input*.

char certificationDate[24] Date of certification.

intlu DPVDatabaseDate DPV database date.

intlu EWS_Denial Number of matches denied due to a matching address found in `Ews.txt`.

char listName[20] Name of the list.

char listProcessorName[25] Name of the list processor.

intlu LOTDatabaseDate eLOT database date.

char LOT_DPCCompanyName[40] Name of the company requesting eLOT & DPC utility certification.

char LOT_DPCConfig[3] eLOT & DPC Utility Configuration.

char LOT_DPCSoftwareName[30] Name of the eLOT & DPC Software.

char LOTVersion[12] eLOT software version.

char version[12] Software version. *Input.*

intlu nCarrt Number of records that are Carrier Routes coded.

intlu nDpbc Number of records that are delivery point coded.

intlu nHighRiseDefault Number of records matched to a default highrise record.

intlu nHighRiseExact Number of records that matched to an exact highrise record. *Input.*

intlu nLACS Number of old (usually rural-route) addresses converted.

intlu nLot Number of records that are eLOT coded.

intlu nRecs Number of records processed.

intlu nRuralRouteDefault Number of records matched to a default rural route record.

intlu nRuralRouteExact Number of records matched to an exact rural record. *Input.*

intlu nSuiteLink Number of records processed through Suite^{Link}. *Input.*

intlu nTotalDPV Deprecated.

intlu nZIP4 Number of records that are ZIP + 4 Coded.

intlu nZIP Number of records that are 5-digit ZIP Coded.

intlu nZ4Changed Number of records skipped (not reprocessed) because the Z4 did not change.

char pSearchPath[256] Location of the GeoStan data.

intlu structVersion Set equal to `GS_GEOSTAN_VERSION`.

char templateName[256] Full path and name of the `cass3553.frm` template.

char version[12] CASS software version.

ntlu zip4DatabaseDate ZIP + 4 database date.

char z4ChangeVersion[12] Z4Change software version.

dataSize Size of the CASS report data structure. *Input* *outputName* Path and name of the report file to write. *Input*.

Return Values

GS_SUCCESS

GS_ERROR

Prerequisites

GsInitWithProps

Notes

Before running a CASS report, verify GeoStan has loaded the GSU, USPS.gdi, and GSZ files (ZIPMove data).

NOTE: See "CASS certification" on page 451 for an example of a CASS 3553 form and more information on obtaining CASS certification.

When you specify a version number for either version, *Z4ChangeVersion* or *LOTVersion*, GeoStan updates the corresponding fields in the template cass3553.frm file. For example, entering a version number for *Z4ChangeVersion* prompts GeoStan to fill these fields on cass3553.frm:

- B. List, 2b. Date List Processed Z4Change
- B. List, 3b. Date of Database Product Used Z4Change
- C. Output, 1b. Total Coded Z4Change Processed

To develop CASS-certified applications in GeoStan, you must have the correct license agreement with Pitney Bowes Business Insight. You must also obtain CASS certification from the USPS for every application. Using GeoStan does not make an application CASS certified.

To receive a return value of GS_SUCESS, the only members of the GsExtendCASSData struct that you MUST set are structVersion and templateName.

Example

```
/* get the file status and file Date of required files */
fileRet = GsFileStatus(process->gs, fileStateCode, &fileDate);

/* if the file status is GS_GSL_EXISTS fill in the data structure */
if( fileRet & GS_GSL_EXISTS )
{
/* fill CASSData structure */
CASSData.structVersion = GsGetLibVersion();

CASSData.nRecs = report.records;
strcpy( CASSData.listName, outListName );
strcpy( CASSData.pSearchPath, process->searchPath );
strcpy( CASSData.LOTVersion, CASSData.version );
strcpy( CASSData.CASS_Z4CompanyName, "Group 1" );
strcpy( CASSData.LOT_DPCCompanyName, "Group 1" );
strcpy( CASSData.CASS_Z4Config, "CAS" );
strcpy( CASSData.LOT_DPCConfig, "CAS" );
strcpy( CASSData.CASS_Z4SoftwareName, programName );

/* if doing z4Change - assign z4ChangeVersion */
if (process->lastZ4ChangeProcessDate[0])
{
strcpy( CASSData.z4ChangeVersion, CASSData.version );
}

strcpy( CASSData.LOT_DPCSoftwareName, programName );
strcpy( CASSData.listProcessorName, "" );

CASSData.zip4DatabaseDate = fileDate;
CASSData.LOTDatabaseDate = fileDate;
}
.
.
.
/* calculate nRecs, nZip, nZip4... and set these in the CASSData structure */
Geocode( )
.
.
.
/* write the CASS report from data in CASSData structure */
retval = GsWriteExtendCASSReport( process->gs, &CASSData, sizeof( CASSData ),
process->CASSFile );

if( retval != GS_SUCCESS )
{
printf( " Error Writing CASS Report! \n" );
}
```

GsZ4Changed

WIN	UNIX	Z/OS

DETERMINES IF THE USPS CHANGED A ZIP + 4 ADDRESS DEFINITION SINCE GEOSTAN LAST PROCESSED THE RECORD.

Syntax

`GsFunStat **GsZ4Changed** ( GsId *gs*, pstr *pZip,* pstr *pZip4,* pstr *pDate*);`

Arguments

gs ID returned by *GsInitWithProps* for the current instance of GeoStan. *Input.*

pZip First five digits of the 9-digit ZIP Code to test. *Input.*

pZip4 Last four digits of the 9-digit ZIP Code to test. *Input.*

pDate Date of the GeoStan ZIP + 4 information file used to process the record, in the format `MMYYYY` (for example, `081998`). *Input.*

Return Values

`GS_ERROR` Usually occurs when GeoStan could not load the GSL file, because:

- GeoStan could not find the GSL file in the path
- The GSL file was a different release level than the GSD file.

`GS_Z4_CHANGE`

`GS_Z4_NO_CHANGE`

Prerequisites

`GsInitWithProps`

Notes

For `ZIP4Change` to work, the GeoStan GSL file must be in a directory listed in the GS_INIT_DATAPATH initialization property of the property list that you pass when calling GsInitWithProps. The Z4Change file, which is generated by the USPS, contains a record for every ZIP + 4 in the country. Each record contains twelve flags that represent the last twelve months, starting with the current release date. Each of these flags has a value of either True or False, indicating if the ZIP + 4 changed for that monthly postal release. The GeoStan GSL file incorporates this information, which **`GsZ4Changed`** references.

Z4Change information is valuable to users who process very large address lists frequently. As you process each record, you can call **`GsZ4Changed`** and quickly tell if the record needs reprocessing. This information can help you quickly identify only those records that need reprocessing.

Your application must store the date of the GeoStan GSL file used to process a record. GeoStan uses this date as the **`GsZ4Changed`** *`pDate`* input parameter. This is the same date printed on the GeoStan CD used for record processing, and is one month later that the release date of the USPS files used on the GeoStan CD.

Use the following code example if you are unsure of the release date. You should run this code when standardizing a batch of records and store the resulting date string as input to **`GsZ4Changed`** in future processing runs.

Example

```
intsu postalDate;
int month, year;
char z4ChangeDate[7];
```

```
/* Get the postal release of the current files. */
GsFileStatus ( gs,"", &postalDate );
month = ((postalDate % 384) / 32) + 1;
year = (postalDate/384) + 1990;
/* Add one month to this date so it matches what is printed on the CD. */
if ( month !=12 )
{
++month;
}
else
{
++year;
month = 1;
}
/* Print the month and year to the date string in the format MMYYYY. */
sprintf( z4changedate, "%02d%d", month, year );
```

CHAPTER 4

Defining COBOL procedures

This chapter lists the public functions in alphabetical order. The following sections provide a quick reference to the data types, copybooks, and procedures; organized by usage.

A COBOL copy member named GEOSTAN has been included with your product which provides constants, status codes, and record layouts for use with GeoStan. You can find it in the COBOL library shipped with the product.

Variables

Common variables for storing and retrieving data

Provides information on the variables used for GSHGET, GSDATSET, GSDATGET, and GSMGET.

Initialization & termination procedures

GSFSTAT

Returns if the file successfully opened, and the date of the USPS data used in generating the primary GSD file.

GSFSTATX

Returns a variety of information about the current GeoStan data set and license.

GSGLIBV

Returns the current version of the GeoStan library.

GSINITWP

Initializes GeoStan.

GSSCACHE

Sets the size of the cache GeoStan uses.

GSSLIC

Identifies license file and key.

GSSRELD

Allows you to specify the oldest acceptable date for GeoStan data files.

GSTERM

Terminates GeoStan.

Last-line lookup

GSCITYDG

Retrieves data for the found city record.

GSCITYFF

Finds the first city matching partial name or valid ZIP Code.

GSCITYFN

Finds the next matching city.

GSFFSTAT

Finds the first state matching name, abbreviation, or valid ZIP Code.

Address lookup

GSDATGET

Returns data for all address and matched elements from GeoStan.

GSDATSET

Passes data for all address elements to GeoStan.

GSFFRANG

Finds the first range object that meets the search criteria.

GSFFSEG

Finds the first segment that meets the search criteria.

GSFFST

Finds the first street object that meets the search criteria.

GSFNRANG

Finds the next range object that meets the search criteria.

GSFNSEG

Finds the next segment that meets the search criteria.

GSFNST

Finds the next street that meets the search criteria.

GSFINDWP

Finds a match with user settable match preferences (properties).

GSGCRDX

Retrieves coordinates for the street segment found via GSFINDWP.

GSMGET

Returns the address elements for the match candidate item specified.

GSMGH

Returns the range handle for the match candidate item specified.

GSNMULT

Returns the number of match candidatees found.

GSSETSEL

Allows you to select a match from a set or match candidatees.

Data querying

GSDBMETA

Retrieves coordinates for the street segment found via GSFINDWP.

GSGETNDB

Returns the number of loaded databases.

GSHGCRDX

Retrieves the coordinates for a street segment object found via GSFFSEG and GSFNRANG.

GSHGET
Retrieves data for objects found via GSFFSEG and GSFNRANG.

Error and message handling

GSERRGET
Retrieves current error information.
GSERRGTX
Retrieves informational messages, error messages, and fatal warnings for the current GeoStan thread.
GSERRHAS
Indicates if any errors occurred or any informational messages are available in the current thread.

Data passing procedures

GSSSELR
Allows GeoStan to use a record found outside of GSSFINDWP as a match.
GSFGF
Finds the first geographic information record with partial matching to input names.
GSFGFX
Finds the first city, county, and or state centroid match from the set of possible matches found.
GSFGN
Finds the next geographic information record with partial matching to input names.
GSFGNX
Finds the next city, county, and or state centroid match from the set of possible matches found.

Utilities

GSCLEAR
Clears the data buffer in the internal data structures.
GSSNDX
Generates a soundex key for use in GSFFSEG.
GSTSTRNG
Tests if a house number falls within a range.
GSVAUXR
Validates an auxiliary file record.
GSVGSDDT
Validates the presence of data for specified states in the GSD file.
GSZ4CH
Determines if the USPS changed a ZIP + 4 address definition since GeoStan last processed the record.

CASS report procedures

GSFCASSH
Writes a CASS 3553 report to the header buffer argument.
GSWECASS
Writes a USPS CASS report based on the specified template.

DPV report procedures

GSDPVGCS
Obtains the complete DPV statistics since the application initialized DPV.
GSDPVGFD
Retrieves the detail record for a DPV false positive report.
GSDPVGFH
Retrieves DPV statistics for the header record for a DPV false positive report.
GSFDFPD
Formats a DPV false positive detail record from `GSDPVGFD.`
GSFDFPH
Formats a DPV false positive header record with data from `GSDPVGFD`.

LACS^{Link} report procedures

GSLACCLS
Clears the LACS^{Link} statistics.
GSLACFPD
Formats a LACS^{Link} false positive detail record from GSLACGFH.
GSLACFPH
Formats a LACS^{Link} false positive header record with data from GSLACGFH.
GSLACGCS
Obtains the complete LACS^{Link} statistics since the application initialized LACS^{Link}.
GSLACGFD
Retrieves the detail record for a LACS^{Link} false positive report.
GSLACGFH
Retrieves LACS^{Link} statistics for the header record for a LACS^{Link} false positive report.

Property setting procedures

GSPSETAS
Sets a property where input name and value are both strings.
GSPSETB
Sets a boolean property.
GSPSETD
Sets a double property.

GSPSETL

Sets an integer property.

GSPSETS

Sets a short integer property.

GSPSETST

Sets a string property.

Property retrieving procedures

GSPGETB

Retrieves a boolean property.

GSPGETD

Retrieves a double property.

GSPGETL

Retrieves a long integer property.

GSPGETST

Retrieves a string property.

GSPGINFO

Retrieves information about a property.

GSPGSTRL

Retrieves the length of the string value of a property.

Property list management

GSPFIND

Finds a property in a property list.

GSPFIRST

Sets property iterator to first property.

GSPFNEXT

Iterates to the next sequential property in the property list.

GSPLSTCR

Creates and initializes a property list for GeoStan initialization.

GSPLSTRD

Destroys a property list.

GSPLSTGC

Retrieves the number of properties in a property list.

GSPLSTWR

Writes the property list to a file or a character string.

GSPREMOV

Removes a property from the property list.

GSPRESET

Resets the property list to its default state.

Common variables for storing and retrieving data

This chapter contains the valid variables used with **GSHGET**, **GSDATSET**, **GSDATGET**, and **GSMGET**.

The following describes the contents of each table column.:

Variable	Lists the defined constant used to access this element.
GSHGET	• Street A bullet indicates this variable is valid when using a Street handle with **GSHGET**. Includes only Street level information. • Segment A bullet indicates this is valid when using a Segment handle with **GSHGET**. Includes Street and Segment information. • Range A bullet indicates this variable is valid when using a Range handle with **GSHGET**. Includes Street, Segment, and Range information.
GSDATSET	A bullet indicates this variable is valid when using **GSDATSET.**
GSDATGET	• Input A bullet indicates this is valid when using input flag with this function. • Output Indicates this variable is valid when using the output flag with this function. Can be: – *A* – Valid for non-intersection matches only – *I* – Valid for intersection matches only – *L* - Valid for street locator matches only – *P* - Centrus point data match. – R - Reverse geocoding only. – *Bullet* – Valid for all types of match
GSMGET	Indicates the valid match types. • *A* – Valid for non-intersection matches. • *I* – Valid for intersection matches only. • *L* - Valid for street locator and relaxed address matches only. • *P* - Centrus point data match. • *Bullet* – Valid for all types of matches.
Length	A number indicating the largest string GeoStan returns for this variable. This length includes a NULL-termination character. Symbolic constants use the naming convention `<<NAME>>-LENGTH`. For example the variable `GS-ADDRLINE` returns the input or output address line. The maximum length of an address line is 61. Therefore, the symbolic constant is `GS-ADDRLINE-LENGTH` and is defined as 61.
Description	A brief explanation of the information returned. If you use an invalid variable, GeoStan returns an Invalid Argument error.

Variable Name	Buffer Length[a]	GSHGET			GSDATSET	GSDATGET			GSMGET
		Street	Segment	Range		Input	Output	Center line	
GS-ADDRLINE First address line or address line for single line addresses. For single line addresses, must include lastline information and can also include address information.	61				●	●	●		●
GS-ADDRLINE-SHORT Shortest possible address line that can be constructed from available short street name and the other address line components.	61						●		
GS-ADDR2 Second address line.	61				●	●	●		
GS-ALIAS Use this to specify alias info instead of normal.		●	●	●					●
GS-ALT-FLAG Base/Alternate record • *B* – Base • A – Alternate	2			●			●		A
GS-APN-ID Assessor's Parcel Number (APN)	46				●		A,P		A,P
GS-AUX-USERDATA User data from the auxiliary file. Blank if no auxiliary file.	301						●		●

Variable Name	Buffer Length[a]	GSHGET			GSDATSET	GSDATGET			GSMGET
		Street	Segment	Range		Input	Output	Center line	
`GS-BEARING` Compass direction, in decimal degrees, from the point data match to the centerline match. Measured clockwise from 0 degrees north.	6							●	
`GS-BLOCK` 15-digit census block ID/census FIPS code, using the syntax `ssccctttttt gbbb` where: • `ss` – 2-digit State FIPS Code • `ccc` – 3-digit County FIPS Code • `tttttt` – 6-digit Census Tract FIPS Code • `g` – Single-digit Block Group FIPS Code, and • `bbb` – Block FIPS Code If the address is on the left side of the street, **`GS-BLOCK`** will contain the equivalent of **`GS-BLOCK-LEFT`**. If the address is on the right, **`GS-BLOCK`** will contain the equivalent of **`GS-BLOCK-RIGHT`**. NOTE: GeoStan does not return a period for the Centrus Tract FIPS Code, this may deviate from the industry standard.	16						A		A
`GS-BLOCK-LEFT` Census block ID from the left side of the street.	16		●	●			A	●	●
`GS-BLOCK-LEFT2` `GS-BLOCK-LEFT` for the second segment in the intersection.	16								I
`GS-BLOCK-RIGHT` Census block ID from the right side of the street.	16		●	●			A	●	●

Variable Name	Buffer Length[a]	GSHGET			GSDATSET	GSDATGET			GSMGET
		Street	Segment	Range		Input	Output	Center line	
`GS-BLOCK-RIGHT2` `GS-BLOCK-RIGHT` for the second segment in the intersection.	16								I
`GS-BLOCK-SFX` Current block suffix for Census 2000 geography.	2		●	●			●		A
`GS-BLOCK-SFX-LEFT` Current left block suffix for Census 2000 geography.	2		●	●			●	●	●
`GS-BLOCK-SFX-LEFT2` `GS-BLOCK-SFX-LEFT` for the second segment in the intersection.	2								I
`GS-BLOCK-SFX-RIGHT` Current right block suffix for Census 2000 geography. *Blank* if the matched record is from point-level data.	2		●	●			●	●	●
`GS-BLOCK-SFX-RIGHT2` `GS-BLOCK-SFX-RIGHT` for the second segment in the intersection.	2								I
`GS-CART` Carrier route.	5			●			A		A
`GS-CBSA-DIVISION-NAME` CBSA division name.	128						●		●
`GS-CBSA-DIVISION-NAME2` `GS-CBSA-DIVISION-NAME` for the second segment in the intersection.	128								I

Variable Name	Buffer Length[a]	GSHGET			GSDATSET	GSDATGET			GSMGET
		Street	Segment	Range		Input	Output	Center line	
`GS-CBSA-DIVISION-NUMBER` CBSA division number.	6						●		●
`GS-CBSA-DIVISION-NUMBER2` `GS-CBSA-DIVISION-NUMBER` for the second segment in the intersection.	6								I
`GS-CBSA-NAME` CBSA name.	128						●		●
`GS-CBSA-NAME2` `GS-CBSA-NAME` for the second segment in the intersection.	128								I
`GS-CBSA-NUMBER` CBSA number.	6						●		●
`GS-CBSA-NUMBER2` `GS-CBSA-NUMBER` for the second segment in the intersection.	6								I
`GS-CHECKDIGIT`[b] Check digit.	2						●		
`GS-CITY` Abbreviated city name from the last line of the input or output address; the value from `GS-NAME-CITY` or `GS-PREF-CITY`.	29			●	●	●	●		●

Variable Name	Buffer Length[a]		GSHGET		GSDATSET		GSDATGET		GSMGET
		Street	Segment	Range		Input	Output	Center line	
GS-CITY-SHORT The output city name that appears in GS_LASTLINE_SHORT. This value is determined by logic similar to GS_CITY. Whenever possible, this city name is 13 characters or less. This output city name is determined by CASS rules. This can be either City State Name, City State Name Abbreviation, or Preferred Last Line City State Name.	29						●		●
GS-CLOSE-MATCH Indicates if the address was a Close match. • T - Match candidate is a Close match to the input address. • F - Match candidate is not a Close match to the input address.							●		●
GS-CMSA CMSA name.	81						●		●
GS-CMSA-NUMBER CMSA number.	5						●		●
GS-COUNTY County FIPS code. (For **GSHGET** and **GSMGET** this is the USPS county for the Range record. For **GSDATAGET** this is the county from the segment record or the USPS county if matched to a USPS only range.)	6			●			●		●
GS-COUNTY2 GS-COUNTY for the second segment in the intersection.	6								|

Variable Name	Buffer Length[a]	GSHGET			GSDATSET	GSDATGET			GSMGET
		Street	Segment	Range		Input	Output	Center line	
GS-COUNTY-FIPS County FIPS code	4					●			
GS-COUNTY-NAME County name.	128			●			●		●
GS-COUNTY-NAME2 GS-COUNTY-NAME for the second segment in the intersection.	128								I
GS-CSA-NAME CSA name.	128						●		●
GS-CSA-NAME2 GS-CSA-NAME for the second segment in the intersection.	128								I
GS-CSA-NUMBER CSA number.	4						●		●
GS-CSA-NUMBER2 GS-CSA-NUMBER for the second segment in the intersection.	4								I
GS-CTYST-KEY USPS city state key (an alphanumeric value that uniquely identifies a locale in the USPS city state product).	7			●			●		●

Variable Name	Buffer Length[a]	GSHGET			GSDATSET	GSDATGET			GSMGET
		Street	Segment	Range		Input	Output	Center line	
`GS-DATATYPE` Type of data used to make the match. • *0* – USPS data • *1* – TIGER data • *2* – Tele Atlas data • *3* – Sanborn point-level data • *4* – Deprecated • *6* – NAVTEQ data • *7* – Tele Atlas point-level data • *8* – Centrus point data • *9* – Auxiliary file data • 10 – User Dictionary	3		●				●	●	●
`GS-DATATYPE2` `GS-DATATYPE` for the second segment in the intersection.	3								I
`GS-DFLT`[b] Indicates the return status of **`GS-HI-RISE-DFLT`** and **`GS-R-RTE-DFLT`** • *Y* – Either **`GS-HI-RISE-DFLT`** or **`GS-R-RTE-DFLT`** returned **Y**. • *Blank* – Both **`GS-HI-RISE-DFLT`** and **`GS-R-RTE-DFLT`** returned **N** or *Blank*.	2						●		A
`GS-DPBC`[b] Delivery Point Bar Code.	3						●		

Variable Name	Buffer Length[a]	GSHGET			GSDATSET	GSDATGET			GSMGET
		Street	Segment	Range		Input	Output	Center line	
`GS-DPV-CONFIRM` Indicates if a match occurred for DPV data. • *N* – Nothing confirmed • *Y* – Everything confirmed (ZIP + 4, primary, and secondary) • *S* – ZIP + 4 and primary (house number) confirmed • *D* – ZIP + 4 and primary (house number) confirmed and a default match (`GS-HI-RISE-DFLT = Y`), secondary did not confirm. • *Blank* – non-matched input address to USPS ZIP + 4 data, or DPV data not loaded	2						●		
`GS-DPV-CMRA` DPV CMRA indicator. • *Y* – Address found in CMRA table • *N* – Address not found in CMRA table • *Blank* – DPV not loaded	2						●		
`GS-DPV-FALSE-POS` DPV false-positive indicator. • *Y* – False-positive match found • *Blank* – False-positive match not found	2						●		
`GS-DPV-FOOTNOTE1` Information about matched DPV records. • *AA* – ZIP + 4 matched • *A1* – Failure to match a ZIP + 4 • *Blank* – Address not presented to hash table or DPV data not loaded	3						●		

Variable Name	Buffer Length[a]	GSHGET			GSDATSET	GSDATGET			GSMGET
		Street	Segment	Range		Input	Output	Center line	
GS-DPV-FOOTNOTE2 Information about matched DPV records. • *BB* – All DPV categories matched • *CC* – DPV matched primary/house number, where the secondary/unit number did not match (present but invalid) • *M1* – Missing primary/house number • *M3* – Invalid primary/house number • *N1* – DPV matched primary/house number, with a missing secondary number • *P1* – Missing PS, RR, or HC Box number • *P3* – Invalid PS, RR or HC Box number • *F1* – All military addresses • *G1* – All general delivery addresses • *U1* – All unique ZIP Code addresses • *Blank* – Address not presented to hash table or DPV data not loaded NOTE: A unique ZIP Code is a ZIP Code assigned to a company, agency, or entity with sufficient mail volume to receive its own ZIP Code.	3						●		

Variable Name	Buffer Length[a]	GSHGET			GSDATSET	GSDATGET			GSMGET
		Street	Segment	Range		Input	Output	Center line	
GS-DPV-FOOTNOTE3 Information about matched DPV records. • *R1* – Matched to CMRA but PMB designator not present. • *RR* – Matched to CMRA and PMB designator present (PMB 123 or # 123). • *Blank* – Address not presented to hash table or DPV data not loaded	3						●		
GS-DPV-FOOTNOTE4 Reserved by USPS for future use.	3						●		
GS-DPV-FOOTNOTE5 Reserved by USPS for future use.	3						●		
GS-DPV-FOOTNOTE6 Reserved by USPS for future use.	3						●		
GS-DPV-NO-STAT • Y - The address is valid for CDS pre-processing. • N - The address is not valid for CDS pre-processing. • Blank - DPV is not loaded or DPV did not confirm.	2								
GS-DPV-SHUTDOWN • Y - Address was found in false-positive table. • N - Address was not found in false-positive table. • Blank - Address was not presented to hash table or DPV data not loaded.	2						●		

Variable Name	Buffer Length[a]	GSHGET			GSDATSET	GSDATGET			GSMGET
		Street	Segment	Range		Input	Output	Center line	
GS-DPV-VACANT • Y - The address is vacant. • N - The address is not vacant. • Blank - DPV is not loaded or DPV did not confirm (so vacancy is irrelevant).	3						●		
GS-EWS-MATCH Indicates if GeoStan made an EWS match. • *Y* – Match denied because it matched to EWS data. • *Blank* – Input record did not match to EWS data.	2						●		
GS-FIRM-NAME Name of firm from the USPS data or the input firm name.	41			●	●	●	A		A
GS-GOVT-FLAG Government building indicator. • *A* – City government building • *B* – Federal government building • *C* – State government building • *D* – Firm-only • *E* – City government building and firm only • *F* – Federal government building and firm only • *G* – State government building and firm only A, B, C, E, F, and G are valid for alternate records only (GS-ALT-FLAG=A). D is valid for both base and alternate records.	2			●			●		A

Variable Name	Buffer Length[a]	GSHGET			GSDATSET	GSDATGET			GSMGET
		Street	Segment	Range		Input	Output	Center line	
`GS-HI-RISE-DFLT` Indicates if GeoStan matched to a highrise. • *N* – Matched to an exact highrise record or a street record • *Y* – Did not match to an exact record. Matched to the USPS default highrise record or a street record. Check the input address for accuracy and completeness. • *Blank* – Does not apply to the input address (for example, PO Boxes and General Delivery addresses) or did not find a match.	2						●		A
`GS-HIRANGE` House number at high end of range.	12			●			A		A
`GS-HIUNIT` High unit number.	12			●			A		A
`GS-HIZIP4` High ZIP + 4 for this range.	5			●					A
`GS-HOUSE-NUMBER` House number of input or output address.	12					●	A		●
`GS-HOUSE-NUMBER2` Second house number for a ranged house number match. If GS_FIND_ADDRESS_RANGE is enabled, the second number in the ranged address.	12					●	A		●

Variable Name	Buffer Length[a]	GSHGET			GSDATSET	GSDATGET			GSMGET
		Street	Segment	Range		Input	Output	Center line	
GS-INTERSECTION Indicates if the address is an intersection. • *T* – Input address is an intersection • *F* – Input address is a street address	2					●	●		●
GS-IS-ALIAS The first character is: • *N* – Normal street match • *A* – Alias match (including buildings, aliases, firms, etc.) The next 2 characters are: • *01* – Basic index, normal address match • *02* – USPS street name alias index • *03* – USPS building index • *05* – Statewide intersection alias (when using the Usw.gsi, Use.gsi, or US.gsi file). • *06* – Spatial data street name alias (when using ent, the Us_pw.gsi, Usw.gsi, Us_pe.gsi, Use.gsi, Us_ps.gsi, Usp.gsi, Us_psw.gsi, or Us_pse.gsi file is required.) • *07* – Alternate index (when using Zip9.gsu, Zip9e.gsu, and Zip9w.gsu) • *08* – LACS^{Link} • *G09* – Auxiliary file match. NOTE: The GIS file is not required if you use **GSHGET**.	4	●	●	●			A	●	●

Variable Name	Buffer Length[a]	GSHGET			GSDATSET	GSDATGET			GSMGET
		Street	Segment	Range		Input	Output	Center line	
`GS-LACS-FLAG` Indicates if GeoStan marked the address for conversion. • *L* – Address marked for LACS conversion. • *Blank* – Address not marked for LACS conversion.	2			●			●		A
`GS-LACSLINK-IND` LACS$^{\text{Link}}$ indicator. • *Y* – Matched LACS$^{\text{Link}}$ record • *N* – LACS$^{\text{Link}}$ match NOT found • *F* – False-positive LACS$^{\text{Link}}$ record • *S* – Secondary information (unit number) removed to make a LACS$^{\text{Link}}$ match • *Blank* – Not processed through LACS$^{\text{Link}}$	2						●		
`GS-LACSLINK-SHUTDOWN` • *Y* – False-positive occurred and LACSLink library shutdown. • *N* – LACSLink library has not shutdown or not loaded.	2						●		

Variable Name	Buffer Length[a]	GSHGET			GSDATSET	GSDATGET			GSMGET
		Street	Segment	Range		Input	Output	Center line	
`GS-LACSLINK-RETCODE` LACS^{Link} return code. • *A* – Matched LACS^{Link} record • *00* – LACS^{Link} match NOT found • *09* – Matched to highrise default, but no LACS^{Link} conversion • *14* – Found LACS^{Link} match, but no LACS^{Link} conversion • *92* – Secondary information (unit number) removed to make a LACS^{Link} match • *Blank* – Not processed through LACS^{Link}	3						●		
`GS-LASTLINE`[c] Complete last line (ZIP + 4 not available with **`GSMGET`** or intersection matches).	61			●	●	●	●		●
`GS-LASTLINE-SHORT` This is the address "lastline". It contains the values of GS_CITY_SHORT, GS_STATE, and GS_ZIP10. Whenever possible, this field is 29 characters or less: • 13-character city name • 2 (comma and space) • 2-character state abbreviation • 2 spaces • 10-digit ZIP code	61						●		
`GS-LAT` Latitude of located point (in millionths of degrees).	11				●		●	●	●

Variable Name	Buffer Length[a]		GSHGET		GSDATSET		GSDATGET		GSMGET
		Street	Segment	Range		Input	Output	Center line	
GS-LINE1 GS-LINE2 GS-LINE3 GS-LINE4 GS-LINE5 GS-LINE6 Used for multiple line entry. GeoStan does not process these as output. PMB-DESIGNATOR and PMB-NUMBER are not returned in this mode.	104					●			
GS-LOC-CODE Location code. For descriptions of location codes, see the *Status Codes* appendix.	5						●		●
GS-LON Longitude of located point (in millionths of degrees).	12				●		●	●	●
GS-LORANGE House number at low end of range.	12			●			A		A
GS-LOT-CODE[b] • *A* – Ascending • *D* – Descending. eLot ascending and descending value. Only available for addresses GeoStan can standardize.	2						●		
GS-LOT-NUM[b] 4-digit eLOT number. Requires an input address GeoStan can standardize.	5						●		
GS-LOUNIT Low unit number.	12			●			A		A

Variable Name	Buffer Length[a]	GSHGET			GSDATSET	GSDATGET			GSMGET
		Street	Segment	Range		Input	Output	Center line	
`GS-LOZIP4` Low ZIP + 4 for this range.	5			●					A
`GS-MAIL-STOP` Returns address information appearing after mail stop designator words: MSC, MS, MAILSTOP, MAIL STOP, ATTN, ATTENTION.	61						●		
`GS-MATCH-CODE` Match code. For descriptions of match codes, see the *Status Codes* appendix.	5						●		●
`GS-MATCHED-DB` Index of database for matched record.							●		●
`GS-MCD-NAME` Minor Civil Division name from the auxiliary file. Blank if no auxiliary file match.	41						●		A
`GS-MCD-NUMBER` Minor Civil Division number from the auxiliary file. Blank if no auxiliary file match.	6						●		A
`GS-METRO-FLAG` Indicates if GS-CBSA-NAME • *Y* – Contains "Metropolitan Statistical Area" • *N* – Is non-blank but does not contain "Metropolitan Statistical Area" • *Blank* – Is blank (the county does not contain a CBSA).	2						●		●

Variable Name	Buffer Length[a]	GSHGET			GSDATSET	GSDATGET			GSMGET
		Street	Segment	Range		Input	Output	Center line	
GS-METRO-FLAG2 GS-METRO-FLAG for the second segment in the intersection.	2								I
GS-MM-RESULT-CODE Returns a MapMarker style result code.	12						●		●
GS-NAME Street name	41	●	●	●		●	●	●	●
GS-NAME2 Second street name in an intersection match.	41					I	I		●
GS-NAME-CITY City name for the matched address from the City State record.	29						●		●
GS-NAME-SHORT The short street name field contains the short street name used to construct the short address line. All attempts are made to abbreviate this name according to the process specified by the USPS in the 30 Character Abbreviation - Cycle M Flow Chart. If an abbreviated address cannot be constructed that is 30 characters or less, this field then contains the same street name value as the GS_NAME field variable return.	41						●		
GS-PARCEN-ELEVATION Elevation of the geocode at the parcel centroid.	7						A,P		A,P

Variable Name	Buffer Length[a]	GSHGET			GSDATSET	GSDATGET			GSMGET
		Street	Segment	Range		Input	Output	Center line	
GS-PERCENT-GEOCODE Percentage along the street segment to the interpolated match. The range is 0.0 - 100.0. The range is always 0.0 for point data.	5						R		A
GS-PMB-DESIGNATOR PMB designator (always "PMB"). NOTE: Not output when using GS-LINE to set the address.	5						●		
GS-PMB-NUMBER PMB number. NOTE: Not output when using GS-LINE to set the address.	9						●		
GS-POINT-ID Unique point ID of the matched record when matched to point-level data. *Blank* if the matched record is not from point-level data. NOTE: This variable is also available for other point datasets, i.e.TeleAtlas.	11		A,P				A,P		A,P
GS-POSTDIR Postfix direction. Can be blank, N, S, E, W, NE, NW, SW, or SE.	3	●	●	●		●	●	●	●
GS-POSTDIR2 Cross street postfix direction.	3					I	I		I

Variable Name	Buffer Length[a]	GSHGET			GSDATSET	GSDATGET			GSMGET
		Street	Segment	Range		Input	Output	Center line	
GS-POSTDIR-SHORT Postdir from the GS-ADDRLINE-SHORT.	3						●		
GS-PREDIR Prefix direction. Can be blank, N, S, E, W, NE, NW, SW, or SE.	3	●	●	●		●	●	●	●
GS-PREDIR2 Cross street prefix direction.	3					I	I		I
GS-PREDIR-SHORT Predir from the GS-ADDRLINE-SHORT.	3						●		●
GS-PREF-CITY Preferred city name for the output ZIP Code of the matched address.	29						●		●
GS-QCITY State , city , or finance numbers.	10	●	●	●			●	●	●
GS-R-RTE-DFLT Match indication for rural routes. • *N* – Matched to an exact rural route record. • *Y* – Did not find an exact record. Matched to the USPS default rural route record. Check the input address for accuracy and completeness. • *Blank* – Does not apply to the input address (for example, street addresses, P.O. Boxes, and General Delivery addresses) or no match found.	2						●		A

Variable Name	Buffer Length[a]	GSHGET			GSDATSET	GSDATGET			GSMGET
		Street	Segment	Range		Input	Output	Center line	
GS-RANGE-PARITY Indicates the parity of the house number in the range. • *E* – Even • *O* – Odd • *B* – Both	41			●					A
GS-REC-TYPE[b] Range record type. • A – Auxiliary file • *F* – Firm • *G* – General delivery • *H* – Highrise • *P* – PO Box • *R* – Rural route/highway contract • *S* – Street • *T* – TIGER file • *U* - User Dictionary	2			●			A		A
GS-RESOLVED-LINE indicates which line in a 2-line address GeoStan used to resolve the address. Relates to **GS-ADDRLINE** and **GS-ADDR2** with **GSDATSET.**	5						●		

Variable Name	Buffer Length[a]	GSHGET			GSDATSET	GSDATGET			GSMGET
		Street	Segment	Range		Input	Output	Center line	
`GS-ROAD-CLASS` Road class code. • *0* – Minor road, main data file • *1* – Major road, main data file • *10* – Minor road, supplemental file • *11* – Major road, supplemental data file	3		●	●			A	●	●
`GS-ROAD-CLASS2` `GS-ROAD-CLASS` for the second segment in the intersection.	3								I
`GS-SEG-HIRANGE` High house number in segment.	12		●	●				●	●
`GS-SEG-HIRANGE2` `GS-SEG-HIRANGE` for the second segment in the intersection.	12								I
`GS-SEG-LORANGE` Low house number in segment.	12		●	●				●	●
`GS-SEG-LORANGE2` `GS-SEG-LORANGE` for the second segment in the intersection.	12								I
`GS-SEGMENT-DIRECTION` • *F* – Numbers are forward • *R* – Numbers are reversed	2		●	●				●	●

Variable Name	Buffer Length[a]	GSHGET			GSDATSET	GSDATGET			GSMGET
		Street	Segment	Range		Input	Output	Center line	
GS-SEGMENT-DIRECTION2 GS-SEGMENT-DIRECTION for the second segment in the intersection.	2								I
GS-SEGMENT-ID Segment ID (TLID) or unique ID from premium data vendors.	11		●	●			A	●	●
GS-SEGMENT-ID2 GS-SEGMENT-ID for the second segment in the intersection.	11								I
GS-SEGMENT-PARITY Odd numbers in the segment are on: • *L* – Left side of the street • *R* – Right side of the street • *B* – Both sides of the street • *U* – Unknown	2		●	●				●	●
GS-SEGMENT-PARITY2 GS-SEGMENT-PARITY for the second segment in the intersection.	2								I
GS-STATE State abbreviation.	3	●	●	●	●	●	●		●
GS-STATE-FIPS State FIPS code.	3				●				

Variable Name	Buffer Length[a]	GSHGET			GSDATSET	GSDATGET			GSMGET
		Street	Segment	Range		Input	Output	Center line	
GS-STREET-SIDE The matched address is on the following side of the street: • *L* – Left side of the street • *R* – Right side of the street • *B* – Both sides of the street • *U* – Unknown side of the street This is relative to the segment end points and the segment direction (GS-SEGMENT-DIRECTION). NOTE: This enum does not reflect the value in the "Side of Street" field of any particular data vendor.	2						●		A
GS-SUITELINK-RET-CODE • A - Suite$^{\text{Link}}$ record match. • 00 - No Suite$^{\text{Link}}$ match. • Blank - This address was not processed through Suite$^{\text{Link}}$.	4						A		
GS-TYPE Street type (also called street suffix).	5	●	●	●		●	●	●	●
GS-TYPE2 Cross street type.	5					I	I		I
GS-TYPE-SHORT Postdir from the GS-ADDRLINE-SHORT.	6						●		
GS-UNIT-NUMBER Unit number.	12					A	A		A

Variable Name	Buffer Length[a]	GSHGET			GSDATSET	GSDATGET			GSMGET
		Street	Segment	Range		Input	Output	Center line	
GS-UNIT-NUMBER2 Second unit number parsed from the address line. Available in CASS and relaxed modes only.	12					A	A		
GS-UNIT-PARITY Indicates the parity of the unit number range (**GS-LOUNIT** and **GS-HIUNIT**). • *E* – Even • *O* – Odd • *B* – Both	2			●					A
GS-UNIT-TYPE Unit type (APT, STE, etc.).	5			●		A	A		A
GS-UNIT-TYPE2 Second unit type parsed from the address line. Available in CASS and relaxed modes only.	5					A	A		
GS-URB-NAME Urbanization name for Puerto Rico.	31				●	●	●		A
GS-ZIP 5-digit ZIP Code.	6			●	●	A	A		●
GS-ZIP4[b] 4-digit ZIP Code extension.	5				●	A	A		A
GS-ZIP9[c] 9-digit ZIP Code (ZIP + 4).	10				●	A	A		A

Variable Name	Buffer Length[a]	GSHGET			GSDATSET	GSDATGET			GSMGET
		Street	Segment	Range		Input	Output	Center line	
`GS-ZIP10`[c] 9-digit ZIP Code (ZIP + 4) with dash separator.	11				●	A	A		A
`GS-ZIP-CARRTSORT` Indicates the type of cart sort allowed. • *A* – Automation cart allowed, optional cart merging allowed. • *B* – Automation cart allowed, optional cart merging not allowed. • *C* – Automation cart not allowed, optional cart merging allowed. • *D* – Automation cart not allowed, optional cart merging not allowed.	2			●			A		●
`GS-ZIP-CITYDELV` Indicates if city-delivery is available. • Y – Post office has city-delivery carrier routes • N – Post office does not have city-delivery	2			●			A		●
`GS-ZIP-CLASS` ZIP Classification Code: • Blank – Standard ZIP Code • M – Military ZIP Code • P – ZIP Code has P.O. Boxes only • U – Unique ZIP Code. (A unique ZIP Code is a ZIP Code assigned to a company, agency, or entity with sufficient mail volume to receive its own ZIP Code.)	2			●			●		●

Variable Name	Buffer Length[a]		GSHGET		GSDATSET		GSDATGET		GSMGET
		Street	Segment	Range		Input	Output	Center line	
GS-ZIP-FACILITY Returns USPS City State Name Facility Code. • *A* – Airport Mail Facility (AMF) • *B* – Branch • *C* – Community Post Office (CPO) • *D* – Area Distribution Center (ADC) • *E* – Sectional Center Facility (SCF) • *F* – Delivery Distribution Center (DDC) • *G* – General Mail Facility (GMF) • *K* – Bulk Mail Center (BMC) • *M* – Money Order Unit • *N* – Non-Postal Community Name, Former Postal Facility, or Place Name • *P* – Post Office • *S* – Station • *U* – Urbanization	2			●			●		●
GS-ZIP-UNIQUE • *Y* – unique ZIP Code. (ZIP Code assigned to a company, agency, or entity with sufficient mail volume to receive its own ZIP Code.) • *Blank* – Not applicable.	2			●			●		●

a. The length column is not applicable.

b. Blank if running in CASS mode and you have not initialized DPV or the output address does not DPV confirm.

c. Last 4 digits are blank if running in CASS mode and you have not initialized DPV or the output address does not DPV confirm. For GsMultipleGet(), GeoStan returns ZIP+4 but only for Address matches.

COBOL procedures

GSCITYDG

RETRIEVES DATA LOCATED WITH GSCITYFF OR GSCITYFN.

Syntax

```
01 GSID                      PIC S9(9) BINARY.
01 GSFUNSTAT                 PIC 9(9) BINARY.
01 GSOPTIONS                 PIC 9(9) BINARY.
01 OUTLEN                    PIC 9(9) BINARY.
01 CITY-NAME                 PIC X(USER LEN).
01 RECNUM                    PIC S9(9) BINARY.
*
CALL “GSCITYDG” USING GSID, RECNUM, GSOPTIONS, CITY-NAME, OUTLEN, GSFUNSTAT.
```

Arguments

GSID ID returned by **GSINITWP** for the current instance of GeoStan. *Input*.

GSFUNSTAT Return value for the procedure. *Output*.

GSOPTIONS Symbolic constant for the data item to retrieve. The following table lists the valid variables. *Input*.

Variable	Size	Description
GS-CITY-CARRTSORT	(2)	• Y – Carrier Route Sort • N – No Carrier Route Sort
GS-CITY-CITYDELV	(2)	• Y – Has city-delivery carrier routes • N – Does not have city-delivery carrier routes
GS-CITY-CLASS	(2)	ZIP Classification Code.
GS-CITY-CTYSTKEY	(7)	6-character USPS City State Key that uniquely identifies a locale in the city/state file.

