AUTODESK® LOCATIONLOGIC

BREW® LBS Extensions
Developer's Guide

Autodesk® Location Services

Autodesk LocationLogic BREW SDK 1.0, compatible with LocationLogic 6.1.
Document Last Revised: June 10, 2005

Contents

About This Guide

This guide provides information for using the Autodesk® LocationLogic BREW™ (Binary Runtime Environment for Wireless™) LBS Extensions to create *location-based* mobile client applications.

This chapter explains what's in this guide and how it's organized.

In this chapter

Audience and Purpose

This guide is intended for experienced developers of BREW applications. It explains how to use the LocationLogic BREW LBS extensions to develop mobile location-based client applications. What you need to know depends on the extension that you will use:

- **Location Proxy Server extension**—You must be familiar with how to use the QUALCOMM® BREW `IPosDet` Interface. `IPosDet` is documented in the QUALCOMM® *BREW API Reference* in the BREW SDK, which you can download from http://brew.qualcomm.com.
- **Geoservices extension**—You must be familiar with basic GIS concepts such as maps, routes, geocoding, and so on. For more information, see the *Autodesk LocationLogic Getting Started* guide.

This guide assumes that you are writing BREW client applications for the LocationLogic platform.

How This Guide Is Organized

This guide is organized as follows:

- Chapter 1, "Introduction," provides a high-level description of the BREW LBS extensions, illustrates where they reside relative to the LocationLogic server and the user's handset, and specifies the minimum system requirements for a BREW-enabled handset.
- Chapter 2, "ILpsPosDet Interface," describes the `ILpsPosDet` Interface: a location privacy management wrapper around the native BREW `IPosDet` Interface that enforces a carrier's location privacy policy and the subscriber's privacy settings, including the complete API reference for all the Interface's functions.
- Chapter 3, "IGeoservice Interface," describes the `IGeoservice` Interface used to access LocationLogic geoservices such as mapping, routing, geocoding, and searching for points of interest. It includes the complete API reference for all Interface functions and data types. Sample code examples show how common application use cases can be implemented with the primary IGeoservice API functions.

Refer to the *Autodesk LocationLogic Glossary* for a listing of terms and definitions relating to LocationLogic and to the GIS (Geographic Information Systems) and wireless industries in general.

Conventions Used in This Guide

This section describes the following conventions used in this guide:

- Text conventions
- Code and syntax conventions

Text Conventions

This guide uses the following text conventions:

- *Italic* is used to introduce new terms. Italic is also used for database column names, file and folder names, and book titles.
- **Bold** is used for any text you must enter, such as at a command line prompt or in a dialog box.
- A `monospace` font is used for all code elements (variable names, data values, method names, and so forth), command lines, scripts, and source code listings.
- ***`Bold italic monospace`*** is used for replaceable elements and placeholders within code listings.

Code and Syntax Conventions

This guide uses the following code and syntax conventions:

- Indentation and line breaks have been added to make examples more legible. However, if you are copying the example code for your own use, do not use line breaks in an actual command.
- Although line breaks are valid if the preceding line ends in a backslash, there should not be leading spaces in an actual command.

Introduction

1

This chapter provides an overview of the BREW® LBS extensions, describes how they work with Autodesk® LocationLogic, and lists the minimum system requirements for a BREW-enabled handset.

In this chapter

About the BREW LBS Extensions

The Autodesk LocationLogic BREW LBS extensions enable rapid development of LBS applications. These extensions are not used as stand-alone applications, and you do not specifically purchase or download them to a device. Instead, your applications use the functions provided by the BREW LBS extensions in the same way as any other BREW function. (For more information about BREW extensions, refer to the following website: http://brew.qualcomm.com/brew/en/developer/resources/ad/extensions.html.)

The following functions are available through the two LocationLogic BREW LBS extensions:

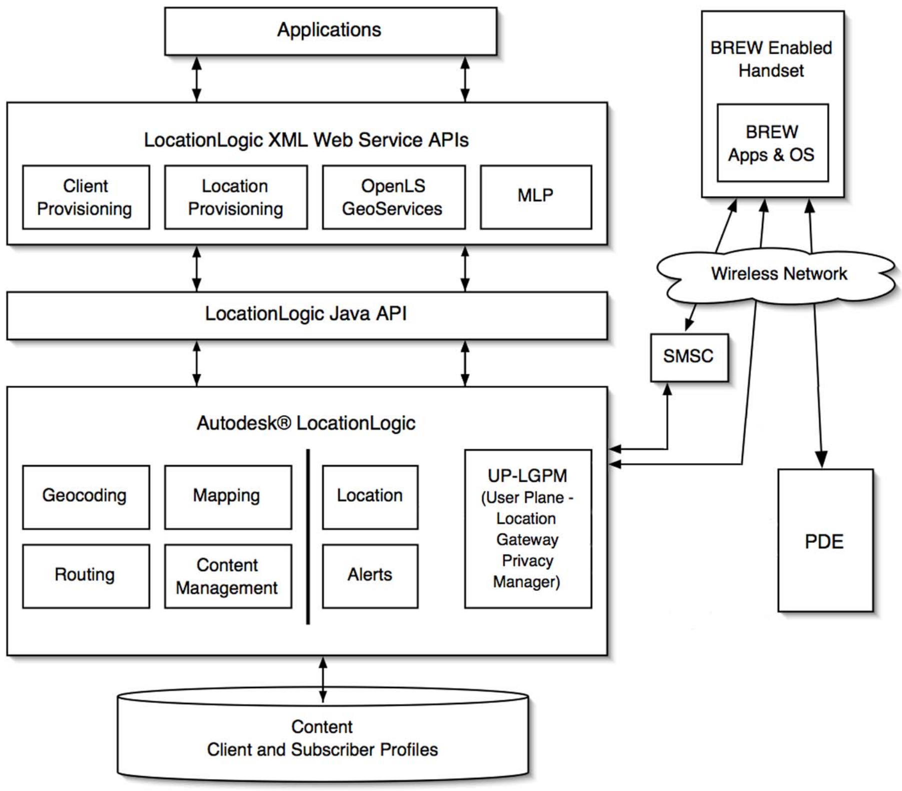
- **Retrieving a mobile device's location**—You use the *LpsPosDet* BREW LBS extension to locate the local handset while respecting privacy settings. The extension supports privacy-related user-interface elements, such as the following:
  - "Ask"-mode consent dialog boxes
  - Secure communications with the PDE (Position Determination Entity) and MPC (Mobile Positioning Center) server
  - Authentication for first-time client registration ("application auto-provisioning")

  For information about the `ILpsPosDet` Interface, see Chapter 2, "ILpsPosDet Interface."
- **Accessing LocationLogic geoservices**—You use the *Geoservices* BREW LBS extension to access the following LocationLogic services:
  - Mapping
  - Routing
  - Geocoding and reverse-geocoding
  - Searching for points of interest

  This extension implements the `IGeoservice` Interface, described in Chapter 3, "IGeoservice Interface."

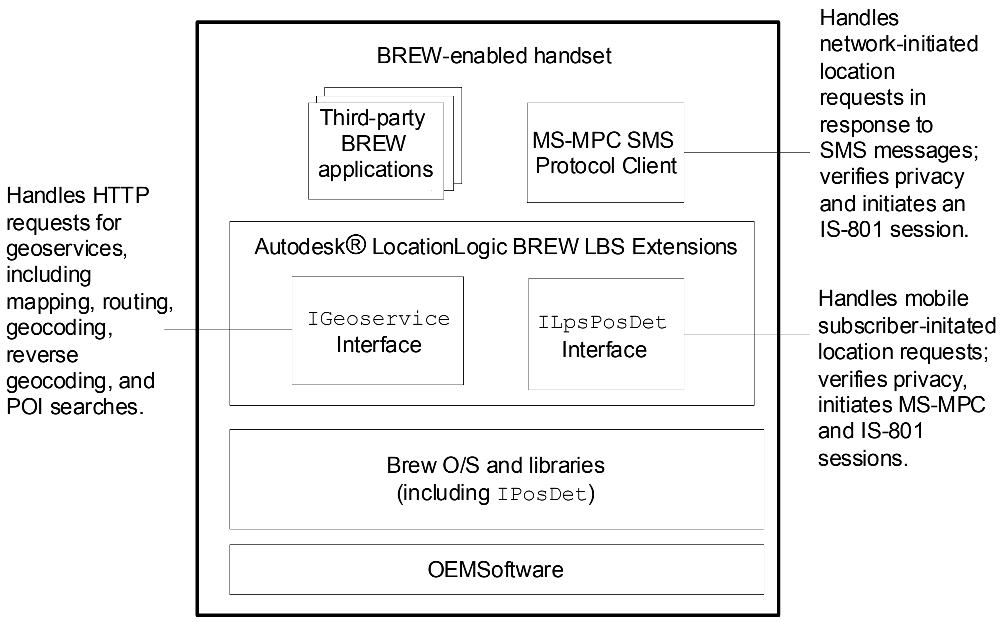
Architectural Overview

The following figure shows how a BREW-enabled handset operates in relation to the LocationLogic platform.



The Autodesk LocationLogic platform and a BREW-enabled handset

The figure below shows how BREW applications, BREW LBS extensions, and other software components reside on a BREW-enabled handset.


BREW-enabled handset
Third-party BREW applications
MS-MPC SMS Protocol Client
Autodesk® LocationLogic BREW LBS Extensions
IGeoservice Interface
ILpsPosDet Interface
Brew O/S and libraries (including IPosDet)
OEMSoftware
Handles HTTP requests for geoservices, including mapping, routing, geocoding, reverse geocoding, and POI searches.
Handles network-initiated location requests in response to SMS messages; verifies privacy and initiates an IS-801 session.
Handles mobile subscriber-initated location requests; verifies privacy, initiates MS-MPC and IS-801 sessions.

The software components of a BREW-enabled handset

Before You Begin

This section describes required system components, and provides information about basic concepts you need to understand before beginning to develop mobile applications using the BREW LBS extensions.

Minimum Handset Capabilities

The minimum system requirements for a BREW-enabled handset using the BREW LBS extensions are:

- BREW 2.1.3 or later
- Fully enabled BREW `IPosDet` Interface
- QUALCOMM gpsOne™ capabilities, including support for mobile subscriber-based and subscriber-assisted GPS positioning; lat/lon, velocity, direction, and accuracy determination; and sector location.

Check with each carrier about specific models of handsets that support these requirements.

Optimization and Performance Guidelines

Your LBS applications should be written in accordance with standard BREW application guidelines. For detailed information, refer to the QUALCOMM BREW Online Knowledge Base, at the following website: http://brew.qualcomm.com/brew/en/developer/resources/ds/okb.html.