Variable	Size	Description
GS-CITY-FACILITY	(2)	Returns USPS City State Name Facility Code: • A – Airport Mail Facility (AMF) • B – Branch • C – Community Post Office (CPO) • D – Area Distribution Center (ADC) • E – Sectional Center Facility (SCF) • F – Delivery Distribution Center (DDC) • G – General Mail Facility (GMF) • K – Bulk Mail Center (BMC) • M – Money Order Unit • N – Non-Postal Community Name, Former Postal Facility, or Place Name • P – Post Office • S – Station • U – Urbanization
GS-CITY-MAILIND	(2)	• 1 – Can use City State Name as last line on mail piece • 0 – Cannot use City State Name as last line on main piece
GS-CITY-NAME	(30)	City name (may be an alternate name)
GS-CITY-PREFNAME	(30)	USPS preferred city name
GS-CITY-QCITY	(10)	City number, using the format `ssffffccc`, where `s` is the state number, `f` is the Finance number, and `c` is the city number
GS-CITY-UNIQUE	(2)	• Y – Unique ZIP Code (ZIP assigned to a single organization) • Blank – Not applicable
GS-CITY-STATE	(3)	State Abbreviation
GS-CITY-ZIP	(6)	ZIP Code

CITY-NAME Location to store the returned data. *Output*.

OUTLEN Maximum size of data to return in buffer. If `OUTLEN` is shorter than the data returned by GeoStan, GeoStan truncates the data and does not generate an error. *Input*.

RECNUM City record number, as returned by **`GSCITYFF`** or **`GSCITYFN`**. *Input*.

Return Values

GS-SUCCESS

GS-ERROR

Prerequisites

GSCITYFF or **GSCITYFN**

Notes

The **GSCITYDG** procedure retrieves data about the city located with **GSCITYFF** or **GSCITYFN**. City information from the USPS City/State file.

GSCITYFF

FINDS THE FIRST CITY MATCHING PARTIAL NAME OR VALID ZIP CODE.

Syntax

```
01 GSID              PIC S9(9) BINARY.
01 GSFUNSTAT         PIC S9(9) BINARY.
01 STATE-NAME        PIC X(USER LEN).
01 CITY-NAME         PIC X(USER LEN).
*
CALL “GSCITYFF” USING GSID, STATE-NAME, CITY-NAME, GSFUNSTAT.
```

Arguments

GSID	ID returned by **GSINITWP** for the current instance of GeoStan. *Input*.
GSFUNSTAT	Return value for the procedure. *Output*.
STATE-NAME	Proper state abbreviation for the searched state, or " " for a ZIP Code search. *Input*.
CITY-NAME	City to search for (may be just a partial string), or a 3-digit or 5-digit ZIP Code. *Input*.

Return Values

Record number of the city located, or zero if GeoStan did not find any cities.

Prerequisites

```
GSCLEAR
```

Notes

This procedure retrieves the record number of the first city in a state that matches the city or ZIP Code search string. For example, if the entered state string is `CA` and the city string is `S`, Geostan finds the first city that begins with `S` in California. If the entered city string is `803` and the state string is null, GeoStan the first city in that sectional center. GeoStan does not return cities in any predefined order.

Before each find procedure, call **`GSCLEAR`** to reset the internal buffers. If you do not reset the buffers, you may receive incorrect results with information from a previous find.

GSCITYFN

FINDS NEXT CITY MATCHING THE PARTIAL NAME OR VALID ZIP CODE.

Syntax

```
01 GSID             PIC S9(9) BINARY.
01 LSTATUS          PIC S9(9) BINARY.
*
CALL “GSCITYFN” USING GSID, LSTATUS.
```

Arguments

GSID	ID returned by **`GSINITWP`** for the current instance of GeoStan. *Input*.
LSTATUS	Return code. *Output*.

Return Values

Record number of the city located, or zero if GeoStan did not find any cities.

Prerequisites

```
GSCITYFF or GSCLEAR
```

Notes

Call this procedure after **`GSCITYFF`**.

Before each find procedure, call **GSCLEAR** to reset the internal buffers. If you do not reset the buffers, you may receive incorrect results with information from a previous find.

GSCLEAR

Clears the data buffer in the internal data structures.

Syntax

```
01 GSID          PIC S9(9) BINARY.
01 GSFUNSTAT     PIC S9(9) BINARY.
*
CALL “GSCLEAR” USING GSID, GSFUNSTAT.
```

Arguments

GSID ID returned by **GSINITWP** for the current instance of GeoStan. *Input*.

GSFUNSTAT Return value for the procedure. *Output*.

Return Values

GS-SUCCESS

GS-ERROR

Notes

Before each find function, call **GSCLEAR** to reset the buffers to null. If you do not reset the buffers, you may receive incorrect results with information from a previous find. This call clears only address element and locational information about the previously processed address and does not affect distance or centroid match type.

GSDATGET

RETURNS DATA FOR ALL ADDRESS AND MATCHED ELEMENTS FROM GEOSTAN.

Syntax

```
01 GSID                       PIC S9(9) BINARY.
01 GSOPTIONS                  PIC 9(9) BINARY.
01 GSSWITCH                   PIC 9(9) BINARY.
01 OUTPUT-STRING              PIC X(USER LEN).
01 OUTLEN                     PIC 9(4) BINARY.
01 GSFUNSTAT                  PIC S9(9) BINARY.
*
CALL “GSDATGET” USING GSID, GSOPTIONS, GSSWITCH, OUTPUT-STRING, OUTLEN,
GSFUNSTAT.
```

Arguments

GSID ID returned by **GSINITWP** for the current instance of GeoStan. *Input*.

GSOPTIONS Flag indicating whether to retrieve original or processed data. *Input*.

To retrieve parsed address components, set *GSOPTIONS* to GS-INPUT. To retrieve matched address information from the parsed input address and the GeoStan database, set *GSOPTIONS* to GS-OUTPUT.

If the input address does not match, setting *GSOPTIONS* to GS-OUTPUT returns the address exactly as entered, except for the last line information, which returns only the parsed last line components.

The parsed last line components correspond to the following variables:

GS-LASTLINE	GS-ZIP	GS-ZIP9
GS-CITY	GS-ZIP4	GS-ZIP10
GS-STATE		

If there is extra data on the input last line (GS-LASTLINE), this data is not retrievable. For example, in the last line “BOULDER CO 80301 US OF A”, “US OF A” is not retrievable from any **GSDATGET()** procedure.

NOTE: For valid variables, see “Common variables for storing and retrieving data” on page 248.

GSSWITCH Symbolic constant for the data item to retrieve. *Input*.

GSFUNSTAT Return value for the procedure. *Output*.

OUTPUT-STRING Location to store the returned data. *Output*.

OUTLEN	Maximum length of data for GeoStan to return. The COBOL copy member "GEOSTAN" lists as constants the recommended buffer size for each item. These sizes are the maximum lengths required to get the full output string. You can allocate a buffer that is smaller or larger than these values. However, if *bufLen* is shorter than the returned data, GeoStan truncates the data and does not generate an error. *Input*.

Return Values

GS-SUCCESS

GS-ERROR

Prerequisites

GSSFINDWP

Notes

This procedure retrieves data elements from internal GeoStan buffers for either the original (input) or matched (output) data elements. To retrieve original data, set *GSOPTIONS* to GS-INPUT. To retrieve matched data, set *GSOPTIONS* to GS-OUTPUT.

GSDATSET

PASSES DATA FOR ALL ADDRESS ELEMENTS TO GEOSTAN.

Syntax

```
01 GSID                        PIC S9(9) BINARY.
01 GSFUNSTAT                   PIC S9(9) BINARY.
01 GSOPTIONS                   PIC 9(9) BINARY.
01 VALUE                       PIC X(LEN).
*
CALL “GSDATSET” USING GSID, GSOPTIONS, VALUE, GSFUNSTAT.
```

Arguments

GSID	ID returned by **GSINITWP** for the current instance of GeoStan. *Input*.
GSFUNSTAT	Return value for the procedure. *Output*.
GSOPTIONS	Symbolic constant for the data item to load. See "Common variables for storing and retrieving data" on page 248" for a list of valid variables. *Input*.

VALUE	String pointer containing the data to be loaded. *Input*.

Return Values

GS-SUCCESS

GS-ERROR

Prerequisites

GSINITWP

Notes

This procedure loads data elements into internal GeoStan buffers. When loading address information as a complete address or last line, GeoStan parses the data into fields. For example, if you entered a last line of “Boulder, CO 80301-1234”, GeoStan parses the data and sets the city, state, ZIP, and ZIP + 4 fields. You can retrieve parsed data using **GSDATGET** and requesting the INPUT to these fields.

If you are passing both an address line and a last line or ZIP Code, be sure to enter the last line or ZIP Code *first* to ensure the greatest accuracy in address standardization.

If you are passing both the address information and the last line information as one input line, enter the address information *first*.

Using the appropriate parameters defined in the COBOL copy member “GEOSTAN”, you can pass single line addresses, two-line addresses, or multiline addresses of up to six lines. For more information, see “Extracting data from GSD files” on page 407.

Do not call **GSSMM** after a call to **GSDATSET**. If you need to change the match mode in mid-process, you must re-enter the data for the current address with **GSDATSET**.

GSDBMETA

RETRIEVES INFORMATION ABOUT EACH DATABASE FILE.

Syntax

```
01 GSID PIC S9(9) BINARY.
01 GSFUNSTAT PIC S9(9) BINARY.
01 DB-NUMBER PIC S9(9) BINARY.
01 ITEM-CODE PIC S9(9) BINARY.
01 STRING-BUFFER PIC X(USER LEN).
01 BUFFER-LENGTH PIC S9(9) BINARY.
*
BUFFER-LENGTH = LENGTH OF STRING-BUFFER.
```

```
CALL "GSDBMETA" USING GSID, ITEM-CODE, DB-NUMBER, STRING-BUFFER, BUFFER-
LENGTH,   GSFUNSTAT.
```

Arguments

GSID ID Returned by GSINITWP for the current instance of GeoStan. Input.

GSFUNSTAT Return value for the procedure. Output.

ITEM-CODE Code for data item requested.   Input.

DB-NUMBER Database file number.   Input.

*STRING-BUFFER*Return value for requested data item. Output.

*BUFFER-LENGTH*Maximum size of returned value string. Input.

The following named code values are defined in the copy member “GEOSTAN” for this function:

GS-STATUS-DATATYPE-NUM	Source data type number.
GS-STATUS-DATATYPE-STR	Source data type descriptive text.
GS-STATUS-DATUM-NUM	Coordinate system code.
GS-STATUS-DATUM-STR	Coordinate system name (Datum)
GS-STATUS-FILE-CHKSUM-NUM	File Header checksum.
GS-STATUS-RECORDS-REMAINING	Metered retrievals remaining (for metered usage license only).
GS-STATUS-GEO-RECORD-TOTAL	Metered retrievals available by license (for metered usage license only).
GS-STATUS-DAYS-REMAINING	Usage days remaining (for temporary license only).
GS-STATUS-GEO-PRECISION	Coordinate precision of data source - number of digits.
GS-STATUS-DB-COPYRIGHT	Copyright declarations for data source.
GS-STATUS-DB-COUNTRY	Country name for data source.
GS-STATUS-DB-VERSION	Database version number.
GS-STATUS-DB-FILE-PATH	File name.

Return Values

Returns.

Prerequisites

`GSINITWP`

Notes

The value in the string buffer is space-padded to the length of the buffer and has no terminator. If the buffer length is shorter than the available value, the returned value will be truncated. The maximum size of any string returned is 255 characters.

GSDPVGCS

OBTAINS THE COMPLETE DPV STATISTICS SINCE THE APPLICATION INITIALIZED DPV.

Syntax

```
01 GSID       PIC S9(9) BINARY.
01 GS-LACS-COMPLETE-STATS
01 RETURN-CODEPIC 9(9) BINARY.
CALL "GSDPVGCS" USING GSID, GS-DPV-COMPLETE-STATS, GS-DCS-SIZE, RETURN-CODE.
```

Arguments

GSID ID returned by **GSINITWP** for the current instance of GeoStan. *Input.*

GS-LACS-COMPLETE-STATS Structure containing the following DPV statistics since the application initialized DPV:

GS-DCS-NCMRA-IN-ERROR Number of records with a CMRA unconfirmed. *Output.*

GS-DCS-NCMRA-VALID Number of records with a CMRA confirmed. *Output.*

GS-DCS-NDPV-IN-ERROR Number of records not confirmed by DPV. *Output.*

GS-DCS-NDPV-NO-STAT-FOUND Not Applicable for this release. *Output.*

GS-DCS-NDPV-SEED-HITS Number of DPV false positive matches. *Output.*

GS-DCS-NDPV-TYPES-ERROR Number of records with a valid primary range, but the secondary range is not confirmed. *Output.*

GS-DCS-NDPV-TYPEY-ERROR Number of records fully confirmed by DPV. *Output.*

GS-DCS-NDPV-ZIPS-ON-FILE Number of distinct ZIP Codes in file. *Output.*

GS-DCS-NFIRM-CMRA-PRESENTED Number of USPS Firm records with PMB presented. *Output.*

GS-DCS-NFIRM-CMRA-VALID Number of USPS Firm records CMRA confirmed with or without PMB. *Output.*

GS-DCS-NFIRM-PRIME-FAIL Number of USPS Firm records without a confirmed primary number. *Output.*

GS-DCS-NFIRM-SECONDARY-FAIL Number of USPS Firm records failed to confirm secondary number. *Output.*

GS-DCS-NFIRM-VALID Number of confirmed USPS Firm records. *Output.*

GS-DCS-NGEN-DEL-VALID Number of confirmed USPS General Delivery records. *Output.*

GS-DCS-NHR-CMRA-PRESENTED Number of USPS Highrise records with PMB presented. *Output.*

GS-DCS-NHR-CMRA-VALID Number of USPS Highrise with confirmed CMRA records with or without PMB. *Output.*

GS-DCS-NHR-PRIME-FAIL Number of USPS Highrise records failed to confirm primary number. *Output.*

GS-DCS-NHR-SECONDARY-FAIL Number of USPS Highrise records failed to confirm secondary number. *Output.*

GS-DCS-NHR-VALID Number of confirmed USPS Highrise records. *Output.*

GS-DCS-NPO-BOX-FAIL Number of USPS PO Box records failed to confirm primary number. *Output.*

GS-DCS-NPO-BOX-VALID Number of confirmed USPS PO Box records. *Output.*

GS-DCS-NRR-CMRA-PRESENTED Number of USPS Rural Route records with PMB presented to the DPV data. *Output.*

GS-DCS-NRR-CMRA-VALID Number of USPS Rural Route records CMRA confirmed with or without PMB. *Output.*

GS-DCS-NRR-HC-PRIME-FAIL Number of USPS Rural Route records failed to confirm primary number. *Output.*

GS-DCS-NRR-HC-VALID Number of USPS Rural Route records confirmed. *Output*

GS-DCS-NST-CMRA-PRESENTED Number of USPS Street records with PMB presented. *Output.*

GS-DCS-NST-CMRA-VALID Number of USPS Street records CMRA confirmed with or without PMB. *Output.*

GS-DCS-NSTREET-PRIME-FAIL Number of USPS Street records failed to confirm primary number. *Output.*

GS-DCS-NSTREET-SECONDARY-FAIL Number of USPS Street records fail to confirm secondary number. *Output.*

GS-DCS-NTOTAL-DPV-PROCESSED Number of records processed through DPV. *Output.*

GS-DCS-NTOTAL-DPV-WITH-ZIP4 Number of records with ZIP + 4 processed through DPV. *Output.*

GS-DCS-FOOTNOTE-AA Number of records with DPV footnote value AA. *Output.*

GS-DCS-FOOTNOTE-A1 Number of records with DPV footnote value A1. *Output.*

GS-DCS-FOOTNOTE-BB Number of records with DPV footnote value BB. *Output.*

GS-DCS-FOOTNOTE-CC Number of records with DPV footnote value CC. *Output.*

GS-DCS-FOOTNOTE-F1 Number of records with DPV footnote value F1. *Output.*

GS-DCS-FOOTNOTE-G1 Number of records with DPV footnote value G1. *Output.*

GS-DCS-FOOTNOTE-M1 Number of records with DPV footnote value M1. *Output.*

GS-DCS-FOOTNOTE-M3 Number of records with DPV footnote value M3. *Output.*

GS-DCS-FOOTNOTE-N1 Number of records with DPV footnote value N1. *Output.*

GS-DCS-FOOTNOTE-P1 Number of records with DPV footnote value P1. *Output.*

GS-DCS-FOOTNOTE-P3 Number of records with DPV footnote value P3. *Output.*

GS-DCS-FOOTNOTE-RR Number of records with DPV footnote value RR. *Output.*

GS-DCS-FOOTNOTE-R1 Number of records with DPV footnote value R1. *Output.*

GS-DCS-FOOTNOTE-U1 Number of records with DPV footnote value U1. *Output.*

RETURN-CODE Size of the **GSDPVGCS** data structure. *Input.*

Return Values

GS-SUCCESS

`GS-ERROR`	Call **`GSERRGTX`** for more information
`GS-WARNING`	Call **`GSERRGTX`** for more information

Prerequisites

`GSINITWP`

GSDPVGFD

Retrieves the detail record for a DPV false positive report.

Syntax

```
01 GSID             PIC S9(9) BINARY.
01 GS-FALSE-POS-DETAIL-DATA
01 RETURN-CODE      PIC 9(9) BINARY.
CALL "GSDPVGFD" USING GSID, GS-FALSE-POS-DEFAILT-DATA, GS-FPDD-SIZE, RETURN-
CODE.
```

Arguments

`GSID` ID returned by **`GSINITWP`** for the current instance of GeoStan. *Input.*

`GS-FALSE-POS-DETAIL-DATA` Retrieves the DPV detail record for a false positive address match using the data passed in **`GSDPVDFD`**. The data members are details provided by GeoStan for the false positive report. This structure contains the following:

`GS-FPPD-ADDRESS-SECONDARY-ABBRV` Unit type (APT, SUITE, LOT). *Output.*

`GS-FPPD-ADDRESS-SECONDARY-NUM` Unit number. *Output.*

`GS-FPPD-MATCHED-PLUS4` ZIP Code extension. *Output*.

`GS-FPPD-MATCHED-ZIP-CODE` ZIP Code. *Output.*

`GS-FPPD-POST-DIRECTIONAL` Street name postdirectional (N, S, E, W). *Output*.

`GS-FPPD-PRIMARY-NUMBER` House number. *Output.*

`GS-FPPD-STREET-NAME` Street name. *Output*.

`GS-FPPD-STREET-PREDIR` Street name predirectional (N, S, E, W). *Output*.

`GS-FPPD-SUFFIX-ABBREVIATION` Street type (AVE, ST, RD). *Output*.

FILLER Reserved for future implementation. *Output.*

RETURN-CODE Size of the **GsFalsePosDetailData** data structure. *Input*.

Return Values

GS-SUCCESS

GS-ERROR Call **GSERRGTX** for more information

GS-WARNING Call **GSERRGTX** for more information

Prerequisites

GSDPVINR

GSDPVGFH

RETRIEVES DPV STATISTICS FOR THE HEADER RECORD FOR A DPV FALSE POSITIVE REPORT.

Syntax

```
01 GSID                        PIC S9(9) BINARY.
GS-FALSE-POS-HEADER-DATA
01 RETURN-CODE                 PIC 9(9) BINARY.
CALL "GSDPVGFH" USING GSID, GS-FALSE-POS-HEADER-DATA, GS-FPHD-SIZE, RETURN-
CODE.
```

Arguments

GSID ID returned by **GSINITWP** for the current instance of GeoStan. *Input.*

GS-FALSE-POS-HEADER-DATA Retrieves DPV statistics from the header record for false positive address matches using the data passed in **GSDPVGFD**. The output data members are statistics for the false positive reports. The input data members include information to correctly complete the false positive report for the USPS. This structure contains the following:

GS-FPHD-MAILERS-ADDRESS-LINE Address of the mailer. *Input.*

GS-FPHD-MAILERS-CITY-NAME City of the mailer. *Input.*

GS-FPHD-MAILERS-COMPANY-NAME Name of the mailer. *Input.*

GS-FPHD-MAILERS-STATE-NAME State of the mailer. *Input*.

GS-FPHD-MAILERS9-DIGITZIP ZIP Code of the mailer. *Input*.

GS-FPHD-NUMBER-FALSE-POS Number of found DPV false positive matches. *Output*.

GS-FPHD-NUMBER-ZIP-ON-FILE Number of distinct ZIP Codes processed through DPV. *Output*.

GS-FPHD-TOTAL-MATCHED Number of records matched with DPV data. *Output*.

GS-FPHD-TOTAL-PROCESSED Number of records processed through DPV. *Output*.

GS-FPHD-TOTAL-ZIP4-MATCHED Number of records that have matched with ZIP + 4. *Output*.

RETURN-CODE Size of the **GSDPVGFD** data structure. *Input*.

Return Values

GS-SUCCESS

GS-ERROR Call **GSERRGTX** for more information

GS-WARNING Call **GSERRGTX** for more information

Prerequisites

GSINITWP with the appropriate DPV initialization properties set.

GSERRGET

RETRIEVES CURRENT ERROR INFORMATION.

Syntax

```
01 GSID                      PIC S9(9) BINARY.
01 GSFUNSTAT                 PIC S9(9) BINARY.
01 MESSAGE-STRING            PIC X(256).
01 DETAIL-STRING             PIC X(256).
*
CALL “GSERRGET” USING GSID, MESSAGE-STRING, DETAIL-STRING, GSFUNSTAT.
```

Arguments

GSID ID returned by **GSINITWP** for the current instance of GeoStan. *Input*.

GSFUNSTAT Return value for the procedure. *Output*.

MESSAGE-STRING Basic explanation for the error; up to 256 bytes in length. *Output*.

DETAIL-STRING Particulars of an error, such as filename; up to 256 bytes in length. *Output*.

NOTE: Always set buffers for *MESSAGE-STRING* and *DETAIL-STRING* to 256 bytes or larger.

Return Values

Error number of the most recent GeoStan error.

-1	No error.
0 through 99	Indicates the actual DOS error values.
100	Unclassified error.
101	Unknown error.
102	Invalid file signature.
103	Table overflow.
104	Insufficient memory.
105	File not found.
106	Invalid argument to a procedure.
107	File is out of date.

Alternates

GSERRGTX

GSERRGTX

RETRIEVES INFORMATIONAL, ERROR, AND FATAL WARNING MESSAGES FOR THE CURRENT THREAD.

Syntax

```
01 GSID                                     PIC S9(9) BINARY.
01 ERROR-MESSAGE                            PIC X(256).
01 DETAILS                                  PIC X(256).
01 GSFUNSTAT                                PIC S9(9) BINARY.

CALL “GSERRGTX” USING GSID, ERROR-MESSAGE, DETAILS, GSFUNSTAT.
```

Arguments

GSID ID returned by **GSINITWP** for the current instance of GeoStan. *Input*.

ERROR-MESSAGE GeoStan error code and descriptive text. *Output*.

DETAILS Descriptive message, such as the file name associated with the error. *Output*.

GSFUNSTAT Return value for the procedure. *Output*.

Return values

Next GeoStan error detected in the current thread. Upon return, error messages contain a brief description and additional text, such as the name of the file or directory associated with the error.

NOTE: Not all errors are fatal. For example, if GeoStan finds an inappropriate GSI file, it reports the error but continues executing.

Prerequisites

GSINITWP and GSERRHAS

GSERRHAS

INDICATES IF ANY ERRORS OCCURRED OR ANY INFORMATION MESSAGES ARE AVAILABLE IN THE CURRENT THREAD.

Syntax

```
01 GSID                             PIC S9(9) BINARY.
01 MOREERRORS                       PIC S9(9) BINARY.

CALL “GSERRHAS” USING GSID, MOREERRORS.
```

Arguments

GSID ID returned by **GSINITWP** if initialization completed, or set to NULL if initialization failed and **GSINITWP** returned NULL. *Input*.

MOREERRORS Zero if there are no more error messages to retrieve (via **GSERRGTX**), and non-zero if there are additional messages. *Output*.

Return Values

non-zero value Errors occurred or GeoStan generated informational messages

zero No errors or informational messages are pending.

Prerequisites

`GSINITWP`

GSFCASSH

Writes a CASS 3553 report to the header buffer argument.

Syntax

```
01 GSID                               PIC S9(9).
01 GSEXTENDCASSDATA
01 HEADER-BUFFER                      PIC X(USER LEN).
01 DATA-SIZE                          PIC S9(9) BINARY.
01 BUFSIZE                            PIC S9(9) BINARY.
01 GSFUNSTAT                          PIC S9(9) BINARY.

CALL "GSFCASSH" USING GSID, GSEXTENDCASSDATA, DATA-SIZE, HEADER-BUFFER,
BUFSIZE, GSFUNSTAT.
```

Arguments

`GSID` ID returned by **`GSINITWP`** for the current instance of GeoStan. *Input*.

`GSEXTENDCASSDATA` Pointer to the CASSS report data structure. The following table contains information about the structure. *Input*.

05	GS-ECD-STRUC-TVERSION	PIC S9(9) BINARY.
05	GS-ECD-NRECS	PIC S9(9) BINARY.
05	GS-ECD-NZIP4	PIC S9(9) BINARY.
05	GS-ECD NZIP	PIC S9(9) BINARY.
05	GS-ECD-NCARRT	PIC S9(9) BINARY.
05	GS-ECD-NDPBC	PIC S9(9) BINARY.
05	GS-ECD-LISTNAME	PIC X(20).
05	GS-ECD-VERSION	PIC X(12).

05	GS-ECD-CERTIFICATION-DATE	PIC X(24).
05	GS-ECD-SEARCH-PATH	PIC X(256).
05	GS-ECD-TEMPLATE-NAME	PIC X(256).
05	GS-ECD-NZ4CHANGED	PIC 9(9) BINARY.
05	GS-ECD-NLOT	PIC 9(9) BINARY.
05	GS-ECD-Z4CHANGE-VERSION	PIC X(12).
05	GS-ECD-LOT-VERSION	PIC X(12).
05	GS-ECD-NHIGHRISE-DEFAULT	PIC S9(9) BINARY.
05	GS-ECD-NHIGHRISE-EXACT	PIC S9(9) BINARY.
05	GS-ECD-NRURALROUTE-DEFAULT	PIC S9(9) BINARY.
05	GS-ECD-NRURALROUTE-EXACT	PIC S9(9) BINARY.
05	GS-ECD-NLACS	PIC S9(9) BINARY.
05	GS-ECD-Z4-COMPANY-NAME	PIC X(40).
05	GS-ECD-DPC-COMPANY-NAME	PIC X(40).
05	GS-ECD-Z4-CONFIG	PIC X(3).
05	GS-ECD-DPC-CONFIG	PIC X(3).
05	GS-ECD-Z4-SOFTWARE-NAME	PIC X(30).
05	GS-ECD-DPC-SOFTWARE-NAME	PIC X(30).
05	GS-ECD-LIST-PROCESSOR-NAME	PIC X(25).
05	FILLER	PIC X(3).
05	GS-ECD-ZIP4-DATABASE-DATE	PIC S9(9) BINARY.
05	GS-ECD-LOT-DATABASE-DATE	PIC S9(9) BINARY.
05	GS-ECD-EWS-DENIAL	PIC S9(9) BINARY.

DATA-SIZE Size of the CASS report data structure. *Input*.

HEADER-BUFFER Buffer containing the CASS header line from the Stage file. *Input*, *Output*.

BUFSIZE Length of the header buffer. *Input*.

GSFUNSTAT Return value for the procedure. *Output*.

Return Values

GS-SUCCESS

```
GS-ERROR

GS-WARNING
```

Prerequisites

```
GSINITWP
```

Notes

When you specify a version number for either version, `GS-ECD-LOT-VERSION or GS-ECD-LOT-VERSION`, GeoStan updates the corresponding fields in the header buffer similar to **`GSWECASS`**. For example, entering a version number for `GS-ECD-LOT-VERSION` prompts GeoStan to fill the following fields in `CASS355.frm`:

- Section “B. List”, Item “2b”: Data List Processed Z4Change
- Section “B. List”, Item “3b”: Data of Database Product used Z4Change.
- Section “C. Output”, Item “1b”: Total coded Z4Change Processed.

To develop CASS certified application in GeoStan, you must have the correct license agreement with Pitney Bowes Business Insight. You must also obtain CASS certification from the USPS for your application. Using GeoStan does not make an application CASS certified. For information on becoming CASS certified, refer Appendix E, “CASS certification.”

GSFDFPD

FORMATS A DPV FALSE POSITIVE DETAIL RECORD FROM GSDPVGFD.

Syntax

```
01 GSID              PIC S9(9) BINARY.
01 GS-FALSE-POS-DETAIL-DATA
01 RETURN-CODE       PIC 9(9) BINARY.
01 HEADER            PIC X(len)
01 HEADER-SIZE       PIC 9(9) BINARY VALUE len.

CALL "GSFDFPD" USING GSID, GS-FALSE-POS-DETAIL-DATA, GS-FPHD-SIZE, HEADER,
HEADER-SIZE, RETURN-CODE.
```

Arguments

`GSID` ID returned by **`GSINITWP`** for the current instance of GeoStan. *Input*

`GS-FALSE-POS-DETAIL-DATA` Pointer to the DPV report data structure. This value is completed by **`GSDPVGFD`**. *Input.*

RETURN-CODE Size of the DPV report data structure. *Input.*

HEADER Buffer containing the DPV false positive detail record after **GSFDPD** successfully completes. When the application writes the false positive report, it writes this buffer to the last line of the file. *Output.*

HEADER-SIZE Length of the detail buffer. *Input.*

Return Values

GS-SUCCESS

GS-ERROR Call **GSERRGTX** for more information.

GS-WARNING Call **GSERRGTX** for more information.

Prerequisites

GSDPVGFD

GSFDFPH

FORMATS A DPV FALSE POSITIVE HEADER RECORD WITH DATA FROM GSDPVGFD.

Syntax

```
01 GSID PIC S9(9) BINARY.
01 GS-FALSE-POS-HEADER-DATA
01 RETURN-CODE PIC 9(9) BINARY.
01 HEADER PIC X(len).
01 HEADER-SIZE PIC 9(9) BINARY VALUE len.

CALL "GSFDFPH" USING GSID, GS-FALSE-POS-HEADER-DATA, GS-FPHD-SIZE, HEADER,
HEADER-SIZE, RETURN-CODE.
```

Arguments

GSID ID returned by **GSINITWP** for the current instance of GeoStan. *Input.*

GS-FALSE-POS-HEADER-DATA Pointer to the DPV report data structure. This value is completed by **GSDPVGFH**. *Input.*

RETURN-CODE Size of the DPV report data structure. *Input.*

`HEADER` Buffer containing the DPV false positive header after **GSFDFPH** successfully completes. When the application writes the false positive report, it writes this buffer to the first line of the file. *Output*.

`HEADER-SIZE` Length of the header buffer. *Input*.

Return Values

`GS-SUCCESS`

`GS-ERROR` Call **GSERRGTX** for more information.

`GS-WARNING` Call **GSERRGTX** for more information.

Prerequisites

```
GSDPVINR
GSDPVGFH
```

GSFFRANG

Finds the first range object that meets the search criteria.

Syntax

```
01 GSID                              PIC S9(9) BINARY.
01 RANGE-HANDLE                      PIC S9(9) BINARY.
01 GSFUNSTAT                         PIC S9(9) BINARY.
*
CALL “GSFFRANG” USING GSID, RANGE-HANDLE, GSFUNSTAT.
```

Arguments

`GSID` ID returned by **GSINITWP** for the current instance of GeoStan. *Input*.

`RANGE-HANDLE` Pointer to a range handle. Returns a valid handle if GeoStan finds a range. *Input*, *Output*.

`GSFUNSTAT` Return value for the procedure. *Output*.

Return Values

`GS-SUCCESS` Found a match. You can retrieve the data for that match using **GSHGET**.

GS-NOT-FOUND	Did not find a match.
GS-ERROR	An error occurred; use **GSERRGET** and **GSERRGTX** to retrieve more information.

Prerequisites

GSINITWP and GSCLEAR

Notes

You must call **GSFFSEG** before calling **GSFFRANG**.

GSFFSEG

FINDS THE FIRST SEGMENT OBJECT THAT MEETS THE SEARCH CRITERIA.

Syntax

```
01 GSID           PIC S9(9) BINARY.
01 GSFUNSTAT      PIC S9(9) BINARY.
01 SEGMENT-HANDLE PIC S9(9) BINARY.
*
CALL “GSFFSEG” USING GSID, SEGMENT-HANDLE, GSFUNSTAT.
```

Arguments

GSID	ID returned by **GSINITWP** for the current instance of GeoStan. *Input*.
GSFUNSTAT	Return value for the procedure. *Output*.
SEGMENT-HANDLE	Pointer to a street/segment handle. Returns a valid handle if GeoStan finds a segment. *Input*, *Output*.

Return Values

GSGS-SUCCESS	Found a match. You can retrieve the data for that match using **GSHGET**.
GS-NOT-FOUND	Did not find a match.
GS-ERROR	An error occurred; use **GSERRGET** and **GSERRGTX** to retrieve more information.

Prerequisites

`GSINITWP` and `GSCLEAR`

Notes

This procedure finds the first street in the current search area that meets the name criteria, and also sets the area and criteria for subsequent segment and range searches.

If you are using **`GSFFSEG`** to find a street that has a number as a component of the street name, such as “US HWY 41,” or “I-95,” enter just the number; the text is not part of the index for such streets.

`GSFFSEG` and **`GSFNRANG`** procedures work together to allow you to access to the entire address directory database.

See Appendix B, “Extracting data from GSD files” for more information on the Find First and Find Next procedures.

GSFFST

FINDS THE FIRST STREET OBJECT THAT MEETS THE SEARCH CRITERIA.

Syntax

```
01 GSID                                       PIC S9(9) BINARY.
01 GSFUNSTAT                                  PIC S9(9) BINARY.
01 STREET-HANDLE                              PIC S9(9) BINARY.
01 GSOPTIONS                                  PIC  9(9) BINARY.
01 LOCALE                                     PIC X(USER LEN).
01 STREET-NAME                                PIC X(USER LEN).
01 STREET-NUMBER                              PIC X(USER LEN).
*
CALL “GSFFST” USING GSID, STREET-HANDLE, GSOPTIONS, LOCALE, STREET-NAME,
STREET-NUMBER, GSFUNSTAT.
```

Arguments

GSID ID returned by **`GSINITWP`** for the current instance of GeoStan. *Input*.

GSFUNSTAT Return value for the procedure. *Output*.

STREET-HANDLE Pointer to a street handle. Returns a valid handle if GeoStan finds a street. *Output*.

GSOPTIONS Set the type of search performed by GeoStan. The following table contains the types of searches available. *Input*.

GS-ZIP-SEARCH	*Required.* GeoStan searches the ZIP Code specified by LOCALE.
GS-CITY-SEARCH	*Required.* GeoStan searches the city and state specified by LOCALE.
GS-SDX-SEARCH	*Optional.* GeoStan searches by Soundex. STREET-NAME is a pointer to a numeric soundex key returned by **GSSNDX**.

LOCALE Sets the search ares. If *GSOPTIONS* is set to GS-ZIP-SEARCH, then *LOCALE* is a valid Zip Code. If *GSOPTIONS* is set to GS-CITY-SEARCH, then *LOCALE* is a valid city and state. *Input*.

STREET-NAME Street name, or partial street name, for which to search. If *GSOPTIONS* is set to GS-SDX-SEARCH, the *STREET-NAME* is a pointer to a numeric soundex key. *Input*.

Limits the search to street names that begin with the name string. If *STREET-NAME* is set to "APPLE," then only streets beginning with Apple are returned, such as Apple or Appleton. If *STREET-NAME* is not specified, GeoStan finds all the streets specified by *LOCALE*.

Return Values

GSGS-SUCCESS Found a match. You can retrieve the data for that match using **GSHGET**.

GS-NOT-FOUND Did not find a match.

GS-LASTLINE-NOT-FOUND Could not find *LOCALE*.

GS-ERROR An error occurred; use **GSERRGET** and **GSERRGTX** to retrieve more information.

Prerequisites

GSINITWP and GSCLEAR

Notes

You must call **GSFFST** before **GSFFSEG**. This procedure also sets the area and criteria for subsequent segment and range searches.

If you are using **GSFFST** to find a street that has a number as a component of the street name, such as "US HWY 41" or "I-95," enter just the number; the text is not part of the index for such streets.

See Appendix B, "Extracting data from GSD files" for more information on the Find First and Find Next procedures.

GSFFSTAT

FINDS THE USPS STATE NAME ABBREVIATION.

Syntax

```
01 GSID                      PIC S9(9) BINARY.
01 GSFUNSTAT                 PIC 9(9) BINARY.
01 OUTLEN                    PIC 9(9) BINARY.
01 STATE-FOUND               PIC X(USER LEN).
01 STATE-PATTERN             PIC X(USER LEN).
*
CALL “GSFFSTAT” USING GSID, STATE-PATTERN, STATE-FOUND, OUTLEN, GSFUNSTAT.
```

Arguments

GSID ID returned by **GSINITWP** for the current instance of GeoStan. *Input*.

GSFUNSTAT Return value for the procedure. *Output*.

OUTLEN Maximum length of data returned in *STATE-FOUND*, including NULL. *Input*.

STATE-FOUND Proper state abbreviation for located state. *Output*.

STATE-PATTERN Search string for state. *Input*.

Can be any of the following items:

- Five-digit ZIP Code.
- First three digits of a ZIP Code known as the Sectional Center Facility (SCF).
- Various state abbreviations and spellings. For example, for New Hampshire, it accepts: N HAMP, N HAMPSHIRE, NEW HAMPSHIRE, NEWHAMPSHIRE, NH, and NHAMPSHIRE.

Return Value

GS-ERROR

GS-SUCCESS

GS-NOT-FOUND

Prerequisites

GSINITWP or **GSCLEAR**

Notes

This procedure returns the proper abbreviation for the requested state. You can request states either by ZIP Code, full state name, or an alternate abbreviation.

Before each find procedure, call **GSCLEAR** to reset the internal buffers. If you do not reset the buffers, you may receive incorrect results with information from a previous find.

GSFGF

FINDS THE FIRST GEOGRAPHIC INFORMATION RECORD WITH PARTIAL MATCHING TO INPUT NAMES.

Syntax

```
01 GSID PIC S9(9) BINARY.
01 GSFUNSTAT PIC S9(9) BINARY.
*
MOVE SPACES TO GS-GEOGRAPHIC-INFO.
MOVE <a City Name> TO GS-INPUT-CITY.
MOVE <a County Name> TO GS-INPUT-COUNTY.
MOVE <a State Name/Abbreviation> TO GS-INPUT-STATE.

CALL "GSFGF" USING GSID, GS-GEOGRAPHIC-INFO, GSFUNSTAT.
```

Arguments

GSID Returned by GSINITWP for the current instance of GeoStan. Input.

GSFUNSTAT Return value for the procedure. *Output.*

GS-INPUT-CITY City Name (may be a partial string). *Input.*

GS-INPUT-COUNTY County Name (may be a partial string). Optional. *Input.*

GS-INPU-STATE Proper state abbreviation or name for the searched state. Input.

Return Values

First Geographic Information Record which has a partial match to the input.

"GSFUNSTAT" contains a value of "GS-SUCCESS" if there are one or more matching records.

The following named items are defined in module the COBOL copy member "GEOSTAN" for this function:

GS-OUTPUT-CITY	Full City name.
GS-OUTPUT-COUNTY	County name.
GS-OUTPUT-STATE	State FIPS abbreviation.
GS-OUTPUT-LAT	Latitude.
GS-OUTPUT-LONG	Longitude.
GS-OUTPUT-RANK	Metropolitan Area Economic and Population Rank.
GS-OUTPUT-RESULT-CODE	GeoStan Match result code.
GS-OUTPUT-LOCATION-CODE	Geostan Location code.
GS-CLOSE-MATCH-FLAG	Indicates close name match.

Prerequisites

```
GSINITWP
```

Notes

This procedure retrieves the first Geographic information record which matches the input city, county, or state names. The match logic may qualify more than one response. Use calls to "GSFGN" to retrieve the additional responses.

The COBOL copy member "GEOSTAN" includes the record "GS-GEOGRAPHIC-INFO" which has the correct format for this function.

Input values may be zero-byte terminated strings or fixed-length with space padding. The returned values are fixed-length with space padding. The optional field "GS-INPUT-COUNTY" should be filled with spaces if it is not used.

GSFGFX

FINDS THE FIRST CITY, COUNTY, AND OR STATE CENTROID MATCH FROM THE SET OF POSSIBLE MATCHES FOUND.

Syntax

```
01 GS-GEOGRAPHIC-INFO-EX.
   05 GS-INPUT-CITY                PIC X(39).
   05 GS-INPUT-COUNTY              PIC X(39).
   05 GS-INPUT-STATE               PIC X(20).
   05 GS-OUTPUT-CITY              PIC X(39).
   05 GS-OUTPUT-COUNTY            PIC X(39).
   05 GS-OUTPUT-STATE             PIC X(20).
   05 GS-OUTPUT-LAT               PIC X(11).
   05 GS-OUTPUT-LONG              PIC X(12).
   05 GS-OUTPUT-RANK              PIC X(2).
   05 GS-OUTPUT-RESULT-CODE       PIC X(11).
   05 GS-OUTPUT-LOCATION-CODE     PIC X(5).
   05 GS-CLOSE-MATCH-FLAG         PIC X.
   05 GS-INPUT-GEO-LIB-VER-EX     PIC 9(9) BINARY.
   05 GS-OUTPUT-FIPS-CODE         PIC X(6).

*
CALL "GSFGFX" USING NAME,.
```

Arguments

GS-INPUT-CITY City Name (may be a partial string). *Input.*

GS-INPUT-COUNTY County Name (may be a partial string). Optional. *Input*.

GS-INPUT-STATE Proper state abbreviation or name for the searched state. Input.

GS-OUTPUT-CITY Output city. *Output*.

GS-OUTPUT-COUNTY Output county. *Output*.

GS-OUTPUT-STATE Output state. *Output*.

GS-OUTPUT-LAT Returned latitude of the geographic centroid. *Output*.

GS-OUTPUT-LONG Returned longitude of the geographic centroid. *Output*.

GS-OUTPUT-RANK Returned geographic rank of the city for city centroid. *Output*.

GS-OUTPUT-RESULT-CODE Result code equivalent (G1 - city centroid, G2 - country centroid, G3 – state centroid). *Output*.

GS-OUTPUT-LOCATION-CODE Location code equivalent (GM - city, GC - county, GS - state). *Output*.

GS-OUTPUT-GEO-LIB-VER-EX GeoStan version. *Input*.

GS-OUTPUT-CLOSE True indicates a close match. *Output*.

GS-OUTPUT-FIPS-CODE FIPS Code. *Output*.

Return Values

```
GS-SUCCESS
GS-ERROR
GS-NOT-FOUND
```

Prerequisites

```
GSINITWP
```

Notes

It is recommended that the user first use the Last-line lookup functions to standardize the city, county and state names. This function only performs minimal fuzzy matching on the input city and county names. The location code returned by this function is to provide users with a location code equivalent and is not retrievable using GsDataGet. It is merely provided to offer a consistent label for the type of address match that is returned and will only consist of one of the three Geographic location codes (GM – City, GC – County and GS – State).

Example

Use the following parameter area defined in the COBOL copy member named GEOSTAN filling in the fields with names beginning with GS-INPUT.

```
01 GS-GEOGRAPHIC-INFO-EX.
   05 GS-INPUT-CITY              PIC X(39).
   05 GS-INPUT-COUNTY            PIC X(39).
   05 GS-INPUT-STATE             PIC X(20).
   05 GS-OUTPUT-CITY             PIC X(39).
   05 GS-OUTPUT-COUNTY           PIC X(39).
   05 GS-OUTPUT-STATE            PIC X(20).
   05 GS-OUTPUT-LAT              PIC X(11).
   05 GS-OUTPUT-LONG             PIC X(12).
   05 GS-OUTPUT-RANK             PIC X(2).
   05 GS-OUTPUT-RESULT-CODE      PIC X(11).
   05 GS-OUTPUT-LOCATION-CODE    PIC X(5).
   05 GS-CLOSE-MATCH-FLAG        PIC X.
   05 GS-INPUT-GEO-LIB-VER-EX    PIC 9(9) BINARY.
   05 GS-OUTPUT-FIPS-CODE        PIC X(6).
```

For a COBOL coding example please see the example in the GeoStan manual for GSFGF. The call is the same except that there are more output fields returned with GSFGFX.Finds the first geographic information record with partial matching to input names.

GSFGN

FINDS THE NEXT GEOGRAPHIC INFORMATION RECORD WITH PARTIAL MATCHING TO INPUT NAMES.

Syntax

```
01 GSID PIC S9(9) BINARY.
01 GSFUNSTAT PIC S9(9) BINARY.
*
CALL "GSFGN" USING GSID, GS-GEOGRAPHIC-INFO, GSFUNSTAT.
```

Arguments

GSID	Returned by GSINITWP for the current instance of GeoStan. Input.
GSFUNSTAT	Return value for the procedure. *Output*.

Return Values

Next Geographic Information Record which has a partial match to the input.

"GSFUNSTAT" contains a value of "GS-SUCCESS" if there are one or more matching records.

See GSFGF for the remainder of the output fields.

Prerequisites

```
GSINITWP
GSFGF (Previous call with return status equal "GS-SUCCESS")
```

Notes

This procedure retrieves the next Geographic information record which matches the input city, county, or state names.   A previous call to "GSFGF" is required, with a returned status of "GS-SUCCESS".

The COBOL copy member "GEOSTAN" includes the record "GS-GEOGRAPHIC-INFO" which has the correct format for this function.

Input values may be zero-byte terminated strings or fixed-length with space padding. The returned values are fixed-length with space padding. The optional field "GS-INPUT-COUNTY" should be filled with spaces if it is not used.

GSFGNX

FINDS THE NEXT CITY, COUNTY, AND OR STATE CENTROID MATCH FROM THE SET OF POSSIBLE MATCHES FOUND.

Syntax

```
01 GS-GEOGRAPHIC-INFO-EX.
   05 GS-INPUT-CITY              PIC X(39).
   05 GS-INPUT-COUNTY            PIC X(39).
   05 GS-INPUT-STATE             PIC X(20).
   05 GS-OUTPUT-CITY             PIC X(39).
   05 GS-OUTPUT-COUNTY           PIC X(39).
   05 GS-OUTPUT-STATE            PIC X(20).
   05 GS-OUTPUT-LAT              PIC X(11).
   05 GS-OUTPUT-LONG             PIC X(12).
   05 GS-OUTPUT-RANK             PIC X(2).
   05 GS-OUTPUT-RESULT-CODE      PIC X(11).
   05 GS-OUTPUT-LOCATION-CODE    PIC X(5).
   05 GS-CLOSE-MATCH-FLAG        PIC X.
   05 GS-INPUT-GEO-LIB-VER-EX    PIC 9(9) BINARY.
   05 GS-OUTPUT-FIPS-CODE        PIC X(6).

*
CALL "GSFGFX" USING NAME,.
```

Arguments

GS-INPUT-CITY City Name (may be a partial string). *Input*.

GS-INPUT-COUNTY County Name (may be a partial string). Optional. *Input*.

GS-INPUT-STATE Proper state abbreviation or name for the searched state. Input.

GS-OUTPUT-CITY Output city. *Output*.

GS-OUTPUT-COUNTY Output county. *Output*.

GS-OUTPUT-STATE Output state. *Output*.

GS-OUTPUT-LAT Returned latitude of the geographic centroid. *Output*.

GS-OUTPUT-LONG Returned longitude of the geographic centroid. *Output*.

GS-OUTPUT-RANK Returned geographic rank of the city for city centroid. *Output*.

GS-OUTPUT-RESULT-CODE Result code equivalent (G1 - city centroid, G2 - country centroid, G3 – state centroid). *Output*.

GS-OUTPUT-LOCATION-CODE Location code equivalent (GM - city, GC - county, GS - state). *Output*.

GS-OUTPUT-GEO-LIB-VER-EX GeoStan version. *Input*.

GS-OUTPUT-CLOSE True indicates a close match. *Output*.

`GS-OUTPUT-FIPS-CODE` FIPS Code. *Output*.

Return Values

```
GS-SUCCESS
GS-ERROR
GS-NOT-FOUND
```

Prerequisites

```
GSINITWP
```

Notes

It is recommended that the user first use the Last-line lookup functions to standardize the city, county and state names. This function only performs minimal fuzzy matching on the input city and county names. The location code returned by this function is to provide users with a location code equivalent and is not retrievable using GsDataGet. It is merely provided to offer a consistent label for the type of address match that is returned and will only consist of one of the three Geographic location codes (GM – City, GC – County and GS – State).

Example

Use the following parameter area defined in the COBOL copy member named GEOSTAN filling in the fields with names beginning with GS-INPUT.

```
01 GS-GEOGRAPHIC-INFO-EX.
   05 GS-INPUT-CITY              PIC X(39).
   05 GS-INPUT-COUNTY            PIC X(39).
   05 GS-INPUT-STATE             PIC X(20).
   05 GS-OUTPUT-CITY             PIC X(39).
   05 GS-OUTPUT-COUNTY           PIC X(39).
   05 GS-OUTPUT-STATE            PIC X(20).
   05 GS-OUTPUT-LAT              PIC X(11).
   05 GS-OUTPUT-LONG             PIC X(12).
   05 GS-OUTPUT-RANK             PIC X(2).
   05 GS-OUTPUT-RESULT-CODE      PIC X(11).
   05 GS-OUTPUT-LOCATION-CODE    PIC X(5).
   05 GS-CLOSE-MATCH-FLAG        PIC X.
   05 GS-INPUT-GEO-LIB-VER-EX    PIC 9(9) BINARY.
   05 GS-OUTPUT-FIPS-CODE        PIC X(6).
```

For a COBOL coding example please see the example in the GeoStan manual for GSFGF. The call is the same except that there are more output fields returned with GSFGFX.Finds the first geographic information record with partial matching to input names.

GSGETNDB

RETURNS THE NUMBER OF LOADED DATABASES.

Syntax

```
01 GSID PIC S9(9) BINARY.
01 DBCOUNT PIC S9(9) BINARY.
*
CALL "GSGETNDB" USING GSID, DBCOUNT.
```

Arguments

GSID ID returned by **GSINITWP** for the current instance of GeoStan. *Input.*

DBCOUNT Number of GeoStan databases loaded. *Output.*

Return Values

Count of GeoStan database files loaded, or zero if GeoStan did not load any databases.

Prerequisites

GSINITWP

GSFINDWP

FINDS A MATCH WITH USER SETTABLE MATCH PREFERENCES (PROPERTIES).

Syntax

```
01 GSID                      PIC S9(9) BINARY.
01 PROPLIST                  PIC 9(9) BINARY.
01 GSFUNSTAT                 PIC 9(9) BINARY.
*
CALL “GSFINDWP” USING GSID, FIND-PROP-LIST, GSFUNSTAT.
```

Arguments

GSID ID returned by **GSINITWP** for the current instance of GeoStan. *Input.*

FIND-PROP-LIST Pointer to property list structure. *Input.*

GSFUNSTAT	Return value for the procedure. *Output*.

Return Value

GS-ERROR

GS-SUCCESS

Prerequisites

GSPLSTCR and GSPSET*

Notes

The application owns the property list, but GeoStan is the active user.

Do not destroy the property list by calling GSPLSTDE while GeoStan is still using that property list.

GSFNRANG

FINDS THE NEXT RANGE OBJECT THAT MEETS THE SEARCH CRITERIA.

Syntax

```
01 GSID             PIC S9(9) BINARY.
01 GSFUNSTAT        PIC S9(9) BINARY.
01 RANGE-HANDLE     PIC S9(9) BINARY.
*
CALL “GSFNRANG” USING GSID, RANGE-HANDLE, GSFUNSTAT.
```

Arguments

GSID	ID returned by **GSINITWP** for the current instance of GeoStan. *Input*.
GSFUNSTAT	Return value for the procedure. *Output*.
RANGE-HANDLE	Pointer to a segment/range handle. Returns a valid handle to the next range object, if there is one. *Input*, *Output*.

Return Values

GS-SUCCESS	Found a match.

`GS-NOT-FOUND`	Did not find a match.
`GS-ERROR`	An error occurred, use **`GSERRGET`** to obtain more information.

Prerequisites

`GSFFSEG` and **`GSCLEAR`**

Notes

This procedure continues to find objects that match the criteria specified in **`GSFFRANG`**.

If GeoStan finds a matching segment, you can retrieve the data for that segment using `GSHGET`.

Before each find procedure, call **`GSCLEAR`** to reset the internal buffers. If you do not reset the buffers, you may receive incorrect results with information from a previous find.

See Appendix B, "Extracting data from GSD files" for more information on the Find First and Find Next procedures.

GSFNSEG

FINDS THE NEXT SEGMENT THAT MEETS THE SEARCH CRITERIA.

Syntax

```
01 GSID              PIC S9(9) BINARY.
01 GSFUNSTAT         PIC S9(9) BINARY.
01 SEGMENT-HANDLE      PIC S9(9) BINARY.
*
CALL "GSFNSEG" USING GSID, SEGMENT-HANDLE, GSFUNSTAT.
```

Arguments

`GSID`	ID returned by **`GSINITWP`** for the current instance of GeoStan. *Input*.
`GSFUNSTAT`	Return value for the procedure. *Output*.
`RANGE-HANDLE`	Pointer to a segment/range handle. Returns a valid handle to the next range object, if there is one. *Input*, *Output*.

Return Values

`GS-SUCCESS` Found a match.

`GS-NOT-FOUND` Did not find a match.

`GS-ERROR` An error occurred, use **`GSERRGET`** to obtain more information.

Prerequisites

`GSFFSEG` `and` **`GSCLEAR`**

Notes

This procedure continues to find objects that match the criteria specified in **`GSFFSETG`**.

If GeoStan finds a matching segment, you can retrieve the data for that segment using `GSHGET`.

Before each find procedure, call **`GSCLEAR`** to reset the internal buffers. If you do not reset the buffers, you may receive incorrect results with information from a previous find.

See Appendix B, "Extracting data from GSD files" for more information on the Find First and Find Next procedures.

GSFNST

FINDS THE NEXT STREET THAT MEETS THE SEARCH CRITERIA.

Syntax

```
01 GSID              PIC S9(9) BINARY.
01 GSFUNSTAT         PIC S9(9) BINARY.
01 STREET-HANDLE     PIC S9(9) BINARY.
*
CALL “GSFNST” USING GSID, STREET-HANDLE, GSFUNSTAT.
```

Arguments

`GSID` ID returned by **`GSINITWP`** for the current instance of GeoStan. *Input*.

`GSFUNSTAT` Return value for the procedure. *Output*.

`STREET-HANDLE` Pointer to a streetrange handle. Returns a valid handle to the next street, if there is one. *Input*, *Output*.

Return Values

GS-SUCCESS Found a match.

GS-NOT-FOUND Did not find a match.

GS-ERROR An error occurred, use **GSERRGET** to obtain more information.

Prerequisites

GSFFST and GSCLEAR

Notes

This procedure continues to find objects that match the criteria specified in **GSFFST**.

If GeoStan finds a matching segment, you can retrieve the data for that segment using GSHGET.

Before each find procedure, call **GSCLEAR** to reset the internal buffers. If you do not reset the buffers, you may receive incorrect results with information from a previous find.

See Appendix B, "Extracting data from GSD files" for more information on the Find First and Find Next procedures.

GSFSTAT

Returns if the file successfully opened, and the date of the USPS data used in generating the primary GSD file.

Syntax

```
01 GSID                         PIC S9(9) BINARY.
01 STATE-CODE                   PIC X(2).
01 GSFUNSTAT                    PIC 9(4) BINARY.
01 BUILD-DATE                   PIC 9(4) BINARY.
*
CALL "GSFSTAT" USING GSID, STATE-CODE, BUILD-DATE, GSFUNSTAT.
```

Arguments

GSID ID returned by **GSINITWP** for the current instance of GeoStan. *Input*.

STATE-CODE State FIPS or abbreviation (this argument is now ignored). *Input*

GSFUNSTAT Return value for the procedure. *Output*.

BUILD-DATE Publish date of the USPS data used in the current GeoStan release. *Output*.

Return Values

Series of bit-flags. The following GS-VARIABLEs are used to test these flags:

GS-FILE-EXISTS	Opened a primary GSD file (us.gsd, use.gsd, usw.gsd).
GS-SUPP-EXISTS	Opened a supplemental GSD file. (ust.gsd, uste.gsd, uswt.gsd)
GS-STATEWIDE-EXISTS	Initialized a state-wide intersection file (us.gsi, use.gsi, usw.gsi).
GS-GSL-EXISTS	Opened eLOT and Z4Change data[a](us.gsl).
GS-EWS-MATCH-EXISTS	loaded a valid, unexpired EWS file is loaded (ews.txt).
GS-ZIPMOVE-EXISTS	Opened a ZIPMove data[b] (us.gsz).
GS-ZIP9-IDX-EXISTS	Opened a ZIP9 index file (ZIP9.gsu, ZIP9e.gsu, ZIP9w.gsu).
GS-AUXILIARY-EXISTS	Opened an auxiliary file (.gsx).
GS-ELV-EXISTS	Opened an elevation file (.elv).
GS-APN-EXISTS	Opened an APN file (.apn).

[a] eLOT data requires an additional license. However, the Z4Change data is always enabled.

[b] You must have an additional license to use this file.

Prerequisites

GSINITWP

Example

The following is an example of how to obtain the month/day/year from the build date:

```
01 GSID                    PIC S9(9) BINARY.
01 STATE-CODE              PIC X(2).
01 GSFUNSTAT               PIC S9(4) BINARY.
01 BUILD-DATE              PIC 9(4) BINARY.

01 BUILD-YEAR              PIC 9(4).
01 BUILD-MONTH             PIC 9(4).
01 BUILD-DAY               PIC 9(4).
CALL “GSFSTAT” USING GSID, STATE-CODE, BUILD-DATE, GSFUNSTAT.
```

1 Divide *BUILD-DATE* by 384 giving *BUILD-YEAR* remainder build-month.

2 Add 1990 to *BUILD-YEAR.*

3 Divide *BUILD-MONTH* by 32 giving *BUILD-MONTH* remainder *BUILD-DAY.*

4 Add 1 to *BUILD-MONTH.*

– or –

5 Compute *BUILD-YEAR* = (BUILD-DATE/384)+1990

6 Compute *BUILD-MONTH* = ((*BUILD-DATE*-((*BUILD-YEAR*-1990)*384)/32)+1

7 Compute *BUILD-DAY* = *BUILD-DATE*-((*BUILD-YEAR*-1990)*384)-((*BUILD-MONTH*-1)*32)

GSFSTATX

RETURNS INFORMATION ABOUT CURRENT GEOSTAN DATA SET AND LICENSE.

Syntax

```
01 GSID                        PIC S9(9) BINARY.
01 OPTION                      PIC S9(9) BINARY.
01 GSFUNSTAT                   PIC 9(9) BINARY.
01 OUTLEN                      PIC 9(9) BINARY.
01 OUTPUT-STRING               PIC X(USER LEN).
*
CALL “GSFSTATX” USING GSID, OPTION, OUTPUT-STRING, OUTLEN, GSFUNSTAT.
```

Arguments

GSID ID returned by **GSINITWP** for the current instance of GeoStan. *Input*.

OPTION Specific information to return. The following table includes the types of information available. *Input.*

GS-STATUS-DATATYPE-NUM	GS-SUCCESS or GS-ERROR. GeoStan places the retrieved information in the buffer. See the *Return Values* section of this procedure for the returned numeric values.
GS-STATUS-DATATYPE-STR	GS-SUCCESS or GS-ERROR. GeoStan places this information in the buffer.
GS-STATUS-DATUM-NUM	Numeric values listed in the *Return Values* section of this procedure.
GS-STATUS-DATUM-STR	The NAD used natively by the data. It does not reflect the datum currently in use by GeoStan. See **GSGDATUM** and **GSSDATUM** for further information on setting the returned NAD.

GS-STATUS-DAYS-REMAINING	• DAYS-UNLIMITED *or* the number of days remaining before the expiration of the license for unmetered licenses and unlimited licenses. • Days remaining before license expiration for metered limited licenses.
GS-STATUS-FILE-CHKSUM-NUM	Calculated value (an integer) used to check data integrity. The OUTPUT-STRING and *OUTLEN* parameters are unused. Set *OUTLEN* to 0.
GS-STATUS-GEO-RECORD-TOTAL	• 0 for unmetered licenses. • Total number or records geocoded for metered licenses
GS-STATUS-RECORDS-REMAINING	• RECORDS-UNLIMITED for unmetered licenses and metered unlimited licenses. • Number or records remaining on the license for metered limited licenses

GSFUNSTAT Return value for the procedure. *Output*.

OUTLEN Maximum size of data that GeoStan returns. If OUTLEN is shorter than the data returned by GeoStan, GeoStan truncates the data and does not generate an error. *Input*.

OUTPUT-STRING The location to store the returned data. *Output*.

Return values

The following table shows the return values for the GS-STATUS-DATATYPE-NUM mode.

GS-STATUS-DATATYPE-NUM	Data Type
0	USPS
1	TIGER
2	TANA
3	Sanborn point-level
4	Deprecated
6	NAVTEQ
7	TANA point
8	Centrus point
9	Auxiliary file

The following table shows the return values for the `GS-STATUS-DATUM-STR` mode.

GS-STATUS-DATUM-STR	Data Type
0	NAD27
1	NAD83 (WGS84 for GTD data)

Prerequisites

`GSINITWP`

GSGCRDX

RETRIEVES COORDINATES FOR THE STREET SEGMENT FOUND VIA GSSFIND.

Syntax

```
01 GSIDPIC S9(9) BINARY.
01 GSFUNSTATPIC 9(4) BINARY.
01 COORDSOCCURS 64 TIMES.
05 COORD-XPIC S9(9) BINARY.
05 COORD-YPIC S9(9) BINARY.
01 MAXPOINTSPIC 9(4) BINARY.
*
CALL “GSGCRDX” USING GSID, COORDS, MAXPOINTS, GSFUNSTAT.
```

Arguments

GSID	ID returned by **GSINIT** for the current instance of GeoStan. *Input*.
GSFUNSTAT	Return value for the procedure. *Output*.
COORDS	Array of coordinates, in x,y (longitude, latitude) order. *Output*.
MAXPOINTS	Maximum number of points that **GSGCRDX** should return; used to prevent writing past the end of *COORDS* buffer. *Input*.

Return Values

Number of points assigned to buffer.

Prerequisites

`GSSFIND`

Notes

This procedure returns an array of coordinates for the current feature found via **GSSFIND**. The maximum number that GeoStan can return is 64 coordinate pairs, each pair consisting of two long integers.

GeoStan scales coordinate pairs to integers with four decimal digits of precision. Thus, GeoStan returns a point at (-98.3, 29.7) as (983000, 297000). This is a different scale from that expected by Spatial+ and similar GIS applications, which typically express coordinates in millionths of degrees. You may need to scale coordinates obtained with this procedure before using them as input to other software libraries or applications.

GSGLIBV

RETURNS THE CURRENT VERSION OF THE GEOSTAN LIBRARY.

Syntax

```
01 GSFUNSTAT       PIC S9(9) BINARY.
*
CALL “GSGLIBV” USING GSFUNSTAT.
```

Arguments

GSFUNSTAT	Return value for the procedure. *Output*.

Return values

- Low Byte = Major version number.
- High Byte = Minor version number.

Prerequisites

GSINITWP

Notes

In general, the major version number changes whenever PBBI adds a new API features, or when the data structures in the GeoStan data files change.

Minor version number changes for each release of GeoStan.

Example

The best way to extract the high and low bytes is to subdefine *GSFUNSTAT* as follows:

```
01 GSFUNSTAT PIC S9(9) BINARY.
01 GS-MINOR-VERSION PIC 9(2) BINARY.
01 GS-MAJOR-VERSION PIC 9(2) BINARY.

   CALL "GSGLIBV" USING GSFUNSTAT.
   DIVIDE GSFUNSTAT BY 256 GIVING GS-MINOR-VERSION
                            REMAINDER MAJOR-VERSION
```

GSHGCRDX

RETRIEVES THE COORDINATES FOR A STREET SEGMENT OBJECT FOUND VIA GSFFSEG AND GSFNRANG.

Syntax

```
01 GSID          PIC S9(9) BINARY.
01 GSFUNSTAT     PIC S9(9) BINARY.
01 SEGMENT-HANDLEPIC S9(9) BINARY.
01 MAXPOINTS     PIC  9(4) BINARY.
01 COORDS        OCCURS <nnn> TIMES.
05 COORD-X   PIC S9(9) BINARY.
05 CORRD-Y   PIC S9(9) BINARY.
*
CALL “GSHGCRDX” USING GSID, SEGMENT-HANDLE, COORDS, MAXPOINTS, GSFUNSTAT.
```

Arguments

GSID ID returned by **GSINIT** for the current instance of GeoStan. *Input*.

GSFUNSTAT return value for the procedure. *Output*.

SEGMENT-HANDLE Handle of the segment object for the returned coordinates. *Input*.

COORDS Array of coordinates, in x,y (longitude, latitude) order. *Output*.

MAXPOINTS Maximum number of points that **GSGCRDX** returns; used to prevent writing past the end of *COORDS* buffer. *Input*.

Return Values

Number of points assigned to the buffer.

Prerequisites

`GSFFSEG` or `GSFNRANG`

Notes

This procedure returns an array of coordinates for the segment identified in *SEGMENT-HANDLE*.

GSHGET

RETRIEVES DATA FOR OBJECTS FOUND VIA GSSFFSG AND GSFNRANG.

Syntax

```
01 GSID                        PIC S9(9) BINARY.
01 GSFUNSTAT                   PIC S9(9) BINARY.
01 RANGE-HANDLE                PIC S9(9) BINARY.
01 GSOPTIONS                   PIC  9(9) BINARY.
01 OUTPUT-STRING               PIC X(USER LEN)
01 OUTLEN                      PIC S9(4) BINARY.

CALL “GSHGET” USING GSID, GSOPTIONS, RANGE-HANDLE,
OUTPUT-STRING, OUTLEN, GSFUNSTAT.
```

Arguments

GSID ID returned by **GSINITWP** for the current instance of GeoStan. *Input*.

GSFUNSTAT Return value for the procedure. *Output*.

RANGE-HANDLE Pointer to the current range handle. *Input*.

GSOPTIONS Variable for the argument you want to retrieve. *Input*.

OUTPUT-STRING Location to store the returned data. *Output*.

OUTLEN Maximum size of the data that GeoStan returns. If *OUTLEN* is shorter than the data returned by GeoStan, GeoStan truncates the data and does not generate an error. *Input*.

Return Values

GS-SUCCESS

GS-ERROR

Prerequisites

`GSFFSEG` or `GSFNRANG`

Notes

This procedure retrieves data from the geocode buffer for a given range handle. If you have a street or segment handle, you must convert the handle to a range handle before you can use this procedure.

GSINITWP

INITIALIZES GEOSTAN USING PROPERTIES.

Syntax

```
01 INIT-PROP-LIST    Defined in GEOSTAN copy member.
01 STATUS-PROP-LIST       Defined in GEOSTAN copy member.
01 GSID              PIC S9(9) BINARY.
CALL 'GSINITWP' USING INIT-PROP-LIST,STATUS-PROP-LIST, GS-ID.
IF GS-ID EQUAL ZERO DISPLAY '** ERROR ** GEOSTAN FAILED TO INITIALIZE’ PERFORM
GET-ERROR-MSG
```

Arguments

*INIT-PROP-LIST, STATUS-PROP-LIST*Pointer to property list structure. *Input.*

GSID ID returned by **GSINITWP** for the current instance of GeoStan. *Input.*

Return Values

GS-SUCCESS

GS-ERROR

Prerequisites

`GSPLSTCR` and `GSPSET*`

Notes

Initializes GeoStan using a property list. Upon return, this function utilized the properties, but left the property list intact. The application owns the property lists. The status property list is guaranteed to contain properties for all defined status properties with their values properly set.

GSINITWP can be used to initialize GeoStan, DPV, and LACSLink with a single call (if the appropriate initialization properties are in the init list). When this function successfully completes, it populates the GS-INIT-GEOSTAN-ID property in the property list referred to by the pInitProps parameter with the actual GeoStan ID.

If you invoke this function with the GeoStan ID property pre-set to a valid GeoStan ID, it will not attempt to re-initialize GeoStan. This is how GSINITWP can be used to initialize DPV and/or LACSLink after GeoStan has already been initialized with a previous call to GSINITWP. If you intend to reuse the same initialization property list to initialize GeoStan several times you must reset the GS-INIT-GEOSTAN-ID property back to zero, or completely remove the property from the list. This informs GeoStan that you want a new GeoStan instance when you call GSINITWP.

These initialization properties are required for GsInitWithProps to function correctly:

- GS-INIT-LICFILENAME
- GS-INIT-DATAPATH
- GS-INIT-PASSWORD

These initialization properties are recommended, but not required:

- GS-INIT-Z4FILE (required for zip centroid matching).
- GS-INIT-OPTIONS_Z9_CODE (required for zip centroid matching).
- GS-INIT-CACHESIZE (to improve GeoStan performance).
- GS-INIT-OPTIONS-ADDR-CODE (required for address matching).
- GS-INIT-OPTIONS-SPATIAL-QUERY (required for reverse geocoding and the MBR-related functions).

All other available initialization properties are entirely optional.

GSLACCLS

CLEARS THE LACS^{Link} GLOBAL STATISTICS.

Syntax

```
01 GSID        PIC 9(9) BINARY.
01 RETURN-CODE PIC 9(9) BINARY

CALL "GSLACCLS" USING GSID, RETURN-CODE.
```

Arguments

GSID ID returned by **GSINITWP** for the current instance of GeoStan. *Input.*

RETURN-CODE Size of the **GSLACGCS** data structure. *Input.*

Return Values

GS-SUCCESS

GS-ERROR Call **GSERRGTX** for more information.

Prerequisites

GSLACINR

GSLACFPD

FORMATS A LACS^{Link} FALSE POSITIVE DETAIL RECORD FROM GSLACGFH.

Syntax

```
01 GSID PIC 9(9) BINARY.
01 GS-FALSE-POS-DETAIL-DATA
01 RETURN-CODE PIC 9(9) BINARY.
01 HEADER      PIC X(len).
01 HEADER-SIZE PIC 9(9) BINARY VALUE len.
CALL "GSLACFPD" USING GSID, GS-FALSE-POS-DETAIL-DATA, GS-FPHD-SIZE, HEADER,
HEADER-SIZE, RETURN-CODE
```

Arguments

GS ID returned by **GSINITWP** for the current instance of GeoStan. *Input.*

GS-FALSE-POS-DETAIL-DATA Pointer to the LACS^{Link} report data structure. This value is completed by **GSLACGFD**. *Input.*

RETURN-CODE Size of the LACS^{Link} report data structure. *Input.*

HEADER Buffer containing the LACS^{Link} false positive header after **GSLACFPD** successfully completes. When the GeoStan application writes the false positive report, it writes this buffer to the last line of the file. *Output.*

HEADER-SIZE Length of the detail buffer. *Input.*

Return Values

GS-SUCCESS

GS-ERROR Call **GSERRGTX** for more information.

GS-WARNING Call **GSERRGTX** for more information.

Prerequisites

GSLACGFD

GSLACFPH

FORMATS A LACS^{Link} FALSE POSITIVE HEADER RECORD WITH DATA FROM GSLACGFH.

Syntax

```
01 GSID PIC 9(9) BINARY.
01 PDATAPIC ????
01 RETURN-CODE PIC 9(9) BINARY.
01 HEADER      PIC X(len).
01 HEADER-SIZE PIC 9(9) BINARY VALUE len.

CALL "GSLACFPH" USING GSID, GS-FALSE-POS-HEADER-DATA, GS-FPHD-SIZE, HEADER,
HEADER-SIZE, RETURN-CODE.
```

Arguments

GSID ID returned by **GSINITWP** for the current instance of GeoStan. *Input.*

GS-FALSE-POS-HEADER-DATA Pointer to the LACS^{Link} report data structure. **GSLACGFD** completes this value. *Input.*

RETURN-CODE Size of **GSLACFPH** data structure. *Input.*

HEADER Buffer containing the LACS^{Link} false positive header after **GSLACFPH** successfully completes. When the GeoStan application writes the false positive report, it writes this buffer to the first line of the file. *Output.*

HEADER-SIZE Length of the header buffer. *Input.*

Return Values

GS-SUCCESS

GS-ERROR Call **GSERRGTX** for more information.

GS-WARNING Call **GSERRGTX** for more information.

Prerequisites

```
GSLACGFD
GS-DPV-FALSE-POS == “Y”
```

GSLACGCS

OBTAINS THE COMPLETE LACS^{Link} STATISTICS SINCE THE APPLICATION INITIALIZED LACS^{Link}.

Syntax

```
01 GSID       PIC 9(9) BINARY.
01 GS-LACS-COMPLETE-STATS
01 RETURN-CODE PIC 9(9) BINARY

CALL "GSLACGCS" USING GSID, GS-LACS-COMPLETE-STATS, GS-LCS-SIZE, RETURN-CODE.
```

Arguments

GSID ID returned by **GSINITWP** for the current instance of GeoStan. *Input.*

GS-LACS-COMPLETE-STATS Retrieves LACS^{Link} statistics since the application initialized LACS^{Link}. This structure contains the following:

GS-LCS-NTOTAL-AMATCHES Total number of records matched through LACS^{Link}. *Output.*

GS-LCS-NTOTAL-PROCESSED Total number of records processed through LACS^{Link}. *Output.*

GS-LCS-NTOTAL-00MATCHES Total number or records not matched through LACS^{Link}. *Output.*

GS-LCS-NTOTAL-09MATCHES Total number of records converted through LACS^{Link}, but no new address provided. *Output.*

GS-LCS-NTOTAL-14MATCHES Total number of records converted through LACS^{Link}. *Output.*

GS-LCS-NTOTAL-93MATCHES Total number of records matched through LACS^{Link} where GeoStan drops the unit number. *Output.*

RETURN-CODE Size of the **GSLACGCS** data structure. *Input.*

Return Values

`GS-SUCCESS`

`GS-ERROR` Call **GSERRGTX** for more information.

`GS-WARNING` Call **GSERRGTX** for more information.

Prerequisites

`GSLACINR`

GSLACGFD

RETRIEVES THE DETAIL RECORD FOR A LACS^{Link} FALSE POSITIVE REPORT.

Syntax

```
01 GSID              PIC 9(9) BINARY.
01 GS-FALSE-POS-DETAIL-DATA
01 RETURN-CODE       PIC 9(9) BINARY.

CALL "GSLACGFD" USING GSID, GS-FALSE-POS-DETAIL-DATA, GS-FPDD-SIZE, RETURN-
CODE.
```

Arguments

GSID ID returned by **GSINITWP** for the current instance of GeoStan. *Input.*

PDATA* Retrieves LACS^{Link} statistics from the detail record for false positive address matches using the data passed in **GSLACGFH. This structure contains the following:

GS-FPDD-ADDRESS-PRIMARY-NUMBER House number. *Output.*

GS-FPDD-ADDRESS-SECONDARY-ABBREVIATION Unit type (APT, SUITE, LOT). *Output.*

GS-FPDD-ADDRESS-SECONDARY-NUMBER Unit number. *Output.*

GS-FPDD-MATCHED-ZIP-CODE ZIP Code. *Output.*

GS-FPDD-MATCHED-PLUS ZIP Code extension. *Output.*

GS-FPDD-POST-DIRECTIONAL Street name postdirectional (N, S, E, W). *Output.*

GS-FPDD-STREET-NAME Name of the street. *Output.*

GS-FPDD-STREET-PREDIR Street name predirectional (N, S, E, W). *Output*.

GS-FPDD-SUFFIX-ABBREVIATION Street type (AVE, ST, RD). *Output*.

FILLER Reserved for future implementation. *Output*.

return-code Size of the **GSLACGFH** data structure. *Input*.

Return Values

GS-SUCCESS

GS-ERROR Call **GSERRGTX** for more information.

GS-WARNING Call **GSERRGTX** for more information.

Prerequisites

GSLACINR
GS-LACSLINK-IND == “F”

GSLACGFH

RETRIEVES LACS^{Link} STATISTICS FOR THE HEADER RECORD FOR A LACS^{Link} FALSE POSITIVE REPORT.

Syntax

```
01 GSID             PIC 9(9) BINARY.
01 GS-FALSE-POS-HEADER-DATA
01 RETURN-CODE      PIC 9(9) BINARY.

CALL "GSLACFPH" USING GSID, GS-FALSE-POS-HEADER-DATA, GS-FPHD-SIZE, RETURN-
CODE.
```

Arguments

GSID ID returned by **GSINITWP** for the current instance of GeoStan. *Input.*

GS-FALSE-POS-HEADER-DATA Retrieves LACS^{Link} statistics from the header record for false positive address matches using the data passed in **GSLACGFD**. This structure contains the following:

GS-FPDD-MAILERS-ADDRESS-LINE Address of the mailer. *Input*.

GS-FPDD-MAILERS-CITY-NAME City of the mailer. *Input*.

GS-FPDD-MAILERS-COMPANY-NAME Name of the mailer. *Input*.

GS-FPDD-MAILERS-STATE-NAME State of the mailer. *Input*.

GS-FPHD-MAILERS9-DIGIT-ZIP ZIP Code of the mailer. *Input*.

GS-FPDD-NTOTAL-PROCESSED Number of records processed through LACS$^{\text{Link}}$. *Output*.

GS-FPDD-NTOTAL-MATCHED Number of records confirmed through LACS$^{\text{Link}}$. *Output*.

GS-FPDD-NUMBER-FALSE-POS Number of LACS$^{\text{Link}}$ false positives found. *Output*.

GS-FPDD-NUMBER-ZIP-ON-FILE Number of distinct ZIP Codes processed through LACS$^{\text{Link}}$. *Output*.

GS-FPDD-TOTAL-ZIP4-MATCHED Number of records that have matched with ZIP + 4. *Output*.

RETURN-CODE Size of the **GSLACGFD** data structure. *Input*

Return Values

GS-SUCCESS

GS-ERROR Call **GSERRGTX** for more information.

GS-WARNING Call **GSERRGTX** for more information.

Prerequisites

```
GSLACINR
GS-LACSLINK-IND == “F”
```

GSMGET

RETURNS THE ADDRESS ELEMENTS FOR THE MATCH CANDIDATE ITEM SPECIFIED.

Syntax

```
01 GSID                        PIC S9(9) BINARY.
01 GSFUNSTAT                   PIC S9(9) BINARY.
01 OUTPUT-STRING               PIC X(USER LEN).
01 OUTLEN                      PIC S9(4) BINARY.
01 GSOPTIONS                   PIC 9(9) BINARY.
01 INDEX                       PIC 9(4) BINARY.
```

```
*
CALL “GSMGET” USING GSID, GSOPTIONS, INDEX, OUTPUT-STRING, OUTLEN, GSFUNSTAT.
```

Arguments

GSID ID returned by **GSINITWP** for the current instance of GeoStan. *Input*.

GSFUNSTAT Return value for the procedure. *Output*.

NOTE: For valid Variables, see in“Common variables for storing and retrieving data” on page 248 .

GSOPTIONS Variable for the argument you want to retrieve. *Input*.

INDEX Entry number (0-based) of the possible match. *Input*.

OUTPUT-STRING Location to store the returned data. *Output*.

OUTLEN Maximum size of the data GeoStan returns. If *OUTLEN* is shorter than the data returned by GeoStan, GeoStan truncates the data and does not generate an error. *Input*.

Return Values

GS-SUCCESS

GS-ERROR

Prerequisites

GSNMULT

Notes

This procedure retrieves data from the GeoStan buffer for match candidatees. GeoStan indicates a match candidate as the **GSSFINDWP** GS-ADDRESS-NOT-RESOLVED return code. It is important to first test for an intersection match, since the variables are different for retrieving intersection and non-intersection matches.

When using any street name variable (GS-NAME, GS-PREDIR, GS-POSTDIR, GS-TYPE) the additional modifier GS-ALIAS is available to request specific alias information, rather than preferred name information. For example, in Boulder, CO, Wallstreet is an alias for Fourmile Canyon. The address 123 Wallstreet, Boulder CO 80301 matches to 123 Fourmile Canyon Dr.

```
MOVE GS-NAME TO GSOPTIONS.
CALL "GSMGET" USING GSID, GSOPTIONS, ...
```

Returns “FOURMILE CANYON” in *OUTPUT-STRING*

```
MOVE GS-ALIAS TO GSOPTIONS.
ADD GS-NAME TO GSOPTIONS.
CALL "GSMGET" USING GSID, GSOPTIONS, ...
```

Returns “WALLSTREET” in *OUTPUT-STRING.*

If you use GS-ALIAS with an variable that does not return alias information (such as GS-ZIP), GeoStan returns the information in the normal format. If GS-IS-ALIAS returns A07, you can only get information based on the returned address, not the alias.

GSMGH

RETURNS THE RANGE HANDLE FOR THE MATCH CANDIDATE ITEM SPECIFIED.

Syntax

```
01 GSID              PIC S9(9) BINARY.
01 GSFUNSTAT         PIC S9(9) BINARY.
01 INDEX             PIC 9(4) BINARY.
01 RANGE-HANDLE      PIC S9(9) BINARY.
*
CALL “GSMGH” USING GSID, INDEX, RANGE-HANDLE, GSFUNSTAT.
```

Arguments

GSID ID returned by **GSINITWP** for the current instance of GeoStan. *Input*.

GSFUNSTAT Return value for the procedure. *Output*.

INDEX Entry number (0-based) of the possible match. *Input*.

RANGE-HANDLE Range handle for the possible match entry. *Output*.

Return Values

GS-SUCCESS

GS-ERROR

Prerequisites

GSNMULT

Notes

This procedure can to extract more information about a possible match, then **GSMGET**. It retrieves the range handle for the possible match indicated by the entry argument. Once you have the correct range handle, you can retrieve information by using **GSHGET**. For a list of elements returned by **GSMGET** or **GSHGET**, see "Common variables for storing and retrieving data" on page 248.

GSNMULT

RETURNS THE NUMBER OF MATCH CANDIDATEES FOUND.

Syntax

```
01 GSID         PIC S9(9) BINARY.
01 GSFUNSTAT    PIC S9(9) BINARY.
*
CALL "GSNMULT" USING GSID, GSFUNSTAT.
```

Arguments

GSID	ID returned by **GSINITWP** for the current instance of GeoStan. *Input*.
GSFUNSTAT	Return value for the procedure. *Output*.

Return Values

A positive value indicates the number of match candidatees. **GS-ERROR** indicates an error.

Prerequisites

GSSFINDWP

Notes

This procedure returns the number of match candidatees found. Use **GSMGET** to retrieve the actual address elements for each possible match. A return code from **GSSFINDWP** indicates a possible match.

GSPFIND

FINDS A PROPERTY IN A PROPERTY LIST.

Syntax

```
01 FIND-PROP-LIST     PIC S9(9) BINARY.
01 PROPENUM           PIC S9(9) BINARY.
01 GSFUNSTAT         PIC S9(9) BINARY.
*
CALL “GSPFIND” USING FIND-PROP-LIST, PROPENUM, GSFUNSTAT.
```

Arguments

*FIND-PROP-LIST*Pointer to property list structure. *Input.*

PROPENUM The ID of the property to find. *Input.*

GSFUNSTAT Return value for the procedure. *Output*.

Return Values

GS-SUCCESS

GS-ERROR

Prerequisites

GSPLSTCR

Notes

GS-SUCCESS returns if the property is found in the property list. GS-ERROR returns if the property is not found.

GSPFIRST

SETS PROPERTY ITERATOR TO FIRST PROPERTY.

Syntax

```
01 FIND-PROP-LIST     PIC S9(9) BINARY.
01 GSFUNSTAT          PIC S9(9) BINARY.
*
CALL “GSPFIRST” USING FIND-PROP-LIST, GSFUNSTAT.
```

Arguments

*FIND-PROP-LIST*Pointer to property list structure. *Input*.

GSFUNSTAT Return value for the procedure. *Output*.

Return Values

GS-SUCCESS

GS-ERROR

Prerequisites

GSPLSTCR

Notes

Prior to iterating through the properties in a property list, this procedure sets the property list iterator to the beginning of the list.

GSPFNEXT

ITERATES TO THE NEXT SEQUENTIAL PROPERTY IN THE PROPERTY LIST.

Syntax

```
01 FIND-PROP-LIST      PIC S9(9) BINARY.
01 PROPENUM       PIC S9(9) BINARY.
01 PROPVALUE      PIC S9(9) BINARY.
01 GSFUNSTAT      PIC S9(9) BINARY.
*
CALL “GSPFNEXT” USING FIND-PROP-LIST, PROPENUM, PROPVALUE, GSFUNSTAT.
```

Arguments

*FIND-PROP-LIST*Pointer to property list structure. *Input*.

PROPENUM The ID of the property to find. *Input*.

PROPVALUE Pointer to union of property values. *Output*.

GSFUNSTAT Return value for the procedure. *Output*.

Return Values

GS-SUCCESS

GS-ERROR

Prerequisites

GSPLSTCR

Notes

GeoStan only populates one of the PropValue union members. The populated union member depends on the property output type. If there is no “next” entry, GeoStan returns an error.

GSPGETB

RETRIEVES A BOOLEAN PROPERTY.

Syntax

```
01 STATUS-PROP-LIST     PIC S9(9) BINARY.
01 PROPENUM         PIC S9(9) BINARY.
01 QBOOL            PIC S9(9) BINARY.
01 GSFUNSTAT        PIC S9(9) BINARY.
*
CALL “GSPGETB” USING STATUS-PROP-LIST, PROPENUM, QBOOL, GSFUNSTAT.
```

Arguments

*STATUS-PROP-LIST*Pointer to property list structure defined in copy member GEOSTAN. *Input.*

PROPENUM The ID of the property to find. *Input.*

QBOOL The boolean value of the property. *Output.*

GSFUNSTAT Return value for the procedure. *Output.*

Return Values

GS-SUCCESS

GS-ERROR

Prerequisites

GSPLSTCR

GSPGETD

RETRIEVES A DOUBLE PROPERTY.

Syntax

```
01 STATUS-PROP-LIST     PIC S9(9) BINARY.
01 PROPENUM             PIC S9(9) BINARY.
01 DOUBLE               PIC S9(9) BINARY.
01 GSFUNSTAT            PIC S9(9) BINARY.
*
CALL “GSPGETD” USING STATUS-PROP-LIST, PROPENUM, DOUBLE, GSFUNSTAT.
```

Arguments

STATUS-PROP-LIST Pointer to property list structure defined in copy member GEOSTAN. *Input.*

PROPENUM The ID of the property to find. *Input.*

DOUBLE The double value of the property. *Output.*

GSFUNSTAT Return value for the procedure. *Output.*

Return Values

GS-SUCCESS

GS-ERROR

Prerequisites

GSPLSTCR

GSPGETL

RETRIEVES AN INTEGER PROPERTY.

Syntax

```
01 STATUS-PROP-LIST     PIC S9(9) BINARY.
01 PROPENUM             PIC S9(9) BINARY.
01 INTL      PIC S9(9) BINARY.
01 GSFUNSTAT            PIC S9(9) BINARY.
*
CALL “GSPGETL” USING STATUS-PROP-LIST, PROPENUM, INTL, GSFUNSTAT.
```

Arguments

*STATUS-PROP-LIST*Pointer to property list structure. *Input.*

PROPENUM The ID of the property to find. *Input.*

INTL Long value. *Output.*

GSFUNSTAT Return value for the procedure. *Output*.

Return Values

GS-SUCCESS

GS-ERROR

Prerequisites

GSPLSTCR

GSPGETST

RETRIEVES A STRING PROPERTY.

Syntax

```
01 STATUS-PROP-LIST     PIC S9(9) BINARY.
01 PROPENUM             PIC S9(9) BINARY.
01 INTL          PIC S9(9) BINARY.
01 GSFUNSTAT            PIC S9(9) BINARY.
*
CALL “GSPGETST” USING STATUS-PROP-LIST, PROPENUM, PROPVALUE, GSFUNSTAT.
```

Arguments

*STATUS-PROP-LIST*Pointer to property list structure. *Input.*

PROPENUM	The ID of the property to find. *Input.*
INTL	Long value. *Output.*
GSFUNSTAT	Return value for the procedure. *Output.*

Return Values

GS-SUCCESS

GS-ERROR

Prerequisites

GSPLSTCR

GSPGINFO

RETRIEVES INFORMATION ABOUT A PROPERTY.

Syntax

```
01 PROPENUM     PIC S9(9) BINARY.
01 INT          PIC S9(9) BINARY.
01 PROPLISTTYPE PIC S9(9) BINARY.
01 PSTR         PIC S9(9) BINARY.
01 GSFUNSTAT    PIC S9(9) BINARY.
*
CALL “GSPGINFO” USING PROPENUM, INT, PROPLISTTYPE, PSTR, GSFUNSTAT.
```

Arguments

PROPENUM	The ID of the property to find. *Input.*
INT	*Output.*
PROPLISTTYPE	Pointer to property list structure. *Input.*
PSTR	*Output.*
GSFUNSTAT	Return value for the procedure. *Output.*

Return Values

```
GS-SUCCESS

GS-ERROR
```

Prerequisites

```
GSPLSTCR
```

GSPGSTRL

Retrieves the length of the string value of a property.