Using the BREW LBS Extensions

The BREW LBS extensions are designed to let you develop client applications easily within the BREW environment while enforcing the customer's location privacy policies. The `ILpsPosDet` Interface in particular is designed to mimic the native BREW `IPosDet` Interface; so if you're already familiar with BREW and `IPosDet`, you should be able to port an existing LBS application rapidly.

Your applications access LocationLogic services by using the functions exposed by the BREW LBS extensions. The applications should be developed in accordance with standard BREW application techniques. For detailed information, refer to the QUALCOMM BREW Online Knowledge Base, at the following website: http://brew.qualcomm.com/brew/en/developer/resources/ds/okb.html.

Setting Up a Device Emulator

The BREW Emulator is included along with the BREW SDK. When using it to emulate your BREW LBS Extension application, consider the following:

- **For IGeoservices**—If you connect to the geoservice server from behind a firewall/router:
  - Create a text file containing the IP address of the router (the external IP address exposed to the internet). Specify the IP address in dotted decimal format (1.2.3.4).
  - Name the text file "*ip.cfg*" and place it in the same folder as the application.
- **For ILpsPosDet**—First Time Through (FTT) provisioning is automatically simulated via privacy settings in a file generated by the extension. The extension places this file, named "*lpsreg*", in the application folder.

  Note Remove this file in order to go through FTT again.

Handling Tracking Sessions

Special consideration needs to be given to tracking sessions. They can be configured for maximum duration, restarted after a timeout, or cancelled.

Configuring—To prevent a permission dialog from appearing on a device multiple times within a tracking session (which require multiple fixes inside a specific time period), set parameters to values that cause the dialog to display only once:

- Set nFixes to a number greater than 1 and nInterval to a number greater than 0 before calling ILPSPOSDET_GetGPSInfo() for the first time.
- Compute the total tracking session time with the following formula:

  (nFixes – 1) * nInterval

Consequently, the permission dialog will be displayed only before the first fix in the tracking session. After the tracking session has timed out, the permission dialog will appear again, if indicated by the application's privacy settings.

Restarting—Ordinarily, a tracking session persists across suspend/resume events and terminates when an application exits or a device is powered off. However, an error can cause a tracking session to time out.

For example, if an application receives an `AEEGPSINFO.status = AEEGPS_ERR_PRIVACY_REFUSED` error from the ILPSPOSDET_GetGPSInfo() callback, then the tracking session has timed out.

ILPSPOSDET_GetGPSInfo() will fail on the first request made after the time window expires.

To continue tracking after a timeout, applications must start a new tracking session. If the subscriber's privacy setting has been set to ASK, a privacy confirmation dialog will be displayed the first time ILPSPOSDET_GetGPSInfo() is called in the new session.

Canceling—To explicitly cancel a tracking session, you must release the ILPSPOSDET interface. You must first call CALLBACK_Cancel() if a callback is active.

ILpsPosDet Interface

2

The `ILpsPosDet` Interface is a location privacy management wrapper around the native BREW `IPosDet` Interface. This chapter describes the `ILpsPosDet` Interface and provides a complete API reference for all functions.

In this chapter

Overview

The ILpsPosDet Interface is a location privacy management wrapper around the native BREW IPosDet Interface, enforces a customer's privacy policies, and provides the following functions:

- Functionality of all IPosDet functions
- All possible combinations of network and MS (Mobile Station) -initiated requests for self:
  - Network-initiated local location
  - MS (Mobile Station) -initiated self location

Most ILpsPosDet functions simply call the corresponding IPosDet function after verifying that GetGPSInfo() has already performed the proper security and privacy checks.

The following table describes the ILpsPosDet functions (listed in alphabetical order). An "X" in the "IPosDet" column means that the ILpsPosDet function has an exact counterpart in the IPosDet Interface.

ILpsPosDet Interface Functions (1 of 2)

Function	IPosDet	Description
ILPSPOSDET_AddRef()	X	Increments the Interface reference count.
ILPSPOSDET_GetGPSConfig()	X	Returns a handset's current GPS configuration parameters.
ILPSPOSDET_GetGPSInfo()	X	Asynchronous function that gets a handset's current location, velocity, altitude, direction of travel, and position-uncertainty. This function does the following: • Communicates with the Location Proxy Server using the IP-based MS-MPC protocol, to determine whether the current application has permission to locate the handset with GPS accuracy, and gets a cached location value if possible. • Initiates an IS-801 session with PDE, if necessary. • Specifies the callback function that will be called after the handset's current location, velocity, altitude, direction of travel and position uncertainty have been determined.

ILpsPosDet Interface Functions (2 of 2)

Function	IPosDet	Description
`ILPSPOSDET_GetOrientation()`	X	Asynchronous function that gets a handset's orientation in the horizontal plane. This function specifies the callback function that will receive the handset's orientation. (This function is supported only on mobile devices with "compass" capability.)
`ILPSPOSDET_GetSectorInfo()`	X	This method always returns "`EUNSUPPORTED`"
`ILPSPOSDET_Init`		This function initializes an instance of the `ILpsPosDet` extension for use by a client application. If the client is not registered, the function will register it on the server.
`ILPSPOSDET_QueryInterface()`	X	Asks the current object for a specific API contract, identified by `ClassID`.
`ILPSPOSDET_Release()`	X	Decrements the Interface reference count.
`ILPSPOSDET_SetGPSConfig()`	X	Sets configuration parameters that control the GPS and location-determination process.

ILpsPosDet API Function Reference

This Interface provides services for position determination using sector information or GPS information for self.

Sector information privileges are required to use the sector-based position determination methods, such as the following:

- `ILPSPOSDET_GetSectorInfo`

Similarly, position determination privileges are required to use the GPS-based position determination methods, such as the following:

- `ILPSPOSDET_SetGPSConfig`
- `ILPSPOSDET_GetGPSConfig`
- `ILPSPOSDET_GetGPSInfo`

The following `ILpsPosDet` Interface methods are asynchronous methods, and use `AEECallback` structures to point to callback functions. Care must be taken to ensure that the callbacks and the information structures passed to these methods by reference remain in scope until the callback returns.

- `ILPSPOSDET_GetSectorInfo`
- `ILPSPOSDET_GetGPSInfo`

Note If multiple requests for sector or GPS information are made without waiting for the previous request callbacks to return, the request will return an error code.

For self determination of sector or GPS information, BREW SDK users can set the GPS emulation in the Tools ➤ GPS Emulation menu to use a pre-recorded NMEA 0183-format file as GPS input, or connect an NMEA-output-capable GPS device. (For more information about the National Marine Electronics Association 0183 communication protocol, refer to the following website: http://www.nmea.org/pub/0183/.)

You can use an offline utility, *NMEALogger.exe*, to record an NMEA file from data coming from a GPS device connected to the serial port of your development environment machine (desktop or laptop computer). This NMEA file can be used later as GPS input.

The following header file is required:

`LpsPosDet.h`

ILPSPOSDET_AddRef

Description

This function increments the reference count of the `ILpsPosDet` Interface object, allowing the object to be shared by multiple callers. The object is freed from memory, and is no longer valid, when the reference count reaches `0` (zero).

Prototype

```
uint32 ILPSPOSDET_AddRef(ILpsPosDet * pILpsPosDet);
```

Parameters

`pILpsPosDet`	[in]	Pointer to the `ILpsPosDet` Interface object

Return Values

Incremented reference count for the object

Comments

A valid object returns a positive reference count.

See Also

- `ILPSPOSDET_Release()`

ILPSPOSDET_GetGPSConfig

Description

This function gets the GPS configuration to be used by the GPS engine.

Prototype

```
int ILPSPOSDET_GetGPSConfig
                (ILpsPosDet *pILpsPosDet,
                 AEEGPSConfig *pAEEGPSConfig);
```

Parameters

`pILpsPosDet`	[in]	Interface pointer
`pAEEGPSConfig`	[out]	Pointer to GPS configuration. Refer to the `AEEGPSConfig` data type description in the QUALCOMM® *BREW API Reference* for details

Return Values

- `SUCCESS`—The function succeeded
- `EPRIVLEVEL`—The caller does not have sufficient privilege levels (`PL_POS_LOCATION`) to invoke this function
- `EBADPARM`—`pConfig` is `NULL`
- `EUNSUPPORTED`—This function is not supported

See Also

- `AEEGPSConfig` data type description in the QUALCOMM® *BREW API Reference*
- `ILPSPOSDET_SetGPSConfig`

Comments

If `ILPOSDET_Init` is not called first, this function will return an `EUNSUPPORTED` error.

ILPSPOSDET_GetGPSInfo

Description

This function returns information for GPS-based position location. It returns latitude, longitude, and altitude information, as well as vector information such as horizontal and vertical velocity, heading, and the uncertainty of the horizontal information.

This is an asynchronous call, and the callback specified by `pAEECallback` is called on completion. If the position request is not satisfactorily answered, the status member of `AEEGPSInfo` will be non-zero values. (Refer to the QUALCOMM® *BREW API Reference* for information about error codes.)

This function enforces the privacy policies recommended by the network operator. Applications invoking this function must be prepared to relinquish control of the screen so that user prompts can be displayed. Applications must suspend any painting to the screen on the event EVT_DIALOG_START, and must redraw on the event EVT_DIALOG_END.

Prototype

```
int ILPSPOSDET_GetGPSInfo
                  (ILpsPosDet *pILpsPosDet,
                   AEEGPSReq aeeGPSReq,
                   AEEGPSAccuracy aeeGPSAccuracy,
                   AEEGPSInfo *pAEEGPSInfo,
                   AEECallback *pAEECallback);
```

Parameters

pILpsPosDet	[in]	Interface pointer
aeeGPSReq	[in]	Request type: • AEEGPS_GETINFO_LOCATION • AEEGPS_GETINFO_VELOCITY • AEEGPS_GETINFO_ALTITUDE The flags can be combined to get multiple types of information.
aeeGPSAccuracy	[in]	Desired level of accuracy for this request; valid values are as specified by the AEEGPS_ACCURACY_* definitions in the BREW SDK.
pAEEGPSInfo	[out]	On input, this must be a valid ptr to the AEEGPSInfo structure. On callback, the members of this struct contain GPS information. The caller must ensure that this structure is valid until the callback specified by pcb gets called.
pAEECallback	[in]	Pointer to callback structure that points to callback function

Return Values

- SUCCESS—The function succeeded
- EPRIVLEVEL—The caller does not have sufficient privilege levels (PL_POS_LOCATION) to invoke this function
- EBADPARM—pAEEGPSInfo or pAEECallback is NULL
- EUNSUPPORTED—This function is not supported
- ENOMEMORY—Out of memory
- EFAILED—General failure
- EITEMBUSY—Previous request is in progress and another request is made

Comments

Tracking Session Timeout

Ordinarily, a tracking session persists across suspend/resume events and terminates after an application exit or when the device power is turned off.