Syntax

```
01 PROPLIST     PIC S9(9) BINARY.
01 PROPENUM     PIC S9(9) BINARY.
01 INTLU        PIC S9(9) BINARY.
01 GSFUNSTAT    PIC S9(9) BINARY.
*
CALL “GSPGSTRL” USING PROPLIST, PROPENUM, INTLU, GSFUNSTAT.
```

Arguments

PROPLIST Pointer to property list structure. *Input.*

PROPENUM The ID of the property to find. *Input.*

INTLU Required string buffer length. *Output.*

GSFUNSTAT Return value for the procedure. *Output*.

Return Values

```
GS-SUCCESS

GS-ERROR
```

Prerequisites

```
GSPLSTCR
```

GSPLSTCR

CREATES AND INITIALIZES A PROPERTY LIST FOR GEOSTAN INITIALIZATION.

Syntax

```
01 INIT-PROP-LIST     PIC S9(9) BINARY.
01 GS-INIT-PROP-LIST-TYPE PIC S9(9) BINARY.
01 GSFUNSTAT    PIC S9(9) BINARY.
*
CALL “GSPLSTRCR” USING INIT-PROP-LIST, GS-INIT-PROP-LIST-TYPE, GSFUNSTAT.
```

Arguments

`INIT-PROP-LIST`Pointer to property list structure defined in copy member GEOSTAN. *Input and Output.*

`GS-INIT-PROP-LIST-TYPE`Pointer to property list structure defined in copy member GEOSTAN. *Input.*

`GSFUNSTAT` Return value for the procedure. *Output.*

Return Values

`GS-SUCCESS`

`GS-ERROR`

Notes

This procedure creates/initializes a property list for use in GSINITWP or GSFINDWP. Any property list created by this function needs to eventually be destroyed with GSPLSTDE.

GSPLSTRD

DESTROYS A PROPERTY LIST.

Syntax

```
01 PROPLIST     PIC S9(9) BINARY.
01 GSFUNSTAT    PIC S9(9) BINARY.
*
CALL “GSPLSTRD” USING PROPLIST, GSFUNSTAT.
```

Arguments

PROPLIST Pointer to property list structure. *Input and Output*.

GSFUNSTAT Return value for the procedure. *Output*.

Return Values

```
GS-SUCCESS

GS-ERROR
```

Prerequisites

```
GSPLSTCR
```

Notes

Do not destroy any property list that GeoStan is actively using. Any list created by GSPLSTCR must eventually be destroyed with GSPLSTRD.

GSPLSTGC

Retrieves the number of properties in a property list.

Syntax

```
01 PROPLIST     PIC S9(9) BINARY.
01 INTL         PIC S9(9) BINARY.
01 GSFUNSTAT    PIC S9(9) BINARY.
*
CALL “GSPLSTGC” USING PROPLIST, INTL, GSFUNSTAT.
```

Arguments

PROPLIST Pointer to property list structure. *Input and Output.*

INTL Returned count. *Output.*

GSFUNSTAT Return value for the procedure. *Output*.

Return Values

`GS-SUCCESS`

`GS-ERROR`

Prerequisites

`GSPLSTCR`

GSPLSTWR

WRITES THE PROPERTY LIST TO A FILE OR A CHARACTER STRING.

Syntax

```
01 PROPLIST     PIC S9(9) BINARY.
01 GS-CONST-STR PIC S9(9) BINARY.
01 PSTR         PIC S9(9) BINARY.
01 INTSU         PIC S9(9) BINARY
01 GSFUNSTAT    PIC S9(9) BINARY.
*
CALL “GSPSETAS” USING PROPLIST, GS-CONST-STR, PSTR, INTSU, GSFUNSTAT.
```

Arguments

PROPLIST Pointer to property list structure. *Input and Output.*

gs-const-str The string name of the property to set. *Input.*

PSTR *Output.*

INTSU *Output.*

GSFUNSTAT Return value for the procedure. *Output.*

Return Values

`GS-SUCCESS`

`GS-ERROR`

Prerequisites

`GSPLSTCR`

Notes

If the outfile argument value is “stdout”, then this function writes the properties to standard out. If the outfiles argument is NULL, then this function writes the properties to the pBuffer string.

GSPREMOV

REMOVES A PROPERTY FROM THE PROPERTY LIST.

Syntax

```
01 PROPLIST     PIC S9(9) BINARY.
01 PROPENUM     PIC S9(9) BINARY.
01 GSFUNSTAT    PIC S9(9) BINARY.
*
CALL “GSPLSTRDE” USING PROPLIST, PROPENUM, GSFUNSTAT.
```

Arguments

PROPLIST	Pointer to property list structure. *Input and Output.*
PROPENUM	Property ID. *Input.*
GSFUNSTAT	Return value for the procedure. *Output.*

Return Values

GS-SUCCESS

GS-ERROR

Prerequisites

GSPLSTCR

GSPRESET

RESETS THE PROPERTY LIST TO ITS DEFAULT STATE.

Syntax

```
01 PROPLIST     PIC S9(9) BINARY.
```

```
01 GSFUNSTAT    PIC S9(9) BINARY.
*
CALL “GSPRESET” USING PROPLIST, GSFUNSTAT.
```

Arguments

PROPLIST Pointer to property list structure. *Input and Output.*

GSFUNSTAT Return value for the procedure. *Output.*

Return Values

GS-SUCCESS

GS-ERROR

Prerequisites

GSPLSTCR

Notes

Depending on the type of property list, this might mean “empty” (init and status properties) or refill it with some needed defaults (find properties).

GSPSETAS

SETS A PROPERTY WHERE INPUT NAME AND VALUE ARE BOTH STRINGS.

NOTE: If you intend to use the same Find property settings for all your address records, try to construct your application to set the Find properties once before processing. It is unnecessary to reset the properties again to the same values for each address record. However, if required, you can change Find property values for individual searches. Performance may be negatively affected by resetting for individual record searches. Reset Find properties between address searches only if changing the Find property value is necessary.

Syntax

```
01 PROPLIST     PIC S9(9) BINARY.
01 GS-CONST-STR PIC S9(9) BINARY.
01 PROPENUM     PIC S9(9) BINARY.
01 GS-CONST-STR PIC S9(9) BINARY.
01 GSFUNSTAT    PIC S9(9) BINARY.
*
CALL “GSPSETAS” USING PROPLIST, GS-CONST-STR, GS-CONST-STR, GSFUNSTAT.
```

Arguments

PROPLIST	Pointer to property list structure. *Input and Output.*
gs-const-str	The string name of the property to set. *Input.*
PROPENUM	Property ID. *Input.*
gs-const-str	The string value of the property. *Input.*
GSFUNSTAT	Return value for the procedure. *Output*.

Return Values

```
GS-SUCCESS

GS-ERROR
```

Prerequisites

```
GSPLSTCR
```

Notes

This procedure performs the conversion to correct property variable ID and property value type. If the input property name is a NULL pointer, the property ID is for use in identifying the property instead. If both property name and property ID are present, preference is given to the property name, ignoring the property ID.

GSPSETB

SETS A BOOLEAN PROPERTY.

NOTE: If you intend to use the same Find property settings for all your address records, try to construct your application to set the Find properties once before processing. It is unnecessary to reset the properties again to the same values for each address record. However, if required, you can change Find property values for individual searches. Performance may be negatively affected by resetting for individual record searches. Reset Find properties between address searches only if changing the Find property value is necessary.

Syntax

```
01 FIND-PROP-LIST     PIC S9(9) BINARY.
01 PROPENUM           PIC S9(9) BINARY.
01 QBOOL              PIC S9(9) BINARY.
```

```
01 GSFUNSTAT          PIC S9(9) BINARY.
*
CALL “GSPSETB” USING FIND-PROP-LIST, PROPENUM, QBOOL, GSFUNSTAT.
```

Arguments

FIND-PROP-LIST Pointer to property list structure defined in copy member GEOSTAN. *Input and Output.*

PROPENUM Property ID. *Input.*

QBOOL Boolean value. *Input.*

GSFUNSTAT Return value for the procedure. *Output.*

Return Values

GS-SUCCESS

GS-ERROR

Prerequisites

GSPLSTCR

GSPSETD

SETS A DOUBLE PROPERTY.

Syntax

```
01 STATUS-PROP-LIST     PIC S9(9) BINARY.
01 PROPENUM             PIC S9(9) BINARY.
01 DOUBLE               PIC S9(9) BINARY.
01 GSFUNSTAT            PIC S9(9) BINARY.
*
CALL “GSPSETD” USING STATUS-PROP-LIST, PROPENUM, DOUBLE, GSFUNSTAT.
```

Arguments

*STATUS-PROP-LIST*Pointer to property list structure defined in copy member GEOSTAN. *Input.*

PROPENUM The ID of the property to find. *Input.*

DOUBLE	The double value of the property. *Output.*
GSFUNSTAT	Return value for the procedure. *Output*.

Return Values

```
GS-SUCCESS

GS-ERROR
```

Prerequisites

```
GSPLSTCR
```

GSPSETL

SETS AN INTEGER PROPERTY.

NOTE: If you intend to use the same Find property settings for all your address records, try to construct your application to set the Find properties once before processing. It is unnecessary to reset the properties again to the same values for each address record. However, if required, you can change Find property values for individual searches. Performance may be negatively affected by resetting for individual record searches. Reset Find properties between address searches only if changing the Find property value is necessary.

Syntax

```
01 PROPLIST     PIC S9(9) BINARY.
01 PROPENUM     PIC S9(9) BINARY.
01 INTL         PIC S9(9) BINARY.
01 GSFUNSTAT    PIC S9(9) BINARY.
*
CALL “GSPSETL” USING PROPLIST, PROPENUM, INTL, GSFUNSTAT.
```

Arguments

PROPLIST	Pointer to property list structure. *Input and Output.*
PROPENUM	Property ID. *Input.*
INTL	Long value. *Input and Output.*
GSFUNSTAT	Return value for the procedure. *Output*.

Return Values

```
GS-SUCCESS

GS-ERROR
```

Prerequisites

```
GSPLSTCR
```

GSPSETS

SETS A SHORT INTEGER PROPERTY.

NOTE: If you intend to use the same Find property settings for all your address records, try to construct your application to set the Find properties once before processing. It is unnecessary to reset the properties again to the same values for each address record. However, if required, you can change Find property values for individual searches. Performance may be negatively affected by resetting for individual record searches. Reset Find properties between address searches only if changing the Find property value is necessary.

Syntax

```
01 PROPLIST     PIC S9(9) BINARY.
01 PROPENUM     PIC S9(9) BINARY.
01 INTS         PIC S9(4) BINARY.
01 GSFUNSTAT    PIC S9(9) BINARY.
*
CALL “GSPSETS” USING PROPLIST, PROPENUM, INTS, GSFUNSTAT.
```

Arguments

PROPLIST	Pointer to property list structure. *Input and Output.*
PROPENUM	Property ID. *Input.*
INTS	Short value. *Input and Output.*
GSFUNSTAT	Return value for the procedure. *Output.*

Return Values

```
GS-SUCCESS

GS-ERROR
```

Prerequisites

GSPLSTCR

GSPSETST

SETS A STRING PROPERTY.

NOTE: If you intend to use the same Find property settings for all your address records, try to construct your application to set the Find properties once before processing. It is unnecessary to reset the properties again to the same values for each address record. However, if required, you can change Find property values for individual searches. Performance may be negatively affected by resetting for individual record searches. Reset Find properties between address searches only if changing the Find property value is necessary.

Syntax

```
01 PROPLIST     PIC S9(9) BINARY.
01 PROPENUM     PIC S9(9) BINARY.
01 C-CHARACTER-STRING.
05 CHAR-STRING  PIC X(user len) VALUE 'your string here'.
05  FILLER      PIC X(01) VALUE X'00'.
*
CALL “GSPSETST” USING PROPLIST, PROPENUM, C-CHARACTER-STRING, GSFUNSTAT.
```

Arguments

PROPLIST	Pointer to property list structure. *Input and Output.*
PROPENUM	Property ID. *Input.*
C-CHARACTER-STRING	String value. *Input.*
GSFUNSTAT	Return value for the procedure. *Output.*

Return Values

GS-SUCCESS

GS-ERROR

Prerequisites

GSPLSTCR

GSSCACHE

SETS THE SIZE OF THE CACHE GEOSTAN USES.

Syntax

```
01 SIZE               PIC S9(9) BINARY.
*
CALL “GSSCACHE” USING SIZE.
```

Arguments

SIZE Size to set cache. *Input*

Value	Cache Size
0	Small
1	Medium
2	Large

Notes

You must call **GSSCACHE** before **GSINITWP**.

A smaller cache may slow the performance of GeoStan. Pitney Bowes Business Insight recommends a cache size of 2 for the best performance.

For changes in cache size settings to take effect, call **GSSCACHE** before **GSINITWP**.

GSSETSEL

ALLOWS YOU TO SELECT A MATCH FROM A SET OR MATCH CANDIDATEES.

Syntax

```
01 GSID              PIC S9(9) BINARY.
01 GSFUNSTAT         PIC S9(9) BINARY.
01 INDEX             PIC 9(4) BINARY.
*
CALL "GSSETSEL" USING GSID, INDEX, GSFUNSTAT.
```

Arguments

GSID ID returned by **GSINITWP** for the current instance of GeoStan. *Input.*

GSFUNSTAT Return value for the procedure. *Output*.

INDEX Index number (0-based) of the possible match. *Input*.

Return Values

GS-SUCCESS

GS-ERROR

Prerequisites

GSMGET

Notes

After **GSFINDWP** returns GS-ADDRESS-NOT-RESOLVED, use **GSNMULT** and **GSMGET** to determine which possible match is the correct match. Then use **GSSETSEL** to load that possible match into the data retrieval buffers and retrieve it using **GSDATGET**.

This procedure returns GS-ERROR if the selection is out of range.

GSSLIC

IDENTIFIES LICENSE FILE AND KEY.

Syntax

```
01 PASSWORD         PIC S9(9) BINARY.
01 FILE-NAME        PIC X(12).
*
CALL “GSSLIC” USING PASSWORD, FILE-NAME.
```

Arguments

PASSWORD Long integer used as a unique key. *Input.*

FILE-NAME Fully qualified file name, including drive and path, of the file containing the license information. *Input.*

Notes

Must call **GSSLIC** before **GSINITWP**.

GSSRELD

Allows you to specify the oldest acceptable date for GeoStan data files.

Syntax

```
01 RELEASE-DATE    PIC X(8).
01 LSTATUS         PIC 9(9) BINARY.
*
CALL "GSSRELD" USING RELEASE-DATE, LSTATUS.
```

Arguments

RELEASE-DATE Oldest allowable date for GeoStan data files, in the format `yyyymmdd`. *Input*.

LSTATUS Return code. *Output*.

Return Values

Date and time given in *RELEASE-DATE* as a binary number. This binary value is the number of seconds that have elapsed since January 1, 1970.

Notes

Must be called before **GSINITWP**.

Ignores any data files with dates older than the *RELEASE-DATE* parameter.

GSSNDX

Generates a soundex key for use in GSFFSET.

Syntax

```
01 LSTATUS       PIC 9(9) BINARY.
01 NAME          PIC X(USERLEN).
*
CALL "GSSNDX" USING NAME, LSTATUS.
```

Arguments

NAME String to convert to a soundex key. *Input*.

LSTATUS Return code. *Output*.

Return Values

Soundex key

Notes

This procedure generates a soundex key for a street name. Use it in conjunction with **GSFFSEG** when you want to perform a soundex search. The soundex key, when combined with a locale (state and city), is the primary index into the GeoStan databases. Searching by soundex allows you to use your own scoring and matching mechanism for geocoding.

NOTE: A Soundex match is a match based on the pronunciation of a field, not on the spelling of a field.

GeoStan uses a modified version of the standard soundex algorithm first published by Donald Knuth. The modifications made by PBBI include special treatment of certain prefixes such as MAC, KN, and WR; special treatment for numeric street names; and an encoding scheme to pack the key into the smallest number of bits.

GSSSELR

ALLOWS GEOSTAN TO USE A RECORD FOUND OUTSIDE OF GSSFINDWP AS A MATCH.

Syntax

```
01 GSID             PIC S9(9) BINARY.
01 GSFUNSTAT        PIC S9(9) BINARY.
01 GSOPTIONS        PIC S9(9) BINARY.
01 RANGE-HANDLE     PIC S9(9) BINARY.
*
CALL "GSSSELR" USING GSID, RANGE-HANDLE, GSOPTIONS,
GSFUNSTAT.
```

Arguments

GSID ID returned by **GSINITWP** for the current instance of GeoStan. *Input*.

GSFUNSTAT Return value for the procedure. *Output*.

RANGE-HANDLE Range pointer of object to return as a match. A range record indicates a range of addresses such as “2800 – 2900 Center Green Drive”. *Input*.

GSOPTIONS The following table contains the valid variables. *Input*.

`GS-ADDR-CODE`	Attempts to standardize and find an address geocode. You must set this switch for address standardization. If this switch is not set, GeoStan uses only the input ZIP and ZIP + 4 address elements.
`GS-WIDE-SEARCH`	GeoStan considers all records matching the first letter of the street name, rather than the Soundex key on the street name. This results in a wider search.
`GS-FINANCE-SEARCH`	When you add GS_FINANCE_SEARCH, GeoStan searches the entire Finance Area for possible streets. These modifiers have no meaning when doing a ZIP centroid match.
`GS-Z9-CODE`	Attempts to find ZIP + 4 centroid match only (no ZIP+2 or ZIP).
`GS-Z7-CODE`	Attempts to find ZIP+2 centroid match only (no ZIP + 4 or ZIP).
`GS-Z5-CODE`	Attempts to find a ZIP centroid match (no ZIP + 4 or ZIP+2).
`GS-Z-CODE`	Attempts to find a match based on all possible ZIP centroids available.

If you specify `GS_ADDR_CODE` and a ZIP centroid option, GeoStan only returns a ZIP Code centroid match if an address geocode is not available.

You must use either `GS_ADDR_CODE` or `GS_Z_CODE` or both.(In general, you should specify both.) These option settings are additive.

If you enter a pre-parsed address, it must contain the USPS abbreviations for street type, predirectionals, and postdirectionals.

Return Values

`GS-SUCCESS`

`GS-ERROR`

Prerequisites

`GSFFSEG`

Notes

This procedure allows you to get a range handle which points to the address range you want to match to. Use this command to indicate that **`GSSFINDWP`** found the match, when in fact the match is set to whichever range handle is passed through the `RANGE-HANDLE` argument. You must specify the *`GSOPTIONS`* variable so that subsequent calls to **`GSDATGET`** have the type of location information to return.

This procedure returns standardized results from a list generated by **GSFFSEG** / **GSFNRANG**. This procedure performs a query procedure that lets the user choose from a list of possible matches.

GSTERM

TERMINATES GEOSTAN.

Syntax

```
01 GSID          PIC S9(9) BINARY.
01 GSFUNSTAT     PIC S9(9) BINARY.
*
CALL "GSTERM" USING GSID, GSFUNSTAT.
```

Arguments

GSID	ID returned by **GSINITWP** for the current instance of GeoStan. *Input*.
GSFUNSTAT	Contains the return value for the procedure. *Output*.

Return Values

GS-SUCCESS

GS-ERROR

Notes

You must call **GSTERM** at the conclusion of a session to close files and release other resources. It closes GeoStan and invalidates GSID. You cannot call any function except **GSINITWP** after **GSTERM**.

GSTSTRNG

DETERMINES WHETHER A HOUSE NUMBER FALLS WITHIN A RANGE.

Syntax

```
01 GSID            PIC S9(9) BINARY.
01 GSFUNSTAT       PIC S9(9) BINARY.
01 NUM-STRING      PIC X(user len).
01 NUM-RANGE-LO    PIC X(user len).
```

```
01 NUM-RANGE-HI    PIC X(user len).
*
CALL “GSTSTRNG” USING GSID, NUM-STRING, NUM-RANGE-LO,
NUM-RANGE-HI, GSFUNSTAT.
```

Arguments

GSID ID returned by **GSINITWP** for the current instance of GeoStan. *Input*.

GSFUNSTAT Return value for the procedure. *Output*.

NUM-STRING House number to test. *Input*.

NUM-STRING-LO Low house number in the range. *Input*.

NUM-STRING-HI High house number in the range. *Input*.

Return Values

Non-zero	Number is within the range.
0	Number is not within the range or the comparison cannot be made.

Prerequisites

GSINITWP

Notes

This procedure tests to see if the house number is within the range defined by NUM-STRING-HI and NUM-STRING-LO. It handles all valid house number patterns, such as

123	123A	A123	1A23	A1B23
1A23B	1-23	AB		

This procedure uses USPS rules defined in Publication 28, “Postal Addressing Standards”, for determining whether number is in range or not. Some patterns are not comparable with other patterns; for example, 123 is not comparable to a range of 1A01 to 1A99.

GSVAUXR

VALIDATES AN AUXILIARY FILE RECORD.

Syntax

```
01 GSID             PIC S9(9) BINARY.
01 GSFUNSTAT        PIC S9(9) BINARY.

*
CALL “GSVARU” USING GSID, GSFUNSTAT
```

Arguments

GSID	ID returned by **GSINITWP** for the current instance of GeoStan. *Input*.
GSFUNSTAT	Return value for the procedure. *Output*.

Return Values

GS-SUCCESS The record is valid and GeoStan will load the record. This includes comment records; records whose first character is a semicolon.

GS-WARNING The record contains a non-fatal error and GeoStan will load the record. Because this record has a problem, an input address may not be able to match to the record in its current state, or the output data from a match to the record may not be valid.

GS-ERROR The record contains a fatal error and GeoStan will NOT load the file.

Prerequisites

GSINITWP

GSVGSDDT

VALIDATES THE PRESENCE OF DATA FOR SPECIFIED STATES IN THE GSD DATA FILE.

Syntax

```
01 GSID                            PIC S9(9) BINARY.
01 STATES                          OCCURS 100 TIMES.
05 STATE-CODE                      PIC X(3).
05 STATE-COUNT                     PIC S9(9) BINARY.
01 GSFUNSTAT                       PIC S9(9) BINARY.

CALL “GSVGSDDT” USING GSID, STATES, GSFUNSTAT.
```

Arguments

GSID ID returned by **GSINITWP** for the current instance of GeoStan. *Input*.

STATES Structure containing the list of state abbreviations and/or FIPS codes. *Input*.

GSFUNSTAT Return value for the procedure. *Output*.

Return Values

GS-SUCCESS All states specified in *STATES* are present in the GSD data.

GS-GSD-DATA-MISSING One or more states specified in *STATES* are not present in the GSD data.

GS-ERROR *STATES* structure is invalid.

Notes

If GeoStan did not find the GSD primary data for all the specified states, this function creates an error in the error list retrievable using **GSERRGTX**. The error message code is 114 and the error message is "NO GSD DATA FOUND FOR STATE" and indicates which states GeoStan did not find.

The state abbreviations and corresponding state FIPS codes are as follows:

State	Abbrev.	FIPS Code[a]	State	Abbrev.	FIPS Code[a]
Alabama	AL	1	New Jersey	NJ	34
Alaska	AK	2	New Mexico	NM	35
Arizona	AZ	4	New York	NY	36
Arkansas	AR	5	North Carolina	NC	37
California	CA	6	North Dakota	ND	38
Colorado	CO	8	Ohio	OH	39
Connecticut	CT	9	Oklahoma	OK	40
Delaware	DE	10	Oregon	OR	41
District of Columbia	DC	11	Pennsylvania	PA	42
Florida	FL	12	Rhode Island	RI	44
Georgia	GA	13	South Carolina	SC	45
Hawaii	HI	15	South Dakota	SD	46

State	Abbrev.	FIPS Code[a]	State	Abbrev.	FIPS Code[a]
Idaho	ID	16	Tennessee	TN	47
Illinois	IL	17	Texas	TX	48
Indiana	IN	18	Utah	UT	49
Iowa	IA	19	Vermont	VT	50
Kansas	KS	20	Virginia	VA	51
Kentucky	KY	21	Washington	WA	53
Louisiana	LA	22	West Virginia	WV	54
Maine	ME	23	Wisconsin	WI	55
Maryland	MD	24	Wyoming	WY	56
Massachusetts	MA	25	American Samoa	AS	60
Michigan	MI	26	Guam	GU	66
Minnesota	MN	27	North Mariana Islands	MP	69
Mississippi	MS	28	Palau	PW	70
Missouri	MO	29	Puerto Rico	PR	72
Montana	MT	30	Virgin Islands	VI	78
Nebraska	NE	31			
Nevada	NV	32			
New Hampshire	NH	33			

[a] Do not specify Minor Islands (UM, 74) or you will receive an error. This is defined as a FIPS code, but the USPS does not generate data for this code.

The following pseudo FIPS codes support APO and FPO addresses:

Address	Abbrev.	FIPS Code
Armed Forces Europe	AE	57
Armed Forces Pacific	AP	58
Armed Forces Americas	AA	59

GSWECASS

WRITES A USPS CASS REPORT BASED ON THE SPECIFIED TEMPLATE.

Syntax

```
01 GSID                   PIC S9(9).
01 GS-EXTEND-CASS-DATA
01 DATA-SIZE              PIC S9(9)          BINARY.
01 OUTPUT-NAME            PIC X(1).
01 GSFUNSTAT              PIC S9(9)          BINARY.

CALL “GSWECASS” USING GSID, GS-EXTEND-CASS-DATA, DATA-SIZE, OUTPUT-NAME,
GSFUNSTAT.
```

Arguments

GSID ID returned by **GSINITWP** for the current instance of GeoStan. *Input.*

GSE-XTEND-CASS-DATA Input structure, defined in the GEOSTAN copybook. *Input*.

DATA-SIZE Size of the CASS report data structure. *Input*.

OUTPUT-NAME Not used.

GSFUNSTAT Return value for the procedure. *Output*.

Return Values

GS-SUCCESS

GS-ERROR

Prerequisites

GSINITWP

Notes

Before running a CASS report, verify you have loaded the GSZ file (ZIPMove data).

This procedure writes the CASS information to the header buffer using the data passed in *GSEXTENDCASSDATA*. This structure contains the following:

```
01 GS-EXTEND-CASS-DATA.
05 GS-ECD-STRUCTVERSION PIC 9(9) BINARY.
05 GS-ECD-NRECS     PIC 9(9) bINARY.
05 GS-ECD-NZIP4     PIC 9(9) BINARY.
```

```
05 GS-ECD-NZIP      PIC 9(9) BINARY.
05 GS-ECD-NCARRT    PIC 9(9) BINARY.
05GS-ECD-NDPBC      PIC 9(9) BINARY.
05 GS-ECD-LISTNAME  PIC X(20).
05 GS-ECD-VERSION   PIC X(12).
05 GS-ECD-CERTIFICATIONDATE PIC X(24).
05 GS-ECD-PSEARCHPATH PIC X(256).
05 GS-ECD-TEMPLATENAME PIC X(256).
05 GS-ECD-NZ4CHANGED PIC 9(9) BINARY.
05 GS-ECD-NLOT      PIC 9(9) BINARY.
05GS-ECD-Z4CHANGEVERSION PIC X(12).
05 GS-ECD-LOTVERSION PIC X(12).
05 GS-ECD-NHIGHRISEDEFAULT PIC 9(9) BINARY.
05 GS-ECD-NHIGHRISEEXACT PIC 9(9) BINARY.
05 GS-ECD-NRURALROUTEDEFAULT PIC 9(9) BINARY.
05 GS-ECD-NRURALROUTEEXACT PIC 9(9) BINARY.
05 GS-ECD-NLACS     PIC 9(9) BINARY.
05 GS-ECD-Z4COMPANYNAME PIC X(40).
05 GS-ECD-LOT-DPCCOMPANYNAME PIC X(40).
05 GS-ECD-LOT-Z4CONFIG PIC X(4).
05 GS-ECD-LOT-DPCCONFIG PIC X(4).
05 GS-ECD-Z4SOFTWARENAME PIC X(30).
05 GS-ECD-DPCSOFTWARENAME PIC X(30).
05 GS-ECD-LISTPROCESSORNAME PIC X(25).
05 GS-ECD-ZIP4DATABASEDATE PIC 9(9) BINARY.
05 GS-ECD-LOTDATABASEDATE PIC 9(9) BINARY.
05 GS-ECD-EWS-DENIAL PIC 9(9) BINARY.
05 GS-ECD-EWS-NTOTALDPV PIC 9(9) BINARY.
05 GS-ECD-EWS-DPVDATABASEDATEPIC 9(9) BINARY.
```

When you specify a version number for either version, Z4ChangeVersion, or LOTVersion, GeoStan updates the corresponding fields in the template cass3553.frm file. For example, entering a version number for Z4ChangeVersion prompts GeoStan to fill these fields on cass3553.frm:

– B. List, 2b. Date list Processed Z4Change

– B. List, 3b. Date of Database Product Used Z4Change

– C. Output, 1b. Total Coded Z4Change Processed.

To develop CASS certified application in GeoStan, you must have the correct license agreement with Pitney Bowes Business Insight. You must also obtain CASS certification from the USPS for every application developed in GeoStan. Using GeoStan does not make an application CASS certified. For information on becoming CASS certified, refer to Appendix E, "CASS certification."

GSZ4CH

DETERMINES IF THE USPS CHANGED A ZIP + 4 ADDRESS DEFINITION SINCE GEOSTAN LAST PROCESSED THE RECORD.

Syntax

```
01 GSID PIC S9(9) BINARY.
01 PZIP PIC X(USER LEN).
01 PZIP4 PIC X(USER LEN).
01 PDATE PIX X(USER LEN).
01 GSFUNSTATPIC S9(9) BINARY.
*
CALL “GSZ4CH” USING GSID, PZIP, PZIP4, PDATE, GSFUNSTAT.
```

Arguments

GSID ID returned by **GSINITWP** for the current instance of GeoStan. *Input.*

PZIP First five digits of the 9-digit ZIP Code to be tested. *Input.*

PZIP4 Last four digits of the 9-digit ZIP Code to be tested. *Input.*

PDATE Date of the GeoStan ZIP + 4 information file used to process the record. The string’s format is MMYYYY (for example, 081998). *Input.*

GSFUNSTAT Return value for the procedure. *Output.*

Return Values

GS-ERROR Common reasons for GS-ERROR are:

- The GSL file was not loaded because it was not found in the path.
- The GSL file was not loaded because it was a different release level than the GSD file.

GS-Z4-CHANGE

GS-Z4-NO-CHANGE

Prerequisites

GSINITWP

Notes

For **ZIP4CH** to work, the GeoStan GSL file must be in a directory listed in the PGPATHS member of GSIINITSTRUCT when calling **GSINITWP**. The Z4Change file, which is generated by the USPS, contains a record for every ZIP + 4 in the country. Each record contains twelve flags that represent the last twelve months, starting with the current release date. Each of these flags has a value of either True or False, indicating if the ZIP + 4 changed for that monthly postal release. The GeoStan GSL file incorporates this information, which **GSZ4Ch** references.

Z4Change information is valuable to users who process very large address lists frequently. As you process each record, you can call **GSZ4CH** and quickly tell if the record needs reprocessing. This information can help you quickly identify only those records that need reprocessing.

Your application must store the date of the GeoStan GSL file used to process a record. GeoStan uses this date as the **GSZ4CH** *PDATE* input parameter. This is the same date printed on the GeoStan CD used for record processing, and is one month later that the release date of the USPS files used on the GeoStan CD.

Use the following code example if you are unsure of the release date. You should run this code when standardizing a batch of records and store the resulting date string as input to **GSZ4CH** in future processing runs.

CHAPTER 5

Java quick reference

The following is a list of the classes that compose the GeoStan Java API. You can find detailed documentation for the Java API in the directory where you installed GeoStan in Geolib\JNI\JAVADOCS.

Class	Description
`CassData`	Formats and writes the CASS header.
`City`	Contains city.
`CityIterator`	Iterator to find the first/next city.
`ConfidenceInfo`	Allows a confidence surface generation engine to produce a confidence surface around a geocode, segment(s), or area.
`Coordinate`	A location with a latitude and longitude.
`DPVCompleteStats`	Obtains the complete DPV statistics since DPV was initialized.
`DPVInit`	Initializes GeoStan for DPV.
`FalsePositiveDetail`	Accesses the false-positive Detail record for DPV and LACS^{Link}.
`FalsePositiveHeader`	Accesses the false-positive Header record for DPV and LACS^{Link}.
FindProps	Contains GeoStan Find properties.
GeoProps	Contains GeoStan base class properties for InitProps, StatusProps, and FindProps (not to be instantiated). This class provides default implementations of many methods. Derived classes with special needs can and do override some of these methods.
`GeoStan`	Wraps the Centrus GeoStan library and accesses Centrus GeoStan functionality from within Java. The Centrus GeoStan library standardizes and geocodes addresses within the United States. The Centrus GeoStan library uses a C API. The GeoStan class uses Java Native Interface (JNI) routines to access the library.

Class	Description
GeoStanBase	Contains definitions all GeoStan input and output fields.
GeoStanBase.CityField	Contains constant field values for City.getData(int).
GeoStanBase.DataSetStatus	Data set status information.
GeoStanBase.Datum	Datum options.
GeoStanBase.Find	GeoStan find options.
GeoStanBase.MatchCode	Match error result codes. GeoStanBase.MATCH_CODE = Ennn indicates an error, or no match. This can occur when the address entered either did not exist in the GeoStan Directory, or the address was badly malformed and could not be parsed correctly. The last three digits of an error code indicate which parts of an address were unable to be matched to the GeoStan Directory.
GeoStanBase.MatchMode	Sets the match strictness.
GeoStanException	GeoStanException is thrown by the GeoStan Product.
GeographicIterator	Iterator to find the first/next Geographic entity (city/county/state).
GeographicIteratorEx	Iterator to find the first/next Geographic entity (city/county/state).
GeographicQuery	Performs geographic geocoding queries.
GeographicQueryEx	Performs geographic geocoding queries.
InitProps	Contains GeoStan Init properties.
LACSCompleteStats	Allows retrieval of the complete stats from LACS^{Link}.
LACSLinkInit	Initializes GeoStan for LACS^{Link}.
MinBoundingRect	Locates a minimum bounding rectangle (MBR) for the specified geographic element.
Range	Contains the range interval for house and unit numbers.
RangeIterator	Iterator to find the first/next Range.
Segment	Street segment container.
SegmentIterator	Iterator to find the first/next segment.
StatusProps	GeoStan Status properties container.

Class	Description
`Street`	Contains streets.
`StreetIterator`	Iterator to find first/next street.

CHAPTER 6

.Net quick reference

The following is a list of the classes that compose the GeoStan .Net API. You can find detailed documentation for the .Net API in the directory where you installed GeoStan in Geolib\DotNet\GsNetInerop.chm.

Class	Description
CASS.CASSData	Formats and writes the CASS header.
CASS.DPVInit	Initializes GeoStan for DPV.
CASS.FalsePositiveDetail	Accesses the false-positive Detail record for DPV and LACS$^{\text{Link}}$.
CASS.FalsePositiveHeader	Accesses the false-positive Header record for DPV and LACS$^{\text{Link}}$.
CASS.LACSLinkInit	Initializes GeoStan for LACS$^{\text{Link}}$.
City	Gets information for a city.
CityField	Contains constant field values for City.getData(int).
CityIterator	Accesses a list of cities.
Coordinate	Contains longitude and latitude values.
CsPropertiesFile	Reads the configuration information for your application.
DataSetStatus	Data set status information.
Datum	Datum options.
Find	GeoStan find options.
FindProps	Contains GeoStan Find properties.
GeoStanException	Thrown by the GeoStan product.
GeographicIterator	Iterator to find the first/next Geographic entity (city/county/state).
GeographicQuery	Performs geographic geocoding queries.

Class	Description
GeoProps	Contains GeoStan base class properties (not to be instantiated). This class provides default implementations of many methods. Derived classes with special needs can and do override some of these methods.
GeoPropsMetaData	Contains information about each of the Init, Find, and Status properties.
GeoStan	Wraps the Centrus GeoStan library and accesses Centrus GeoStan functionality from within .Net. The Centrus GeoStan library standardizes and geocodes addresses within the United States.
GeoStanBase	GeoStan base class that defines constants, enums, and utility classes used by the GeoStan .NET API.
InitProps	Contains GeoStan Init properties.
LACSCompleteStats	Allows retrieval of the complete stats from LACS^{Link}.
LACSLinkInit	Initializes GeoStan for LACS^{Link}.
MatchCode	Contains match error result codes.
MinBoundingRect	Locates a minimum bounding rectangle (MBR) for the specified geographic element.
Range	Contains the range interval for house and unit numbers.
RangeIterator	Iterator for the Range class.
Segment	Street segment container.
SegmentIterator	Iterator to find the first/next segment.
StatusProps	Contains GeoStan Status properties.
Street	Street container.
StreetIterator	Iterator to find first/next street.

APPENDIX A

Deprecated functions

This appendix discusses the deprecated C functions and COBOL procedures.

Data types for C APIs

Please see “Data types” on page 68.

Common enums for storing and retrieving data for C APIs

For more information on the following section, see “Common enums for storing and retrieving data” on page 68.

This section contains the valid enums used with **`GsHandleGet,`** **`GsDataSet`** **`GsDataGet`**, and **`GsMultipleGet`**.

EnumGeoStan static int	Length[a]	`GsHandleGet`			`GsDataSet`	`GsDataGet`			`GsMultipleGet`
		Street	Segment	Range		Input	Output	Center line	
`GS_CENTERLINE_OFFSET` Offset distance from the street center for a centerline match.	4					●			
`GS_DIST` Offset distance in feet from centerline and squeeze distance from intersection.	3				●	●			
`GS_OFFSET_DIST` Left and right offset distance from the street segments. The range is 0 - 999 feet, with a default of 50 feet.	4				●	●			

EnumGeoStan static int	Length[a]	GsHandleGet			GsDataSet	GsDataGet			GsMultipleGet
		Street	Segment	Range		Input	Output	Center line	
GS_MSA MSA or PMSA name.	66						●		
GS_MSA_NUMBER MSA or PMSA number.	5						●		
GS_NAME2_SHORT Shortest available intersection cross street street name.	41					I	I		I
GS_NEAREST_DIST Used differently with reverse geocoding and centerline matching: • *Reverse geocoding* – Distance, in feet, from the input location to the matched street segment, point address, or intersection. • *Centerline* – Distance, in feet, from the point-level match to the centerline match.	8						●	●	●
GS_SEARCH_DIST Radius, in feet, that GeoStan searches for a reverse geocode match. The range is 0 - 5280 feet, with a default value of 150 feet.	5				●	●			

EnumGeoStan static int	Length[a]	GsHandleGet			GsDataSet	GsDataGet			GsMultipleGet
		Street	Segment	Range		Input	Output	Center line	
GS_SCORE NOT SUPPORTED.. GeoStan matching score.	12								●
GS_SQUEEZE_DIST Distance, in feet, to offset address-level geocodes from the street end points. Default is 50 feet. If the squeeze distance is more than half the segment length, GeoStan sets the distance to the midpoint of the segment.	11				●	●			

[a] The length column is not applicable.

Deprecated C APIs

GsFind

WIN	UNIX	OS/390

DEPRECATED. MATCHES THE CURRENT ADDRESS LOADED VIA GSDATASET.

Syntax

```
GsFunStat GsFind( GsId gs, GsEnum options );
```

Arguments

gs ID returned by **GsInit_r** for the current instance of GeoStan. *Input*.

options **GsFind** options. The following table contains valid enums. *Input*.

Address Processing	
GS_ADDR_CODE	Attempts to standardize and find an address geocode. Set this switch if you want the address parsed and standardized. If this switch is not set, GeoStan uses only the input ZIP and ZIP + 4 address elements.

GS_GEO_CODE	Finds a geocode when address standardization is not possible. If GeoStan cannot standardize an address, it uses the input ZIP or ZIP + 4 to find a centroid match.
Search Options	
GS_WIDE_SEARCH	Considers all records matching the first letter of the street name, rather than the Soundex key on the street name, which allows a wider search. NOTE: This option has no effect when performing a ZIP centroid match.
GS_FINANCE_SEARCH	Searches the entire Finance Area for possible streets. NOTE: This option has no effect when performing a ZIP centroid match.
GS_BUILDING_SEARCH	Enables matching to building names even when no unit numbers are present
GS_ALWAYS_FIND_CANDIDATES	Enables GeoStan to keep multiple candidate records when matching with point-level data for use with centerline matching. Not valid when using the reverse geocoding options.
Geocode Levels	
GS_Z9_CODE	Attempts to find a ZIP + 4 centroid match only. Not valid when using the reverse geocoding options.
GS_Z7_CODE	Attempts to find a ZIP+2 centroid match only (no ZIP + 4 or ZIP). Not valid when using the reverse geocoding options.
GS_Z5_CODE	Attempts to find a ZIP centroid match (no ZIP + 4 or ZIP+2). Not valid when using the reverse geocoding options.
GS_Z_CODE	Attempts to find any ZIP centroid match. Not valid when using the reverse geocoding options.
Multi-Line Address Processing	
GS_PREFER_POBOX	Sets the preference to P.O. Box addresses instead of street addresses for multi-line input addresses. See "Specifying a preference for street name or P.O. Box" on page 36 for more information. Ignored if processing in CASS mode.
GS_PREFER_STREET	Sets the preference to street addresses instead of P.O. Box addresses for multi-line input addresses. Ignored if processing in CASS mode.
Reverse Geocoding Processing	
GS_NEAREST_ADDRESS	Specifies that GeoStan can match to addresses interpolated on street segments or to point data locations.
GS_NEAREST_INTERSECTION	Specifies that GeoStan can match to intersections.
GS_NEAREST_UNRANGED	Specifies that GeoStan can match a street segment with no number ranges. Enabled with GS_NEAREST_ADDRESS. Ignored for point data and intersection matches.
Reverse APN Lookup	
GS_APN_SEARCH	Specifies that GeoStan match an input FIPS and APN code to receive information on the corresponding parcel
NOTE: You can use GS_NEAREST_ADDRESS and GS_NEAREST_INTERSECTION together to specify reverse geocoding to both addresses and intersections.	

For Address processing, you must use either `GS_ADDR_CODE` or `GS_Z_CODE` (or both). These settings are additive. For most purposes, you should specify the `GS_ADDR_CODE`, `GS_Z_CODE`, and `GS_GEO_CODE` switch.

If you specify `GS_ADDR_CODE` and one of the Geocode level options, GeoStan only returns a ZIP Code centroid match if the address is standardized but an address geocode is not available.

For reverse geocoding, you need to set the reverse geocoding processing options: `GS_NEAREST_ADDRESS, GS_NEAREST_INTERSECTION`, and `GS_NEAREST_UNRANGED`. The address processing and multi-line address processing options are not valid for reverse geocoding.

Return Values

`GS_ADDRESS_NOT_RESOLVED` GeoStan cannot resolve a match candidate.

`GS_ADDRESS_NOT_FOUND` GeoStan did not find an address match or if you have a metered license and the GeoStan record count is depleted.

For metered licenses, **`GsFind`** decrements the GeoStan record counter in the license each time it returns a non-error match code. If the counter reaches zero or below, **`GsFind`** returns `GS_ADDRESS_NOT_FOUND` and sets the match code to `E015`. If you have an unmetered license, no decrementing occurs, and processing is unlimited.

`GS_ERROR` Low-level errors; use **`GsErrorGet`** to retrieve the error information.

`GS_LASTLINE_NOT_FOUND` GeoStan did not find a match for city/state or ZIP Code.

`GS_SUCCESS` GeoStan found a match.

Prerequisites

`GsDataSet`

Notes

If GeoStan cannot standardize and address, you can still retrieve normalized address information, match codes, carrier routes, or other elements with **`GsDataGet.`** You can retrieve alias information by calling **`GsMultipleGet`** with an index of `0`.

Before each find function, call **`GsClear`** to reset the internal buffers. If you do not reset the buffers, you may receive incorrect results with information from a previous find. If you use both the reverse geocode and address line matching enums in the same call, GeoStan displays an error. These types of finds are mutually exclusive.

GsGetCoords

WIN	UNIX	OS/390

DEPRECATED. USE GSGETCOORDSEX FOR GREATER PRECISION.

Syntax

```
ints GsGetCoords( GsId gs, pintl pCoords, intsu maxPoints );
```

Arguments

gs	ID returned by **`GsInit_r`** for the current instance of GeoStan. *Input*.
pCoords	Array of coordinates, in `x,y` (longitude, latitude) order. *Output*.
maxPoints	Maximum number of points to return; used to prevent writing past the end of the *pCoords* buffer. *Input*.