However, if an application receives an `AEEGPSINFO.status = AEEGPS_ERR_PRIVACY_REFUSED` error from the ILPSPOSDET_GetGPSInfo() callback, then the tracking session has timed out. ILPSPOSDET_GetGPSInfo() will fail on the first request made after the time window expires.

Note See the notes in "Configuring and Cancelling a Tracking Session," on page 28 for information about how to set the maximum duration of a tracking session.

To continue tracking after a timeout, applications must start a new tracking session. If the subscriber's privacy setting has been set to `ASK`, a privacy confirmation dialog will be displayed the first time ILPSPOSDET_GetGPSInfo() is called in the new session.

Suspension of Application during GetGPSInfo Request

If an application is suspended during an ongoing request to GetGPSInfo(), the application must cancel all callbacks by using the CALLBACK_Cancel macro.

After the callbacks have been cancelled, the application must do the following, regardless of single-shot or tracking mode:

- Call the CALLBACK_Init() macro
- Call ILPSPOSDET_GetGPSInfo()

Possible Status Values

The result of the operation is indicated in the `status` member of AEEGPSInfo when callback is invoked. The possible values of `status` are:

- `AEEGPS_ERR_NO_ERR`—Indicates that the request is completely answered.
- `AEEGPS_ERR_GENERAL_FAILURE`—Indicates that reason for failure is a result of either device is low on resources or device is busy with other operations to handle this request or device is shutting down.
- `AEEGPS_ERR_TIMEOUT`—Request timed out.
- `AEEGPS_ERR_INFO_UNAVAIL`—Indicates that either partial or no information is available for the request. Check the fValid field to retrieve any partial information available.

- AEEGPS_ERR_PRIVACY_REFUSED—Indicates that the request is refused to protect the device's privacy. This occurs when the Privacy enforced on the device decided to reject the position requests of the application.
- EPRIVLEVEL—Indicates either that the requester application is not authorized to get GPS Information, or that the user denied permission on the consent dialog box.
- ENOMEMORY—Out of memory
- EFAILED—General failure
- EFAILED_EULADECLINED—EULA was declined
- EFAILED_FTTCANCELLED—FTT process was cancelled
- EFAILED_LPSCONNECTION—Unable to connect to LPS; download EULA or save the privacy settings
- If ILPOSDET_Init is not called first, this function will return an EUNSUPPORTEDerror.

ILPSPOSDET_GetOrientation

Description

This function returns a device's orientation, in the horizontal plane. This is an asynchronous call, and the callback specified by `pAEECallback` is called on completion.

Prototype

```
int ILPSPOSDET_GetOrientation
                  (ILpsPosDet *pILpsPosDet,
                   AEEOrientationReq aeeOrientationReq,
                   void *pOrInfo,
                   AEECallback *pAEECallback);
```

Parameters

`pILpsPosDet`	[in]	Interface pointer
`aeeOrientationReq`	[in]	Requested information; the only valid value is `AEEORIENTATION_REQ_AZIMUTH`
`pOrInfo`	[out]	On input, this must be a non-`NULL` ptr to valid memory, with the first two bytes (`wSize`) indicating the space available in bytes. The space should be a minimum of that required to place the response corresponding to the request. On callback, the members of this struct contain orientation information corresponding to the request. The caller (application) must ensure that this memory is valid until the callback specified by `pAEECallback` gets called.
`pAEECallback`	[in]	Pointer to callback structure that points to callback function

Return Values

- `SUCCESS`—Function succeeded
- `EPRIVLEVEL`—The caller (application) does not have sufficient privilege levels (`PL_POS_LOCATION`) to invoke this function
- `EBADPARM`—`pGPSInfo` or `pcb` is `NULL`
- `EUNSUPPORTED`—This function is not supported
- `EFAILED`—General failure

Comments

If `ILPOSDET_Init` is not called first, this function will return an `EUNSUPPORTED` error.

ILPSPOSDET_GetSectorInfo

Description

This is an asynchronous call; the callback specified in the `pAEECallback` data is called on completion. If the sector request is not satisfactorily answered, the status member of `SectorInfo` will be non-zero, with one of `AEEGPS_ERR_`* values. (Refer to the QUALCOMM® *BREW API Reference* for information about error codes.)

This function returns information for sector-based position location. It returns information about the `SystemID`, `NetworkID`, `BaseStationID`, `BaseStationClass` and best `Pilot`. In order to invoke this function, the caller (application) must have the `PL_SECTORINFO` privilege level; without this privilege level, this function will fail.

Prototype

```
int ILPSPOSDET_GetSectorInfo
                 (ILpsPosDet *pILpsPosDet,
                  AEESectorInfo * pAEESectorInfo);
```

Parameters

`pILpsPosDet`	[in]	Interface pointer
`pAEESectorInfo`	[out]	Pointer to `AEESectorInfo` data structure, containing the desired sector information
`pAEECallback`	[in]	Pointer to callback structure that points to callback function

Return Values

- `SUCCESS`—Function succeeded
- `EPRIVLEVEL`—The caller does not have sufficient privilege levels to invoke this function
- `EBADPARM`—`pSecInfo` is `NULL`
- `EUNSUPPORTED`—Function is not supported
- `EFAILED`—General failure

Comments

The current implementation always returns `EUNSUPPORTED`.

See Also

- `AEESectorInfo` data type description in the QUALCOMM® *BREW API Reference*

ILPSPOSDET_Init

Description

This function initializes an instance of the ILpsPosDet extension library for use by a client application. If the client is not registered, the function will register it on the server. During the registration process, the library will open a dialog box, so any applications invoking ILPSPOSDET_Init must be prepared to relinquish the screen. Therefore, after invoking `ILPSPOSDET_Init`, an application must suspend any painting to the screen on the events `EVT_DIALOG_START` and `EVT_DIALOG_INIT`, and must redraw on the event `EVT_DIALOG_END`.

Prototype

```
int LpsPosDet_Init( uint32 unClientID, AECHAR* pwszClientPwd,
                    AEECallback* pAEECallback, int* pnStatus)
```

Parameters

`unClientID`	[in]	Client ID provided by Verizon. This is a unique application ID.
`pwszClientPwd`	[in]	Password
`pAEECallback`	[in]	Pointer to the callback structure which gets called on completion of the registrations process
`pnStatus`	[out]	In the callback, if the initialization fails (for example, because of a wrong client ID), `pnStatus=EFAILED`. If the initialization succeeds, `pnStatus=SUCCESS`. In this implementation, the callback function is always called.

Return Values

- `SUCCESS`—The function succeeded
- `EBADPARM`—`pszClientPwd`, `pAEECallback` or `pnStatus` is `NULL`
- `EFAILED_LPSCONNECTION`—Unable to connect to LPS, download EULA or save the privacy settings.

  Note This message may be returned when an invalid ClientID has been specified, or if the EULA is not available.
- `EFAILED`—General failure

Comments

This method should be the first one called after the extension is loaded. If other methods are called first, they will return `EUNSUPPORTED`.

You should check the `pnstatus` inside your callback function.

ILPSPOSDET_QueryInterface

Description

This function asks an object for another API contract from the object in question.

Prototype

```
int ILPSPOSDET_QueryInterface
                 (ILpsPosDet * pILpsPosDet,
                  AEECLSID aeeCLSID,
                  void ** ppo);
```

Parameters

`pILpsPosDet`	[in]	Pointer to the `ILpsPosDet` Interface object
`aeeCLSID`	[in]	Requested class ID exposed by the object
`ppo`	[out]	Returned object. Filled by this method

Return Values

- `SUCCESS`—Interface found
- `ENOMEMORY`—Insufficient memory
- `ECLASSNOTSUPPORT`—Requested interface is unsupported

Comments

- The pointer in `*ppo` is set to the new Interface (with `refcount` positive), or `NULL` if the `ClassID` is not supported by the object.
- `The value of ppo` cannot be `NULL`.

ILPSPOSDET_Release

Description

This function decrements the reference count of the `ILpsPosDet` Interface object. The object is freed from memory, and is no longer valid, when the reference count reaches `0` (zero).

Prototype

```
uint32 ILPSPOSDET_Release(ILpsPosDet * pILpsPosDet);
```

Parameters

`pILpsPosDet`	[in]	Pointer to the `ILpsPosDet` Interface object

Return Values

- Decremented reference count for the object
- `0` (zero), if the object has been freed and is no longer valid

See Also

- `ILPSPOSDET_AddRef`()

ILPSPOSDET_SetGPSConfig

Description

This function sets the GPS configuration to be used by the GPS engine.

Prototype

```
int ILPSPOSDET_SetGPSConfig
                 (ILpsPosDet *pILpsPosDet,
                  AEEGPSConfig *pAEEGPSConfig);
```

Parameters

`pILpsPosDet`	[in]	Interface pointer
`pAEEGPSConfig`	[in]	Pointer to GPS configuration. Refer to the `AEEGPSConfig` data type description in the QUALCOMM® *BREW API Reference* for details

Return Values

- `SUCCESS`—Function succeeded
- `EPRIVLEVEL`—The caller does not have sufficient privilege levels to invoke this function
- `EBADPARM`—`pConfig` is `NULL`, or the data passed in the *`pConfig` is invalid (checks validity of `mode`, `server.svrType`, `optim`)
- `EUNSUPPORTED`—This function is not supported

Comments

Until this function is called, the position determination engine will be configured with default settings. Default settings can be obtained using `ILPSPOSDET_GetGPSConfig()`. Only the position determination requests following a call to `ILPSPOSDET_SetGPSConfig()` will use the new configuration.

Configuring and Cancelling a Tracking Session

Tracking session configuration—For single location fixes, every time `GetGPSInfo()` is called, a permission dialog appears if the application's location privacy permissions have been set to `Always Ask`. To prevent the permission dialog from appearing multiple times in tracking sessions, which require multiple fixes inside a specific time period, a tracking application should set `nFixes` to a number greater than 1 and `nInterval` to a number greater than 0 before calling `GetGPSInfo()` for the first time.

Consequently, the permission dialog will be displayed only before the first fix in the tracking session. After the tracking session has timed out, the permission dialog will appear again, if indicated by the application's privacy settings.

The total tracking session time is computed by using the following formula:

```
(nFixes - 1) * nInterval
```

Canceling a tracking session—To explicitly cancel a tracking session, you must release the ILPSPOSDET interface. You must first call `CALLBACK_Cancel()` if a callback is active

See Also

- `AEEGPSConfig` data type description in the QUALCOMM® *BREW API Reference*
- `ILPSPOSDET_GetGPSConfig`

ILpsPosDet Data Types

Data types for ILpsPosDet are identical to data types for IPosDet. Please see the official Qualcomm IPosDet data type documentation at:

http://brew.qualcomm.com/brew/en/.

3 IGeoservice Interface

The `IGeoservice` Interface (included in the Geoservices BREW LBS extension) is used to access LocationLogic geoservices, such as mapping, routing, geocoding, and searching for points of interest. This chapter describes the `IGeoservice` Interface and includes an API reference for all functions and data types. It concludes with detailed code examples that show how common application use cases can be implemented with the primary IGeoservice API functions.

In this chapter

Overview

The Geoservices LBS BREW extension, which lets you access Autodesk LocationLogic geoservices, implements the `IGeoservice` Interface.

The following table describes the `IGeoservice` functions. Note that several functions are asynchronous, so callback functions are used.

IGeoservice Interface Functions (1 of 2)

Function	Description
`IGEOSERVICE_AddRef()`	Increments the Interface reference count.
`IGEOSERVICE_DetermineRoute()`	Asynchronous function that generates and returns a route object. The callback function executes after the route is calculated and the appropriate portions are sent back to the handset.
`IGEOSERVICE_DetermineRouteResponseFree()`	Releases memory allocated to a `DetermineRoute` response structure when it replies to a `DetermineRoute` request.
`IGEOSERVICE_FindFeatures()`	Asynchronous function that searches for points of interest and other geographical features. The callback function executes after the search finishes and a list of features is sent back to the handset.
`IGEOSERVICE_FindFeaturesResponseFree()`	Releases memory allocated to a FindFeatures response structure when it replies to a `FindFeatures` request.

IGeoservice Interface Functions (2 of 2)

Function	Description
`IGEOSERVICE_Geocode()`	Asynchronous forward-geocoding function that returns a point location, or a list of partially matched street addresses.
`IGEOSERVICE_GeocodeResponseFree()`	Releases memory allocated to a Geocode response structure when it replies to a `Geocode` request.
`IGEOSERVICE_Init()`	Initializes the Geoservice extension with ClientID, Password, and language. It is called once before any other function in the `IGeoservice` interface.
`IGEOSERVICE_PortrayMap()`	Asynchronous function that generates and returns a map image. The callback function executes after the map image has been transferred to the handset.
`IGEOSERVICE_PortrayMapResponseFree()`	Releases memory allocated to a `PortrayMap` response structure when it replies to a `PortrayMap` request.
`IGEOSERVICE_Release()`	Decrements the Interface reference count.
`IGEOSERVICE_ReverseGeocode()`	Asynchronous reverse-geocoding function that returns a street address near a specified point location.
`IGEOSERVICE_ReverseGeocodeResponseFree()`	Releases memory allocated to a `ReverseGeocode` response structure when it replies to a `ReverseGeocode` request.

IGeoservice API Function Reference

The `IGeoservice` Interface provides access to the following LocationLogic services:

- Generating maps and routes
- Geocoding and reverse-geocoding
- Searching for points of interest

The exposed functions use BREW `IPosDet` parameter conventions for expressing latitude and longitude.

Note Applications must call `GEOSERVICE_Init()` before using the individual functions.

The following header file is required

IGeoservice.h

IGEOSERVICE_AddRef

Description

This function increments the reference count of the `IGeoservice` Interface object, allowing the object to be shared by multiple callers. The object is freed from memory when the reference count reaches `0` (zero).

Prototype

```
uint32 IGEOSERVICE_AddRef(IGeoservice * pIGeoservice);
```

Parameters

`pIGeoservice`	[in]	Pointer to the `IGeoservice` interface object

Return Values

Incremented reference count for the object.

Comments

A valid object returns a positive reference count.

See Also

- `IGEOSERVICE_Release()`

IGEOSERVICE_DetermineRoute

Description

This function calculates a travel route. It returns route instructions, route geometry and a route map.

A route connects a start location to a destination location and can include zero or more intermediate locations ("waypoints"). You can request a route that avoids potentially unfavorable conditions such as toll plazas, tunnels, and bridges. As a result, the actual path of a route can vary depending upon the time of day and the preferences specified.

Prototype

```
GSVCExceptions IGEOSERVICE_DetermineRoute
                (IGeoservice *po,
                 GSVCDetermineRouteRequest *req,
                 GSVCDetermineRouteResponse** ppResp,
                 GSVCExceptions* peResponseStatus,
                 AEECallback *cb);
```

Parameters

`po`	[in]	Interface pointer
`req`	[in]	Pointer to `GSVCDetermineRouteRequest`
`ppResp`	[out]	Pointer to pointer to `GSVCDetermineRouteResponse`
`peStatus`	[out]	Address of `GSVCExceptions` enum where Response Status will be returned
`cb`	[in]	Pointer to callback structure

Return Values

`GSVCExceptions`:

- `GSVC_SUCCESSFUL` is returned if there is no problem during the query initialization phase of the request.
- If there is a query initialization problem, a corresponding value will be returned.

- `GSVC_EUNSUPPORTED` is returned if this function is called before `IGEOSERVICE_Init`.

Comments

This function receives an `IGeoservice` interface pointer and a pointer to `GSVCDetermineRouteRequest`. On completion of the `GSVCDetermineRouteRequest`, the function creates and populates the `GSVCDetermineRouteResponse` structure, delegating control to the callback function registered in `cb` by the user. The resultant status of this method is cached in `*peStatus`.

See Also

- `AEECallback`
- `GSVCDetermineRouteResponse`
- `GSVCDetermineRouteRequest`

IGEOSERVICE_DetermineRouteResponseFree

Description

This function releases memory allocated to a `GSVCDetermineRouteResponse` structure by `IGeoservice` when it replies to an `IGEOSERVICE_DetermineRoute` request.

Prototype

```
void IGEOSERVICE_DetermineRouteResponseFree
                (IGeoservice *p,
                 GSVCDetermineRouteResponse* pDRResp);
```

Parameters

`p`	[in]	Interface pointer
`pDRResp`	[in]	Pointer to `GSVCDetermineRouteResponse` structure to be released

Return Values

`None`

Comments

This function is used to release all the memory allocated for an instance of the `GSVCDetermineRouteResponse` structure. The extension allocates memory for a `GSVCDetermineRouteResponse` while processing an `GSVCDetermineRouteRequest`. This memory must be deallocated by the developer by explicitly calling this function when the response is no longer needed.

See Also

- `GSVCDetermineRouteResponse`

IGEOSERVICE_FindFeatures

Description

This function is used to find specific or nearby features by using both geographic and property filters. The response contains a list of returned features.

Prototype

```
void IGEOSERVICE_FindFeatures
                (IGeoservice *po,
                GSVCFindFeaturesRequest *req,
                GSVCFindFeaturesResponse **ppResp,
                GSVCExceptions* peStatus,
                 AEECallback *cb);
```

Parameters

`po`	[in]	Interface pointer
`req`	[in]	Pointer to `GSVCFindFeaturesRequest`
`ppResp`	[out]	Pointer to pointer to `GSVCFindFeaturesResponse`
`peStatus`	[out]	Address of `GSVCExceptions` enum where Response Status will be returned
`cb`	[in]	Pointer to callback structure

Return Values

`GSVCExceptions`:

- `GSVC_SUCCESSFUL` is returned if there is no problem during the query initialization phase of the request.
- If there is a query initialization problem, a corresponding value will be returned.
- `GSVC_EUNSUPPORTED` is returned if this function is called before `IGEOSERVICE_Init`.

Comments

This function receives an `IGeoservice` pointer and a pointer to `GSVCFindFeaturesRequest`. On completion of the `GSVCFindFeaturesRequest,` the function creates and populates a `GSVCFindFeaturesResponse` structure, delegating control to the callback function registered in `cb` by the user. The resultant status of this method is cached in `*peStatus`.

See Also

- `AEECallback`
- `GSVCFindFeatureRequest`
- `GSVCFindFeatureResponse`

IGEOSERVICE_FindFeaturesResponseFree

Description

This function releases memory allocated to a `GSVCFindFeaturesResponse` structure by `IGeoservice` when it replies to an `IGEOSERVICE_FindFeatures` request.

Prototype

```
void IGEOSERVICE_FindFeaturesResponseFree
                (IGeoservice *p,
                 GSVCFindFeaturesResponse* pFFesp);
```

Parameters

`p`	[in]	Interface pointer
`pFFesp`	[in]	Pointer to `GSVCFindFeaturesResponse` structure to be released

Return Values

`None`

Comments

This function is used to release all the memory allocated for an instance of the `GSVCFindFeaturesResponse` structure. The extension allocates memory for a `GSVCFindFeaturesResponse` while processing a `GSVCFindFeaturesRequest`. This memory must be deallocated by the developer by explicitly calling this function when the response is no longer needed.

See Also

- `GSVCFindFeaturesResponse`

IGEOSERVICE_Geocode

Description

This function is used to convert the postal address into a pair (lat, lon) of geographical coordinates.

Prototype

```
GSVCExceptions IGEOSERVICE_Geocode
                 (IGeoservice *po,
                  GSVCGeocodeRequest *req,
                  GSVCGeocodeResponse** ppResponse,
                  GSVCExceptions* peStatus,
                  AEECallback *cb);
```

Parameters

`po`	[in]	Interface pointer
`req`	[in]	Pointer to `GSVCGeocodeRequest` structure
`ppResponse`	[out]	Pointer to pointer to `GSVCGeocodeResponse` structure
`peStatus`	[out]	Address of `GSVCExceptions` enum where Response Status will be returned
`cb`	[in]	Pointer to callback structure

Return Values

`GSVCExceptions`:

- `GSVC_SUCCESSFUL` is returned if there is no problem during the query initialization phase of the request.
- If there is a query initialization problem, a corresponding value will be returned.
- `GSVC_EUNSUPPORTED` is returned if this function is called before `IGEOSERVICE_Init`.

Comments

This function receives an `IGeoservice` pointer and a pointer to `GSVCGeocodeRequest`. On completion of the `GSVCGeocodeRequest`, the function creates and populates a `GSVCGeocodeResponse` structure, delegating control to the callback function registered in `cb` by the user. The resultant status of this method is cached in `*peStatus`.

See Also

- `GSVCGeocodeRequest`
- `GSVCGeocodeResponse`
- `AEECallback`

IGEOSERVICE_GeocodeResponseFree

Description

This function releases memory allocated to a `GSVCGeocodeResponse` structure by `IGeoservice` when it replies to an `IGEOSERVICE_Geocode` request.

Prototype

```
void IGEOSERVICE_GeocodeResponseFree
                (IGeoservice *p,
                 GSVCGeocodeResponse* pGResp);
```

Parameters

`p`	[in]	Interface pointer
`pGResp`	[in]	Pointer to `GSVCGeocodeResponse` structure to be released

Return Values

`None`

Comments

This function is used to release all the memory allocated for an instance of the `GSVCGeocodeResponse` structure. The extension allocates memory for a `GSVCGeocodeResponse` while processing an `GSVCGeocodeRequest`. This memory must be deallocated by the developer by explicitly calling this function when the response is no longer needed.