Return Values

Number of points assigned to buffer.

Prerequisites

`GsFind`

Notes

This function returns an array of coordinates for the current feature found via **`GsFind`**. The maximum number that GeoStan can return is 64 coordinate pairs, each pair consisting of two long integers.

GeoStan scales coordinate pairs to integers with four decimal digits of precision. For example, GeoStan returns a point at (-98.3, 29.7) as (983000, 297000). This is a different scale from that expected by Spatial+ and similar GIS applications, which typically express coordinates in millionths of degrees. You may need to scale coordinates obtained with this function before using them as input to other software libraries or applications.

GsGetDatum

WIN	UNIX	OS/390

DEPRECATED. RETURNS THE CURRENT DATUM SETTING.

Syntax

```
intlu GsGetDatum ( GsId gs );
```

Arguments

gs	ID returned by **GsInit_r** for the current instance of GeoStan. *Input.*

Return Values

Current datum setting. Valid settings are DATUM_NAD27 and DATUM_NAD83 (default).

Prerequisites

GsSetDatum

Notes

This function returns the datum currently used by GeoStan in calculating address coordinates. This setting affects only the numeric coordinates returned by **GsDataGet** for the latitude and longitude of an address.

A datum is a mathematical model of the Earth used to calculate the coordinates on any map, chart, or survey system. The North American Datum (NAD) is the official reference ellipsoid used for the primary geodetic network in North America.

Although the return values are DATUM_NAD27 and DATUM_NAD83, note that GDT uses WGS84 instead of NAD83. These two coordinate systems are compatible.

GsHandleGetCoords

WIN	UNIX	OS/390

DEPRECATED. ONLY USED WITH GSGETCOORDS. IF USING GSGETCOORDSEX, USE GSHANDLEGETCOORDSEX.

Syntax

```
ints GsHandleGetCoords( GsId gs, GsSegmentHandle *pHandle,
pintl pCoords, intsu maxPoints );
```

Arguments

gs	ID returned by **GsInit_r** for the current instance of GeoStan. *Input.*
**pHandle*	Handle of the segment object for the coordinates returned. *Input.*

pCoords Array of coordinates, in x,y (longitude, latitude) order. *Output*.

maxPoints Maximum number of points that **GsGetCoords** returns; used to prevent writing past the end of *pCoords* buffer. *Input*.

Return Values

Number of points assigned to the buffer.

Prerequisites

GsFindFirst or **GsFindNext**

Notes

This function returns an array of coordinates for the segment to which *handle* points.

Example

```
/*This example retrieves coordinates for a street segment found using
GsFindFirstSegment.*/

#define MAX_POINTS 50
.
.
.
ints NumCoords;
intl Coords[MAX_POINTS *2];

NumCoords = GsHandleGetCoords(gs, &hSegment, Coords, MAX_POINTS);
for (int i = 0; i < NumCoords * 2; i += 2)
printf("Lon = %ld, Lat = %ld\n", Coords[i], Coords[i+1]);
```

GsInit_r

WIN	UNIX	OS/390

Deprecated. Initializes GeoStan.

Syntax

```
GsId GsInit_r ( GsInitStruct *gis );
```

Arguments

gis Contains the following:

gsVersion Set to `GS_GEOSTAN_VERSION`. *Input*.

options The following table contains valid enums. *Input*.

GS_FILE_ADDR_CODE	Loads the files necessary for address geocoding.
GS_FILE_Z9_CODE	Loads the Z9 file for centroid geocoding.
GS_FILE_SPATIAL_QUERY	Loads spatial query files. If GeoStan cannot load the spatial query files, **GsInit_r** fails. You can verify that GeoStan loads the files by checking the **IStatus** parameter in **GsInit_r**.
GS_FILE_ELEVATION_CODE	Loads the optional elevation files.
GS_FILE_APN_CODE	Loads the optional Assessor's Parcel Number (APN) files.

Except for a few cases, you should always specify `GS_FILE_ADDR_CODE` and `GS_FILE_Z9_CODE`. For example:

```
GS_FILE_ADDR_CODE | GS_FILE_Z9_CODE
```

pGeoPaths List of paths to search for necessary files. *Input*.

pZ4Dir Name of the ZIP + 4 directory file. *Input*.

licFilePassword Password for the license file. *Input*.

licFileName String `[GS_MAX_STR_LEN]` that specifies the path and name of the license file, for example, `c:\license\geostan.lic`. *Input*.

cacheSize Relative cache size used by GeoStan; either 0, 1, or 2. *Input*.

relDate String `[GS_MAX_STR_LEN]` that specifies the latest data to use in initialization, in the format `YYYYMMDD`. *Input*.

status Pointer to a long (32 bit) integer that stores details on the successfully initialized components. *Output*.

GeoStan uses the following constants to test each significant bit:

GS_FILE_CBSA_DIR	Successfully loaded the CBSA lookup file (cbsac.dir).
GS_FILE_CITY_DIR	Successfully loaded the City lookup file (Ctyst.dir).
GS_FILE_EWS	Successfully loaded the EWS file (ews.txt).
GS_FILE_EXPIRED	All GSD files have expired.
GS_FILE_AUXILIARY	Successfully loaded the auxiliary file (.gax).
GS_FILE_GEO_DIR	Successfully loaded the GeoStan directory file (.gsd).
GS_FILE_LICENSE	Successfully loaded the GeoStan license file.
GS_FILE_LOT	Successfully loaded the eLOT and Z4Change file (Us.gsl).
GS_FILE_PARSE_TABLES	Successfully loaded the parsing tables (Parse.dir).
GS_FILE_SPATIAL_QUERY	Successfully loaded the spatial query file (finmbr.dat).
GS_FILE_ZIP4CENT_DIR	Successfully loaded the ZIP + 4 centroid file (Us.z9).
GS_FILE_ZIP9_IDX	Successfully loaded the ZIP9 index file (*.gsu).
GS_FILE_ZIPMOVE	Successfully loaded the ZIPMove file (Us.gsz).
GS_FILE_APN	Successfully loaded the APN file.
GS_FILE_ELEVATION	Successfully loaded the elevation file.
GS_FILE_MEM_LIM_EXCDD	Exceeded the requested file memory limit.
GS_FILE_MEM_SYS_EXCDD	Exceeded system resources preventing file memory mapping.

Return Values

Valid GsId System initialized correctly.

NULL GeoStan failed to initialize. This can occur for one of the following reasons:

- GeoStan did not find the necessary files. Check the Status argument to view the files GeoStan found.
- GeoStan could not find the license files or the passwords were incorrect.
- There is not enough memory for GeoStan to initialize.
- All available GSD files have expired. Indicated by GS_FILE_EXPIRED in the status argument.

GS_WARNING If you are using **GsSetFilememoryLimit**, you can only call **GsSetFilememoryLimit** once per process. When calling **GsInit_r** more than once, you do not need to call **GsSetFileMemoryLimit** again.

Notes

GsInit_r is the recommended way to start a GeoStan application. It sets the license and expiration date, and initializes the library.

You can use **GsSetFilememoryLimit** to control the amount of memory used for mapped files. If you are using **GsSetFilememoryLimit**, call **GsSetFilememoryLimit** before calling **GsInit_r**.

Example

```
bool showAllMessages = true;
GsId gs;
char message[256];
char details[256];
intlu lDays;
GsInitStruct GIS;

/* Initialize GeoStan */

/* Complete the initialization structure */
GIS.gsVersion = GS_GEOSTAN_VERSION;
GIS.options = GS_FILE_ADDR_CODE|GS_FILE_Z9_CODE;
strcpy(GIS.pGeoPaths, "c:\\geostan\\datasets;c:\\geostan");
strcpy(GIS.pZ4Dir, "c:\\geostan\\datasets\\us.z9");
GIS.licFilePassword=43218765;
strcpy(GIS.licFileName, "c:\\license\\geostan.lic");
GIS.cacheSize=2;
strcpy(GIS.relDate,"");
GIS.status=0;

/* Initialize the library */
gs = GsInit_r(&GIS);

if ( gs == 0 || showAllMessages )
{
 while ( GsErrorHas(gs) )
 {
      GsErrorGetEx(gs, message, details);
         printf ("%s\n %s\n", message, details);
 }

     if ( gs == 0 )
     {
        /* GeoStan failed to initialize */
        printf("Unable to initialize GeoStan library,
        status = %04X.\n", GIS.status);
        return(1);
     }
}

/* GeoStan is initialized */
/* retrieve information about your GeoStan license */
lDays = GsFileStatusEx(gs, GS_STATUS_DAYS_REMAINING, 0, 0);
if ( lDays == DAYS_UNLIMITED )
{
   printf("License is non-expiring.\n");
}
else
{
   printf("License will expire in %d days.\n", lDays);
}

/* retrieve information about your GeoStan data */
lDays = GsFileStatusEx(gs, GS_STATUS_DATATYPE_STR, message, 256);
if ( lDays == GS_SUCCESS )
{
   printf("GeoStan data type is %s.\n", message);
}
else
{
   printf("Failed to retrieve information about data.\n");
}

/* do your geocoding ...*/

/* close the library */
GsTerm(gs);
return(0);
```

WIN	UNIX	OS/390

GsInitDpv_r

DEPRECATED. INITIALIZES GEOSTAN FOR DPV

Syntax

```
GsFunStat GsInitDpv_r (GsId gs, DpvInitStruct *dpvInitStruct,
int structSize);
```

Arguments

gs ID returned by **GsInit_r** for the current instance of GeoStan. *Input*.

**dpvInitStruct*
This structure contains the following:

dataAccess Indicates the type of files to load and how to access the files. The following table contains the possible values. *Input*.

`DPV_DATA_FULL_FILEIO`	*Default.* Files are not loaded into memory. Table access is through file I/O.
`DPV_DATA_FULL_MEMORY`	Files are loaded into memory. Use this option to gain performance improvement by reducing repetitive file I/O.
`DPV_DATA_SPLIT_FILEIO`	Use if your file is sorted by ZIP Code and you have limited memory. Uses a split data format that separates the DPV data file into multiple smaller files, based on the first 2 digits of the ZIP Code. If you sort your mailing file by ZIP Code, you can use this value to bring the relevant portion of the DPV file into memory. This process uses 32 MB of storage, but reduces the number of I/O requests that normally occurs when you use the full DPV data file.

`memoryBufferSize` Number of MBs used to load buffered files into memory. *Input.*

This option is only valid if `dataAccess` is not `DPV_DATA_FULL_MEMORY`. The following table contains the possible values.

`0`	Default. Do not load DPV files into memory. Access is through file I/O.
`1-999`	Number of MBs to allocate for buffered I/O.

`options` Reserved for future implementation. *Input.*

`pDirectory` String that specifies the directory containing DPV data. *Input.*

`securityKey` Security key to initialize DPV functionality. *Input.*

`Status` Indicates the successful initialization of DPV. The following table contains the constants used to test each significant bit. *Output.*

`GS_DPV_FILE_SECURITY`	The DPV security file initialized successfully.
`GS_DPV_FILE_ALL`	The DPV data initialized successfully.

`structSize` Size of the **`dpvInitStruct`** data structure. *Input.*

`structVersion` Set to `GS_GEOSTAN_VERSION`. *Input.*

Return Values

`GS_SUCCESS` Initialized successfully.

`GS_ERROR` Failed to initialize. Call `GsErrorGetEx` for information.

Prerequisites

See notes

Notes

If you are using **GsSetFileMemoryLimit** to control the amount of memory used for memory mapped files, you need to call **GsInitDpv_r** after **GsSetFileMemoryLimit** and before **GsInit_r**.

If you are not using **GsSetFileMemoryLimit**, you need to call **GsInit_r** before calling **GsInitDpv_r**.

GsInitLACSLink_r

WIN	UNIX	OS/390

DEPRECATED. INITIALIZES GEOSTAN FOR LACS^Link^.

Syntax

```
GsFunStat GsInitLACSLink_r ( GsId gs,
LACSLinkInitStruct *lacsInitStruct, init structSize);
```

Arguments

gs ID returned by **GsInit_r** for the current instance of GeoStan. *Input*.

**lacsInitStruct* This structure contains the following:

structVersion LACS^Link^ version number. Set to **GS_GEOSTAN_VERSION**. *Input*.

pDirectory String that specifies the directory containing LACS^Link^ data. *Input*.

status Indicates the successful initialization of LACS^Link.^ The following table contains the constants used to test each significant bit. *Output*.

GS_LACSLINK_FILE_SECURITY	Successfully loaded the LACS^Link^ license.
GS_LACSLINK_FILE_ALL	Successfully loaded the LACS^Link^ data

options Reserved for future implementation. *Input*.

securityKey Security key to initialize LACS^Link^ functionality. *Input*.

structSize Size of the `lacsInitStruct` data structure. *Input*.

Return Values

`GS_SUCCESS`	Initialized successfully.
`GS_ERROR`	Failed to initialize. Call `GsErrorGetEx` for information.

Prerequisites

See notes

Notes

If you are using **`GsSetFileMemoryLimit`** to control the amount of memory used for memory mapped files, you need to call **`GsInitLACSLink_r`** after **`GsSetFileMemoryLimit`** and before **`GsInit_r`**.

If you are not using **`GsSetFileMemoryLimit`**, you need to call **`GsInit_r`** before calling **`GsInitLACSLink_r`**.

GsSetDatum

WIN	UNIX	OS/390

DEPRECATED. SPECIFIES THE DATUM GEOSTAN USES TO CALCULATE ADDRESS COORDINATES.

Syntax

```
intlu GsSetDatum ( GsId gs, intlu iNewDatum );
```

Arguments

`gs`	ID returned by **`GsInit_r`** for the current instance of GeoStan. *Input.*
`iNewDatum`	Enum value indicating the datum to use in returning address coordinates. Valid enums are `DATUM_NAD27` and `DATUM_NAD83` (default). *Input*.

NOTE: A datum is a mathematical model of the Earth used to calculate the coordinates on any map, chart, or survey system. The North American Datum (NAD) is the official reference ellipsoid used for the primary geodetic network in North America.

Return Values

`GS_SUCCESS`	GeoStan used the requested datum to calculate the existing coordinates (no conversion required), or if all of the files for the requested datum are present.

`GS_ERROR`	GeoStan did not find any of the files for the requested datum. GeoStan continues to use the current datum, and does not reset. Call **`GsErrorGet`** to find out which datum files did not load.

Prerequisites

`GsInit_r`

Notes

Use **`GsSetDatum`** to select the datum you want GeoStan to use when returning address coordinates via **`GsDataGet`**.This setting affects only the numeric coordinates returned by **`GsDataGet`** for the latitude and longitude of an address.

You must call **`GsSetDatum`** before calling **`GsFind`**. GeoStan does not cache or modify the lat and lon values if you call **`GsSetDatum`** after **`GsFind`**.

Although the arguments are `DATUM_NAD27` and `DATUM_NAD83`, note that GDT uses WGS84 instead of NAD83. These two coordinate systems are compatible.

GsSetFileMemoryLimit

WIN	UNIX

DEPRECATED. CONTROLS THE AMOUNT OF MEMORY USED FOR MEMORY MAPPED FILES.

Syntax

```
int GsSetFileMemoryLimit (intlu megabytes)
```

Arguments

`megabytes`	Amount of memory to budget for memory mapped files. 0-4294 megabytes. If you input 0, GeoStan uses a small cache and reads from the disk. If you use the maximum amount, GeoStan maps as many files as possible until the limit is exhausted. *Input.*

Return Values

`GS_SUCCESS`	
`GS_ERROR`	You must call **`GsSetFileMemoryLimit`** before calling **`GsInit_r`**.

Prerequisites

None

Notes

`GsSetFileMemoryLimit` helps you control the amount of memory GeoStan uses by providing you with a way to set the memory limit for mapped files. Mapped files are the gsd, gsu, gsi, cbsac.dir, finmbr.dat, and us.gsl files.

Call `GsSetFileMemoryLimit` before `GsInit_r`. `GsSetFileMemoryLimit` returns a warning if called more than once per process.

If you are using DPV and LACS$^{\text{Link}}$, you must call `GsInitDpv_r` and `GsInitLACSLink_r` after `GsSetFileMemoryLimit` and before `GsInit_r`. Although the DPV and LACS$^{\text{Link}}$ files are not memory mapped files, these libraries load large files that can cause `GsInit_r` to fail if not enough memory is reserved.

To determine the amount of virtual bytes used by your process, set the file limit to 0 and run your process.

GsSetMatchMode

WIN	UNIX	OS/390

Deprecated. Controls the matching mode used by GsFind.

Syntax

```
ints GsSetMatchMode( GsId gs, ints MatchMode );
```

Arguments

`gs` ID returned by `GsInit_r` for the current instance of GeoStan. *Input.*

`MatchMode` Enum value indicating match mode type. The following table contains valid enums. *Input*.

GS_MODE_EXACT	Requires an exact name match. Generates the fewest number of possibles to search.
GS_MODE_CLOSE	(default) Requires a very close name match. Generates a moderate number of possibles to search

GS_MODE_RELAX	Requires a close name match. Generates the largest number of possibles to search.
GS_MODE_CASS	Requires a close name match. Generates the largest number of possibles to search. This setting imposes additional rules to ensure compliance with the USPS regulations for CASS software. For example, this mode disables intersection matching, and matching to the User Dictionary matches for standardization.

Return Values

Old match mode

– or –

`-1` if the new match mode is invalid

Prerequisites

`GsInit_r` or **`GsClear`**

Notes

`GsSetMatchMode` affects how **`GsDataSet`** performs. For this reason, only call this function immediately after **`GsInit_r`** or **`GsClear`**. The results are undefined if you call this function with **`GsDataSet`**.

To emulate the Centrus Desktop modes, use the settings in the following table.

Centrus Desktop (Match Mode)	GsSetMatchMode	GsFind
Tight	GS_MODE_EXACT	1 | 2
Close	GS_MODE_RELAX	1 | 2
Extended	GS_MODE_RELAX	1 | 2 | 3 | 4
CASS	GS_MODE_CASS	1 | 2 | 3
GeoStan Close Mode	GS_MODE_CLOSE	1 | 2

`GsFind` then uses as its second argument the enums listed (in combination with a logical OR). For example, in CASS mode, the second argument to **`GsFind`** is:

`GS_ADDR_CODE | GS_Z9_CODE | GS_FINANCE_SEARCH`

Enums for the second argument in **`GsFind`** are:

1	GS_ADDR_CODE
2	GS_Z9_CODE

3	GS_FINANCE_SEARCH
4	GS_WIDE_SEARCH

NOTE: For information on EWS, see Appendix E, "CASS certification". For information on DPV, see Appendix F, "USPS Link products".

Changing **`GsSetMatchMode`** does not require the presence of the following files; however, to achieve strict CASS compliance, these files must be present:

- `Us.gsz`
- `Use.gsd` and `Usw.gsd`
- `Use.gsu` and `Usw.gsu`
- DPV data
- EWS data (The file must be named `ews.txt`)

GsSetMixedCase

WIN	UNIX	OS/390

DEPRECATED. SETS THE CASING OF THE RETURNED INFORMATION.

Syntax

```
ints GsSetMixedCase( GsId gs, ints bMixedCase );
```

Arguments

`gs` ID returned by **`GsInit_r`** for the current instance of GeoStan. *Input.*

`bMixedCase` Sets the casing. *Input.*

- 0 = (default) Upper case
- 1 = Mixed case

Return Values

Current setting for this option (the setting before the change).

Prerequisites

`GsInit_r`

Notes

This function determines if GeoStan returns standardized data in upper or mixed case. It affects Firm name, all Address components, and City name. The USPS prefers upper case.

If you specify mixed case, generally only the first letter is set to upper case.

GsSetStreetCentroid

WIN	UNIX	OS/390

DEPRECATED. USED TO ENABLE STREET LOCATOR GEOCODING AS AN AUTOMATIC GEOCODING FALLBACK.

Syntax

```
qbool GsSetStreetCentroid( GsId gs, qbool GsStrCen );
```

Arguments

gs — ID returned by **GsInit_r** for the current instance of GeoStan. *Input*.

GsStrCen — Specifies whether or not to return a street segment geocode as an automatic geocoding fallback. *Input*.

- 0 = (default) Street locator disabled
- 1 = Return street segment geocode

Return Values

Current setting for this option (the setting before the change).

Prerequisites

GsInit_r

Notes

When this feature is enabled, if a street name is encountered while geocoding, and there is no matching address range, GeoStan will attempt to locate the street within the input ZIP Code or city if there is no input ZIP Code. If GeoStan is able to locate the street, it will return a geocode along the matched street segment rather than the geocode for the entered ZIP Code or ZIP + 4.

If a street number is entered, GeoStan will return the coordinates of the end point of the closest numeric street segment within the input ZIP Code. When there is no input ZIP Code, the closest numeric street segment of all the ZIP Codes within the input city will be returned.

If no street number is entered, the centroid of a matching street segment within the input ZIP Code will be returned. The centroid of a street segment for all the ZIP Codes within the input city will be returned when there is no input ZIP Code.

match candidatees may be returned. More information about the returned segment(s) can be obtained for either a single or a match candidate by calling GsMultipleGet().

This option is not available in CASS mode.

Common variables for storing and retrieving data for COBOL functions

NOTE: For more information on the following section, see“Common variables for storing and retrieving data” on page 248 .

This chapter contains the valid enums used with `GSHGET, GSDATASET, GSDATAGET,` and `GSMGET.`

Variable name	Buffer length	GSHGET			GSDATASET	GSDATAGET			GSMGET
		Street	Segment	Range		Input	Output	Center line	
`GS-CENTERLINE-OFFSET` Offset distance from the street center for a centerline match.	4					●			
`GS-DIST` Offset distance in feet from centerline and squeeze distance from intersection.	3				●	●			
`GS-OFFSE-DIST` Left and right offset distance from the street segments. The range is 0 - 999 feet, with a default of 50 feet.	4				●	●			

Variable name	Buffer length		GSHGET		GSDATASET		GSDATAGET		GSMGET
		Street	Segment	Range		Input	Output	Center line	
GS-SEARCH-DIST Radius, in feet, that GeoStan searches for a reverse geocode match. The range is 0 - 5280 feet, with a default value of 150 feet.	5				●	●			
GS-SCORE DEPRECATED. GeoStan matching score.	12								●
GS-SQUEEZE-DIST Distance, in feet, to offset address-level geocodes from the street end points. Default is 50 feet. If the squeeze distance is more than half the segment length, GeoStan sets the distance to the midpoint of the segment.	11				●	●			

Deprecated COBOL procedures

GSSFIND

Deprecated. Matches the current address loaded via GSDATSET.

Syntax

```
01 GSID          PIC S9(9) BINARY.
01 GSFUNSTAT     PIC S9(9) BINARY.
01 GSOPTIONS     PIC 9(9) BINARY.
*
```

```
CALL "GSSFIND" USING GSID, GSOPTIONS, GSFUNSTAT.
```

Arguments

GSID Returned by **GSINIT** for the current instance of GeoStan. *Input*.

GSFUNSTAT Return value for the procedure. *Output*.

GSOPTIONS The following table contains valid variables. *Input*.

Processing Instructions	
GS-ADDR-CODE	Attempts to standardize and find an address geocode. Set this switch if you want the address parsed and standardized. If this switch is not set, GeoStan uses only the input ZIP and ZIP+4 address elements.
GS-GEO-CODE	Finds a geocode when address standardization is not possible . If GeoStan cannot standardize an address, it uses the input ZIP or ZIP + 4 to find a centroid match.
Search Options	
GS-WIDE-SEARCH	Considers all records matching the first letter of the street name, rather than the Soundex key on the street name, which allows a wider search. This option has no effect when performing a ZIP centroid match.
GS-FINANCE-SEARCH	GeoStan searches the entire Finance Area for possible streets. This option has no effect when performing a ZIP centroid match.
GS-BUILDING-SEARCH	Enables matching to building names even when no unit numbers are present.
Geocode Levels	
GS-Z9-CODE	Attempts to find ZIP+4 centroid match only.
GS-Z7-CODE	Attempts to find ZIP+2 centroid match only (no ZIP + 4 or ZIP).
GS-Z5-CODE	Attempts to find a ZIP centroid match (no ZIP + 4 or ZIP+2).
GS-Z-CODE	Attempts to find a ZIP Code centroid match.
Multi-Line Address Processes	
GS-PREFER-POBOX	Sets the preference to a P.O. Box instead of a street address (multi-line input address). See "Specifying a preference for street name or P.O. Box" on page 36 for more information. Ignored when processing in CASS mode.
GS-PREFER-STREET	Sets the preference to a street address instead of a P.O. Box (multi-line input address). Ignored when processing in CASS mode.
Reverse Geocoding Processing	
GS-NEAREST-ADDRESS	Specifies that GeoStan can match to addresses interpolated on street segments or to point data locations.
GS-NEAREST-INTERSECTION	Specifies that GeoStan can match to intersections.
GS-NEAREST-UNRANGED	Specifies that GeoStan can match a street segment with no number ranges. Enabled with GS-NEAREST-ADDRESS. Ignored for point data and intersection matches.
NOTE: You can use GS_NEAREST_ADDRESS and GS_NEAREST_UNRANGED together to specify reverse geocoding to both addresses and intersections.	

You must use either `GS-ADDR-CODE` or one of the geocode level options (or both). These option settings are additive. For most purposes, you should specify the `GS-ADDR-CODE, GS-GEO-CODE`, and one of the geocode level options. If you specify `GS-ADDR-CODE` and one of the geocode level options, GeoStan returns a ZIP Code centroid match only if the address is standardized but an address geocode is not available.

Return Values

`GS-SUCCESS` Found a match.

`GS-ERROR` Low-level error; use **`GSERRGET`** to retrieve the error information.

`GS-ADDRESS-NOT-FOUND` Did not find an address match or you have a metered license and the GeoStan record count is depleted.

`GS-ADDRESS-NOT-RESOLVED` GeoStan cannot resolve which possible match is a match.

`GS-LASTLINE-NOT-FOUND` Did not find a match for city/state or ZIP Code.

Prerequisites

`GSDATSET`

Notes

If GeoStan could not standardize an address, you can still retrieve normalized address information, match codes, carrier routes, or other elements with **`GSDATGET.`** You can also return alias information by calling **`GSMGET`** with an index of 0.

If you enter a pre-parsed address, it must contain the USPS abbreviations for street type, predirectionals, and postdirectionals.

Before each find procedure, call **`GSCLEAR`** to reset the internal buffers. If you do not reset the buffers, you may receive incorrect results with information from a previous find.

If you use both the reverse geocode and address line matching variables in the same call, GeoStan displays an error. These types of finds are mutually exclusive.

GSGCRD

DEPRECATED. RETRIEVES COORDINATES FOR THE STREET SEGMENT FOUND VIA GSSFIND.

Syntax

```
01 GSIDPIC S9(9) BINARY.
01 GSFUNSTATPIC 9(4) BINARY.
01 COORDSOCCURS 64 TIMES.
05 COORD-XPIC S9(9) BINARY.
05 COORD-YPIC S9(9) BINARY.
01 MAXPOINTSPIC 9(4) BINARY.
*
CALL “GSGCRD” USING GSID, COORDS, MAXPOINTS, GSFUNSTAT.
```

Arguments

`GSID`	ID returned by **`GSINIT`** for the current instance of GeoStan. *Input*.
`GSFUNSTAT`	Return value for the procedure. *Output*.
`COORDS`	Array of coordinates, in x,y (longitude, latitude) order. *Output*.
`MAXPOINTS`	Maximum number of points that **`GSGCRD`** should return; used to prevent writing past the end of *`COORDS`* buffer. *Input*.

Return Values

Number of points assigned to buffer.

Prerequisites

`GSSFIND`

Notes

This procedure returns an array of coordinates for the current feature found via **`GSSFIND`**. The maximum number that GeoStan can return is 64 coordinate pairs, each pair consisting of two long integers.

GeoStan scales coordinate pairs to integers with four decimal digits of precision. Thus, GeoStan returns a point at (-98.3, 29.7) as (983000, 297000). This is a different scale from that expected by Spatial+ and similar GIS applications, which typically express coordinates in millionths of degrees. You may need to scale coordinates obtained with this procedure before using them as input to other software libraries or applications.

GSGDATUM

Deprecated. Returns the current datum setting.

Syntax

```
01 GSID                      PIC S9(9) BINARY.
01 GSFUNSTAT                 PIC S9(9) BINARY.
*
CALL “GSGDATUM” USING GSID, GSFUNSTAT.
```

Arguments

GSID ID returned by **GSINIT** for the current instance of GeoStan. *Input*.

GSFUNSTAT Return value for the procedure. *Output*.

Return Values

Returns the current datum setting. Valid settings are DATUM-NAD27 and DATUM-NAD83.

Prerequisites

GSSDATUM

Notes

This procedure returns the datum currently used by GeoStan in calculating address coordinates. This setting affects only the numeric coordinates returned by **GSDATGET** for the latitude and longitude of an address.

A datum is a mathematical model of the Earth used to calculate the coordinates on any map, chart, or survey system. The North American Datum (NAD) is the official reference ellipsoid used for the primary geodetic network in North America.

Although the return values are DATUM_NAD27 and DATUM_NAD83, note that GTD uses WGS84 instead of NAD83. These two coordinate systems are compatible.

GSHGCRD

DEPRECATED. RETRIEVES THE COORDINATES FOR A STREET SEGMENT OBJECT FOUND VIA GSFFSEG AND GSFNRANG.

Syntax

```
01 GSID           PIC S9(9) BINARY.
01 GSFUNSTAT      PIC S9(9) BINARY.
01 SEGMENT-HANDLEPIC S9(9) BINARY.
01 MAXPOINTS      PIC  9(4) BINARY.
01 COORDS         OCCURS <nnn> TIMES.
05 COORD-X    PIC S9(9) BINARY.
```

```
05 CORRD-Y   PIC S9(9) BINARY.
*
CALL “GSHGCRD” USING GSID, SEGMENT-HANDLE, COORDS, MAXPOINTS, GSFUNSTAT.
```

Arguments

GSID ID returned by **GSINIT** for the current instance of GeoStan. *Input*.

GSFUNSTAT return value for the procedure. *Output*.

SEGMENT-HANDLE Handle of the segment object for the returned coordinates. *Input*.

COORDS Array of coordinates, in x,y (longitude, latitude) order. *Output*.

MAXPOINTS Maximum number of points that **GSGCRD** returns; used to prevent writing past the end of *COORDS* buffer. *Input*.

Return Values

Number of points assigned to the buffer.

Prerequisites

GSFFSEG or GSFNRANG

Notes

This procedure returns an array of coordinates for the segment identified in *SEGMENT-HANDLE*.

GSINIT

DEPRECATED. INITIALIZES GEOSTAN.

Syntax

```
01 GSID         PIC S9(9) BINARY.
01 LSTATUS      PIC  9(9) BINARY.
01 GSOPTIONS    PIC  9(9) BINARY.
* PPATH AND Z4DIR: ONLY VALID VALUES ARE BLANK OR "HIPER"
* THEY ARE ONLY USED IN GSINIT
*
01 PPATH        PIC X(12).
01 Z4DIR        PIC X(12).
*
```

```
CALL “GSINIT” USING GSOPTIONS, PPATH, Z4DIR, LSTATUS, GSID.
```

Arguments

GSID ID returned by **GSINIT** for the current instance of GeoStan. *Input*.

GSOPTIONS Specifies which components of GeoStan to initialize.

GS-FILE-ADDR-CODE	Loads the files necessary for address geocoding.
GS-FILE-Z9-CODE	Loads the Z9 file for centroid geocoding.
GS-FILE-SPATIAL-QUERY	Loads spatial query files. If GeoStan cannot load the spatial query files, **GSINIT** fails. You can verify that GeoStan loads the files by checking the LStatus parameter in **GSINIT**.
GS-FILE-ELEVATION-CODE	Loads the optional elevation files.
GS-FILE-APN-CODE	Loads the optional Assessor’s Parcel Number (APN) files.

Except for a few cases, you should always specify GS-FILE-ADDR-CODE and GS-FILE-Z9-CODE. *Input*.

PPATH List of paths to search for necessary files. Set this argument to either “ ” or to “HIPER”, which loads the GSD file into hyperspace (OS/390 extended memory). Other values are ignored. *Input*.

Z4DIR Name of the ZIP + 4 directory file. Set this argument to either “ ” or to “HIPER”, which loads the Z9 file into hyperspace (OS/390 extended memory). GeoStan ignores other values. *Input*.

LSTATUS Pointer to a long (32-bit) integer that specifics which components GeoStan successfully initialized. *Output*.

GeoStan uses the following constants to test each significant bit:

GS-FILE-CBSA-DIR	Successfully loaded The CBSA lookup file (cbsac.dir).
GS-FILE-CITY-DIR	Successfully loaded the City lookup file (Ctyst.dir).
GS-FILE-EWS	Successfully loaded the EWS file (ews.txt).
GS-FILE-EXPIRED	All GSD files have expired (see Return Values section below).
GS-FILE-AUXILIARY	Successfully loaded the auxiliary file (.gax).
GS-FILE-GEO-DIR	Successfully loaded the GeoStan directory file (*.gsd).
GS-FILE-LICENSE	Successfully loaded the GeoStan license file.

GS-FILE-LOT	Successfully loaded the eLOT and Z4Change file (Us.gsl)
GS-FILE-PARSE-TABLES	Successfully loaded the parsing tables (Parse.dir).
GS-FILE-SPATIAL-QUERY	Successfully loaded the spatial query file (finmbr.dat).
GS-FILE-ZIP4-CENT-DIR	Successfully loaded the ZIP + 4 centroid file (Z4.dir).
GS-FILE-ZIPMOVE	Successfully loaded the ZIPMove file (Us.gsz).
GS-FILE-ZIP9-IDX	Successfully loaded the ZIP9 index file (Zip9.gsu).

Return Values

Returns a valid *GSID* if the system initializes correctly.

Returns 0 if GeoStan did not initialize.

GSINIT can fail for any of the following reasons:

- GeoStan did not find the necessary files. Check *LSTATUS* for the files GeoStan successfully found (and by omission, the files not found).
- Not enough memory for GeoStan to initialize.
- All available GSD files have expired. In this situation, **GSINIT** returns *GS-FILE-EXPIRED* in the *LSTATUS* argument.

Prerequisites

GSSLIC

Notes

Must be called before **GSSCACHE**, **GSSLIC**, or **GSSRELD**.

If you are using the **GSSCACHE** and **GSSRELD** procedures, You must call these before **GSINIT**.

You must call **GSINIT** before any other GeoStan procedure that uses the handle that **GSINIT** returns.

GSDPVINR

Deprecated. Initializes GeoStan for DPV.

Syntax

```
01 GSID                PIC S9(9) BINARY VALUE 0.
01 GSFUNSTAT           PIC S9(9) BINARY.
```

```
01 STRUCT-SIZE          PIC S9(9) BINARY.
01 W-HAVE-MESSAGE       PIC S9(9) BINARY VALUE +0.
01 W-ERROR-MSG          PIC X(256) VALUE LOW-VALUES.
01 W-ERROR-DETAIL       PIC X(256) VALUE LOW-VALUES.
*
CALL 'GSDPVINR' USING GSID, GS-DPV-INIT-STRUCT, STRUCT-SIZE, GSFUNSTAT.
```

Arguments

GSID ID returned by **GSINIT** for the current instance of GeoStan. *Input*.

GS-DPV-INIT-STRUCT Also defined in GEOSTAN.H. This structure contains the following:

GS-DIS-STRUCT-VERSION Set to GS-GEOSTAN-VERSION. *Input*.

GS-DIS-OPTIONS Reserved for future implementation. *Input*.

GS-DIS-PDIRECTORY String [GS-MAX-STR-LN] that specifies the directory containing the DPV file. Optional. *Input*.

GS-DIS-SECURITY-KEY Security key to initialize DPV functionality. *Input*.

GS-DIS-STATUS Indicates the successful initialization of DPV. the following table contains the constants used to test each significant bit. *Output*.

GS-DPV-FILE-SECURITY	The DPV security file for initializing the DPV functionality.
GS-DPV-FILE-ALL	The DPV data initialized successfully.

GS-DIS-DATA-ACCESS Indicates the type of files to load and how to access the files. The following table contains the possible values. *Input*.

DPV-DATA-FULL-FILEIO	Default. Files not loaded into memory. All table access is through file I/O. Use this option for OS/390.
DPV-DATA-FULL-MEMORY	All files loaded into memory. Use this option to gain performance improvement by reducing repetitive file I/O.
DPV-DATA-SPLIT-FILEIO	Separates the DPV data file into multiple smaller files, based on the first 2 digits of the ZIP Code. If you sort your mailing file by ZIP Code, you can bring the relevant portion of the DPV file into memory. This process uses 32 MB of storage, but reduces the number of I/O requests that normally occurs when you use the full DPV data file. Use this option if your file is sorted by ZIP Code and you have limited memory.

GS-DIS-MEOMORY-BUFFER-SIZE Number of megabytes used to load buffered files into memory. *Input*.

This option is only valid if `dataAccess != DPV-DATA-FULL-MEMORY`. The following are possible values:

`0`	Default. Do not load DPV files into memory. Access is through file I/o. **NOTE:** Use this option for OS/390.
`1-999`	Number of megabytes to allocate for buffered I/o.
`DPV-DATA-SPLIT-FILEIO`	Use if your file is sorted by ZIP Code and you have limited memory. Uses a split data format that separates the DPV data file into multiple smaller files, based on the first 2 digits of the ZIP Code. If you sort your mailing file by ZIP Code, you can use this value to bring the relevant portion of the DPV file into memory. This process uses 32 MB of storage, but reduces the number of I/O requests that normally occurs when you use the full DPV data file.

`STRUCT-SIZE` The size of the **`DPVINITSTRUCT`** data structure. *Input.*

`GSFUNSTAT` Return value for the procedure. *Output*.

Return Values

`GS-ERROR` Call **`GSERRGTX`** for more information

`GS-WARNING` Call **`GSERRGTX`** for more information

Prerequisites

`GSINIT`

Example

First initialize GeoStan for DPV.

```
**** WORKING STORAGE VARIABLES ******************
01 GSID                 PIC S9(9) BINARY VALUE 0.
01 GSFUNSTAT            PIC S9(9) BINARY.
01 STRUCT-SIZE          PIC S9(9) BINARY.
01 W-HAVE-MESSAGE       PIC S9(9) BINARY VALUE +0.
01 W-ERROR-MSG          PIC X(256) VALUE LOW-VALUES.
01 W-ERROR-DETAIL       PIC X(256) VALUE LOW-VALUES.

COPY GSCONST.

**** SAMPLE CODE TO INITIALIZE DPV **************
**** PLACE AFTER CALL TO GSINIT    **************
MOVE '(YOUR DPV LICENSE KEY)' TO GS-DIS-SECURITY-KEY.
MOVE GS-GEOSTAN-VERSION       TO GS-DIS-STRUCT-VERSION.
MOVE DPV-DATA-FULL-FILEIO     TO GS-DIS-DATA-ACCESS.
MOVE ZERO                     TO GS-DIS-MEMORY-BUFFER-SIZE.
MOVE GS-DPV-COB-STRUCT-SIZE   TO STRUCT-SIZE.
MOVE 'NULL'                   TO GS-DIS-PDIRECTORY.
MOVE X'00'                    TO GS-DIS-PDIRECTORY(5:1).

CALL 'GSDPVINR' USING GSID, GS-DPV-INIT-STRUCT, STRUCT-SIZE, GSFUNSTAT.

IF GSFUNSTAT = GS-SUCCESS
DISPLAY 'DPV INITIALIZED SUCCESSFULLY'
ELSE
DISPLAY '*********************************'
DISPLAY 'DPV FAILED TO INITIALIZE'
DISPLAY 'GSFUNSTAT IS:', GSFUNSTAT
DISPLAY 'GSID IS:', GSID
DISPLAY 'GS-DPV-INIT-STRUCT IS:', GS-DPV-INIT-STRUCT
DISPLAY 'STRUCT-SIZE IS:', STRUCT-SIZE
DISPLAY '*********************************'
CALL 'GSERRHAS' USING GSID, W-HAVE-MESSAGE
PERFORM UNTIL W-HAVE-MESSAGE IS EQUAL TO ZERO
CALL 'GSERRGTX' USING GSID, W-ERROR-MSG, W-ERROR-DETAIL,  GSFUNSTAT
DISPLAY 'W-ERROR-MSG:', W-ERROR-MSG
DISPLAY 'W-ERROR-DETAIL:', W-ERROR-DETAIL
CALL 'GSERRHAS' USING GSID, W-HAVE-MESSAGE
END-PERFORM.
```

After you have initialized DPV and called `GSSFIND` for an address, you can then call `GSDATGET` and request any of the following output fields, which are included in the copy member `GSCONST`:

• GS-DPBC	• GS-DPV-CONFIRM
• GS-DPV-CMRA	• GS-DPV-FALSE-POS
• GS-DPV-FOOTNOTE1	• GS-DPV-FOOTNOTE2
• GS-DPV-FOOTNOTE3	• GS-DPV-FOOTNOTE4
• GS-DPV-FOOTNOTE5	• GS-DPV-FOOTNOTE6

GSSDATUM

DEPRECATED. SPECIFIES THE DATUM GEOSTAN USES TO CALCULATE ADDRESS COORDINATES.

Syntax

```
01 GSID               PIC 9(09) BINARY.
01 NEWDATUM           PIC 9(09) BINARY.
01 GSFUNSTAT          PIC S9(09) BINARY.
*
CALL “GSSDATUM” USING GSID, NEWDATUM, GSFUNSTAT.
```

Arguments

GSID	ID returned by **GSINIT** for the current instance of GeoStan. *Input*.
NEWDATUM	Variable value indicating the datum to use in returning address coordinates. Valid variables are DATUM-NAD87 and DATUM-NAD83. *Input*.
GSFUNSTAT	Return value for the procedure. *Output*.

Return Values

GS-SUCCESS	GeoStan used the requested datum to calculated the existing coordinates (no conversion required), or the files for the requested datum were present.
GS-ERROR	Did not find all of the files for the requested datum. GeoStan does not reset the datum, and continues to use the current datum. Call **GSERRGET** to find the datum files that were not loaded.

Prerequisites

GSINIT

Notes

A datum is a mathematical model of the Earth used to calculate the coordinates on any map, chart, or survey system. The North American Datum (NAD) is the official reference ellipsoid used for the primary geodetic network in North America.

Use **GSSDATUM** to select the datum you want GeoStan to use when returning address coordinates via **GSDATGET.**

You must call **GSSDATUM** before calling **GSSFIND** to ensure that the lat and lon values are cached and are not modified.

This setting affects only the numeric coordinates returned by **GSDATGET** for the latitude and longitude of an address.

Although the arguments are DATUM_NAD27 and DATUM_NAD83, note that GTD uses WGS84 instead of NAD83. These two coordinate systems are compatible.

GSLACINR

DEPRECATED. INITIALIZE GEOSTAN FOR LACS^{Link}.

Syntax

```
01 GSID             PIC S9(9) BINARY VALUE 0.
```

```
01 GSFUNSTAT       PIC S9(9) BINARY.
01 STRUCT-SIZE     PIC S9(9) BINARY.
01 W-HAVE-MESSAGE  PIC S9(9) BINARY.
01 W-ERROR-MSG     PIC X(256) VALUE LOW-VALUES.
01 W-ERROR-DETAIL  PIC X(256) VALUE LOW-VALUES.
*
CALL 'GSLACINR' USING GSID, GS-LACSLINK-INIT-STRUCT, STRUCT-SIZE, GSFUNSTAT.
```

Arguments

GSID ID returned by **GSINIT** for the current instance of GeoStan. *Input.*

GS-LACSLINK-INIT-STRUCT Also defined in GEOSTAN.H. This structure contains the following:

GS-LIS-STRUCT-VERSION LACS^{Link} version number. Set to GS-GEOSTAN-VERSION. *Input.*

GS-LIS-OPTIONS Reserved for future implementation. *Input.*

GS-LIS-STATUS Indicates the successful initialization of LACS^{Link}. The following table contains the constants used to test each significant bit. *Output.*

GS-LACSLINK-FILE-LICENSE	LACS^{Link} license loaded successfully.
GS-LACSLINK-FILE-ALL	LACS^{Link} data loaded successfully.

GS-LIST-SECURITY-KEY Security key to initialize LACS^{Link} functionality. *Input.*

STRUCTSIZE Size of the LACSINITSTRUCT data structure. *Input.*

Return Values

GS-SUCCESS Initialized successfully.

GS-ERROR Failed to initialize.

Prerequisites

GSINIT

Example

Initialize GeoStan for LACS^{Link}.

```
***********  WORKING STORAGE VARIABLES ***********************
01 GSID            PIC S9(9) BINARY VALUE 0.
01 GSFUNSTAT       PIC S9(9) BINARY.
01 STRUCT-SIZE     PIC S9(9) BINARY.
```

```
01 W-HAVE-MESSAGE  PIC S9(9) BINARY.
01 W-ERROR-MSG     PIC X(256) VALUE LOW-VALUES.
01 W-ERROR-DETAIL  PIC X(256) VALUE LOW-VALUES.

COPY GSCONST.

*******  SAMPLE CODE TO INITIALIZE LACS/LINK *****************
**** PLACE AFTER CALL TO GSINIT    ***************************
MOVE GS-GEOSTAN-VERSION       TO GS-LIS-STRUCT-VERSION.
MOVE GS-LIS-COB-STRUCT-SIZE   TO STRUCT-SIZE.

CALL 'GSLACINR' USING GSID, GS-LACSLINK-INIT-STRUCT, STRUCT-SIZE, GSFUNSTAT.

IF GSFUNSTAT = GS-SUCCESS
DISPLAY 'LACS INITIALIZED SUCCESSFULLY'
ELSE
DISPLAY '*********************************'
DISPLAY 'LACS FAILED TO INITIALIZE'
DISPLAY 'GSFUNSTAT IS:', GSFUNSTAT
DISPLAY 'GSID IS:', GSID
DISPLAY 'GS-LACSLINK-INIT-STRUCT:', GS-DPV-INIT-STRUCT
DISPLAY 'STRUCT-SIZE IS:', STRUCT-SIZE
DISPLAY '*********************************'
CALL 'GSERRHAS' USING GSID, W-HAVE-MESSAGE
PERFORM UNTIL W-HAVE-MESSAGE IS EQUAL TO ZERO
CALL 'GSERRGTX' USING GSID, W-ERROR-MSG, W-ERROR-DETAIL,  GSFUNSTAT
DISPLAY 'W-ERROR-MSG:', W-ERROR-MSG
DISPLAY 'W-ERROR-DETAIL:', W-ERROR-DETAIL
CALL 'GSERRHAS' USING GSID, W-HAVE-MESSAGE
END-PERFORM
```

After you have initialized LACS^{Link} and called GSSFIND for an address, you can then call GSDATGET and request any of the following output fields, which are included in the copy member GSCONST:

- GS-LACS-FLAG
- GS-LACSLINK-IND
- GS-LACSLINK-RETCODE

GSSMIXED

DEPRECATED. SETS THE CASING OF THE RETURNED INFORMATION.

Syntax

```
01 GSID              PIC S9(9) BINARY.
01 GSFUNSTAT         PIC 9(4) BINARY.
01 MIXED-CASE        PIC 9(4) BINARY.
*
CALL "GSSMIXED" USING GSID, MIXED-CASE, GSFUNSTAT.
```

Arguments

GSID ID returned by **GSINIT** for the current instance of GeoStan. *Input*.

GSFUNSTAT Return value for the procedure. *Output*.

MIXED-CASE Sets the casing of the returned value; either 0 for upper case (default) or1 for mixed case. *Input*.

Return Values

Current setting for this option (the setting before the change).

Prerequisites

GSINIT

Notes

Casing affects the firm name, all address components, and City name. The USPS prefers upper case.

If you specify mixed case, generally only the first letter is set to upper case.

GSSMM

Deprecated. Controls the matching mode used by GSSFIND.

Syntax

```
01 GSIDPIC S9(9) BINARY.
01 MATCH-MODEPIC 9(4) BINARY.
01 GSFUNSTATPIC 9(4) BINARY.
*
CALL "GSSMM" USING GSID, MATCH-MODE, GSFUNSTAT.
```

Arguments

GSID ID returned by **GSINIT** for the current instance of GeoStan. *Input*.

MATCH-MODE Variable value indicating match mode type. The following table contains the valid variables. *Input*.

GS-MODE-EXACT	Requires an exact name match. Generates the fewest number of possibles to search.
GS-MODE-CLOSE	*Default*. Requires a very close name match. Generates a moderate number of possibles to search
GS-MODE-RELAX	Requires a close name match. Generates the largest number of possibles to search.
GS-MODE-CASS	Requires a close name match. Generates the largest number of possibles to search. This setting imposes additional rules to ensure compliance with the USPS regulations for CASS software. For example, this mode disables intersection matching, and matching to the street network file for standardization.

GSFUNSTAT Contains the return value for the procedure. *Output*.

Return Values

Old match mode, or a -1 if the new match mode entered is invalid.

Prerequisites

GSINIT or GSCLEAR

Notes

GSSMM affects how **GSDATSET** performs. For this reason, call this procedure only immediately after **GSINIT**, or after a **GSCLEAR**. If you call this procedure after loading data with **GSDATSET**, the results are undefined.

GSSTRCEN

DEPRECATED. USED TO ENABLE STREET LOCATOR GEOCODING AS AN AUTOMATIC GEOCODING FALLBACK.

Syntax

```
01 GSID               PIC S9(9) BINARY.
01 GSSTRCEN           PIC S9(9) BINARY.
01 GSFUNSTAT          PIC S9(9) BINARY.
*
CALL "GSSTRCEN" USING GSID, GSSTRCEN, GSFUNSTAT.
```

Arguments

GSID ID returned by **GSINIT** for the current instance of GeoStan. *Input*.

GSSTRCEN Specifies whether or not to return a street segment geocode as an automatic geocoding fallback. *Input*.

- 0 = (default) Street locator disabled
- 1 = Return street segment geocode

GSFUNSTAT Return value for the procedure. *Output*.

Return Values

Current setting for this option (the setting before the change).

Prerequisites

GSINIT

Notes

When this feature is enabled, if a street name is encountered while geocoding, and there is no matching address range, GeoStan will attempt to locate the street within the input ZIP Code or city if there is no input ZIP Code. If GeoStan is able to locate the street, it will return a geocode along the matched street segment rather than the geocode for the entered ZIP Code or ZIP + 4.

If a street number is entered, GeoStan will return the coordinates of the end point of the closest numeric street segment within the input ZIP Code. When there is no input Zip Code, the closest numeric street segment of all the Zip Codes within the input city will be returned.

If no street number is entered, the centroid of a matching street segment within the input Zip Code will be returned. The centroid of a street segment for all the Zip Codes within the input city will be returned when there is no input Zip Code.

match candidatees may be returned. More information about the returned segment(s) can be obtained for either a single or a match candidate by calling GSMGet().

This option is not available in CASS mode.

Appendix B

Extracting data from GSD files

You can access all of the information contained within the GSD files via the functions in GeoStan. While you need sophisticated matching algorithms to facilitate address standardization and geocoding, there may be other purposes that require data access in a more direct method. The **`GsFindFirst__`**, **`GsFindNext__`**, and **`GsHandleGet`** functions provide a way to directly access information. This section documents how to use these function, and provide information on the following topics:

- Extracting data directly from GSD files
- Data structure and concepts
- Street, segment, and range handles
- FindFirst and FindNext functions
- Search query
- Code example

NOTE: **`GsFindFirst__`**, **`GsFindNext__`**, and **`GsHandleGet`** are advanced features not normally used for most standardization and geocoding applications.

Extracting data directly from GSD files

Address data can be extracted directly from GSD files. All street elements may be extracted, including shape points. Such usage may be limited depending upon your license agreement with PBBI.

The locale to be searched can be set by City and State, Finance Area, or ZIP Code. You can filter the returned information by street name (partial strings are allowed) and house number. For example, all segments beginning with the letter “M” in ZIP Code 80301 can be returned, as well as all segments beginning with “GOVER” in the Finance Area containing ZIP Code 11001.

Data structure and concepts

The data contained within the GSD files comes primarily from two sources, premium street network data, such as the TIGER Census files, and the USPS ZIP + 4 files. GeoStan merges these files to create a complete and timely database that you can use to perform address standardization and geocoding.

NOTE: The following discussion specifically mentions TIGER data; however, the same information holds true for each of the premium geographic data vendors with which PBBI partners.

The data within the GSD files is organized into the following objects:

- Street Objects

Street Objects are the highest level and contain information about the street as a whole, such as the predirectional, name, type, and postdirectional. Each Street Object can contain one or more Segment Objects.

- Segment Objects

A Segment Object relates to only one Street Object. Segment objects contain the latitude and longitude of the segment corners, shape points (to help define a street that is not a straight line), the left and right Census Block codes, and the maximum and minimum ranges for that segment. GeoStan derives the data for a Segment Object from the TIGER files. Each Segment Object can contain zero or more Range Objects.

- Range Objects

A Range Object relates to only one Segment Object, and therefore only one Street Object. A Range Object contains low and high house ranges (including alphanumeric ranges), low and high unit ranges, the unit type (such as an apt. or ste.), ZIP Code, ZIP + 4 low and ZIP + 4 high, carrier route, and other USPS data. GeoStan obtains the data for a Range Object from the USPS files.

If an address does not have any Range Objects, the USPS does not have data for the address. PBBI data places street segments without ranges in the `Ustw.gsd` and `Uste.gsd` files.

There can be overlap within Range Objects. For example, one Range Object may contain house numbers between 100 to 198, while another Range Object may contain house numbers between 150 and 198. This frequently happens with high-rise Range Objects.

Because records may exist in one source that are not present in the other, there are occasions in which GeoStan may present only TIGER or USPS information. If PBBI was unable to match any TIGER records to a USPS record, a Segment Object may not contain any of the location and Census information. In this case, address geocode and Census Block information is not available. Conversely, there are records within TIGER that PBBI cannot match to any USPS records. In this case, no Range Objects may exist for that segment, and therefore GeoStan cannot standardize the address and cannot return the information available in the Range Object. This type of match does not produce a mailable address.

Street, segment, and range handles

Because of the hierarchical nature of the GSD data, Pitney Bowes Business Insight can create handles to any of the Street, Segment, and Range Object types. These objects must contain the reference to the higher object in order for GeoStan to create the handle. For example, a range handle must contain a segment handle, which in turn must contain a street handle.

A handle can be promoted to either predecessor. For example,

```
hStreet = hSegment.hStreet;
hStreet = hRange.hSegment.hStreet;
```

are acceptable, but

```
hSegment = hStreet;
```

is undefined.

However, the statement

```
hSegment.hStreet = hMyStreet;
```

is valid and is used to promote a street handle to a segment handle for entry into the Find routine. The segment handle of `hSegment` is still undefined, but that `hSegment.hStreet` can be used as a valid street handle (assuming `hMyStreet` was a valid handle to begin with).

NOTE: All handles are invalid after a call to GSCLEAR.

FindFirst and FindNext functions

To search the GSD files, begin with a call to **`GsFindFirstStreet`**. This function sets the locale and filtering options, as well as returns a street handle for the first street. The next call is normally to **`GsFindFirstSegment`**, which takes as an argument the street handle received from **`GsFindFirstStreet`** promoted to a segment handle, as detailed in "Street, segment, and range handles" on page 409. Often developers use a series of three nested while statements to traverse all possible entries.

Search query

With **`GsFindFirstStreet`** you can define options that narrow the matches returned by GeoStan.

When you use **`GsFindFirstStreet`** you use the city or ZIP Code to set the finance area of a search. A finance area can contain more than one ZIP Code or city. Therefore, GeoStan may return results that are in the finance area specified by the locale you entered, but the result may not be in the actual ZIP Code or city you entered.

You can narrow the number of matches GeoStan returns by setting the whole or partial street name. For example, if you are searching for “Apple,” GeoStan returns the streets within the finance area with the names of “APPLE,” “APPLETON,” and “APPLE ORCHARD.” You can set the street name based on the actual spelling of the name, or by the soundex value of the name.

You can further narrow the matches returned by GeoStan by indicating a house number. If you specify a house number, GeoStan only returns matches that have ranges that include the house number.

Code example

The following example searches ZIP Code 80301 for all streets beginning with “A.” Note how you can convert a street or segment handle to a range handle for use with **GsHandleGet** to return Street or Segment information.

```
   /* This example searches ZIP Code 80301 for streets beginning
      with “A” and no specific house number. */

   char                 tempstr[60];
   GsStreetHandle       hStreet;
   GsSegmentHandle      hSegment;
   GsRangeHandle        hRange;

   /* Use GsFindFirstStreet to get the first street handle that matches the criteria.
   */
   int iStat = GsFindFirstStreet( gs, &hStreet, GS_ZIP_SEARCH, “80301”, “A”, ““ );
   while( iStat == GS_SUCCESS )
   {
      /* We got a valid street handle which we convert to a range handle so that we
      can use GsHandleGet retrieve information to show the user. Even though we
      converted to a range handle, it really is still just a street handle, so we
      can only access those elements that are valid at the street level. See Appendix
      A for a complete list of valid elements for each handle type. */
      hSegment.hStreet = hStreet;
      hRange.hSegment = hSegment;
      GsHandleGet( gs, GS_PREDIR, &hRange, tempstr, sizeof(tempstr) );
      printf( “Street: %s “, tempstr );
      GsHandleGet( gs, GS_NAME, &hRange, tempstr, sizeof(tempstr) );
      printf( “%s “, tempstr );
      GsHandleGet( gs, GS_TYPE, &hRange, tempstr, sizeof(tempstr) );
      printf( “%s “, tempstr );
      GsHandleGet( gs, GS_POSTDIR, &hRange, tempstr, sizeof(tempstr) );
      printf( “%s\n”, tempstr );

      /* Find the first valid segment for the current street. */
      iStat = GsFindFirstSegment( gs, &hSegment );
      while( iStat == GS_SUCCESS )
      {
/* We’ve got a valid segment, convert to a range handle to get information to display.
   */
hRange.hSegment = hSegment;
printf( “   Segment: “ );
GsHandleGet( gs, GS_BLOCK_LEFT, &hRange, tempstr, sizeof(tempstr) );
printf( “%s “, tempstr );
GsHandleGet( gs, GS_BLOCK_RIGHT, &hRange, tempstr, sizeof(tempstr) );
printf( “%s “, tempstr );
GsHandleGet( gs, GS_SEGMENT_PARITY, &hRange, tempstr, sizeof(tempstr) );
printf( “%s “, tempstr );
GsHandleGet( gs, GS_SEGMENT_DIRECTION, &hRange, tempstr, sizeof(tempstr) );
printf( “%s \n”, tempstr );

/* Get the first valid range for the current segment. */
iStat = GsFindFirstRange( gs, &hRange );
while( iStat == GS_SUCCESS )
   {
   GsHandleGet( gs, GS_LORANGE, &hRange, tempstr, sizeof(tempstr) );
   printf( “      Range: %s “, tempstr );
   GsHandleGet( gs, GS_HIRANGE, &hRange, tempstr, sizeof(tempstr) );
   printf( “%s “, tempstr );
   GsHandleGet( gs, GS_BLOCK_LEFT, &hRange, tempstr,sizeof(tempstr) );
   printf( “%s “, tempstr );
   GsHandleGet( gs, GS_PREDIR, &hRange, tempstr,
      sizeof(tempstr) );
   printf( “%s “, tempstr );
   GsHandleGet( gs, GS_NAME, &hRange, tempstr,
      sizeof(tempstr) );
   printf( “%s “, tempstr );
   GsHandleGet( gs, GS_TYPE, &hRange, tempstr,
      sizeof(tempstr) );
   printf( “%s “, tempstr );
   GsHandleGet( gs, GS_POSTDIR, &hRange, tempstr, sizeof(tempstr) );
   printf( “%s “, tempstr );
   GsHandleGet( gs, GS_ZIP, &hRange, tempstr,sizeof(tempstr) );
   printf( “%s “, tempstr );
   GsHandleGet( gs, GS_LOUNIT, &hRange, tempstr,
      sizeof(tempstr) );
   printf( “%s “, tempstr );
   GsHandleGet( gs, GS_HIUNIT, &hRange, tempstr,sizeof(tempstr) );
   printf( “%s\n”, tempstr );

   /* Get the next valid range for this segment.*/
   iStat = GsFindNextRange( gs, &hRange );
}
if ( iStat == GS_ERROR ) break;

/* Get the next valid segment for this street. */
iStat = GsFindNextSegment( gs, &hSegment );
      }
      if ( iStat == GS_ERROR ) break;

      /* Get the next street that meets the initial criteria. */
      iStat = GsFindNextStreet( gs, &hStreet );
   }
```

APPENDIX C

Sample applications

This appendix discusses the sample applications, `geotest` and `geocoder`, provided with GeoStan.

NOTE: The following instructions are of using the sample application on Windows unless otherwise specified.

Geotest

`Geotest` is a sample program that demonstrates the functionality of GeoStan. `Geotest` is located in the 32bit (Windows) or samples (UNIX) directory. The sample code is located in the samples (UNIX) or Ms_c (Windows) directory.

You can run `geotest` without any arguments using the `test.ini` file or in interactive mode with arguments on the command line.

Configuring geotest

If you do not specify any arguments on the command line when initializing `geotest`, `geotest` initializes using the information in the `test.ini` file. You need to modify the `test.ini` file to fit the configuration of your system. You can find this file in the directory where you installed GeoStan in Test_fls.

The format of the `test.ini` file consists of the following lines:

Line 1	Search path for the Centrus data files.
Line 2	Search path and file name for the ZIP + 4 file (`Us.z9`).
Line 3	Search path and file name for GeoStan license file (`Geostan.lic`).
Line 4	Password for your GeoStan license.
Line 5	*(Optional)* DPV data directory.
Line 6	*(Optional)* Security key to initialize DPV functionality.

Line 7 *(Optional)* LACS^{Link} data directory.

Line 8 *(Optional)* Security key to initialize LACS^{Link} functionality.

Line 9 *(Optional)* Amount of memory GeoStan uses for memory mapped files. For more information, see “GsSetFileMemoryLimit” on page 384.

Line 10 *(Optional)* Sets a select number of processing options.

- A – Loads the files for address geocoding.
- Z – Loads the file for centroid geocoding.
- S – Loads spatial query files.
- E – Loads elevation files.
- P – Loads Assessor’s Parcel Number (APN) files.
- D - Loads the debug log.

Line 11 *(Optional)* Suite^{Link} data directory.

Line 12 *(Optional)* To exit the main menu, enter q or Q.

Line 13 *(Optional)* Sets the cache size (0, 1, or 3).

Line 14 *(Optional)* Sets the path to a file that contains all of your find properties.

For example:

```
C:\geostan\datasets
C:\geostan\dtatsets\us.z9
C:\licensefiles\geostan.lic
43218765
C:\geostan\datasets\DPV
9999-9999-9999-9999
C:\geostan\datasets\LACS
9999-9999-9999-9999
```

Using geotest

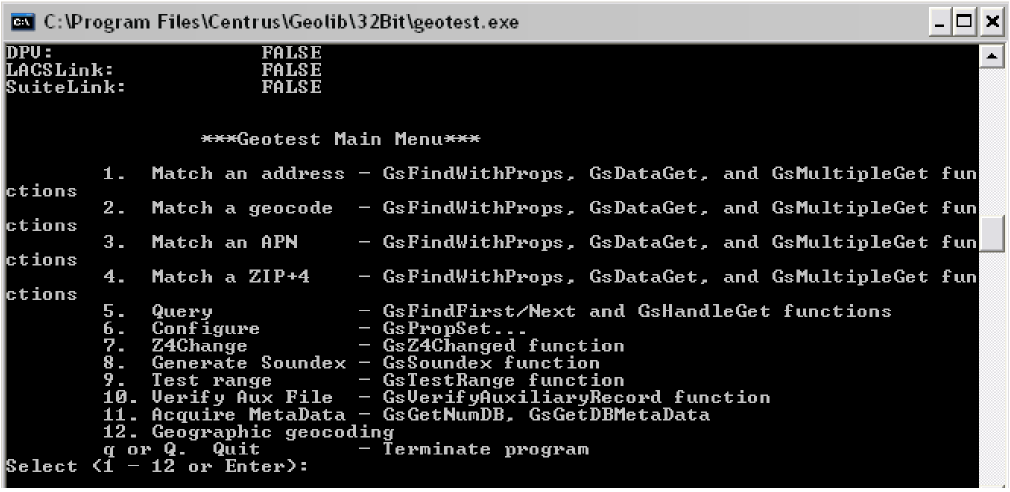
You can run `geotest` using `geotest.exe` located in the platform-specific file where you installed GeoStan, or you can run `geotest` by specifying arguments on a command line.

The following is an example of running `geotest` from the command line.

```
geotest searchpath z9path licensefile password -dp dpvpath -dk dpvSecuritykey -
lp lacslinkpath -lk lacslinksecuritykey -fml memorylimit -fp Find Properties Path
-tp SuiteLink Path -gc geocoder init flags -c cache size -sp search path -z z9file
-l license file -lpw license password -log turn on logging -q quit main menu -i
Init options (like 10 above)
```

If any of the setup criteria does not initialize correctly, a list of problems displays. Check and correct all of the listed problems and rerun `geotest`.

The Geotest Main Menu displays when all configurations are correct (sample menu shown below).



```
DPV:                 FALSE
LACSLink:            FALSE
SuiteLink:           FALSE


                ***Geotest Main Menu***

        1.   Match an address - GsFindWithProps, GsDataGet, and GsMultipleGet fun
ctions
        2.   Match a geocode  - GsFindWithProps, GsDataGet, and GsMultipleGet fun
ctions
        3.   Match an APN     - GsFindWithProps, GsDataGet, and GsMultipleGet fun
ctions
        4.   Match a ZIP+4    - GsFindWithProps, GsDataGet, and GsMultipleGet fun
ctions
        5.   Query            - GsFindFirst/Next and GsHandleGet functions
        6.   Configure        - GsPropSet...
        7.   Z4Change         - GsZ4Changed function
        8.   Generate Soundex - GsSoundex function
        9.   Test range       - GsTestRange function
        10.  Verify Aux File  - GsVerifyAuxiliaryRecord function
        11.  Acquire MetaData - GsGetNumDB, GsGetDBMetaData
        12.  Geographic geocoding
        q or Q.  Quit         - Terminate program
Select (1 - 12 or Enter):
```

The following contains a description of each menu item.

Match an address Prompts for an address to standardize and geocode. Returns a matched address or a list of possible matches.

Match a geocode Prompts for a geocode (latitude and longitude) to convert to an address. Returns the closest match. Returns a possible match; the address may not physically exist or may not accept mail delivery.

Match an APN Prompts for an APN, state FIPS code, and county FIPS code.

Match a ZIP + 4 Prompts for information to attempt a ZIP Code match.

Query Prompts for a ZIP Code or city and state, then asks if you want to search by street by name or soundex. If you want to output to a file, enter the file name at the `Name` prompt.

Configure Provides a way for you to configure the matching and output of `geotest`.

Z4Change Prompts for ZIP Code information, then indicates if the ZIP + 4 has changed since you last processed the address.

Generate Soundex Prompts for a string and returns a soundex value.

Test range Prompts for the house number and the low and high house number in the range to determine if the house number falls within a range.

Verify Aux File Prompts for the path of the auxiliary file to verify if any error exist in the file.

Acquire Metadata Provides information regarding the input datasets.

Geographic Geocoding Prompts for a city, a county, and a state to provide returns for CloseFlag, City\County\State, State, ResultCode, FIPS, LocationCode, Lat, and Long.

Verify Aux File Prompts for the path of the auxiliary file to verify if any error exist in the file.

q or Q Quits the application.

NOTE: You can press enter to exit `geotest` at any time.

Geocoder

`Geocoder` is a sample CASS-certified program that performs batch processing on fixed-length text files; standardizing and geocoding addresses during processing. You can modify `geocoder` to create your own application. The sample code is located in the samples (UNIX) or Ms_c (Windows) directory.

NOTE: PBBI does not support `geocoder` if you have made any changes to the source code. If you make changes to `geocoder`, you need to certify the application with the USPS for CASS compliance.

Using geocoder

`Geocoder` is located in the platform-specific directory where you installed GeoStan. You need to modify `Gcode.fmt` to match your file system structure prior to running `geocoder`. Sample format files and input files are located in the `Test_fls` directory (Windows) and `Common/Test_fls` directory (UNIX).

To execute `geocoder`, type the following at the command line:

```
geocoder formatFile inputFile outputFile [logFile]
```

`formatFile` Contains format initialization data used in processing the input file; similar to a Windows .fmt file. You can specify a dash (-) or `stdin` in place of the format file. There is no blank line required at the end of the file.

`inputFile` Path and name of the input addresses to process. The file must be a fixed length ASCII text file. You can specify a dash (-) or `stdin` in place of the format file.

`outputFile` Path and name of a new text file to hold the processed records. The output file normally has the same format as the input file, with the possible addition of fields at the end of each record. You can specify a dash (-), `stdout`, or `stderr` in place of the format file.

`logfile`

(*Optional*). Name of the log file where `geocoder` writes all messages to the specified log file instead of `stdout`. If specified, this must be the fourth parameter on the command line. `geocoder` disables the progress messages unless there is an override in the format file. You can specify a dash (-), `stdout`, or `stderr` in place of the log file.

The following are optional parameters you can enter at the command line for different processing of your input and output.

/l Limit on the number of input records to process. `geocoder` exits after reading the specified number of records.

/z Date of the Centrus data last used to process the file, in the format MMyyyy. `Geocoder` looks at the last date on which it processed a particular list, and compares each address to a Z4Change file to see if it has changed.

/r <Report File name> To specify a report file name, enter r/ <Report File name>, where the desired file name replaces what is between the <> in the command line. The /r command line option overrides the "reportFile" keyword in the format file.

The following are processing examples.

Scenario	Command
Z4 Change processing enabled	`geocoder formatFile inputFile outputFile /z mmyyyy`
Z4 change processing and the log file enabled	`geocoder formatFile inputFile outputFile logFile /z mmyyyy`
Input file read from stdin, output file written to stdout, logfile enabled, and Z4 change processing enabled	`geocoder formatFile - - logfile /z 062006`
Specify a processing limit of 100 records for test purposes	`geocoder formatFile - - /l 100`
Naming the report file.	`geocoder formatFile - - /r <Report File name>`

Using keywords in the format file

The following sections provide information on using control, input, and output keywords in the format file.

When using keywords in the format file, adhere to the following:

- Keywords are not case sensitive.
- Values are case sensitive on UNIX when specifying a path or filename
- Lines that begin with open brackets ([ ) or semicolons (;) are comment lines
- The application ignores blank lines.

- Do not put quotation marks around strings.

Control keywords

Control keywords use the syntax: `keyword = value`. The following table contains the control keywords you can use in the format file.

Keyword	Explanation
addressrange	Permits searching ranges of street addresses. • 0 = Disabled (standard default). • 1 = Enabled.
addrPntInterp	Enable Address Point Interpolation. • 0 = Disabled (standard default). • 1 = Enabled.
alternateLookupPref	• 1 = Prefer street lookup first (custom default). • 2 = Prefer firm lookup first. • 3 = Street only lookup (standard default).
alwaysFindCandidates	• 0 – *Default.* Do not search for more than one candidate. • 1 – enables continued search for candidates GS_ALWAYS_FIND_CANDIDATES or see GS_FIND_CENTRLN_PROJ_OF_POINT.
apnData	• 0 - *Default.* Do not load APN data. • 1 - Load APN data.
apnSearch	Performs an address lookup based on the Assessor's Parcel Number (APN). • 0 - *Default.* Disable Reverse APN searching. • 1 - Enable Reverse APN searching. – Only valid inputs are in APNID, inCountyFips, and inStateFips.
audit	• 0 – *Default.* No records copied to an audit file during processing. • 1 to 100000 – Writes every *n*th record to an audit output file indicated by `audFile`. See "Audit Report" on page 432 for more information. NOTE: This is an optional keyword that may also require a filename definition as an additional optional keyword.

Keyword	Explanation
audFile	Fully qualified path and filename for audit record redirection, which is required if not set to 0. A keyword is required if set to 1. NOTE: On z/OS you do not need to include a filename; code only the letter x.
CASSFile	Fully qualified path and filename for CASS Report. (Required if `matchMode` is set to 4.) NOTE: On z/OS you do not need to include a filename; code only the letter x.
CASSTemplateFile	Fully qualified filename of the CASS 3553 form used to generate the report. (Required in CASS mode.) NOTE: On z/OS you do not need to include a filename; code only the letter x.
DPVdirectory	Directory containing DPV data. NOTE: On z/OS you do not need to include a filename; code only the letter x.
DPVFind	Enable DPV Find. • 0 = Disabled • 1 = Enabled (standard default).
DPVsecurityKey	DPV security key.
elevationData	• 0 - *Default*. Do not load elevation data. • 1 - Load elevation data.
enhancedBuildingSearch	• 0 – *Default.* Disables enhanced building name searches. • 1 – Enables enhanced building name searches. NOTE: Not available with User Dictionaries.
firstLetterExpanded	Expands address search abilities to account for a missing or wrong first letter in the street name. • 0 = Disabled (standard default). • 1 = Enabled.
inReclen	*Required.* Record length of input file. Minimum line length to read if `inRecType` is 1.
inRecType	• 0 – Records are not delimited. (Must be 0 on z/OS.) • 1 – *Default.* Records are terminated with `cr/lf` (DOS/Windows) or `lf` (UNIX).

Keyword	Explanation
keepMultiMatch	• If the input address multi-matches (GS_ADDRESS_NOT_RESOLVED), return all candidate matches, each on a separate output line. This will increase the output length of the output file. • 0 = Disabled (default). • n = Enabled and returnes a maximum of n candidates. NOTE: resolveMultiMatch and keepMultiMatch are mutually exclusive.
keepNoMatch	• 0 – Do not output records that do not match. • 1 – *Default.* Output all records.
LACSdirectory	Fully qualified path for LACS^{Link} data. NOTE: On z/OS you do not need to include a filename; code only the letter x.
lacsFind	Enable LACS^{Link} Find. • 0 = Disabled. • 1 = Enabled (standard default).
LACSsecuritykey	LACS^{Link} security key.
licensePath	*Required.* Path to your GeoStan license file (`geostan.lic`).
MailerAddress	Address of your organization. Required for DPV and LACS^{Link} initialization.
MailerCity	City where your organization resides. Required for DPV and LACS^{Link} initialization.
MailerName	Name of your organization. Required for DPV and LACS^{Link} initialization.
MailerState	2-character state abbreviation where your organization resides. Required for DPV and LACS^{Link} initialization.
MailerZip	9-digit ZIP Code where your organization resides. Required for DPV and LACS^{Link} initialization.
matchMode	• 0 – Exact (GS_MODE_EXACT). • 1 – Close (GS_MODE_CLOSE). • 2 – *Default.* Relaxed (GS_MODE_RELAX). • 4 – CASS (GS_MODE_CASS). • 7 - Custom (GS_MODE_CUSTOM).
mixedCase	• 0 – *Default.* Return results in upper case. (USPS preferred.) • 1 – Return results in mixed case.

Keyword	Explanation
multiLine	• 0 – *Default.* Address is not multi-line. • 1 – Address is multi-line. Set your preferred address line using `multiPref`. NOTE: This is an optional keyword that may also require a filename definition as an additional optional keyword.
multiPref	• 0 – *Default.* If 2-line address, select first line. If multi-line address, select bottom-most line. • 1 – If 2-line or multi-line address, prefer P.O. Box. Ignored if processing in CASS mode. • 2 – If 2-line or multi-line address, prefer street address. Ignored if processing in CASS mode.
mustMatchAddrNumber	Custom matching option to ensure house number is matched exactly. • 0 = Disabled. • 1 = Enabled (default).
mustMatchMainAddr	Custom matching option to ensure street name, predir, postdir, and type match exactly. • 0 = Disabled (default). • 1 = Enabled.
mustMatchCity	Custom matching option to ensure city matches exactly. • 0 = Disabled (default). • 1 = Enabled.
mustMatchState	Custom matching option to ensure state matches exactly. • 0 = Disabled (default). • 1 = Enabled.
mustMatchZipCode	Custom matching option to ensure zipcode matches exactly. • 0 = Disabled (default). • 1 = Enabled.
nonStand	Redirects non-standardized records to a non-standard output file, putting only standardized records in the original output files as specified on the command line. May be non-existant, 0 (Off) (default) or 1 (On). NOTE: See "Audit Report" on page 432 for more information.

Keyword	Explanation
nStdFile	Fully qualified path and filename for non-standardized record output. (Required if `nonStand` is greater than 0.) NOTE: On z/OS you do not need to include a filename; code only the letter x.
offsetDist	Deprecated. The number of feet GeoStan will squeeze streets when interpolating the geocode for a house number. In addition, the number of feet used to set the geocode off the street. Use squeezeDist and setbackDist keywords instead.
centerlineOffset	Centerline Offset Distance in feet, GS_FIND_CENTERLINEOFFSET.
outReclen	*Required.* Record length of output file.
outRecType	• 0 – Records are not delimited. (Must be 0 on z/OS) • 1 – *Default.* Records are terminated with `cr/lf` (DOS/Windows) or `lf` (UNIX).
password	*Required.* GeoStan password to enable operation. NOTE: On z/OS you do not need to include a filename; code only the letter x.
path	*Required.* Search path for GeoStan data files; delimited with semicolons. For example `c:/;d:/geocode`. NOTE: On z/OS you do not need to include a filename; code only the letter x.
prefZipOverCity	Prefer ZIP over city when address is the same. • 0 – Disables (default). • 1 – Enabled.
printCandidates	Return all candidate matches, each on a separate line. This will increase the legnth of the output file. • 0 – Disables (default). • 1 – Enabled and returns a maximum of n candidates NOTE: printCandidates and keepMultiMatch are mutually exclusive.
removeRecordDelimiter	• 0 – Do not remove line feed characters. • 1 – If `inRecType` is 0 (indicating records are not delimited), look for and remove line feed characters anyway

Keyword	Explanation
reportFile	Fully qualified path and file name for statistics report file. If this keyword is not present, `geocoder` writes the report file `geocoder.rpt` to the current working directory. **NOTE:** On z/OS you do not need to include a filename; code only the letter x.
resolveMultiMatch	If the input address multi-matches (GS_ADDRESS_NOT_RESOLVED), select the first candidate to resolve the match. • 0 – Disables (default). • 1 – Enabled. **NOTE:** resolveMultiMatch and keepMultiMatch are mutually exclusive.
reverseGeoAddress	Specifies how addresses and unranged streets are processed with reverse geocoding. • 0 – *Default.* Does not match to an address. • 1 – Matches to points and interpolated addresses, but does not match to any unranged segments. • 2 – Matches to all types of street segments.
reverseGeointersection	Specifies how GeoStan handles reverse geocoding intersections close to the input geocode. • 0 – Does not return intersections. • 1 – Returns intersections.
searchDist	Deprecated. Used for reverse geocoding. It defines, in feet, the search area.
setbackDist	The number of feet GeoStan will set the geocode back off the street segment when interpolating a geocode from a house number on a segment. Incompatible with offsetDist. Sets the value for GS_FIND_STREET_OFFSET.
showMessages	• 0 – Supresses messages from GsErrorGetEx (default). • 1 – Prints messages from GsErrorGetEx.
showProgress	Specifies the frequency with which the application writes status messages indicating the number of processed records. • 0 – No progress message. • 10 – Default. Display messages every 10 records. **NOTE:** By default, the program sets this value to 0 if the command line specifies a log file or if the application is running under z/OS.

Keyword	Explanation
stopAt	The number of sequential records you want processed. stopAt=10000 will end after 10K input records have been processed.
streetCentroid	Specifies whether or not to return a street segment geocode as an automatic geocoding fallback. • 0 – *Default.* Street locator disabled. • 1 – Return street segment geocode.
squeezeDist	This keyword has been deprecated. The number of feet GeoStan squeezes street segments when interpolating geocodes from house numbers. Reverse Geocoding also uses this same value to reverse interpolate a house number from a geocode along the same segment. Incompatible with offsetDist. Sets the value for GS_FIND_CORNER_OFFSET.
suiteLinkDirectory	Directory containing Suite^{Link} data.
suitelinkFind	Enable Suite^{Link} Find. • 0 = Disabled. • 1 = Enabled (standard default).
userDictionaryPref	Obsolete, warning will be logged.
uMatchOnly	• 0 – *Default.* Process all records whether previously matched or not. Used when processing in CASS mode. • 1 – Process only records that the application has not previously processed successfully. Pass input directly to output without processing if `outLat` exists (not blank), `outLon` exists (not blank), `outMatchCode` exists (not error and not blank), `outLocCode` exists (not error and not blank), and `matchMode` does not equal 4.
useCentroids	• 0 – Do not assign centroid geocodes. • 1 – *Default.* Use if no address geocode is available. • 2 – Use ZIP centroid geocoding only (no address matching). • Requires z9dir if non 0.
usePathOrder	• 0 – Uses the default database search order, determined by the GeoStan database priority ranking. Value 0 is the default and the behavior is therefore backwards compatible with previous versions of geocoder. • 1 – Sets the search order according to how the directories in the path keyword are specified. See also GS_FIND_DB_ORDER.

Keyword	Explanation
z4ChangeDate	Date of the Centrus data media used to process the address file, in the format *mmddyy*. For z/OS only.
z9dir	*Optional.* Full name and path of the Z9 file, for example `d:/us.z9`. See useCentroids above.

Input and output keywords

All input and output fields have the following syntax:

```
keyword=start_pos, length
```

start_pos Offset for the first character of the field. The first position of the specified offset is one.

length Length of the field.

For example, `inFirm=10,50` indicates the firm field starts at position 10 and is 50 characters wide.

`geocoder` does not have field validation. Before running large jobs, Pitney Bowes Business Insight recommends testing the setup file on a small subset of records from the larger file.

If your input file does not contain the fields necessary to store the output data, you can add fields to the output record by specifying a larger `outReclen` size and adding the necessary output fields to the end of the record structure. For example, if your input record length is `200`, you can set the output record length to 240 and have `outAddress = 201, 40`.

NOTE: If you are processing in CASS mode, `geocoder` alerts you if you do not output required CASS fields and have the parameter `CASSFile` initialized with the path and filename for the CASS report file. Once complete, `geocoder` writes the data to the file specified.

The following table contains information on the input fields.

Field	Explanation
inAddress	Address line. When using this field, set `multiLine` to 0.
inAddr2	Second address line. When using this field, set `multiLine` to 0. Use `multiPref` to specify a preference for P.O. Box or street address. Ignore if processing in CASS mode.
inCity	City name.
inFirm	Firm name.
inLastline	Combined city, state, and ZIP Code.

Field	Explanation
inLat	Latitude of the geocode for the desired address.
inLine1 - inLine6	Multiline mode input fields. When using, set multiLine to 1.
inLon	Longitude of the geocode for the desired address.
inAPNID	Assessors parcel number (APN). NOTE: You can only specify the `inAPNID`, `inCountyFIPS`, and `inStateFIps` keywords when requesting an APN match. Specifying any other keywords when using these values results in an error.
inCountyFips	3-digit County FIPS Code. NOTE: You can only specify the `inAPNID`, `inCountyFIPS`, and `inStateFIps` keywords when requesting an APN match. Specifying any other keywords when using these values results in an error.
inStateFips	2-digit State FIPS Code. NOTE: You can only specify the `inAPNID`, `inCountyFIPS`, and `inStateFIps` keywords when requesting an APN match. Specifying any other keywords when using these values results in an error.
inState	State name or abbreviation.
inUrb	Urbanization name. Only valid for Puerto Rico.
inZip	ZIP Code.
inZip4	ZIP + 4.

The following table contains information on the output fields.

Field	Length (with null terminator)	Description
`centerlineBearing`	3	Compass direction, in decimal degrees, from the point data match to the centerline match. Measured clockwise from 0 degrees north.
`centerlineDataType`	2	Data type used to make the centerline match • 0 – USPS • 1 – Tiger • 2 – TANA • 3 – Sanborn point-level • 6 – NAVTEQ • 7 – TANA point-level • 8 – Centrus point • 9 – Auxiliary file
`centerlineDist`	8	Distance between the input point and the centerline in feet
`centerlineIsAlias`	3	Matched centerline record located via an alias index
`centerlineLat`	11	Latitude in millionths, for a centerline match
`centerlineLon`	11	Longitude in millionths, for a centerline match
`centerlineName`	40	Primary street name, for a centerline match
`centerlinePostDir`	2	Street postfix directional, for a centerline match
`centerlinePreDir`	2	Street prefix directional, for a centerline match
`centerlineSegDir`	2	• *F* – Numbers are forward on a centerline match • *R* – Numbers are reversed on a centerline match
`centerlineSegID`	11	Unique segment ID from data vendors, specific to a centerline match
`centerlineType`	4	Street type (street suffix), for a centerline match
`outAddress`	60	Address line
`outAddressShort`	`60`	Address line less than 30 characters when possible.
`outAddr2`	60	Second address line

Field	Length (with null terminator)	Description
`outAPNID`	46	Assessor's Parcel Number (APN)
`outAuxUserData`	301	User data from the auxiliary file. Blank if no auxiliary file.
`outBlock`	15	15-digit census block ID
`outBlockSfx`	1	Single character Census Block ID suffix
`outCandidateNum`	2	Returns the candidate number when keepMultiMatch or printCandidates is enabled. Value will be blank or 0-n to indicate candidate number in the possible match list.
`outCarrt`	4	Carrier route
`outCASSHeader`	2	• 0 – *Default*. Do not write the CASS header to the geocoder output file • 1 – Write the CASS header to the geocoder output file
`outCBSADivName`	127	CBSA division name
`outCBSADivNumber`	5	CBSA division number
`outCBSAMetroFlag`	1	• Y – CBSA is Metropolitan Statistical Area • N – CBSA is Micropolitan Statistical Area • Blank – Data is unavailable
`outCBSAName`	127	CBSA name
`outCBSANumber`	5	CBSA number
`outCSAName`	127	CSA name
`outCSANumber`	3	CSA number
`outChkDigit`	1	Check digit
`outCity`	28	City name
`outCityName`	28	City name for the matched address
`outCityPref`	28	Preferred city name for the matched address
`outCityShort`	28	Short City Name less than13 characters when possible.
`outCmsaNumber`	4	CMSA number
`outCmsaName`	127	CMSA name
`outCounty`	5	County FIPS code

Field	Length (with null terminator)	Description
outCountyName	127	County name
outDatatype	2	Data type used to make the match: • 0 – USPS data • 1 – Tiger data • 2 – TANA data • 3 – Sanborn point-level data • 6 – NAVTEQ data • 7 – TANA point-level data • 8 – Centrus point data • 9 – Auxiliary file data • 10 - User Dictionary
outDflt	1	Matched to Highrise or Rural Route
outDPBC	2	Delivery point bar code
outDPVConfirm	1	DPV confirmation indicator
outDPVVacant	2	DPV Vacant Indicator.
outDPVCMRA	1	DPV CMRA indicator
outDPVFalsePositive	1	DPV false-positive indicator
outDPVFootnote1	2	DPV Footnote 1
outDPVFootnote2	2	DPV Footnote 2
outDPVFootnote3	2	DPV Footnote 3
outDPVFootnote4	2	DPV Footnote 4
outDPVFootnote5	2	DPV Footnote 5
outDPVFootnote6	2	DPV Footnote 6
outDPVNoStat		DPV no stat indicator.
outEWS	1	EWS denied match
outFirm	40	Firm name
outHiRiseDflt	1	Highrise default flag
outHouseNumber	11	House number of entered address
outHouseNumber2	11	House number of entered address.
outIsAlias	3	Matched record located via an alias index

Field	Length (with null terminator)	Description
`outIntersect`	1	Intersection flag
`outLastLine`	60	Complete last line (city, state, ZIP + 4)
`outLastLineShort`	60	Complete Last Line (city,state, zip+4) less than 30 characters when possible.
`outLACS`	1	LACS^{Link} code
`outLACSRetCode`	2	LACS^{Link} return code
`outLACSInd`	1	LACS^{Link} indicator
`outLat`	10	Latitude in millionths
`outLocCode`	4	Location code
`outLon`	11	Longitude in millionths
`outLOTCode`	1	LOT code
`outLOTNum`	4	LOT number
`outMailStop`	60	Mail Stop
`outMatchCode`	4	Match code
`outMatchedDb`	5	Database matched against. Use in conjunction with GsGetDBMetadata.
`outMCDName`	41	Minor Civil Division name from the auxiliary file. Blank if no auxiliary file match.
`outMCDNumber`	6	Minor Civil Division number from the auxiliary file. Blank if no auxiliary file match.
`outMsaName`	30	MSA name
`outMsaNumber`	4	MSA number
`outName`	40	Primary street name
`outName2`	40	Second street name (cross street)
`outName2Short`	40	Intersection Cross Street Short Name .
`outNameShort`	40	Street Short Name corr to outAddressShort.
`outNearestDist`	8	Distance, in feet, from the input location to the matched street segment, point address, or intersection.
`outNumMultiple`		Return value from GsNumMultiple() not GsDataGet() call.