See Also

- `GSVCGeocodeResponse`

IGEOSERVICE_Init

Description

This function initializes the Geoservice extension with ClientID, Password, and language. It is called once before any other function in the IGeoservice interface. As this function only sets the internal state, no authentication is performed.

Prototype

```
GSVCExceptions IGEOSERVICE_Init(IGeoservice * po, GSVCConfig*
pConf);
```

Parameters

`po`	[in]	Interface pointer
`pConf`	[in]	Pointer to Config structure

Return Values

`GSVCExceptions`:

- `GSVC_SUCCESSFUL` is returned if there is no problem during initialization.
- If there is a initialization problem, a corresponding value will be returned.

IGEOSERVICE_PortrayMap

Description

Returns a map stored inside a `GSVCPortrayMapResponse` instance.

Prototype

```
GSVCExceptions IGEOSERVICE_PortrayMap
                 (IGeoservice *po,
                  GSVCPortrayMapRequest *req,
                  GSVCPortrayMapResponse** ppResp,
                  GSVCExceptions* peStatus,
                  AEECallback *cb);
```

Parameters

`po`	[in]	Interface pointer
`req`	[in]	Pointer to `GSVCPortrayMapRequest`
`ppResponse`	[out]	Pointer to pointer to `GSVCPortrayMapResponse`
`peStatus`	[out]	Address of `GSVCException` enum where Response Status will be returned
`cb`	[in]	Pointer to callback structure

Return Values

`GSVCExceptions`:

- `GSVC_SUCCESSFUL` is returned if there is no problem during the query initialization phase of the request.
- If there is a query initialization problem, a corresponding value will be returned.
- `GSVC_EUNSUPPORTED` is returned if this function is called before `IGEOSERVICE_Init`.

Comments

This function receives an `IGeoservice` pointer and a pointer to `GSVCPortrayMapRequest`. On completion of the `GSVCPortrayMapRequest`, the function creates and populates a `GSVCPortrayMapResponse` structure, delegating control to the callback function registered in `cb` by the user. The resultant status of this method is cached in `*peStatus`.

See Also

- `AEECallback`
- `GSVCPortrayMapRequest`
- `GSVCPortrayMapResponse`

IGEOSERVICE_PortrayMapResponseFree

Description

This function releases memory allocated to a `GSVCPortrayMapResponse` structure by `IGeoservice` when it replies to an `IGEOSERVICE_PortrayMap` request.

Prototype

```
void IGEOSERVICE_PortrayMapResponseFree
                 (IGeoservice *p,
                  GSVCPortrayMapResponse* pPMResp);
```

Parameters

`p`	[in]	Interface pointer
`pPMResp`	[in]	Pointer to `GSVCPortrayMapResponse` structure to be released

Return Values

`None`

Comments

This function is used to release all the memory allocated for an instance of the `GSVCPortrayMapResponse` structure. The extension allocates memory for a `GSVCPortrayMapResponse` while processing an `GSVCPortrayMapRequest`. This memory must be deallocated by the developer by explicitly calling this function when the response is no longer needed.

See Also

- `GSVCPortrayMapResponse`

IGEOSERVICE_Release

Description

This function decrements the reference count of the `IGeoservice` Interface object. The object is freed from memory, and is no longer valid, when the reference count reaches `0` (zero).

Prototype

```
uint32 IGEOSERVICE_Release(IGeoservice * pIGeoservice);
```

Parameters

`pIGeoservice`	[in]	Pointer to the IGeoservice Interface object

Return Values

Decremented reference count for the object. A value of `0` (zero) is returned if the object has been freed and is no longer valid.

See Also

- `IGEOSERVICE_AddRef()`

IGEOSERVICE_ReverseGeocode

Description

This function converts a pair of geographical coordinates (lat, lon) into a postal address.

Prototype

```
GSVCExceptions IGEOSERVICE_ReverseGeocode
                 (IGeoservice *po,
                  GSVCReverseGeocodeRequest *req,
                  GSVCReverseGeocodeResponse** resp,
                  GSVCExceptions* peStatus,
                  AEECallback *cb);
```

Parameters

po	[in]	Interface pointer
req	[in]	Pointer to GSVCReverseGeocodeRequest
resp	[out]	Pointer to pointer to GSVCReverseGeocodeResponse structure
*peStatus	[out]	Address of GSVCException enum where Response Status will be returned
cb	[in]	Pointer to callback structure

Return Values

GSVCExceptions:

- GSVC_SUCCESSFUL is returned if there is no problem during the query initialization phase of the request.
- If there is a query initialization problem, a corresponding value will be returned.
- GSVC_EUNSUPPORTED is returned if this function is called before IGEOSERVICE_Init.

Comments

This function receives an `IGeoservice` pointer and a pointer to `GSVCReverseGeocodeRequest`. On completion of the `GSVCReverseGeocodeRequest,` the function creates and populates a `GSVCReverseGeocodeResponse structure`, delegating control to the callback function registered in `cb` by the user. The resultant status of this method is cached in `*peStatus`.

See Also

- `GSVCReverseGeocodeRequest`
- `GSVCReverseGeocodeResponse`
- `AEECallback`

IGEOSERVICE_ReverseGeocodeResponseFree

Description

This function releases memory allocated to a `GSVCReverseGeocodeResponse` structure by `IGeoservice` when it replies to an `IGEOSERVICE_ReverseGeocode` request.

Prototype

```
void IGEOSERVICE_ReverseGeocodeResponseFree
                (IGeoservice *p,
                 GSVCReverseGeocodeResponse* pRGResp);
```

Parameters

`p`	[in]	Interface pointer
`pRGResp`	[in]	Pointer to `GSVCReverseGeocodeResponse` structure to be released

Return Values

`None`

Comments

This function is used to release all the memory allocated for an instance of the `GSVCReverseGeocodeResponse` structure. The extension allocates memory for a `GSVCReverseGeocodeResponse` while processing an `GSVCReverseGeocodeRequest`. This memory must be deallocated by the developer by explicitly calling this function when the response is no longer needed.

See Also

- `GSVCReverseGeocodeResponse`

IGeoservice Data Types

This section describes the data types used by the `IGeoservice` Interface functions. These data types define the format and content of the data that is passed by applications to the `IGeoservice` Interface functions, and received by the applications. Type definitions for the `IGeoservice` Interface data structures are contained in the *IGeoservice.h* header file. The description of each `IGeoservice` Interface function contains links to the descriptions of all relevant data structures.

GSVCAddress

Description

This structure contains a postal address, which appears in various contexts in the `IGeoService` extension.

Definition

```
typedef struct _GSVCAddress
   {
      AECHAR* pwszStreetAddress;
      AECHAR* pwszMunicipality;
      AECHAR* pwszCountrySubdivision;
      AECHAR* pwszPostalCode;
      AECHAR* pwszCountry;

   }GSVCAddress;
```

Members

Members of `GSVCAddress`

`pwszStreetAddress`	Street name and number
`pwszMunicipality`	Municipality (e.g., City)
`pwszCountrySubdivision`	Country Subdivision (e.g., State, Province)
`pwszPostalCode`	Postal Code
`pwszCountry`	Country

GSVCAddressType

Description

`GSVCAddressType` is an enumeration used to specify the type of address being passed in a request or response.

Definition

```
typedef enum _GSVCAddressType
   {
     GSVC_LOCATION_POINT = 1,
     GSVC_LOCATION_ADDRESS = 2

   }GSVCAddressType;
```

Members

Members of `GSVCAddressType`

`GSVC_LOCATION_POINT`	Specifies that the address is in point format (lat/lon)
`GSVC_LOCATION_ADDRESS`	Specifies that the address is in detail format (street address)

GSVCAreaFilter

Description

Common base class for all geographic filters to use when finding points of interest (POIs) with GSVCFindFeaturesRequest. A search can find POIs closest to a specified location (GSVCNearestFilter), within a specified area of interest (GSVCWithinBoundaryFilter), or within a specified distance of a location (GSVCWithinDistanceFilter).

Definition

```
typedef struct _GSVCAreaFilter
   {
      GSVCAreaFilterType  eAreaFilterType;
      void*  pAreaFilterType;

   }GSVCAreaFilter;
```

Members

Members of GSVCAreaFilter

eAreaFilterType	Type of geographic filter that will be used in GSVCFindFeaturesRequest.
pAreaFilterType	Pointer to GSVCNearestFilter, GSVCWithinBoundaryFilter, or GSVCWithinDistanceFilter, depending upon value of eAreaFilterType.

See Also

- GSVCAreaFilterType
- GSVCWithinBoundaryFilter
- GSVCWithinDistanceFilter
- GSVCNearestFilter

GSVCAreaFilterType

Description

This enumeration defines the type of geographic filter that restricts the search area.

Definition

```
typedef enum _GSVCAreaFilterType
        {
          GSVC_FILTER_NEAREST=1,
          GSVC_FILTER_WITHINBOUNDARY=2,
          GSVC_FILTER_WITHINDISTANCE=3

         }GSVCAreaFilterType;
```

Members

Members of `GSVCAreaFilterType`

`GSVC_FILTER_NEAREST`	Value representing a geographic filter that finds points of interest (POIs) nearest a specified location in a `GSVCFindFeatureRequest` search
`GSVC_FILTER_WITHINBOUNDARY`	Value representing a geographic filter that finds points of interest (POIs) within a specified area of interest (AOI) in a `GSVCFindFeatureRequest` search
`GSVC_FILTER_WITHINDISTANCE`	Value representing a geographic filter that finds points of interest (POIs) within a specified distance of a location or along a route in a `GSVCFindFeatureRequest` search

Comments

This enumeration is used to identify the type of filter that will be used in a `GSVCFindFeatureRequest` search.

GSVCAreaOfInterest

Description

Common base class for all areas of interest (AOIs). An `AreaOfInterest` is a geographic area defined by a rectangle (`GSVCBoundingBox`), polygon (`GSVCPolygon`), or circle (`GSVCCircleByCenterPoint`).

Definition

```
typedef  struct _GSVCAreaOfInterest
   {
      GSVCContextType  eContextType;

   }GSVCAreaOfInterest;
```

Members

Members of `GSVCAreaOfInterest`

`eContextType`	Map context

Comments

This structure will be used as a base class for all areas of interest to avoid when calculating a route with `GSVCDetermineRouteRequest`, to search when finding points of interest (`GSVCPOI`) with `GSVCFindFeaturesRequest`, or to define the map context in the `GSVCPortrayMapRequest`.

See Also

- `GSVCContextType`

GSVCAvoid

Description

This structure specifies the geographic areas or feature to avoid when calculating the route with `GSVCDetermineRouteRequest`. A route can avoid specific street addresses (`GSVCAvoidAddress`), areas of interest (`GSVCAvoidAOI`), features (`GSVCAvoidFeature`), or points (`GSVCAvoidPoint`).