Field	Length (with null terminator)	Description
`outParCenElevation`	7	Elevation, in feet, at the parcel centroid
`outPMBDesignator`	5	Private mail box designator; always “PMB”
`outPMBNumber`	9	Private mail box number
`outPercentGeocode`	5	Percentage along the street segment to the interpolated match. The range is 0.0 - 100.0. The range is always 0.0 for point data and intersections.
`outPointID`	10	Returns the unique point ID of an address
`outPostDir`	2	Street postdirectional
`outPostDir2`	2	Cross street postdirectional
`outPostDir2Short`	2	Intersection Cross Street Short Post-directional.
`outPostDirShort`	2	Street Short Post-directional corr to outAddressShort.
`outPreDir`	2	Primary street predirectional
`outPreDir2`	2	Secondary street predirectional
`outPreDir2Short`	2	Intersection Cross Street Short Pre-directional.
`outPreDirShort`	2	Street Short Pre-directional corr to outAddressShort.
`outRecordType`	1	USPS range record type
`outResultCode`	11	Result code.
`outRRteDflt`	1	Rural route default flag
`outSegID`	11	Unique segment ID from data vendor
`outSegDirection`	2	• *F* – Numbers are forward • *R* – Numbers are reversed
`outState`	2	2-letter state abbreviation
`outStreetSide`	2	Side of the street of the matched address • *L* – Left side • *R* – Right side • *B* – Both sides • *U* – Unknown side
`outSuileLinkRetCode`	3	Suite^{Link} Return Code.

Field	Length (with null terminator)	Description
`outTimeGsFind`	10	Runtime for GsFind for each address processed by Geocoder.
`outType`	4	Street type (also called street suffix)
`outType2`	4	Cross street type (also called street suffix)
`outType2Short`	4	Intersection Cross Street Short Type.
`outTypeShort`	4	Street Short Type corr to outAddressShort.
`outUnitNumber`	11	Unit number
`outUnitNumber2`	11	Second unit number in the address line
`outUnitType`	4	Unit type (designator)
`outUnitType2`	4	Second unit type in the address line
`outUrbName`	30	Urbanization name for Puerto Rico
`outZip`	5	ZIP Code
`outZIP4`	4	ZIP + 4
`outZ4Changed`	2	• N – ZIP + 4 has not changed • Y – ZIP + 4 changed
`outZIP9`	9	9-digit ZIP Code
`outZIP10`	10	9-digit ZIP Code with hyphen

NOTE: When you redirect output data and specify a CASS header in the format file, `geocoder` writes the CASS header as the last record of the output file instead of the first record.

Generating Report Files

The following sections provide information on generating report files.

Audit Report

If the keyword `audit` is greater than 0 in the FMT file, `geocoder` outputs the audit file. For example if `audit = 1000`, `geocoder` writes every 1000th record to the audit output file. The file is named using the path and filename initialized with `audFile` in the format file.

The information shown in both the Input Address and Output Results sections is dependent on the keywords you specified in the FMT file. The match code and location code is included at the end of each audit entry for troubleshooting purposes.

Non-standardized File

If the keyword nonStand is greater than 0 in the format file geocoder generates the non-standardized output file. The file is named using the path and filename initialized as the nStdFile keyword in the format file. geocoder sends all records that receive an Exxx match code to this file instead of to the output file. Initializing the nonStand keyword overrides a keyword keepNoMatch setting of 0 and outputs the non-standardized records to nStdFile.

Statistics Report

geocoder creates a status report file (geocoder.rpt by default) in the current working directory with every pass of the data. The file contains a statistical compilation of the most recent pass of geocoder. Each successive geocoder pass with the same format file overwrites the previous status report file. If you want to maintain historical data, backup this file prior to the next geocoder pass or modify the reportFile keyword in the format file.

APPENDIX D

Status codes

This appendix contains the different status codes used within GeoStan through `GS_MATCH_CODE` and `GS_LOC_CODE` used with `GsDataGet`.

For more information on Map Marker result codes, please refer to Map Marker documentation.

Match codes

GeoStan returns match codes that indicate the portions of the address that matched or did not match to the GeoStan Directory file. If GeoStan could not make a match, the match code begins with `E` and the remaining digits indicate why the address did not match. The digits do not specifically refer to which address elements did not match, but rather why the address did not match.

The following table contains the match code values. You can find a description of the hex digits for the different match codes in the table following the match code table.

Code	Description
`Ahh`	Same as Shh, but indicates match to an alias name record or an alternate record.
`Chh`	Street address did not match, but located a street segment based on the input ZIP Code or city.
`D00`	Matched to a small town with P.O. Box or General Delivery only.
`Ghh`	Matched to an auxiliary file.
`Hhh`	House number was changed.

Code	Description
Jhh	Matched to a User Dictionary.
Nxx	Matched to the nearest address. Used with reverse geocoding. The following are the only values for N: • NS0 – Nearest street center match (nearest street segment interpolated) • NS1 – Nearest unranged street segment • NP0 – Nearest point address • NX0 – Nearest intersection
P	Successful Reverse APN lookup match.
Qhh	Matched to USPS range records with unique ZIP Codes. CASS rules prohibit altering an input ZIP if it matches a unique ZIP Code value.
Rhh	Matched to a ranged address.
Shh	Matched to USPS data. This is considered the best address match, because it matched directly against the USPS list of addresses. S is returned for a small number of addresses when the matched address has a blank ZIP + 4.
Thh	Matched to a street segment record. Street segment records do not contain ZIP Code information. If you enter a ZIP Code, the application returns the ZIP Code you entered. If the input city and state has only one ZIP Code, the application returns that ZIP Code.
Uhh	Matched to USPS data but cannot resolve the ZIP + 4 code without the firm name or other information. CASS mode returns an E023 (multiple candidates for a match) error code.
Xhhh	Matched to an intersection of two streets, for example, “Clay St & Michigan Ave.” The first hex digit refers to the last line information, the second hex digit refers to the first street in the intersection, and the third hex digit refers to the second street in the intersection. **NOTE:** The USPS does not allow intersections as a valid deliverable address.
Yhhh	Same as Xhhh, but an alias name record was used for one or both streets.
Z[a]	No address given, but verified the provided ZIP Code .

[a] Zh may be returned if Correct Last Line is set to True. For more information see Correct last line match codes and Using correct last line.

The following table contains the description of the hex digits for the match code values.

Code	In first hex position means:	In second and third hex position means:
0	No change in last line.	No change in address line.
1	ZIP Code changed.	Street type changed.
2	City changed.	Predirectional changed.
3	City and ZIP Code changed.	Street type and predirectional changed.

Code	In first hex position means:	In second and third hex position means:
4	State changed.	Postdirectional changed.
5	State and ZIP Code changed.	Street type and postdirectional changed.
6	State and City changed.	Predirectional and postdirectional changed.
7	State, City, and ZIP Code changed.	Street type, predirectional, and postdirectional changed.
8	ZIP + 4 changed.	Street name changed.
9	ZIP and ZIP + 4 changed.	Street name and street type changed.
A	City and ZIP + 4 changed.	Street name and predirectional changed.
B	City, ZIP, and ZIP + 4 changed.	Street name, street type, and predirectional changed.
C	State and ZIP + 4 changed.	Street name and postdirectional changed.
D	State, ZIP, and ZIP + 4 changed.	Street name, street type, and postdirectional changed.
E	State, City, and ZIP + 4 changed.	Street name, predirectional, and postdirectional changed.
F	State, City, ZIP, and ZIP + 4 changed.	Street name, street type, predirectional, and postdirectional changed.

The following table describes the values returned when the application cannot find a match code

Code		Description
Ennn[a]		Indicates an error, or no match. This can occur when the address entered does not exist in the database, or the address is badly formed and cannot be parsed correctly. The last three digits of an error code indicate which parts of an address the application could not match to the database.
	nnn = 000	No match made.
	nnn = 001	Low level error.
	nnn = 002	Could not find data file.
	nnn = 003	Incorrect GSD file signature or version ID.
	nnn = 004	GSD file out of date. Only occurs in CASS mode.
	nnn = 010	No city and state or ZIP Code found.
	nnn = 011	Input ZIP not in the directory.

Code		Description
	nnn = 012	Input city not in the directory.
	nnn = 013	Input city not unique in the directory.
	nnn = 014	Out of licensed area. Only occurs if using Group 1 licensing technology.
	nnn = 015	Record count is depleted and license has expired.
	nnn = 020	No matching streets found in directory.
	nnn = 021	No matching cross streets for an intersection match.
	nnn = 022	No matching segments.
	nnn = 023	Unresolved match.
	nnn = 024	No matching segments. (Same as 022.)
	nnn = 025	Too many possible cross streets for intersection matching.
	nnn = 026	No address found when attempting a multiline match.
	nnn = 027	Invalid directional attempted.
	nnn = 028	Record also matched EWS data, therefore the application denied the match.
	nnn = 029	No matching range, single street segment found
	nnn = 030	No matching range, multiple street segments found

[a] Ehnn may be returned if Correct Last Line is set to True. For more information see Correct last line match codes and Using correct last line.

Correct last line match codes

As mentioned in Using correct last line, when set to True, GS_FIND_CORRECT_LASTLINE corrects elements of the output last line, providing a good ZIP Code or close match on the soundex even if the address would not match or was non-existent.

The feature works when GS_FIND_ADDRCODE is True and the address does not match a candidate or when GS_FIND_Z_CODE is True and only last line information is input. The match codes returned are similar to Z and Ennn in that the first letter remains the same with the second digit changing.

Code		Description
Zh		No address given, but verified the provided ZIP Code .
	h = 0	No change in last line.
	h = 1	ZIP Code changed.
	h = 2	City changed.
	h = 3	City and ZIP Code changed.
	h = 4	State changed.
	h = 5	State and ZIP Code changed.
	h = 6	State and City changed.
	h = 7	State, City, and ZIP Code changed.
	h = 8	ZIP + 4 changed.
	h = 9	ZIP and ZIP + 4 changed.
	h = A	City and ZIP + 4 changed.
	h = B	City, ZIP, and ZIP + 4 changed.
	h = C	State and ZIP + 4 changed.
	h = D	State, ZIP, and ZIP + 4 changed.
	h = E	State, City, and ZIP + 4 changed.
Ehnn		Indicates an error, or no match. This can occur when the address entered does not exist in the database, or the address is badly formed and cannot be parsed correctly. The second digit of the error code is a hex digit which details the changes that were made to the last line information to correct the last line. The last two digits of an error code indicate which parts of an address the application could not match to the database.
	h = 0	No change in last line.
	h = 1	ZIP Code changed.
	h = 2	City changed.
	h = 3	City and ZIP Code changed.
	h = 4	State changed.
	h = 5	State and ZIP Code changed.
	h = 6	State and City changed.

Code		Description
	h = 7	State, City, and ZIP Code changed.
	h = 8	ZIP + 4 changed.
	h = 9	ZIP and ZIP + 4 changed.
	h = A	City and ZIP + 4 changed.
	h = B	City, ZIP, and ZIP + 4 changed.
	h = C	State and ZIP + 4 changed.
	h = D	State, ZIP, and ZIP + 4 changed.
	h = E	State, City, and ZIP + 4 changed.
	nn = 00	No match made.
	nn = 01	Low level error.
	nn = 02	Could not find data file.
	nn = 03	Incorrect GSD file signature or version ID.
	nn = 04	GSD file out of date. Only occurs in CASS mode.
	nn = 10	No city and state or ZIP Code found.
	nn = 11	Input ZIP not in the directory.
	nn = 12	Input city not in the directory.
	nn = 13	Input city not unique in the directory.
	nn = 14	Out of licensed area. Only occurs if using Group 1 licensing technology.
	nn = 15	Record count is depleted and license has expired.
	nn = 20	No matching streets found in directory.
	nn = 21	No matching cross streets for an intersection match.
	nn = 22	No matching segments.
	nn = 23	Unresolved match.
	nn = 24	No matching segments. (Same as 022.)
	nn = 25	Too many possible cross streets for intersection matching.
	nn = 26	No address found when attempting a multiline match.
	nn = 27	Invalid directional attempted.
	nn = 28	Record also matched EWS data, therefore the application denied the match.

Code		Description
	`nn = 29`	No matching range, single street segment found
	`nn = 30`	No matching range, multiple street segments found

Location codes

GeoStan returns location codes that indicate the accuracy of the assigned geocode.

A Location Code of E indicates a location code is not available. This usually occurs when you have requested ZIP Code centroids of a high quality, and one is not available for that match. It can occur infrequently when GeoStan does not have a 5-digit centroid location. GeoStan can also return an E location code type when it cannot standardize an input address and there is no input ZIP Code. In this case, do not assume the ZIP Code returned with the non-standardized address is the correct ZIP Code because GeoStan did not standardize the address; therefore, GeoStan does not return geocoding or Census Block information.

Address location codes

Address location codes detail the known qualities about the geocode. An address location code has the following characters.

1st character	Always an A indicating an address location.	
2nd character	May be one of the following	
	C	Interpolated address point location.
	G	Auxiliary file data location
	I	Application infers the correct segment from the candidate records
	P	Point-level data location
	R	Location represents a ranged address.
	S	Location on a street range
	X	Location on an intersection of two streets
3rd and 4th characters	Digit indicating other qualities about the location.	

The following table contains the address codes.

Code		Description
AGn		Indicates an auxiliary file for a geocode match where n is one of the following values:
	n = 0	The geocode represents the center of a parcel or building.
	n = 1	The geocode is an interpolated address along a segment.
	n = 2	The geocode is an interpolated address along a segment, and the side of the street cannot be determined from the data provided in the auxiliary file record.
	n = 3	The geocode is the midpoint of the street segment.
APnn		Indicates a point-level geocode match representing the center of a parcel or building, where nn is one of the following values:
	nn = 00	User Dictionary centroid. Geocode returned by a User Dictionary.
	nn = 02	Parcel centroid Indicates the center of an assessor's parcel (tract or lot) polygon. When the center of an irregularly shaped parcel falls outside of its polygon, the centroid is manually repositioned to fall inside the polygon as closely as possible to the actual center.
	nn = 04	Address point Represents field-collected GPS points with field-collected address data.
	nn = 05	Structure centroid Indicates the center of a building footprint polygon, where the building receives mail or has telephone service. Usually a residential address consists of a single building. For houses with outbuildings (detached garages, shed, barns, etc.), only the residences have a structure point. Condominiums and duplexes have multiple points for each building. Larger buildings, such as apartment complexes, typically receive mail at one address for each building and therefore individual apartments are not represented as discrete structure points. Shopping malls, industrial complexes, and academic or medical center campuses where one building accepts mail for the entire complex are represented as one point. When addresses are assigned to multiple buildings within one complex, each addressed structure is represented by a point. If the center of a structure falls outside of its polygon, the center is manually repositioned to fall inside the polygon.
	nn = 07	Manually placed Address points are manually placed to coincide with the midpoint of an assessor's parcel's street frontage at a distance from the center line.

Code		Description
	nn = 08	Front door point Represents the designated primary entrance to a building. If a building has multiple entrances and there is no designated primary entrance or the primary entrance cannot readily be determined, the primary entrance is chosen based on proximity to the main access street and availability of parking.
	nn = 09	Driveway offset point Represents a point located on the primary access road (most commonly a driveway) at a perpendicular distance of between 33-98 feet (10-30 meters) from the main roadway.
	nn = 10	Street access point Represents the primary point of access from the street network. This address point type is located where the driveway or other access road intersects the main roadway.
	nn = 21	Base parcel point The Centrus point data includes individual parcels that may be "stacked".These stacked parcels are individually identified by their unit or suite number, and GeoStan is able to match to this unit number and return the correct APN.If an input address is for a building or complex, without a unit number.The "base" parcel information returns and will not standardize to a unit number or return additional information such as an APN.
AIn		The correct segment is inferred from the candidate records at match time.
ASn		House range address geocode. This is the most accurate street interpolated geocode available.
AIn, ASn, and ACnh share the same qualities for the 3rd character n as follows:		
	n = 0	Best location.
	n = 1	Street side is unknown. The Census FIPS Block ID is assigned from the left side; however, there is no assigned offset and the point is placed directly on the street.
	n = 2	Indicates one or both of the following: • The address is interpolated onto a TIGER segment that did not initially contain address ranges. • The original segment name changed to match the USPS spelling. This specifically refers to street type, predirectional, and postdirectional. NOTE: Only the second case is valid for non-TIGER data because segment range interpolation is only completed for TIGER data.
	n = 3	Both 1 and 2.
	n = 7	Placeholder. Used when starting and ending points of segments contain the same value and shape data is not available.
ACnh		

Code		Description
	The ACnh 4th digit characteristics are as follows:	
	h = 0	Represents the interpolation between 2 points, both coming from User Dictionaries.
	h = 1	Represents the interpolation between 2 points. The low boundary came from a User Dictionary and the high boundary, from a non-User Dictionary.
	h = 2	Represents the interpolation between 1 point and 1 street segment end point, both coming from User Dictionaries.
	h = 3	Represents the interpolation between 1 point (low boundary) and 1 street segment end point (high boundary). The low boundary came from a User Dictionary and the high boundary from a non-User Dictionary.
	h = 4	Represents the interpolation between 2 points. The low boundary came from a non-User Dictionary and the high boundary from a User Dictionary.
	h = 5	Represents the interpolation between 2 points, both coming from non-User Dictionaries.
	h = 6	Represents the interpolation between 1 point (low boundary) and 1 street segment end point (high boundary). The low boundary came from a non-User Dictionary and the high boundary from a User Dictionary.
	h = 7	Represents the interpolation between 1 point and 1 street segment end point and both came from non-User Dictionaries.
	h = 8	Represents the interpolation between 1 street segment end point and1 point, both coming from User Dictionaries.
	h = 9	Represents the interpolation between 1 street segment end point (low boundary) and1 point (high boundary). The low boundary came from a User Dictionary and the high boundary from a non-User Dictionary.
	h = A	Represents the interpolation between 2 street segment end points, both coming from User Dictionaries.
	h = B	Represents the interpolation between 2 street segment end points. The low boundary came from a User Dictionary and the high boundary from a non-User Dictionary.
	h = C	Represents the interpolation between 1 street segment end point (low boundary) and1 point (high boundary). The low boundary came from a non-User Dictionary and the high boundary from a User Dictionary.
	h = D	Represents the interpolation between 1 street segment end point and1 point, both coming from non-User Dictionary.
	h = E	Represents the interpolation between 2 street segment end points. The low boundary came from a non-User Dictionary and the high boundary from a User Dictionary.

Code		Description
	h = F	Represents the interpolation between 2 street segment end points, both coming from non-User Dictionaries.
ARn		Ranged address geocode, where n is one of the following:
	n = 1	The geocode is placed along a single street segment, midway between the interpolated location of the first and second input house numbers in the range.
	n = 2	The geocode is placed along a single street segment, midway between the interpolated location of the first and second input house numbers in the range, and the side of the street is unknown. The Census FIPS Block ID is assigned from the left side; however, there is no assigned offset and the point is placed directly on the street.
	n = 4	The input range spans multiple USPS segments. The geocode is placed on the endpoint of the segment which corresponds to the first input house number, closest to the end nearest the second input house number.
	n = 7	Placeholder. Used when the starting and ending points of the matched segment contain the same value and shape data is not available.
AXn		Intersection geocode, where n is one of the following:
	n = 3	Standard single-point intersection computed from the center lines of street segments.
	n = 8	Interpolated (divided-road) intersection geocode. Attempts to return a centroid for the intersection.

Street centroid location codes

Street centroid location codes indicate the Census ID accuracy and the position of the geocode on the returned street segment. A street centroid location code has the following characters.

1st character	Always C indicating a location derived from a street segment.
2nd character	Census ID accuracy based on the search area used to obtain matching Street Segment.
3rd character	Location of geocode on the returned street segment.

The following table contains the values and descriptions for the location codes.

Character position	Code	Description
2nd Character		
	B	Block Group accuracy (most accurate). Based on input ZIP Code.
	T	Census Tract accuracy. Based on input ZIP Code.
	C	Unclassified Census accuracy. Normally accurate to at least the County level. Based on input ZIP Code.
	F	Unknown Census accuracy. Based on Finance area.
	P	Unknown Census accuracy. Based on input City.
3rd Character		
	C	Segment Centroid.
	L	Segment low-range end point.
	H	Segment high-range end point.

ZIP + 4 centroid location codes

ZIP + 4® centroid location codes indicate the quality of two location attributes: Census ID accuracy and positional accuracy. A ZIP + 4 centroid location code has the following characters.

1st character	Always Z indicating a location derived from a ZIP centroid.
2nd character	Census ID accuracy.
3rd character	Location type.
4th character	How the location and Census ID was defined. Provided for completeness, but may not be useful for most applications.

The following table contains the values and descriptions for the location codes.

Character position	Code	Description
2nd Character		
	B	Block Group accuracy (most accurate).

Character position	Code	Description
	T	Census Tract accuracy.
	C	Unclassified Census accuracy. Normally accurate to at least the County level.
3rd Character		
	5	Location of the Post Office that delivers mail to the address, a 5-digit ZIP Code centroid, or a location based upon locale (city). See the 4th character for a precise indication of locational accuracy.
	7	Location based upon a ZIP + 2 centroid. These locations can represent a multiple block area in urban locations, or a slightly larger area in rural settings.
	9	Location based upon a ZIP + 4 centroid. These are the most accurate centroids and normally place the location on the correct block face. For a small number of records, the location may be the middle of the entire street on which the ZIP + 4 falls. See the 4th character for a precise indication of locational accuracy.
4th Character		
	A	Address matched to a single segment. Location assigned in the middle of the matched street segment, offset to the proper side of the street.
	a	Address matched to a single segment, but the correct side of the street is unknown. Location assigned in the middle of the matched street segment, offset to the left side of the street, as address ranges increase.
	B	Address matched to multiple segments, all segments have the same Block Group. Location assigned to the middle of the matched street segment with the most house number ranges within this ZIP + 4. Location offset to the proper side of the street.
	b	Same as methodology B except the correct side of the street is unknown. Location assigned in the middle of the matched street segment, offset to the left side of the street, as address ranges increase.
	C	Address matched to multiple segments, with all segments having the same Census Tract. Returns the Block Group representing the most households in this ZIP + 4. Location assigned to t he middle of the matched street segment with the most house number ranges within this ZIP + 4. Location offset to the proper side of the street.
	c	Same as methodology C except the correct side of the street is unknown. Location assigned in the middle of the matched street segment, offset to the left side of the street, as address ranges increase.

Character position	Code	Description
	D	Address matched to multiple segments, with all segments having the same County. Returns the Block Group representing the most households in this ZIP + 4. Location assigned to the middle of the matched street segment with the most house number ranges within this ZIP + 4. Location offset to the proper side of the street.
	d	Same as methodology D except the correct side of the street is unknown. Location assigned in the middle of the matched street segment, offset to the left side of the street, as address ranges increase.
	E	Street name matched; no house ranges available. All matched segments have the same Block Group. Location placed on the segment closest to the center of the matched segments. In most cases, this is on the mid-point of the entire street.
	F	Street name matched; no house ranges available. All matched segments have the same Census Tract. Location placed on the segment closest to the center of the matched segments. In most cases, this is on the mid-point of the entire street.
	G	Street name matched (no house ranges available). All matched segments have the same County. Location placed on the segment closest to the center of the matched segments. In most cases, this is on the mid-point of the entire street.
	H	Same as methodology G, but some segments are not in the same County. Used for less than .05% of the centroids.
	I	Created ZIP + 2 cluster centroid as defined by methodologies A, a, B, and b. All centroids in this ZIP + 2 cluster have the same Block Group. Location assigned to the ZIP + 2 centroid.
	J	Created ZIP + 2 cluster centroid as defined by methodologies A, a, B, b, C, and c. All centroids in this ZIP + 2 cluster have the same Census Tract. Location assigned to the ZIP + 2 centroid.
	K	Created ZIP + 2 cluster centroid as defined by methodologies A, a, B, b, C, c, D, and d. Location assigned to the ZIP + 2 centroid.
	L	Created ZIP + 2 cluster centroid as defined by methodology E. All centroids in this ZIP + 2 cluster have the same Block Group. Location assigned to the ZIP + 2 centroid.
	M	Created ZIP+2 cluster centroid as defined by methodology E and F. All centroids in this ZIP + 2 cluster have the same Census Tract. Location assigned to the ZIP + 2 centroid.
	N	Created ZIP + 2 cluster centroid as defined by methodology E, F, G, and H. Location assigned to the ZIP + 2 centroid.

Character position	Code	Description
	V	Over 95% of addresses in this ZIP Code are in a single Census Tract. Location assigned to the ZIP Code centroid.
	W	Over 80% of addresses in this ZIP Code are in a single Census Tract. Reasonable Census Tract accuracy. Location assigned to the ZIP Code centroid.
	X	Less than 80% of addresses in this ZIP Code are in a single Census Tract. Census ID is uncertain. Location assigned to the ZIP Code centroid.
	Y	Rural or sparsely populated area. Census code is uncertain. Location based upon the USGS places file.
	Z	P.O. Box or General Delivery addresses. Census code is uncertain. Location based upon the Post Office location that delivers the mail to that address

Geographic centroid location codes

Geographic centroid location codes indicate the quality of two location attributes: .the geographic location and area type

1st character	Always G indicating a location derived from a geographic centroid.
2nd character	Geographic area type.

The following table contains the values and descriptions for the location codes.

Character position	Code	Description
2nd Character		
	M	Municipality (city).
	C	County.
	S	State.

APPENDIX E

CASS certification

The USPS® Coding Accuracy Support System (CASS™) is a service offered to mailers, service bureaus, and software vendors that improves the accuracy of delivery point codes, ZIP+4 Codes, 5-digit ZIP Codes, and carrier route information on mail. CASS provides a common platform to measure the quality of address matching software and useful diagnostics to correct software problems.

NOTE: For more information on CASS contact the USPS National Customer Support center at http://ribbs.usps.gov/files/cass/ or 1-800-642-2914.

To receive greater discounts on bulk mail postage, you can process your address list with USPS-certified address standardization software; reffered to as CASS certification. To qualify for the CASS certification™ discount, you must generate a CASS report with the standardization software during processing and submit the completed form with the mailing.

CASS report

USPS-certified address standardization software generates the CASS form PS3553 report, commonly refered to as the CASS report, during mailing list processing.

To generate a CASS report, the software compiles numerous statistics during processing, such as the number of 5-digit ZIP Codes and ZIP+4 Codes assigned to records in the address file. The application writes this information, along with other information required by the USPS, to an ASCII text file.

NOTE: You must use the DPV, Suite^{Link}, and LACS^{Link} options to produce a CASS form PS 3553.

You can use the Use.gsd and Usw.gse files on the Centrus Data Product Suite installation media for 135 days, but after 105 days you cannot create a CASS report.

For more information about CASS, visit the USPS National Customer Support Center (NCSC) Web site at http://ribbs.usps.gov/files/cass/, or call the NCSC at 1-800-642- 2914.

Obtaining USPS CASS certification for applications

This section outlines the process for obtaining Coding Accuracy Support System (CASS) certification for software from the United States Postal Service (USPS). Currently, the USPS certifies only software programs (executables) and not software libraries. CASS certification for applications created from code libraries must be obtained directly from the USPS, by using the following procedure.

Procedure for CASS-Certifying an Application Program

NOTE: Testing the application against the Stage 1 file is optional, but recommended.

1 Obtain a “Stage 1” file from the USPS National Customer Support Center (NCSC).

  This file can be obtained by either:

  - Downloading it from the NCSC Web site at http://www.usps.gov/ncsc/.
  - FAXing an order using the appropriate USPS form (the USPS returns the magnetic media of your choice). You may obtain a copy of this form by downloading it as an Adobe Acrobat file from the NCSC Web site, or by calling the National Customer Support Center at 1-800-642-2914.

  The Stage 1 file is a fixed-field, flat ASCII file containing approximately 100,000 addresses to be standardized. The USPS keeps an ASCII version of the Stage 1 file on their NCSC Web site at http://www.usps.gov/ncsc/.

2 Process the Stage 1 file using your standardization application. This testing process determines if the application outputs correctly standardized addresses in the proper format.

3 Compare your standardized addresses with those that the USPS provides. The Stage 1 file contains the correct standardized addresses.

4 Modify your application as necessary and repeat the standardization run as many times as needed.

5 Request a Stage 2 file after you are satisfied that your application meets the USPS standards.

  The Stage 2 file contains approximately 100,000 test addresses extracted and presented in the same makeup and mixture as the Stage 1 file—only no answers are provided. Return the Stage 2 standardized addresses to the USPS for grading. This grade determines if the application will be CASS certified.

  The USPS either ships this file to you on your preferred medium (for example tape or diskette), or creates a custom file for you to download from the Internet.

6 Process and return the Stage 2 file to the USPS within 10 days for grading.

  You can only return the file via USPS methods (mail or upload). The USPS rejects packages delivered by other carriers (such as FedEx). If you upload the file, notify the CASS department with an e-mail message.

  It generally takes a 1 to 3 days for the USPS to grade the file results.

If your standardized addresses pass USPS standards, the USPS sends you a certificate and the results of the grading.

If your standardized addresses fail USPS standards, the USPS notifies you and shows which addresses failed and the categories returned. Programs typically fail the CASS certification process because data is output incorrectly.

NOTE: You must obtain CASS certification for every release of your software application (that is, any time you recompile it and distribute it for general use).

For detailed information about the CASS certification process, review the documentation found on the Web site http://www.usps.gov/ncsc/programs/. You may also get information about the CASS/MASS program by calling the NCSC at 1-800-642-2914.

Modifying the CASS 3553 report template

After you have CASS-certified your application, you must also customize the CASS 3553 report template your application uses. Edit the template CASS3553.frm (which is an ASCII text file) by changing the “CASS CERTIFIED COMPANY NAME” and “CASS CERTIFIED SOFTWARE NAME” fields to the appropriate values.

APPENDIX F

USPS Link products

This appendix provides information on Delivery Point Validation (DPV), Locatable Address Conversion System process (LACS^{Link}), and Suite^{Link} available with GeoStan.

NOTE: GeoStan requires the DPV and LACS^{Link} options in CASS mode to receive ZIP + 4 and ZIP + 4 related output (DPBC, USPS record type, etc.,). GeoStan also requires the DPV Suite^{Link}, and LACS^{Link} options to produce a CASS form PS 3553.

DPV overview

Delivery Point Validation (DPV™) is a United States Postal Service (USPS®) technology that validates the accuracy of address information down to the physical delivery point. DPV is only available through a CASS-certified vendor, such as PBBI.

Previous address-matching software could only validate that an address fell within the low-to-high address range for the named street. By incorporating the DPV technology, you can resolve multiple matches and determine if the actual address exists. Using DPV reduces undeliverable-as-addressed (UAA) mail that results from inaccurate addresses, reducing postage costs and other business costs associated with inaccurate address information.

DPV also provides unique address attributes to help produce more targeted mailing lists. For example, DPV can indicate if a location is vacant and can identify commercial mail receiving agencies (CMRAs) and private mail boxes.

Although DPV can validate the accuracy of an existing address, you cannot use DPV to create address lists. DPV is a secure dataset of USPS addresses. For example, you can validate that 123 Elm Street Apartment 6 exists, but you cannot ask who lives in Apartment 6 or if there is an Apartment 7 at the same street address.

With DPV, your application automatically processes *every* ZIP+4 coded record against the DPV files. Using DPV may increase your ZIP+4 match rate, but may also increase processing time. Therefore, you may not wish to use DPV if you are not CASS certifying.

LACS^{Link} overview

The Locatable Address Conversion System (LACS) converts rural addresses to city-style addressees. LACS^{Link} is a USPS technology that provides mailers with an automated process to correct address lists for areas that have undergone LACS processing. Address list conversions occur when the LACS process modifies, changes, or replaces an address. This usually occurs due to one of the following: the conversion of rural routes and box numbers to city-style addresses, the renaming or renumbering of existing city-style addresses to avoid duplication, or the establishment of new delivery addresses.

LACS$^{\text{Link}}$ is a secure dataset of USPS addresses. Although LACS$^{\text{Link}}$ can validate the accuracy of an existing address, you cannot use LACS$^{\text{Link}}$ to create address lists.

NOTE: LACS$^{\text{Link}}$ is not run in multiple match searches.

False positive addresses overview

False positive addresses, also known as seed records, are addressees the USPS monitors to ensure users are not attempting to create a mailing list from the DPV or LACS$^{\text{Link}}$ data.

NOTE: Per the USPS regulations, PBBI must contact the USPS with the name and address of the organization for every false positive address encountered. If multiple incidents of artificial address detection occurs, the USPS may ask PBBI to suspend a customer's DPV or LACS$^{\text{Link}}$ processing capability.

If you encounter a false positive, you will receive a message. Processing continues to the end of your job, but further DPV or LACS$^{\text{Link}}$ processing is disabled. DPV or LACS$^{\text{Link}}$ processing is not available for subsequent jobs until you have reported the false-positive address encounter to PBBI and have received a new security key.

A message similar to the following appears when you encounter a false positive address:

DPV	DPV processing was terminated due to the detection of what is determined to be an artificially created address. No address beyond this point has been DPV validated. In Accordance with the License Agreement between USPS and PBBI, DPV shall be used to validate legitimately obtained addresses only, and shall not be used for the purpose of artificially creating address lists. The written Agreement between PBBI and the PBBI customer shall also include the same restriction against using DPV to artificially create address lists. Continuing use of DPV requires compliance with all terms of the License Agreement. If you believe this address was identified in error, please contact PBBI.
LACS/Link	LACS/Link processing was terminated due to the LACS/Link DEVELOPER LICENSEE PERFORMANCE REQUIREMENTS detection of what is determined to be an artificially created address. No address beyond this point has been LACS/Link processed. In accordance with the License Agreement between USPS and PBBI, LACS/Link shall be used to convert legitimately obtained addresses only, and shall not be used for the purpose of artificially creating address lists. The written Agreement between PBBI and the PBBI customer shall also include this same restriction against using LACS/Link to artificially create address lists. Continuing use of LACS/Link requires compliance with all terms of the License Agreement. If you believe this address was identified in error, please contact PBBI.

When implementing DPV and LACS^{Link} you need to create a false positive file that contains the Header Record and Detail Record information. You must provide this file to obtain a new security file from the PBBI Web site or from technical support.

For information purposes, the following tables contain the layout of the header and detail records of the false positive files for DPV and LACS^{Link}. The header record contains the mailer information from the Mailer Parameter Record and statistics gathered by the application.

Position	Length	Description	Format
1-40	40	Company name	alphanumeric
41-98	58	Address line	alphanumeric
99-126	28	City name	alphanumeric
127-128	2	State abbreviation	alphabetical
129-137	9	9-digit ZIP code	numeric
138-146	9	Total records DPV/LACS^{Link} processed	numeric
147-155	9	Total records DPV/LACS^{Link} matched	numeric
156-164	9	% match rate to DPV	numeric
165-173	9	% match rate to ZIP+4	numeric
174-178	5	Number of ZIP codes on file	numeric
179-180	2	Number of false-positives	numeric

NOTE: Positions 156-180 in the previous table do not exist in the LACS^{Link} false-positive header record.

The detail record contains false positive record information.

Position	Length	Description	Format
1-2	2	Street pre-directional	alphanumeric
3-30	28	Street name	alphanumeric
31-34	4	Street suffix abbreviation	alphanumeric
35-36	2	Street post-directional	alphanumeric
37-46	10	Address primary number	alphanumeric
47-50	4	Address secondary abbreviation	alphanumeric
51-58	8	Address secondary number	numeric

Position	Length	Description	Format
59-63	5	Matched ZIP code	numeric
64-67	4	Matched ZIP+4	numeric
68-180	113	Filler	

Data expiration

The USPS has determined that the ZIP+4 Directory data, DPV data, and LACS^{Link} data expire in 105 days for CASS processing. The date is measured from the release of the Postal database, which is the 15th of the month indicated on the PBBI data CD. For example, the June data release is good for 105 days from the 15th of June. However, in non-CASS processing modes, ZIP+4 data expires in 135 days.

Implementing LACS^{Link} and DPV

LACS^{Link} and DPV processing utilizes additional data. PBBI provides this data on separate CDs from the traditional GeoStan data, and is dependent on your contract with PBBI. For more information on installing DPV and LACS^{Link}data, see the *DPV and LACS/Link Release Notes.*

NOTE: DPV and LACS^{Link} are optional when processing records in CASS mode. However, you must use DPV and LACS^{Link} data for CASS certification.

When you implement DPV and LACS^{Link} you must first initialize GeoStan. After you have initialized GeoStan, you can initialize DPV and LACS^{Link}.

If you initialize DPV, GeoStan automatically uses DPV to resolve match candidatees. DPV will not delivery point validate unless you specifically request DPV output. If you do not specifically request DPV output DPV will never hit a false-positive address.

If you initialize LACS^{Link}, GeoStan automatically uses LACS^{Link} to convert addresses according to the guidelines created by the USPS

False positive report example code

The following code is an example of how to implement DPV and LACS^{Link} false positive reporting using the C language:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define GEOSTAN_PROPERTIES
#include "geostan.h"
int
```

```
main(int argc, pstr * argv)
{

   GsId gs;
   GsFunStat retcode;
   char buffer[GS_MAX_STR_LEN];
   char buffer2[GS_MAX_STR_LEN];
   FILE * DPVfalsPosOut;   /* DPV false positive file pointer */
   FILE * LACSfalsPosOut;  /* LACSLink false positive file pointer */
   GsFalsePosHeaderData FPheaderData; /* DPV/LACSLink false positive
                                      header record */
   GsFalsePosDetailData FPdetailData; /* DPV/LACSLink false positive
                                      detail record */
   char * mailerName = "PITNEY BOWES SOFTWARE, INC";
   char * mailerAddress = "4750 WALNUT ST STE 200";
   char * mailerCity = "BOULDER";
   char * mailerState = "CO";
   char * mailerZip = "803012532";

    PropList initProps;
   PropList statusProps;
   PropList findProps;
   qbool bVal;

    GsPropListCreate( &initProps, GS_INIT_PROP_LIST_TYPE );

    // Replace paths and keys with your installation
   GsPropSetStr( &initProps, GS_INIT_DPV_SECURITYKEY,
      "1237-5678-9abc-def0" );
   GsPropSetStr( &initProps, GS_INIT_DPV_DIRECTORY,
      "C:\\Program Files\\Centrus\\Datasets\\Current" );
   GsPropSetLong( &initProps, GS_INIT_DPV_DATA_ACCESS,
      DPV_DATA_FULL_FILEIO );
   GsPropSetBool( &initProps, GS_INIT_DPV, TRUE );
   GsPropSetStr( &initProps, GS_INIT_LACSLINK_SECURITY_KEY,
      "1237-5678-9abc-def0" );
   GsPropSetStr( &initProps, GS_INIT_LACSLINK_DIRECTORY,
      "C:\\Program Files\\Centrus\\Datasets\\Current" );
   GsPropSetBool( &initProps, GS_INIT_LACSLINK, TRUE );
   GsPropSetStr( &initProps, GS_INIT_SUITELINK_DIRECTORY,
      "C:\\Program Files\\Centrus\\Datasets\\Current" );
   GsPropSetBool( &initProps, GS_INIT_SUITELINK, TRUE );
   GsPropSetLong( &initProps, GS_INIT_GSVERSION,
      GS_GEOSTAN_VERSION );
   GsPropSetBool( &initProps, GS_INIT_OPTIONS_ADDR_CODE, TRUE );
   GsPropSetBool( &initProps, GS_INIT_OPTIONS_Z9_CODE, TRUE );
   GsPropSetStr( &initProps, GS_INIT_DATAPATH,
      "C:\\Program Files\\Centrus\\Datasets\\Current" );
   GsPropSetStr( &initProps, GS_INIT_Z4FILE,
      "C:\\Program Files\\Centrus\\Datasets\\Current\\us.z9" );
   GsPropSetLong( &initProps, GS_INIT_PASSWORD, 12345678 );
   GsPropSetLong( &initProps, GS_INIT_CACHESIZE, 2 );
   GsPropSetStr( &initProps, GS_INIT_LICFILENAME,
      "C:\\Program Files\\Centrus\\Geolib\\Geostan.lic" );

   GsPropListWrite( &initProps, "stdout", NULL, 0 );
   GsPropListCreate( &statusProps, GS_STATUS_PROP_LIST_TYPE );
```

```
 /* initialize GeoStan */
gs = GsInitWithProps( &initProps, &statusProps );
GsPropListWrite( &statusProps, "stdout", 0, 0 );
while ( GsErrorHas(gs) )
{

   GsErrorGetEx (gs, buffer, buffer2);
   printf ("%s\n%s\n", buffer, buffer2);
}

if ( gs == 0 )
{

   printf( "GeoStan failed to initialize.\n" );
   exit(1);
}

 // Verify DPV loaded correctly
retcode = GsPropGetBool( &statusProps,
   GS_STATUS_DPV_FILE_SECURITY, &bVal );
if ( GS_SUCCESS == retcode )
{

   if ( !bVal )
   {
      printf( "DPV security key failed to verify.\n" );
   }
   else if ( GS_SUCCESS == GsPropGetBool(&statusProps,
      GS_STATUS_DPV_FILE_ALL, &bVal) )
   {
      if ( !bVal )
         printf( "DPV data failed to initialize.\n" );
   }

}


retcode = GsPropGetBool( &statusProps,
   GS_STATUS_LACSLINK_FILE_SECUR, &bVal );
if ( GS_SUCCESS == retcode )
{
   if ( !bVal )
   {
      printf( "LACSLink security key failed to verify.\n" );
   }
   else if ( GS_SUCCESS == GsPropGetBool(&statusProps,
      GS_STATUS_LACSLINK_FILE_ALL, &bVal) )
   {
      if ( !bVal )
         printf( "LACSLink data failed to initialize.\n" );
   }

}


 retcode = GsPropGetBool( &statusProps,
   GS_STATUS_SUITELINK_FILE_ALL, &bVal );
if ( GS_SUCCESS == retcode )
```

```
{
   if ( !bVal )

      printf( "SuiteLink data failed to initialize.\n" );
}


GsPropListCreate( &findProps, GS_FIND_PROP_LIST_TYPE );
GsPropSetBool( &findProps, GS_FIND_ADDRCODE, TRUE );
GsPropSetBool( &findProps, GS_FIND_Z9_CODE, TRUE );
GsPropSetBool( &findProps, GS_FIND_FINANCE_SEARCH, TRUE );
GsPropSetLong( &findProps, GS_FIND_MATCH_MODE, GS_MODE_CASS );

GsClear( gs );
GsDataSet( gs, GS_FIRM_NAME, "DIXIES DAISYS FLOWER SERVICE");
GsDataSet( gs, GS_ADDRLINE, "74203 PERRANIA");
GsDataSet( gs, GS_LASTLINE, "GRANT CO 80448");
retcode = GsFindWithProps( gs, &findProps );

GsPropListWrite( &findProps, "stdout", 0, 0 );
while ( GsErrorHas(gs) )
{

   GsErrorGetEx (gs, buffer, buffer2);
   printf ("%s\n%s\n", buffer, buffer2);
}

 GsDataGet(gs, GS_OUTPUT, GS_DPV_FALSE_POS, buffer,
   sizeof(buffer));
if ( *buffer == 'Y' )
{

   /*
      A DPV false positive occurred.
      Write a false positive report
   */
   DPVfalsPosOut = fopen("DPVfalsePos.rpt", "w");
   if ( DPVfalsPosOut )
   {

      /* Get the false positive header data */
      memset(&FPheaderData, 0, sizeof(FPheaderData));
      strcpy(FPheaderData.MailersCompanyName, mailerName);
      strcpy(FPheaderData.MailersAddressLine, mailerAddress);
      strcpy(FPheaderData.MailersCityName, mailerCity);
      strcpy(FPheaderData.MailersStateName, mailerState);
      strcpy(FPheaderData.Mailers9DigitZip, mailerZip);
      GsDpvGetFalsePosHeaderStats(gs, &FPheaderData,
         sizeof(FPheaderData));
      /* Format the false positive header data */
      retcode = GsFormatDpvFalsePosHeader( gs,&FPheaderData,
         sizeof(FPheaderData), buffer, sizeof(buffer) );
      /* Write the header to the false positive file */
      if (retcode == GS_SUCCESS)
      {

         fprintf( DPVfalsPosOut,"%s\n", buffer );
      }
      else
      {

         GsErrorGetEx(gs, buffer, buffer2);
         printf( "Error calling GsFormatDpvFalsePosHeader:"
            "\n%s\n%s\n",              buffer, buffer2 );
      }