Definition

```
typedef struct _GSVCAvoid
   {
      GSVCAvoidType  eAvoidType;
      void*  pAvoid;

   }GSVCAvoid;
```

Members

Members of `GSVCAvoid`

`eAvoidType`	Enumeration value that specifies the type of geographic area or feature to avoid.
`pAvoid`	Points to the structure `GSVCAvoidAddress`, `GSVCAvoidAreaOfInterest`, `GSVCAvoidFeature`, or `GSVCAvoidPoint`, depending upon the value of `eAvoidType`.

See Also

- `GSVCAvoidFeature`
- `GSVCAvoidPoint`
- `GSVCAvoidAddress`
- `GSVCAvoidAreaOfInterest`

GSVCAvoidAddress

Description

Represents a street address (`GSVCAddress`) to avoid when calculating a route with `GSVCDetermineRouteRequest`.

Definition

```
typedef struct _GSVCAddress GSVCAvoidAddress
```

See Also

- `GSVCAddress`

GSVCAvoidAreaOfInterest

Description

Represents an area of interest (`GSVCAreaOfInterest`) to avoid when calculating a route with `GSVCDetermineRouteRequest`.

Definition

```
typedef struct _GSVCAreaOfInterest GSVCAvoidAreaOfInterest
```

See Also

- `GSVCAreaOfInterest`

GSVCAvoidFeature

Description

Enumerates the type of feature (bridges or toll roads, for example) to avoid when calculating a route with `GSVCDetermineRouteRequest`.

Definition

```
typedef struct _GSVCAvoidFeature
   {
      GSVCAvoidFeatureType  eAvoidFeatureType;

   }GSVCAvoidFeature;
```

Members

Members of `GSVCAvoidFeature`

`eAvoidFeatureType`	Enumeration that contains the feature type to avoid.

See Also

- `GSVCAvoidFeatureType`

GSVCAvoidFeatureType

Description

This enumeration identifies features to be avoided while calculating a route in `GSVCDetermineRouteRequest`.

Definition

```
typedef enum _GSVCAvoidFeatureType
     {
        GSVC_AVOIDFEATURE_TOLLWAY =1,
        GSVC_AVOIDFEATURE_BRIDGE =2,
        GSVC_AVOIDFEATURE_TUNNEL =3,
        GSVC_AVOIDFEATURE_FERRY =4

     }GSVCAvoidFeatureType;
```

Members

Members of `GSVCAvoidFeatureType`

`GSVC_AVOIDFEATURE_TOLLWAY`	Avoid street network links that represent toll roads during route calculation
`GSVC_AVOIDFEATURE_BRIDGE`	Avoid street network links that represent bridges during route calculation
`GSVC_AVOIDFEATURE_TUNNEL`	Avoid street network links that represent tunnels during route calculation
`GSVC_AVOIDFEATURE_FERRY`	Avoid street network links that represent ferry crossings during route calculation

GSVCAvoidPoint

Description

Represents a point (`GSVCPoint`) to avoid when calculating a route with `GSVCDetermineRouteRequest`.

Definition

```
typedef struct _GSVCPoint GSVCAvoidPoint
```

See Also

- `GSVCPoint`

GSVCAvoidType

Description

This enumeration identifies types of geographic areas or features to avoid when calculating a route.

Definition

```
typedef enum _GSVCAvoidType
   {
      GSVC_AVOID_POINT= 1,
      GSVC_AVOID_ADDRESS= 2,
      GSVC_AVOID_FEATURE= 3,
      GSVC_AVOID_AREAOFINTEREST= 4

   }GSVCAvoidType;
```

Members

Members of `GSVCAvoidType`

`GSVC_AVOID_POINT`	`GSVCAvoidPoint` Type Identifier
`GSVC_AVOID_ADDRESS`	`GSVCAvoidAddress` Type Identifier
`GSVC_AVOID_FEATURE`	`GSVCAvoidFeature` Type Identifier
`GSVC_AVOID_AREAOFINTEREST`	`GSVCAvoidAreaOfInterest` Type Identifier

GSVCBoundingBox

Description

Represents a rectangular area of interest (`GSVCAreaOfInterest`) defined by the lower-left and upper-right corners of a rectangle. The rectangle is a geographic area with points expressed in latitude-longitude—or for hotspots, a rectangle expressed by points in pixels, where *x* is the lon value and *y* is the lat value.

Definition

```
typedef struct _GSVCBoundingBox
   {
      GSVCAreaOfInterest  AOIBase;
      GSVCPoint  Lowerleft;
      GSVCPoint  UpperRight;

   }GSVCBoundingBox;
```

Members

Members of `GSVCBoundingBox`

`AOIBase`	AOI base class. The AOI context type must be `GSVC_BOUNDING_BOX`.
`Lowerleft`	`GSVCPoint` that contains the lower-left point of the bounding-box rectangle. Cannot be null.
`UpperRight`	`GSVCPoint` that contains the upper-right point of the bounding-box rectangle. Cannot be null.

See Also

- `GSVCAreaOfInterest`
- `GSVCPoint`

GSVCCircleByCenterPoint

Description

Represents a circular area of interest (`GSVCAreaOfInterest`) defined by the center point and radius of a circle.

Definition

```
typedef struct _GSVCCircleByCenterPoint
   {
     GSVCAreaOfInterest  AOIBase;
     GSVCPoint  centerPoint;
     uint32   unDistance;

   }GSVCCircleByCenterPoint;
```

Members

Members of `GSVCCircleByCenterPoint`

`AOIBase`	AOI base class. Must point to `GSVC_CIRCLE_BY_CENTER_PT`.
`centerPoint`	`GSVCPOINT` that contains the center of the circle. Cannot be null.
`unDistance`	The radius of the circle (in meters).

See Also

- `GSVCAreaOfInterest`
- `GSVCPoint`

GSVCConfig

Description

Represents a configuration for the server, defined by ClientID and Password.

Definition

```
typedef struct _GSVCConfig
     {
       AECHAR ClientID[CLIENTID_LENGTH];
       AECHAR PassWord[PASSWORD_LENGTH];
       AECHAR Language[LANGUAGE_LENGTH];
       uint32 unTimeout;

     }GSVCConfig;
```

Members

Members of `GSVCAvoidType`

`ClientID[CLIENTID_LENGTH]`	ClientID
`PassWord[PASSWORD_LENGTH]`	Password
`Language[LANGUAGE_LENGTH]`	Language to be used
`unTimeout`	Timeout for web transaction

GSVCContextType

Description

This enumeration sets the map context that defines a geographic area to cover in the map image.

Definition

```
typedef enum _GSVCContextType
            {
              GSVC_UNSUPPORTED=-1,
              GSVC_BOUNDING_BOX=1,
              GSVC_CIRCLE_BY_CENTER_PT=2,
              GSVC_POLYGON=3

            }GSVCContextType;
```

Members

Members of `GSVCContextType`

`GSVC_UNSUPPORTED`	Specifies that the context type is not supported
`GSVC_BOUNDING_BOX`	Specifies Bounding Box as the context type
`GSVC_CIRCLE_BY_CENTER_PT`	Specifies Circle by Center Point as the context type
`GSVC_POLYGON`	Specifies Polygon as the context type ; do not use when rendering maps.

Comments

This enumeration indicates the context type of the image generated by the server.

GSVCDetermineRouteRequest

Description

Represents a request to calculate a travel route. A route connects a start location to a destination location, and can include zero or more intermediate locations. When calculating the route, you can specify that it may avoid locations associated with traffic incidents or other unfavorable conditions. You also can specify that it take into account time-dependent traffic conditions such as lane closures, turn restrictions, and traffic data from dynamic content feeds. The actual path the route follows can vary by time of day, owing to traffic conditions and travel preferences.

Definition

```
typedef struct _GSVCDetermineRouteRequest
   {
      GSVCRequest  RequestBase;
      GSVCRoutePlan*  pRoutePlan;
      boolean  bReturnRouteGeometry;
      boolean  bReturnRouteHandle;
      AECHAR*  pwszRouteHandle;
      GSVCPortrayMapRequest*  pMapRequest;

   }GSVCDetermineRouteRequest;
```

Members

Members of `GSVCDetermineRouteRequest`

`RequestBase`	Base class for the request.
`pRoutePlan`	Contains the criteria that determines a route: • Start point • End point • Waypoints • Travel preferences • Locations and features to avoid • Travel start time
`bReturnRouteGeometry`	Set to `TRUE` to return the route geometry in the response, or FALSE to suppress it.
`bReturnRouteHandle`	A route handle is a unique identifier that expires by default in 15 minutes. Route handles are returned in the response and often are used to request additional information about a route or to request an alternate route. Set to `TRUE` to store the calculated route temporarily on the server, or `FALSE` to delete the route from the route cache.
`pwszRouteHandle`	Route handle (unique identifier) of a previously calculated route.
`pMapRequest`	Defines the map image properties of the generated route map returned in the response. Set this value to `NULL` to suppress the map image.

Comments

You cannot define the map name, map context, overlays, or hotspots for a map request within an `GSVCDetermineRouteRequest`. The map name, map context, overlay array, and hotspot flags will be ignored if set.

The map image returned with an `GSVCDetermineRouteRequest` will use the default map for the associated application and will have a bounding box context that contains the extents of the route. The route start point, end point, and route linestring overlays will use the default symbols and highlight style.

GSVCDetermineRouteResponse

Description

Represents a response to a `GSVCDetermineRouteRequest` to calculate a route.

Definition

```
typedef struct _GSVCDetermineRouteResponse
   {
      GSVCResponse  ResponseBase;
      AECHAR*  pwszRouteHandle;
      int32  nDuration;
      uint32  unDistance;
      GSVCBoundingBox*  pBbox;
      GSVCLineString*  pRouteGeometry;
      GSVCRouteInstruction**  ppRouteInstruction;
      size_t  nRouteInstruction;
      GSVCPortrayMapResponse*  pMapResponse;

   }GSVCDetermineRouteResponse;
```

Members

Members of `GSVCDetermineRouteResponse` **(1 of 2)**

`ResponseBase`	Base class for the response.
`pwszRouteHandle`	A route handle is a unique identifier for this route supplied by the server. By default, this route will be cached for 15 minutes. Route handles are often are used to request additional information about a route or to request an alternate route.
`nDuration`	The time required to traverse the route (in seconds)
`unDistance`	The distance in meters.
`pBbox`	The rectangular geographic area bounding the complete route.
`pRouteGeometry`	The route geometry linestring, ordered from start to end.

Members of `GSVCDetermineRouteResponse` **(2 of 2)**

`ppRouteInstruction`	An array of `GSVCRouteInstruction` instances representing the textual travel instructions for the route.
`nRouteInstruction`	Number of elements in the array.
`pMapResponse`	Contains the route map and related information if a route map was requested with the `GSVCDetermineRoute` request. If route map was not requested, this value will be `NULL`.

GSVCError

Description

This structure provides detailed information about a particular error.

Definition

```
typedef struct _GSVCError
  {
    int32 nErrorCode;
    AECHAR* pwszMessage;

  }GVSCError;
```

Members

Members of `GSVCError`

`nErrorCode`	The code for this error
`pwszMessage`	The message for this error

GSVCExceptions

Description

`GSVCExceptions` specifies the exceptions that can occur in this extension.

Definition

```
typedef enum _GSVCExceptions
{
   GSVC_SUCCESSFUL = 0,
   GSVC_EXCEPTION_LOW_MEMORY = -1,
   GSVC_EXCEPTION_NULL_POINTER = -2,
   GSVC_EXCEPTION_POINTER_NOT_INITIALIZED = -3,
   GSVC_EXCEPTION_BADPARAM = -4,
   GSVC_EXCEPTION_PRIVLEVEL = -5,
   GSVC_EXCEPTION_PARSE= -6,
   GSVC_EXCEPTION_UNKNOWN_REQUEST = -7,
   GSVC_EXCEPTION_UNKNOWN_RESPONSE = -8,
   GSVC_EXCEPTION_UTF_DATA_FORMAT = -9,
   GSVC_EXCEPTION_GENERAL_FAILURE = -10,
   GSVC_EXCEPTION_CANCEL = -11,
   GSVC_EXCEPTION_CONNECTION_TIMED_OUT = -12,
   GSVC_EXCEPTION_IMPROPER_STREAM = -13,
   GSVC_EXCEPTION_BAD_PROTOCOL = -16,
   GSVC_EXCEPTION_NET = -17,
   GSVC_EXCEPTION_FILE = -18,
   GSVC_EXCEPTION_ENCRYPTION = -19,
   GSVC_EXCEPTION_DECRYPTION = -20,
   GSVC_EXCEPTION_IO = -21,
   GSVC_EXCEPTION_WRONG_ARRAY_INDEX = -22,
   GSVC_EXCEPTION_RESPONSE_ERROR = -23,
   GSVC_EXCEPTION_NULL_REQUEST = -24,
   GSVC_EXCEPTION_AUTHORISATION = -25,
   GSVC_EXCEPTION_NUMBER_FORMAT = -26,
   GSVC_EXCEPTION_BUSY = -27,
   GSVC_EXCEPTION_WEB_PROTOCOL = -1281,
   GSVC_EXCEPTION_WEB_BAD_URL = -1282,
   GSVC_EXCEPTION_WEB_BAD_HOSTNAME b= -1283,
   GSVC_EXCEPTION_WEB_BAD_PORT = -1284,
   GSVC_EXCEPTION_WEB_UNSUPSCHEME = -1285,
   GSVC_EXCEPTION_WEB_DNSCONFIG = -1286,
   GSVC_EXCEPTION_WEB_DNSTIMEOUT = -1287,
   GSVC_EXCEPTION_WEB_ADDRUNKNOWN = -1288,
   GSVC_EXCEPTION_WEB_CONNECT = -1289,
   GSVC_EXCEPTION_WEB_SEND = -1290,
   GSVC_EXCEPTION_WEB_RECV = -1291,
   GSVC_EXCEPTION_WEB_BADRESPONSE = -1292,
   GSVC_EXCEPTION_WEB_BODYLENGTH = -1293,
   GSVC_EXCEPTION_WEB_PROXYSPEC = -1294,
   GSVC_EXCEPTION_WEB_SSL = -1295,
   GSVC_EUNSUPPORTED = 20,
}GSVCExceptions;
```

GSVCFindFeaturesRequest

Description

Represents a request to find nearby or specific features. You can query for features by using both geographic and property filters. A *feature* is a real-world object, such as a restaurant, hotel, street segment, or cell sector.

The response is a `GSVCFindFeaturesResponse` instance.

Definition

```
typedef struct _GSVCFindFeaturesRequest
   {
      GSVCRequest  RequestBase;
      AECHAR*  pwszDirectoryType;
      GSVCAreaFilter*  pAreaFilter;
      GSVCPOIProperty**  ppPropertyFilter;
      size_t  nPropertyFilter;
      GSVCSortCriteria  eSortCriteria;
      GSVCSortDirection  eSortDirection;
      int16  nMaxResponses;
      GSVCPropertyName*  ppwszReturnProperties;
      size_t  nReturnProperties;

   }GSVCFindFeaturesRequest;
```

Members

Members of `GSVCFindFeaturesRequest`

`RequestBase`	Contains the request ID.
`pwszDirectoryType`	Feature category path for the request.
`pAreaFilter`	Geographic filter that restricts the search area.
`ppPropertyFilter`	Array of `GSVCPOIProperty` instances that restricts the search to specific property values. Set to `NULL` for an unrestricted property-value search.
`nPropertyFilter`	Length of the property filter array.
`eSortCriteria`	Criteria for POI sorting— either `GSVC_SORTCRITERIA_NAME` or `GSVC_SORTCRITERIA_DISTANCE`.
`eSortDirection`	Sorting direction of POIs in the response (ascending/ descending)
`nMaxResponses`	Maximum number of POIs to return in the response. Default value is `5`. Set this value to greater than or equal to 1 to limit the search to a specified number of responses. If a value < 1 is specified, all matching POIs found are returned. It is recommended that you set this to a value of > 1; for example, 5.
`ppwszReturnProperties`	Array of `GSVCPropertyName` that specifies the properties to return with each POI in the response
`nReturnProperties`	Length of the return properties array.

See Also

- `GSVCRequest`
- `GSVCAreaFilter`
- `GSVCPOIProperty`
- `GSVCSortDirection`
- `GSVCSortCriteria`

GSVCFindFeaturesResponse

Description

Represents response to an `GSVCFindFeaturesRequest` to find nearby or specific features.

Definition

```
typedef struct _GSVCFindFeaturesResponse
   {
      GSVCResponse  ResponseBase;
      GSVCPOI**  ppPointOfInterest;
      size_t  nNumPois;

   }GSVCFindFeaturesResponse;
```

Members

Members of `GSVCFindFeaturesResponse`

`ResponseBase`	Response error (if any) from the server.
`ppPointOfInterest`	An array of pointers to `GSVCPOI` instances representing the features (POIs) found.
`nNumPois`	Number of POIs returned. This value will be `0` if no features were found.

Comments

In the case of a server error, all the members of `GSVCFindFeaturesResponse`, except `ResponseBase`, will be `NULL`.

See Also

- `GSVCFindFeaturesRequest`
- `GSVCPOI`

GSVCGeocodeMatch

Description

This structure represents either a geocoded match in a `GSVCGeocodeResponse` or a reverse geocoded match in a `GSVCReverseGeocodeResponse`.

Definition

```
typedef struct _GSVCGeocodeMatch
   {
      GSVCPoint  point;
      GSVCAddress*  pAddress;
      byte  nMatchCodeAccuracy;
      GSVCMatchCodeType  eMatchCodeType;
      uint32  unSrchCenterDistance;

   }GSVCGeocodeMatch;
```

Members

Members of `GSVCGeocodeMatch`

`point`	Geographic location, returned in both geocoding and reverse geocoding matches
`pAddress`	Address, returned in both geocoding and reverse geocoding matches
`nMatchCodeAccuracy`	Match accuracy, as a byte between 0 and 255 (inclusive), with higher numbers indicating better accuracy; returned only in geocoding matches
`eMatchCodeType`	Type of this match; returned only in geocoding matches
`unSrchCenterDistance`	Distance, in meters, between the original point and the reverse geocoded address of this match; returned only in reverse geocoding matches

See Also

- `GSVCPoint`
- `GSVCAddress`
- `GSVCMatchCodeType`

GSVCGeocodeRequest

Description

This structure represents a request to geocode one postal address.

Definition

```
typedef struct _GSVCGeocodeRequest
   {
      GSVCRequest RequestBase;
      uint16 unMaxMatches ;
      GSVCAddress* pAddress;

   }GSVCGeocodeRequest
```

Members

Members of `GSVCGeocodeRequest`

`RequestBase`	Holds Request ID
`nMaxMatches`	Maximum number of matches that could be fetched for this request
`pAddress`	Address to be geocoded

See Also

- `GSVCRequest`
- `GSVCAddress`

GSVCGeocodeResponse

Description

This structure represents a response to a `GSVCGeocodeRequest`.

Definition

```
typedef struct _GSVCGeocodeResponse
  {
    GSVCResponse ResponseBase;
    GSVCGeocodeMatch* pGeocodeMatches;
    size_t    nNoOfMatches;

  }GSVCGeocodeResponse;
```

Members

Members of `GSVCGeocodeResponse`

`ResponseBase`	Holds error response (if any) from the server
`pGeocodeMatches`	Array of GeocodeMatch instances representing candidate matches for the address, ordered by match accuracy
`nNoOfMatches`	Length of array of GeocodeMatches

Comments

If an error occurs, the `pGeocodeMatches` member is `NULL` and the error information is contained in the `ResponseBase` structure.

See Also

- `GSVCResponse`
- `GSVCGeocodeMatch`

GSVCGeocodeSearchMode

Description

Enumerates the geocoding search modes.

Definition

```
typedef enum _GSVCGeocodeSearchMode
   {
      GSVC_SEARCH_MODE_AUTO = 0,
      GSVC_SEARCH_MODE_TOKEN = 1,
      GSVC_SEARCH_MODE_STEMMING = 2

   }GSVCGeocodeSearchMode;
```

Members

Members of `GSVCGeocodeSearchMode`

`GSVC_SEARCH_MODE_AUTO`	Automatic search mode
`GSVC_SEARCH_MODE_TOKEN`	Tokenized search mode
`GSVC_SEARCH_MODE_STEMMING`	Stemming search mode

GSVCGeoRegContext

Description

This structure represents the four points at the corners of a map, as well as the map display scale, which is a string representing a double. If the `GeoRegContext` boolean variable is set to `TRUE` at the time of a request, the `GeoRegContext` information will follow at the time of the response.

Definition

```
typedef struct _GSVCGeoRegContext
  {
    GSVCPoint UpperLeft;
    GSVCPoint UpperRight;
    GSVCPoint LowerLeft;
    GSVCPoint LowerRight;
    AECHAR*   pwszMapDisplayScale;
    size_t    nNoOfMatches;

  }GSVCGeoRegContext;
```

GSVCHotSpot

Description

Represents a bounding box in pixel coordinates of a point of interest (POI) or point (`GSVCPoint`) overlaid on a map image. A hotspot also can contain the coordinates of a label associated with a point of interest.