      /* Get the false positive detail data */
      retcode = GsDpvGetFalsePosDetail(gs, &FPdetailData,
         sizeof(FPdetailData));
      if (retcode != GS_SUCCESS)
      {

         GsErrorGetEx(gs, buffer, buffer2);
         printf( "Error calling GsDpvGetFalsePosDetail:"
            "\n%s\n%s\n",
            buffer, buffer2);
      }

      /* Format the false positive detail data */
      retcode = GsFormatDpvFalsePosDetail(gs, &FPdetailData,
         sizeof(FPdetailData), buffer, sizeof(buffer));
      /* Write the detail to the false positive file */
      if (retcode == GS_SUCCESS)
      {

         fprintf( DPVfalsPosOut,"%s\n", buffer );
      }
      else
      {
         GsErrorGetEx(gs, buffer, buffer2);
         printf( "Error calling GsFormatDpvFalsePosDetail:"
            "\n%s\n%s\n",
            buffer, buffer2);
      }

      fclose(DPVfalsPosOut);
   }
   else
   {

      printf("Failed to open DPV false positive file "
         "(errno =%d).\n", errno);
   }

}


GsClear( gs );
GsDataSet( gs, GS_ADDRLINE, "RR 1 BOX 6300");
GsDataSet( gs, GS_LASTLINE, "MOUNT SIDNEY VA 24467");
retcode = GsFindWithProps( gs, &findProps );

GsDataGet(gs, GS_OUTPUT, GS_LACSLINK_IND, buffer,
   sizeof(buffer));
if (*buffer == 'F')
{
```

```
/*
   A LACSLink false positive occurred.
   Write a false positive report */
LACSfalsPosOut = fopen("LACSfalsePos.rpt", "w");
if ( LACSfalsPosOut )
{

   /* Get the false positive header data */
   memset(&FPheaderData, 0, sizeof(FPheaderData));
   strcpy(FPheaderData.MailersCompanyName, mailerName);
   strcpy(FPheaderData.MailersAddressLine, mailerAddress);
   strcpy(FPheaderData.MailersCityName, mailerCity);
   strcpy(FPheaderData.MailersStateName, mailerState);
   strcpy(FPheaderData.Mailers9DigitZip, mailerZip);
   GsLACSGetFalsePosHeaderStats(gs, &FPheaderData,
      sizeof(FPheaderData));
   /* Format the false positive header data */
   retcode = GsFormatLACSFalsePosHeader(gs, &FPheaderData,
                         sizeof(FPheaderData), buffer,
                         sizeof(buffer));
   /* Write the header to the false positive file */
   if (retcode == GS_SUCCESS)
   {

      fprintf( LACSfalsPosOut,"%s\n", buffer );
   }
   else
   {

      GsErrorGetEx(gs, buffer, buffer2);
      printf("Error calling GsFormatLACSFalsePosHeader:"
         "\n%s\n%s\n", buffer,
         buffer2);
   }

   /* Get the false positive detail data */
   retcode = GsLACSGetFalsePosDetail(gs, &FPdetailData,
      sizeof(FPdetailData));
   if (retcode != GS_SUCCESS)
   {

      GsErrorGetEx(gs, buffer, buffer2);
      printf("Error calling GsLACSGetFalsePosDetail:"
         "\n%s\n%s\n", buffer,
         buffer2);
   }

   /* Format the false positive detail data */
   retcode = GsFormatLACSFalsePosDetail(gs, &FPdetailData,
      sizeof(FPdetailData), buffer, sizeof(buffer));
   /* Write the detail to the false positive file */
   if (retcode == GS_SUCCESS)
   {

      fprintf( LACSfalsPosOut,"%s\n", buffer );
   }
   else
```

```
            {

               GsErrorGetEx(gs, buffer, buffer2);
               printf("Error calling GsFormatDpvFalsePosDetail:"
                  "\n%s\n%s\n", buffer,
                  buffer2);
            }

            fclose(LACSfalsPosOut);
         }
         else
         {

            printf("Failed to open LACSLink false positive file"
               " (errno =%d).\n", errno);
         }

      }


      GsTerm( gs );
      GsPropListDestroy( &findProps );
      GsPropListDestroy( &initProps );
      GsPropListDestroy( &statusProps );

      return 0;
   }
```

Reporting a false positive address

You report false positive address matches and obtain a new security key in the same manner for both DPV and LACS^{Link}. You must provide the false positive report file to PBBI to obtain a replacement security key.

To report a false positive address match and obtain a new security key:

1. Enter the PBBI Support Web site, http://www.g1.com/support/login.asp, in a browser.

2. Log into the site, by entering your user ID and password.

   NOTE: If you do not know your user ID and password for the support Web site, select the *Need Your User ID or Password* link. Enter your email address as instructed. PBBI will email the user ID and password if your email address exactly matches the email in the PBBI customer database.

3. Click *My Products* from the column on the left of the PBBI Web site. A screen appears with a listing of all of your PBBI products.

4. Click on the appropriate product name. A screen appears with the platforms available for the product.

5 Select *View Details* from the right-most column. A screen appears with detailed information for the platform.

6 Select *Download DPV* or *Download LACS^{Link}* in the Database section. A window appears asking you for specific information.

7 Enter your old license key and attach your false-positive file by clicking the Browse button.

8 When prompted, save the file that contains the new security key to your machine.

You can now use the new security key located in the file you downloaded from the PBBI Web site when prompted by your application.

If you cannot access the PBBI Web site or need assistance, contact Technical Support at **800-367-6950**. Have your false-positive file ready to provide to the PBBI representative.

Understanding Suite^{Link}

The purpose of Suite$^{Link™}$ is to improve business addressing by adding known secondary (suite) numbers to allow delivery sequencing where it would otherwise not be possible. Suite^{Link} uses the input business name, street number location, and 9 digit ZIP+4 to return a unit descriptor (i.e. "STE") and unit number for that business.

As an example, when entering the following address with Suite^{Link} enabled in CASS mode...

UT Animal Research
910 Madison Ave
Memphis TN 38103

GeoStan returns the following:

UT Animal Research
910 Madison Ave STE 823
Memphis TN 38103

Or

UT Animal Research
910 Madison Ave #823
Memphis TN 38103

If you have licensed the Suite^{Link} processing option, you must install the Suite^{Link} data and set the Suite^{Link} initialization properties for GeoStan to process your address through Suite^{Link}. For more information on Suite^{Link} enums and functions, see the following sections:

- “Common enums for storing and retrieving data” on page 68 for C and page 248 for COBOL.
- GsFindWithProps properties

Suite^{Link} is required for CASS certification.

APPENDIX G

User defined data files

This chapter includes information on both auxiliary files and using User Dictionaries.

User Dictionary

This chapter includes information on creating User Dictionaries, source data requirements and required fields, and other information specific to working with User Dictionaries.

NOTE: User Dictionaries are not for use with CASS geocoding.

Understanding User Dictionary capabilities and requirements

The capabilities of User Dictionaries and the basic requirements for creating them are as follows.

- All fields supported by normal street geocoding can be included in User Dictionaries.
- Landmarks and place names are supported in User Dictionaries. Postal or geographic centroid geocoding are not supported in User Dictionaries.

- User Dictionaries support address browsing using partial street names or landmarks and place names.
- GSDs are necessary to create the User Dictionary. This is because the GSDs have some internal structure that must be available when creating a User Dictionary.

The results from a User Dictionary are similar to that from the GSD. For address matches where the first letter of the match code would be 'S', a User Dictionary match has the letter 'J'. The value of the GS_REC_TYPE is 'U'. Also, the enum GS_DATATYPE returns a new value for the User Dictionary record matches, see User Dictionary (GsFileStatusEx) for more information.

For example: SE9 is a match code for a match that comes from a GSD, while JE9 is for a match that comes from a User Dictionary. See Status codes for a complete description of match codes.

Source data requirements

The source data for User Dictionaries includes street data but can also include place names and intersections.

To create a User Dictionary, your source data must conform to the following requirements:

- Source records must include required fields, and these fields are mapped during the User Dictionary creation process. If a value of a required field is empty for a particular record, then that record will not be imported into the User Dictionary. Required fields may vary for different countries. The MapInfo table must contain specific fields, which GeoStan then uses to convert the table into the dictionary format. These input fields are described in Required Input Fields on page.
- Source records must be in a MapInfo table (TAB file). The TAB file requirements vary for different countries.
- Segments must have two or more defined endpoints to be loaded into a User Dictionary. Segments without endpoints are ignored.
- Segments that make up intersections must have one or more end points in the intersection for GeoStan to recognize it as an intersection. Source records can be either point objects or segments.
- Each row in the table is equivalent to a street segment.

Required input fields

You must specify the field names in the MapInfo table (TAB file) in order for the table to be translated into a User Dictionary. Certain fields are required and must be present in the MapInfo table. Other fields are optional, but are strongly recommended because there may be negative consequences if they are omitted. This is described in Optional (Recommended)

Input Fields on page Optional (recommended) input fields. If any of the required fields are missing, a missing field error code is returned.

The following table describes the required input fields.

Required fields	Description	Maximum field length
Left start address	Start of address range on left side of street.	10
Right start address	Start of address range on right side of street.	10
Left end address	End of address range on left side of street.	10
Right end address	End of address range on right side of street.	10
Street name	Name of street.	30
State abbreviation	Two-character state abbreviation.	2
Left ZIP Code	ZIP Code for left side of street.	5
Right ZIP Code	ZIP Code for the right side of the street.	5

Optional (recommended) input fields

The Left and Right Odd/Even Indicator fields are used to specify whether the sides of the street segment contain odd or even address ranges. Although these indicators are not required for creating a User Dictionary, it is important to use the Odd/Even Indicators when your data contains odd/even address numbers.

When the Odd/Even Indicator is specified, but is inconsistent with address numbers, the indicator is set to Both.

When the Odd/Even Indicator is not specified and both Start Address and End Address have values, the indicator is set to Both, unless the start and end address numbers are the same number. In that case, the indicator is set to Odd if the address numbers are odd, and set to Even if the address numbers are even.

When the Odd/Even Indicator is not specified and both Start Address and End Address have values, the indicator is set to Both (odd and even).

NOTE: If your table contains Odd/Even indicator information, we strongly recommend that you use the Odd/Even indicator fields. These fields ensure that your geocoded addresses are located on the correct side of the street. Omitting the fields when your data contains Odd/Even information may produce incorrect results.

The following table describes the optional input fields.

Optional fields	Description	Maximum field length
Left Odd/Even indicator*	Left side of the street contains only odd or even address ranges (O=odd, E=even, B=both)	1
Right Odd/Even indicator*	Right side of the street contains only odd or even address ranges (O=odd, E=even, B=both)	1
City*	City name	28
Left ZIP + 4 Code	4-digit ZIP + 4 add-on for left side of street.	4
Right ZIP + 4 Code	4-digit ZIP + 4 add-on for right side of street.	4
Left Census Block	Census Block ID for left side of street	15
Right Census Block	Census Block ID for right side of street	15
Place Name	Place name	40

* These fields are highly recommended.

User Dictionary file names and formats

GeoStan has some requirements for User Dictionary files that you must be aware of before you create a User Dictionary:

- Each User Dictionary has a base name of eight characters or fewer.
- Each User Dictionary resides in its own directory.
- The maximum length of a path to a User Dictionary is 1024 characters.
- The ZIP Code range in the MapInfo table for a User Dictionary is unlimited.

Because each User Dictionary resides in its own directory, User Dictionaries may share the same name. However, it is generally good practice to use a unique name for each User Dictionary.

Some of the output files are tied to the base name. The other output files have constant names. For example, the output files for a dictionary called ud1 are the following:

```
postinfo.jdr
postinfo.jdx
lastline.jdr
post2sac.mmj
```

```
geo2sac.mmj
sac2fn_ud.mmj
ud1.jdr
ud1.jdx
ud1.bdx
```

If your data includes place names, the dictionary contains the following files:

```
ud1.pdx
ud1.pbx
```

The dictionary also contains these log files:

```
ud1.log
ud1.err
```

Additional User Dictionary considerations

See the following topics for more information when working with User Dictionaries.

Data Access License

You must still have a valid access license to the data contained in the GSD when you are geocoding against your User Dictionary. For example, if you create a dictionary of New York streets and addresses, you must purchase the New York or entire U.S. GSD.

Use without GSD data files

To utilize a User Dictionary without the use of GSDs, the files listed below are required:

- ctyst.dir - The USPS City State table.
- parse.dir - The GeoStan dictionary

To perform postal centroid geocoding, in addition to a GSD or a User Dictionary and the files listed above, the following files are necessary:

- us.z9 - Postal centroid information.
- cbsac.dir - Required only if county names or CBSA/CSA data are needed.

CASS standards

You cannot geocode to CASS standards using a User Dictionary. This also means that the ParcelPrecision Dictionary cannot be used during CASS geocoding.

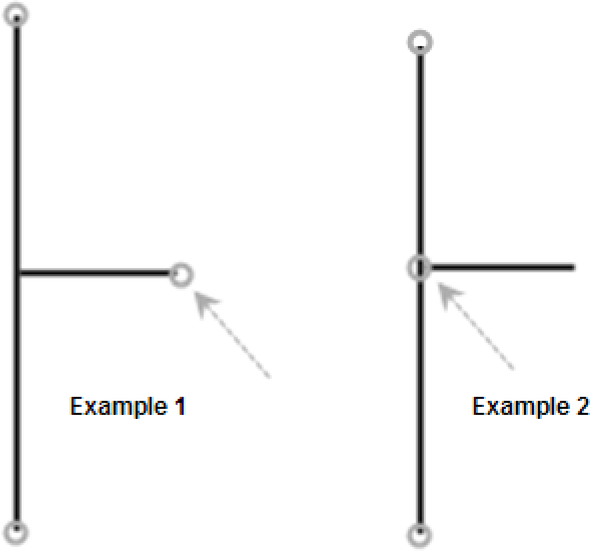
Address Range Order

GeoStan determines the order of the address range based on a comparison of the start and end addresses. The comparison produces the following results:

- If the end is greater than the start, the range is ascending.
- If the start is greater than the end, the range is descending.
- If the start is equal to the end, the range is ascending.

Street intersections and User Dictionaries

When geocoding to street intersections with a User Dictionary, GeoStan cannot recognize the intersections if one or more of the segments that make up the intersection does not have an end point at the intersection. This can happen when you create the User Dictionary from a customized street table in which some segments that terminate at intersections do not have end points (Example 1).



Example 1: Intersection in User Dictionary does not have end points for all segments. GeoStan does not recognize this as an intersection.

Example 2: Intersection in TIGER-based GSD includes end points for all segments. GeoStan geocodes to this intersection.

City lookup

GeoStan relies on USPS data to determine addresses. If a new address was input, it might not have been recognized despite the address being valid if it was not yet valid according to the USPS. An example of an input address that would not match against a UD:

1 Second Street
Stickville, NY 11111

In this example, the city is fictitious and the zip is made up. This would fail to match even with a UD record having that city and that zip, because they are not found in the USPS data. But a user may possess a UD with such a city and zip.

When matching to a UD record, GeoStan, if necessary, corrects the city name and/or zip code to the data that is in the UD record. GeoStan is now able to obtain matches for non-USPS cities and zips that were prevented from succeeding or which required temporary workarounds.

Using User Dictionaries with address point interpolation

An important part of the process of creating a User Dictionary is to specify a mapping of fields from your source data. See the *MapInfo User Dictionary Utility Product Guide*, for a complete discussion. There are two main categories of data fields: required and optional.

Of the optional fields, there are two that have an impact on the address point interpolation feature. These are the ″Left Odd/Even″ and ″Right Odd/Even″ fields. If these are not populated, the results from address point interpolation is less accurate.

Please be aware that aforementioned fields are not populated by source data obtained via MapInfo StreetPro. You must modify the source TAB file by adding the ″Left Odd/Even″ and ″Right Odd/Even″ indicator fields, and create queries to populate them. Source data obtained from other products, or your own data, may have similar issues.

To add the ″Left Odd/Even″ and ″Right Odd/Even″ indicator fields to a source TAB file, you must add them and then run a series of SQL update queries to populate them. The fields should be filled in with “O” (odd), “E” (even), or “B” (both). Below are the steps for adding these fields:

1 Add two 1-char columns to your TAB file.

   Naming each column, for example, Ind_Right and Ind_Left.

2 Perform the following updates to populate these fields:

   - Update <tablename>

   Set Ind_Left=″E″, Ind_Right=″O″

   Where From_Left mod 2=0 AND To_Left mod 2=0

   - Update <tablename>

Set Ind_Left="O", Ind_Right="E"

Where From_Left mod 2=1 AND To_Left mod 2=1

- Update <tablename>

Set Ind_Left="B", Ind_Right="B"

Where From_Left="" AND To_Left=""

NOTE: These example queries are simplified for illustrative purposes. Your actual queries may need to be more complex.

Preferring User Dictionary Matches

If you select Prefer User Dictionary Matches, candidates from the User Dictionary are given a higher score than a similar candidate from the GSD. The GS_FIND_DB_ORDER property is designed for situations where you feel that your User Dictionary is superior to the GSD, and therefore you prefer matches from the User Dictionary over matches from the GSD whenever possible.

GeoStan supports the creation and use of User Dictionaries based on your own source data. A User Dictionary can be used independently or as a supplement to the supplied GSD.

You can geocode using:

- User Dictionary (or multiple User Dictionaries) alone.
- Standard GSD.
- A combination of GSD and User Dictionaries

The results from a User Dictionary are similar to that from the GSD. For address matches where the first letter of the match code would be 'S', a User Dictionary match has the letter 'J'. The value of the GS_REC_TYPE is 'U'. Also, the enum GS_DATATYPE returns a new value for the User Dictionary record matches, see User Dictionary (GsFileStatusEx) for more information.

For example: SE9 is a match code for a match that comes from a GSD, while JE9 is for a match that comes from a User Dictionary. See Status codes for a complete description of match codes.

Auxiliary files

PBBI updates its data regularly to incorporate new rules by government entities and enhancements by third-party data providers. In some cases, your organization may have newer information that PBBI has not yet incorporated into the data files. Auxiliary files provide a way for you to process your input records against a file that includes these changes.

Creating your auxiliary files

This section contains information on creating auxiliary files, and contains the following topics:

- Auxiliary file requirements
- Record types
- Auxiliary file organization
- Default values

NOTE: On MVS, you must convert your auxiliary files to a VSAM KSDS data set. Use the JCL `yourprefix.GEOSTAN.CNTL(AFVSAM)` to create and load the VSAM file.

Auxiliary file requirements

GeoStan requires that the auxiliary file comply with the following:

- File must be a fixed-width text file
  - On Win32 and UNIX, text file must be ASCII
  - On MVS, each text file must be in EDCDIC, and the DDNAME for the auxiliary file must end in a number (for example, AUXFIL1)
- File must have a .gax extension on Win32 and UNIX
- File must have less than 500,000 records
- File must follow the column field order and lengths specified in "Auxiliary file layout" on page 480

Record types

You can include two types of records in your auxiliary file.

Street Records

A street record contains a range of one or more addresses on a street. To be a valid street record the record must have the following fields:

- ZIP Code
- Street name
- Street type abbreviation, if part of the address
- Predirectional abbreviation, if part of the address
- Postdirectional abbreviation, if part of the address
- Low house number within the street segment

- High house number within the street segment
- Beginning longitude of the street segment
- Beginning latitude of the street segment

In addition, a street record may NOT have:

- Secondary address information, such as unit numbers
- Mailstops
- Private mail boxes (PMBs)

Landmark Records

A landmark record represents a single site. To be a valid landmark record the record must have the following fields:

- ZIP Code
- Name of the landmark – placed in the street name field
- Beginning latitude of the landmark
- Beginning longitude of the landmark

In addition, a landmark record may NOT have the following fields:

- Street type abbreviation
- Predirectional abbreviation
- Postdirectional abbreviation
- Low house number
- High house number

During processing GeoStan ignores any record that does not comply with the preceding requirements.

Auxiliary file organization

You must comply with the following organizational rules when creating your auxiliary file.

- Use semicolons in the first column to indicate a row is a comment, not a data record; GeoStan ignores rows that begin with a semicolon.
- Order the records within the file by descending ZIP Code then descending street name for optimal performance.
- All records must represent one or both sides of a street. However, if using both sides of the street is not compatible with GeoTax.
- All records must represent segments that are straight lines. Records cannot represent a non-straight segment.
- If house numbers are present in the record, the house number range must be valid according to USPS rules documented in Publication 28.
- The numeric fields, such as ZIP Codes, must contain all numbers.
- Latitude and Longitude values must be in millionths of decimal degrees.

- Records cannot contain PO Box addresses.

Default values

GeoStan uses the following defaults if you do not include the values in the auxiliary file:

- House number parity = B (both odds and evens)
- Segment direction = F (forward) or A (ascending), these are interchangeable.
- Side of street = U (unknown)

Matching to auxiliary files

This section provides information on the matching performed by GeoStan to auxiliary files, and contains the following topics:

- Matching overview
- Record type matching rules
- Unavailable GeoStan features and functions
- Auxiliary match output

Matching overview

GeoStan performs the following steps when matching an input address to an auxiliary file.

1 GeoStan determines if there is an auxiliary file present.

GeoStan only accepts one auxiliary file. If more than one auxiliary files is present, GeoStan attempts to match against the first file. GeoStan ignores any additional auxiliary files for matching, regardless if Geostan found a match to the first auxiliary file.

If a record within the auxiliary files is invalid, GeoStan returns a message indicating the auxiliary file has an invalid record. GeoStan continues to process input addresses against the auxiliary file, but will not match to the invalid auxiliary file record.

2 If an auxiliary file is present, GeoStan first attempts to match to the auxiliary file.

GeoStan assumes that the auxiliary file is the most accurate data set and first attempts to find a match to the input address in the auxiliary file. If GeoStan cannot find a match in the auxiliary file, it continues to process as normal against the traditional GeoStan datasets.

NOTE: GeoStan only matches your input address to your auxiliary file if there is an exact match. Therefore, your input address list should be as clean as possible; free of misspellings and incomplete addresses.

3 If GeoStan finds an exact record match to the auxiliary file, it standardizes the match to USPS regulations and returns the output of the auxiliary file match.

NOTE: You cannot update the auxiliary file while GeoStan is running. If you want to update the auxiliary file, you need to terminate GeoStan before attempting to replace or edit the file.

Record type matching rules

When attempting a match against an auxiliary file, GeoStan abides by the following rules:

Street record match

- The input house number must fall within or be equal to the low and high house number values of the auxiliary record.
- The input house number must agree with the parity of the auxiliary record.
- The input ZIP Code must exactly match the ZIP Code of the auxiliary record.

Landmark record match

- The input data must contain both a ZIP Code and address line, and they must exactly match the values on the auxiliary record.
- The input address cannot have any other data, such as a house number, unit number, or Private Mail Box (PMB).

NOTE: GeoStan only matches the ZIP Code against the auxiliary file. GeoStan does not verify that the ZIP Code of the input address record is correct for the city and state. You should validate this information in your input address before processing against the auxiliary file.

Unavailable GeoStan features and functions

The following contains the features and functions that do not apply when GeoStan makes an auxiliary file match.

- GeoStan does not match to:
  - two-line addresses
  - multi-line addresses
  - intersection addresses
  - dual addresses
- You cannot use auxiliary file matching when processing in CASS mode
- GeoStan does not perform EWS, ZIPMove, LACS^{Link}, or DPV processing on auxiliary matches

- You cannot create an auxiliary file for the reverse geocoding option
- You can only access the auxiliary file with processing through the Find function. You cannot access the auxiliary file through the Find First/Next or MBR functions
- You can only accesses the auxiliary file logic using the address code option of the Find function; not the geocode option.
- The following are not executed:

C	COBOL	Java	.Net
GsGetCoords	GSSETSEL	GeoStan.getCoords	Coordinate
GsSetSelection	GSSSELR	GeoStan.select	GeoStan.select
GsSetSelectionRange	GSHGET	Range.select	Range.select
GsHandleGet	GSMGET	Street.getData	Street.getData
GsHandleGetCoords	GSMGH	Segment.getData	Segment.getData
GsMultipleGetHandle	GSHGCRD	Range.getData	Range.getData
		GeoStan.getRange	GeoStan.getRange
		Segment.getCoords	Segment.Coords

Auxiliary match output

Several standard GeoStan outputs do not apply to an auxiliary match since GeoStan matches to an exact auxiliary match and does not perform any additional validation for the match.

GeoStan provides special match codes and location code values for auxiliary matches. See Appendix D, "Status codes" for more information.

When GeoStan finds a match to an auxiliary file, the default output follows the following conventions:

- GeoStan formats the output of auxiliary file match as a street-style address.
- GeoStan follows the casing setting you indicate by the casing function. GeoStan does not maintain the casing in the auxiliary file for mixed cased values. For example, GeoStan returns O'Donnell as ODONNELL or Odonnell depending on the setting of the casing function.

NOTE: GeoStan does not change the casing for the User Data field.

- GeoStan removes spaces at the beginning and ending of fields in the auxiliary file.

NOTE: GeoStan does not remove spaces for the User Data field.

Auxiliary file layout

		Required				
Field	Description	For Street Segment Match	For Landmark Match	Requires Exact Match	Length	Position
ZIP Code	5-digit ZIP Code.	X	X	X	5	1-5
Street name	Name of the street or landmark.	X	X	X	30	6-35
Street type abbreviation	Street type. Also called street suffix. See the USPS Publication 28 for a complete list of supported street types.			X	4	36-39
Predirectional	USPS street name predirectional abbreviation. Supported values are N, E, S, W, NE, NW, SE, and SW.			X	2	40-41
Postdirectional	USPS street name postdirectional abbreviations. Supported values are N, E, S, W, NE, NW, SE, and SW.			X	2	42-43
RESERVED	RESERVED				4	44-47

Field	Description	Required		Requires Exact Match	Length	Position
		For Street Segment Match	For Landmark Match			
Low house number	Low house number of the address range.	X			11	48-58
High house number	High house number of the address range.	X			11	59-69
House number parity[a]	Parity of the house number in the range. • *E* – Even • *O* – Odd • *B* – Both				1	70
Segment direction	Direction the house numbers progress along the segment: • *F* – Forward (*default*) or A - Ascending • *R* – Reverse or D - Descending				1	71
RESERVED	RESERVED				1	72
FIPS state	US government FIPS state code.				2	73-74
FIPS county	US government FIPS county code.				3	75-77
Census tract	US Census tract number.				6	78-83
Census block group	US Census block group number.				1	84
Census block ID	US Census block ID number.				3	85-87
RESERVED	RESERVED				5	88-92
State abbreviation	USPS state abbreviation.				2	93-94
County name	Name of the county.				25	95-119
MCD code	Minor Civil Division code.				5	120-124
MCD name	Minor Civil Division name.				40	125-164
CBSA code	Core Based Statistical Area code.				5	165-169
CBSA name	Core Based Statistical Area name.				49	170-218

Field	Description	Required For Street Segment Match	 For Landmark Match	Requires Exact Match	Length	Position
RESERVED	RESERVED				5	219-223
City Name	City name. Overrides the city/state preferred city name upon a return.				40	224-263
RESERVED	RESERVED				237	264-500
User-defined data	User-defined data.				300	501-800
Record ID Number	User-defined unique record identifier.				10	801-810
Side of street	Side of the street for the address: • *L* – Left side • *R* – Right side • *B* – Both sides • *U* – Unknown side (*default*) This is relative to the segment end points and the segment direction.				1	811
Beginning longitude	Beginning longitude of the street segment in millionths of degrees.	X	X		11	812-822
Beginning latitude	Beginning latitude of the street segment in millionths of degrees.	X	X		10	823-832
Ending longitude	Ending longitude of the street segment in millionths of degrees.				11	833-843
Ending latitude	Ending latitude of the street segment in millionths of degrees.				10	844-853

a. For even and odd house number parity records, this specifies on which side of the street the house lays. For records containing both even and odd house numbers, the odd house numbers are on the specified side of the street, and the even house numbers are on the other side. This is a factor when using street offset.

GLOSSARY

ADC Area Distribution Center. A mail processing facility that receives and distributes mail destined for specific ZIP Code areas under the Managed Mail Program (MMP). An ADC is one of the points within the national MMP distribution network.

address elements The components of a street address, including house number, prefix direction, street name, street type, and postfix direction. These elements are parsed by GeoStan and should not be entered separately.

address geocoding See geocode, geocoding.

address standardization Address standardization is the process of taking an address and verifying that each component meets U.S. Postal Service guidelines for addresses. For example, when properly abbreviated, “123 Main Avenue” appears as “123 Main Ave.” During standardization, minor misspellings, dropped address elements, and abbreviations are corrected and the correct city, state, and ZIP Code are provided.

alias A recognized alternate for a street name maintained by association in the database.

alias information Data returned with certain enums when it exists. Not returned by all enums even if specifically requested.

alternate record Additional or differing information that may be available about a specific address but that differs from the base record. See the enums table for necessary flag settings.

AMC/AMF Airport Mail Center/Airport Mail Facility. A postal facility at an airport that receives, distributes, and dispatches mail transported by air.

base record The principle, rather than an alternate, record within the database.

block assignments (or blockface) For the assignment of ZIP + 4 codes, one side of a street, from one intersection to the next.

BMC Bulk Mail Center. A highly mechanized mail processing plant that distributes Standard Mail (A) and Periodicals in bulk form and Package Services (B) in piece and bulk form.

carrier route The addresses to which a carrier delivers mail. In common usage, a carrier route includes city routes, rural routes, highway contract routes, post office box sections, and general delivery units.

CASS Coding Accuracy Support System. A service offered to mailers, service bureaus, and software vendors that improves the accuracy of delivery point codes, ZIP + 4 codes, 5-digit ZIP Codes, and carrier route information on mail. CASS provides a common platform to measure the quality of address matching software and useful diagnostics to correct software problems.

CBSA A statistical geographic entity consisting of the county or counties associated with at least one core (urbanized area or urban cluster) of at least 10,000 population, plus adjacent counties having a high degree of social and economic integration with the core as measured through commuting ties with the counties containing the core. Metropolitan and Micropolitan Statistical Areas are the two categories of Core Based Statistical Areas.

CBSA Division A subdivision of CBSA.

Census block ID The 15-digit identification number used to specify a particular aggregate or block of addresses associated through census processes.

Census FIPS Code/Census ID See FIPS code.

centroid The calculated center of an area. The coordinates that define a centroid are the average of the sets of coordinates that describe the area.

centroid match An address that has, through geocoding, been found to match a defined geocentroid.

city state key A six-character USPS key that uniquely identifies a city name in the city/state file. Each city has a unique city state key.

city state name facility code The character (A-G, K, M, N, P, S, or U) that specifies the type of postal facility.

CMSA name, CMSA number Consolidated Metropolitan Statistical Area. The name represents the largest city in a statistical area. The number represents a 4-digit FIPS code.

County The primary legal division of every state except Alaska and Louisiana. A number of geographic entities are not legally designated as a county, but are recognized by the U.S. Census Bureau as equivalent to a county for data presentation purposes. These include the boroughs, city and boroughs, municipality, and census areas in Alaska; parishes in Louisiana; and cities that are independent of any county in Maryland, Missouri, Nevada, and Virginia. They also include the municipios in Puerto Rico, districts and islands in American Samoa, municipalities in the Northern Mariana Islands, and islands in the Virgin Islands of the United States. Because they contain no primary legal divisions, the Census Bureau treats the District of Columbia and Guam each as equivalent to a county (as well as equivalent to a state) for data presentation purposes. In American Samoa, a county is a minor civil division.

coordinates See latitude/longitude coordinates.

CPO Community Post Office. A contract postal unit that provides service in small communities where independent post offices have been discontinued. A CPO bears its community's name and ZIP Code as part of a recognized address.

CRIS Carrier Route Information System. The official city delivery scheme that lists all city and non-city delivery post offices, which is available to mailers in a standardized format. It contains schemes for city routes, rural routes, highway contract routes, post office box sections, and general delivery units. The data are formatted by ZIP Code, street name, and street number range. Delivery statistics (possible deliveries) for each carrier route are also included in the file.

CRS Carrier Route Sort. also carrier route presort mail — Mail that the mailer arranges by carrier route to qualify for discount postage rates. The mail requires no primary or secondary distribution. The term is a general descriptor of the available rates for this type of preparation, including Enhanced Carrier Route Standard Mail, automation carrier route First-Class Mail, carrier route Periodicals, and carrier route Bound Printed Matter. Except for automation rates, this mail usually does not bear a barcode. (See also sequence.) (Also called Enhanced Carrier Route Standard Mail and route-sequenced mail.)

CSA A geographic entity consisting of two or more adjacent Core Based Statistical Areas (CBSAs) with employment interchange measures of at least 15. Pairs of CBSAs with employment interchange measures of at least 25 combine automatically. Pairs of CBSAs with employment interchange measures of at least 15, but less than 25, may combine if local opinion in both areas favors combination.

datum A mathematical model of the Earth used to calculate the coordinates on any map, chart, or survey system. Surveyors take an ellipsoid model of the Earth and fix it to a base point. The North American Datum (NAD) is the official reference ellipsoid used for the primary geodetic network in North America.

DDC Delivery Distribution Center. The location where mail is sorted into pigeonhole cases, trays, sacks, machine bins, or pouches in order to group pieces with a common destination for transportation to the post office of address. It may be done by manual, mechanical, or automated means. The term is also applied to the distributed mail itself.

directionals A geographic address line component that precedes (predirectional) or follows (postdirectional) the street name.

DMM Domestic Mail Manual. The USPS manual that contains the basic standards governing U.S. domestic mail services; descriptions of the mail classes and special services and conditions governing their uses; and standards for rate eligibility and mail preparation. Domestic mail is classified by size, weight, content, service, and other factors.

DPBC The Delivery Point Bar Code is a POSTNET barcode that consists of 62 bars with beginning and ending frame bars and 5 bars each for the 9 digits of the ZIP + 4 code, the last 2 digits of the primary street address number (or post office box, and so on), and a correction digit. The DPBC allows automated sorting of mail to the carrier level in walk sequence.

DPC certified Delivery point code certified. A software or hardware device that meets U.S.P.S. standards for evaluating a properly standardized ZIP + 4 code address and determines the correct 2-digit DPC and checkdigit.

eLOT The Enhanced Line of Travel (eLOT) Product was developed to provide mailers the ability to sort their mailings in approximate carrier-casing sequence. To aid in mail sorting, eLOT contains an eLOT sequence number field and an ascending/ descending code. The eLOT sequence number indicates the first occurrence of delivery made to the add-on range within the carrier route, and the ascending/ descending code indicates the approximate delivery order within the sequence number. Mailers can use eLOT processing to qualify for enhanced carrier route presort discounts.

Finance Area A Finance Area is an area defined by the U.S. Postal Service from which it collects cost and statistical data. A Finance Area is frequently used for area searches, since it covers some or all of the ZIP Code areas in a town or city.

finance number An assigned six-digit number that identifies and installation for processing it's financial data. The first two digits are the state code and the next four are uniquely assigned from 0001 through 9999 to each installation in alphabetical order.

FIPS code Federal Information Processing Standards code. A FIPS Code, also called a Census ID, uniquely identifies each piece of Census geography. The syntax of the FIPS code is as follows:

`ssccctttt.ttgbbb` where:
`ss` = the two-digit State Census FIPS Code
`ccc` = the three digit County Census FIPS Code
`tttt.tt` = the 6 digit Census Tract Census FIPS Code
`g` = the single digit Block Group Census FIPS Code
`bbb` = the Block Census FIPS Code

GDT Geographic Data Technology data. Produced by Tele Atlas, a premium vendor of street segment files.

geocode, geocoding A geocode is the geographic information associated with a unique address or centroid, such as longitude and latitude. Geocoding is the process of assigning data based upon location information. GeoStan uses an address or ZIP Code to assign latitude, longitude, and Census FIPS information.

GIS Geographic Information System. A computer-based tool for enhancing geographic data by analyzing both the physical location in space and the set of characteristics associated with a location.

GMF General Mail Facility.

GSD files GeoStan directory files.

GsEnums Enumerated types in the GeoStan API. These enums are prefixed with "GS_" and are defined in the geostan.h file.

GSL file USPS eLOT and Z4Change data. This files is used to assign line of travel (LOT) codes to addresses.

GSU files GSU files contain information to match addresses based on unique ZIP Code and additional highrise unit information.

GSX files Geographic spatial index. These files are used by spatial functions in GeoStan.

GSZ file GeoStan ZIPMove file contains USPS ZIPMove data.

handle A reference to an object that is required by the Library and is not to be manipulated directly by the developer. The handle is generated when the library is initialized and is required for many library functions.

intersection matches Intersections matches are indicated by an x___ match code. For example, 28th Street and Valmont intersections may be standardized and geocoded and return demographic information. Intersections do not represent a valid address for mailings.

LACS Locatable Address Conversion System. This system corrects addresses electronically for areas that have undergone permanent address conversions. The address conversion occurred as a result of the 911 system implementation and involves renumbering and renaming rural route and highway contract route information as city-style addresses with street number and name.

lat/lon; latitude/longitude coordinates Longitude and latitude coordinates are always in degrees, and are always represented as 64-bit doubles. Positive numbers represent the Eastern and Northern hemispheres, respectively, and negative numbers represent the Western and Southern hemispheres. For example, the point 140W by 30N would be represented as −140.0,30.0. The library always assumes that the longitude coordinate is the horizontal direction and the latitude coordinate is the vertical direction. Support is not provided for user coordinates.

location code Location codes indicate the accuracy of the assigned geocode.

mail stop designator This designator indicates a routing code used by a company for internal mail delivery.

MASS Multiline (OCR) Accuracy Support System. A tool similar to Coding Accuracy Support System (CASS) that accesses and checks the address matching software used by customers' multiline optical character readers (OCRs).

match code Indicates the portions of the address that matched or did not match with the address information in the GeoStan data files.

match mode The algorithm used by GeoStan to match an input address to an address in the data files.

match rates The number of input addresses that correspond (can be matched) to address information in data files.

MBR Minimum bounding rectangle. A geographic region defined by and minimum and maximum latitude and longitude.

Metropolitan Statistical Area A Core Based Statistical Area associated with at least one urbanized area that has a population of at least 50,000. The Metropolitan Statistical Area comprises the central county or counties containing the core, plus adjacent outlying counties having a high degree of social and economic integration with the central county as measured through commuting.

Micropolitan Statistical Area A Core Based Statistical Area associated with at least one urban cluster that has a population of at least 10,000, but less than 50,000. The Micropolitan Statistical Area comprises the central county or counties containing the core, plus adjacent outlying counties having a high degree of social and economic integration with the central county as measured through commuting.

MLOCR Multiline Optical Character Reader. An optical character reader that reads and interprets more than one line of the delivery address on a mailpiece.

MSA name/ number Metropolitan Statistical Area. The name represents the name of the largest central city and the number is the 4-digit FIPS code.

match candidate resolution The process of resolving an address match when more than one street segment has been identified as corresponding to the input address.

NAD The North American Datum (NAD) is the official reference ellipsoid used for the primary geodetic network in North America.

NAD27 NAD27 has its origin at Meades Ranch, Kansas. NAD27 does not include the Alaskan islands and Hawaii. Latitudes and longitudes that are surveyed in the NAD27 system are valid only in reference to NAD27 and do not tie to any maps outside the U.S.

NAD83 NAD83 is earth-centered and defined with satellite and terrestrial data. NAD83 is compatible with the World Geodetic System 1984 (WGS84), the terrestrial reference frame associated with the NAVSTAR Global Positioning System (GPS) now used extensively for navigation and surveying. Note that GDT uses WGS84 instead of NAD83. These two coordinate systems are compatible.

NAVTEQ A premium vendor of street segment data.

NCSC National Customer Support Center. The U.S.P.S. CASS support center can be reached at www.usps.gov/ncsc.

object A basic functional unit of a library. A library contains functions that allow the user to create, manipulate, and destroy objects. C programmers access objects through handles that are provided through object creation functions.

OCR Optical Character Reader. An automated mail sorting machine that interprets the address information on a letter-size mailpiece and sprays the corresponding ZIP Code information onto the piece as a barcode. The OCR consists of a mail feed unit, transport unit, stacker modules, computer with a control system, video monitor, and printer.

package A group of addressed pieces assembled and secured together to make up a basic unit of bulk mail for mail processing. The term is not correctly applied when referring to unsecured groups of pieces placed in trays and identified by separator cards, although package labels and other package identification methods may be used for unsecured groups of pieces as permitted by standard.

piece An individually addressed mailpiece. This definition also applies when piece is used in eligibility standards. Quantities indicated for optional or required sortings always refer to pieces unless specifically excepted.

postdirectional (postdir) See directionals.

POSTNET Postal Numeric Encoding Technique. The barcode system for encoding the delivery point information and ZIP + 4 code information on letter-size and flat-size mailpieces. Also see delivery point barcode.

predirectional (predir) See directionals.

record matching algorithm Programmed logic that allows evaluation of the results of all field matching algorithms to determine whether two records match (i.e., are duplicates).

road class code A key in the street segment file that identifies a road as major or minor according to the Census Feature Classification Code.

RR Rural Route. A delivery route served by a rural carrier.

SCF Sectional Center Facility. A postal facility that serves as the processing and distribution center (P&DC) for post offices in a designated geographic area as defined by the first three digits of the ZIP Codes of those offices. Some SCFs serve more than one 3-digit ZIP Code range.

soundex algorithm A type of field matching algorithm that compares two fields based on their pronunciation.

soundex key Generated by the GsSoundex function. Used to search the database by employing a soundex algorithm.

spatial query functions Used to extract data from the GSD files. These functions specify the area to be searched through a minimum bounding rectangle rather than through city/state/ZIP or finance area.

stage 1 file A sample address file provided by the U.S.P.S to determine if software/hardware meets postal requirements for CASS.

stage 2 file An address file provided by the U.S.P.S. that is used to grade software/hardware to determine if it meets postal requirements for CASS.

street network files Files provided by vendors (other than U.S.P.S.) the contain address and geocode information.

Tele Atlas A premium data vendor of street segment files.

TIGER files Topographically Integrated Geographic Encoding and Referencing. A digital database of geographic features covering the entire United States.

TLID TIGER/Line® Identification Number.
The TIGER/Line® files use a permanent 10-digit TLID to uniquely identify a complete chain for the Nation. The 10-digit TLID will not exceed the value 231-1 (2,147,483,647) and represents the same complete chain in all versions of this file, beginning with the TIGER/Line® Precensus Files, 1990. The minimum value is 100,001. Topological changes to the complete chain causes the TLIDs to change. For instance, when updates split an existing complete chain, each of the new parts receives a new TLID; the old TLID is not reused.

As distributed, TIGER/Line® files are grouped by county (or statistically equivalent entity). A complete chain representing a segment of the boundary between two neighboring counties may have the same TLID code in both counties or it may have different TLID codes even though the complete chain represents the exact same feature on the ground.

unit designator Indicates the type of unit (e.g., apartment, unit).

USPS data files Files provided by the post office containing address and ZIP Code information.

walk sequence The order in which a city carrier delivers mail for a route. This order is required for most carrier route presort mail.

ZIP + 4 directory file Address records that contain the ZIP + 4 codes for all delivery points, in an electronic form.

ZIP + 4 centroid geocoding See geocoding.

ZIP Code Zone Improvement Plan Code. Established in 1963 the five-digit numeric code of which the first three digits identify the delivery area of a sectional center facility or a major-city post office serving the delivery address area. The next two (the fourth and fifth) digits identify the delivery area of an associate post office, post office branch, or post office station. All post offices are assigned at least one unique 5-digit code. ZIP Code is a USPS trademark.

ZIP + 4 is an enhanced code consisting of the 5-digit ZIP Code and four additional digits that identify a specific range of delivery addresses. The nine-digit numeric code, established in 1981, composed of two parts: (a) The initial code: the first five digits that identify the sectional center facility and delivery area associated with the address, followed by a hyphen; and (b) the four-digit expanded code: the first two additional digits designate the sector and the last two digits designate the segment. ZIP + 4 is also a USPS trademark.

INDEX

Pitney Bowes
Business Insight

Pitney Bowes Software
4200 Parliament Pl Ste 600
Lanham, MD 20706-1844
www.g1.com

Main: +1 (301) 731-2300
Sales: +1 (888) 413-6763
Support: +1 (800) 367-6950