A hotspot bounding box is represented using the standard geographic (`GSVCBoundingBox`) and Point types; therefore *x* pixel values represent longitude and *y* pixel values represent latitude.

Definition

```
typedef struct _GSVCHotSpot
   {
      GSVCBoundingBox*  pBoundingBox;
      GSVCBoundingBox*  pLabelBBox;
      AECHAR  pwszId;

   }GSVCHotSpot;
```

Members

Members of `GSVCHotSpot`

`pBoundingBox`	Represents the bounding box of the overlaid location associated with this hotspot, in pixel image coordinates.
`pLabelBBox`	Represents the bounding box of the label associated with this hotspot, in pixel image coordinates.
`pwszId`	Represents the unique identifier of this hotspot.

See Also

- `GSVCBoundingBox`

GSVCImageType

Description

This enumeration is used to specify the image format the server will generate.

Definition

```
typedef enum _GSVCImageType
    {
      GSVC_IMAGETYPE_PNG8,=0,
      GSVC_IMAGETYPE_PNG2,=1,
      GSVC_IMAGETYPE_PNG24,=2,
      GSVC_IMAGETYPE_WBMP1U,=3,
      GSVC_IMAGETYPE_BMP1=4,
      GSVC_IMAGETYPE_BMP8=5,
      GSVC_IMAGETYPE_GIF=6

    }GSVCImageType;
```

Members

Members of `GSVCImageType`

`GSVC_IMAGETYPE_PNG2`	2-bit grayscale Portable Network Graphics (PNG)
`GSVC_IMAGETYPE_PNG8`	8-bit color PNG (256 colors)
`GSVC_IMAGETYPE_PNG24`	24-bit color PNG (16,777,216 colors)
`GSVC_IMAGETYPE_WBMP1U`	1-bit uncompressed black-and-white bitmap (BMP)
`GSVC_IMAGETYPE_BMP1`	1-bit compressed black-and-white BMP
`GSVC_IMAGETYPE_BMP8`	8-bit color BMP (256 colors)
`GSVC_IMAGETYPE_GIF`	8-bit palette-based GIF (256 colors)

Comments

The default value is `GSVC_IMAGETYPE_PNG8`.

GSVCLineString

Description

Represents a piecewise linear path defined by a list of ordered points that are connected by straight line segments. Two or more points define a linestring. Linestrings are often used to represent routes.

Definition

```
typedef struct _GSVCLineString
   {
      GSVCPoint*  pPoints;
      size_t  nPoints;

   }GSVCLineString;
```

Members

Members of `GSVCLineString`

`pPoints`	Array of points in the linestring.
`nPoints`	Number of points in the `pPoints` array. This value cannot be set to less than 2.

See Also

- `GSVCPoint`

GSVCLocation

Description

This structure contains a location, which can be represented by a postal address (`GSVCAddress`) or a geographic point (`GSVCPoint`).

Definition

```
typedef struct _GSVCLocation
   {
      GSVCAddressType  eAddressType;
      void*  pLocation;

   }GSVCLocation;
```

Members

Members of `GSVCLocation`

`eAddressType`	Specifies the type of location (point or address).
`pLocation`	Pointer to `GSVCAddress` or `GSVCPoint`.

See Also

- `GSVCAddressType`
- `GSVCAddress`
- `GSVCPoint`

GSVCMatchCodeType

Description

This enumeration defines the type of match to be used with `GSVCGeocodeMatch`.

Definition

```
typedef enum  _GSVCMatchCodeType
  {
    GSVC_MATCH_TYPE_EXACT= 0,
    GSVC_MATCH_TYPE_STREET_NUMBER_RANGE= 1,
    GSVC_MATCH_TYPE_POSTAL_CODE= 2,
    GSVC_MATCH_TYPE_MUNICIPALITY_CENTROID= 3,
    GSVC_MATCH_TYPE_NONE= 4

   }GSVCMatchCodeType;
```

Members

Members of `GSVCMatchCodeType`

`GSVC_MATCH_TYPE_EXACT`	House number matched exactly
`GSVC_MATCH_TYPE_STREET_NUMBER_RANGE`	Street name or a street intersection matched exactly.
`GSVC_MATCH_TYPE_POSTAL_CODE`	Postal code matched exactly
`GSVC_MATCH_TYPE_MUNICIPALITY_CENTROID`	Municipality matched, either exactly or approximately.
`GSVC_MATCH_TYPE_NONE`	Unknown match type

Comments

The match code type is only returned in Geocoding; it is not returned for ReverseGeocoding.

GSVCNearestCriteria

Description

This enumeration sets the criteria to determine the basis on which points of interest (POIs) are considered to be nearest to the location of the client.

Definition

```
typedef enum _GSVCNearestCriteria
        {
          GSVC_NEARESTCRITERIA_PROXIMITY=1,
          GSVC_NEARESTCRITERIA_FASTEST=2,
          GSVC_NEARESTCRITERIA_SHORTEST=3

        }GSVCNearestCriteria;
```

Members

Members of `GSVCNearestCriteria`

`GSVC_NEARESTCRITERIA_ PROXIMITY`	*Closest proximity* will be used to determine which POIs are nearest
`GSVC_NEARESTCRITERIA_ FASTEST`	*Fastest travel time* will be used to determine which POIs are nearest
`GSVC_NEARESTCRITERIA_ SHORTEST`	*Shortest travel distance* will be used to determine which POIs are nearest

GSVCNearestFilter

Description

Represents a geographic filter that finds points of interest (POIs) nearest a specified location in a `GSVCFindFeaturesRequest` search. The criteria that determine which POIs are nearest can be the straight-line distance, fastest travel time, or shortest travel distance.

Definition

```
typedef struct _GSVCNearestFilter
   {
      GSVCNearestCriteria  eNearestCriteria;
      GSVCLocation*  pLocation;

   }GSVCNearestFilter;
```

Members

Members of `GSVCNearestFilter`

`eNearestCriteria`	Enumeration indicating the criteria that determine the nearest POIs.
`pLocation`	Pointer to `GCVCPOINT` or `GSVCAddress`. Cannot be `NULL`.

See Also

- `GSVCNearestCriteria`

GSVCOverlay

Description

This structure defines overlays. An overlay displays locations and routes geographically by overlaying them on a base map image. An overlay can show a point (`GSVCPointOverlay`), point of interest (`GSVCPOIOverlay`), or route (`GSVCRouteOverlay`). Map hotspots (`GSVCHotSpot`) are returned when the hotspot id is not `NULL`.

Definition

```
typedef struct _GSVCOverlay
       {
        GSVCOverlayType eOverlayType;
        AECHAR*   pwszHotSpotId;
        GSVCOverlayStyle pwszOverlayStyle;

          } GSVCOverlay;
```

Members

Members of `GSVCOverlay`

`pwszHotSpotId`	Unique identifier of this overlay, required to request hotspots with `GSVCHotSpot`.
`pwszOverlayStyle`	Contains the overlay style that will be used on the map.
`eOverLaytype`	Type of overLay to be used.

Comments

You must set a unique identifier to request hotspots and your application must guarantee the identifier's uniqueness within each `PortrayMapRequest`. Note that you can reuse identifiers across different requests. For route overlay, specify a style of `1`, which is used to highlight the route.

See Also

- `GSVCOverlayStyle`
- `GSVCOverlayType`

GSVCOverlayStyle

Description

This structure specifies the style, or icon, of an overlay to be rendered on top of a base map.

Definition

```
typedef AECHAR* GSVCOverlayStyle;
```

Comments

Default overlay styles are shown in the following table.

Symbols for LocationLogic Maps (1 of 2)

Type	Symbol	Name

Friend	🚶 🚶 🚶 🚶 🚶	`LL - Red Friend`
		`LL - Black Friend`
		`LL - Green Friend`
		`LL - Blue Friend`
		`LL - Orange Friend`

Golf	⛳	`LL - Golf`
Recreation (color)	🏃	`LL - Recreation`
Recreation (black-and-white)	🏃	`LL - Recreation`
Shopping (color)	🛒	`LL - Shopping`
Shopping (black-and-white)	🛒	`LL - shopping`
Historical POI (color)	✻	`LL - Historical`
Historical POI (black-and-white)	✻	`LL - Historical`
Hospital (color)	✚	`LL - Hospital - Cross - Red`

Symbols for LocationLogic Maps (2 of 2)

Type	Symbol	Name
Hospital (black-and-white)	✚	`LL - Hospital - Cross`
Airport	✈	`LL - Airport`

Traffic Incident	! ! ! !	`LL - Incident - Black/White`
		`LL - Incident - Red`
		`LL - Incident - Orange`
		`LL - Incident - Green`

Start point (route)	Ⓢ	`LL - Start`
End point (route)	Ⓔ	`LL - End`
Location marker (you are here)	⚑	`LL - Black_Red Flag`
Step markers and POI markers (1-100)	➊	`LL - 1 ... LL - 99`
	⚑	`LL - Red Flag`
	⚑	`LL - Black Flag`

GSVCOverlayType

Description

This enumeration specifies the type of an overlay rendered on top of a base map.

Definition

```
typedef enum _GSVCOverlayType
          {
            GSVC_POINT_OVERLAY = 1,
            GSVC_POI_OVERLAY   = 2,
            GSVC_ROUTE_OVERLAY = 3

          }GSVCOverlayType;
```

Members

Members of `GSVCOverlayType`

`GSVC_POINT_OVERLAY`	Specifies Point as the overlay type
`GSVC_POI_OVERLAY`	Specifies Point of Interest as the overlay type
`GSVC_ROUTE_OVERLAY`	Specifies Route as the overlay type

Comments

This enumeration is used to specify the type of overlay on the map generated by server.

GSVCPOI

Description

Represents a point of interest (`GSVCPOI`). A point of interest is a vendor-supplied feature, such as a business, facility, restaurant, or any place that has a location. Each POI has properties (`GSVCPOIProperty`) that describe or identify it; not all properties are available for all POIs.

Definition

```
typedef struct _GSVCPOI
   {
      GSVCPOIProperty*  pProperties;
      size_t  nProperties;

   }GSVCPOI;
```

Members

Members of `GSVCPOI`

`pProperties`	Array of `GSVCPOIProperty` containing POI properties.
`nProperties`	Number of elements in the POI properties array.

See Also

- `GSVCPOIProperty`

GSVCPOIOverlay

Description

Represents a map overlay (`GSVCOverlay`) defined by a point of interest.

Definition

```
typedef struct _GSVCPOIOverlay
   {
      GSVCOverlay  OverlayBase;
      GSVCPOI  Poi;
      AECHAR*  pwszLabel;

   }GSVCPOIOverlay;
```

Members

Members of `GSVCPOIOverlay`

`OverlayBase`	Instance of `GSVCOverlay` that contains the overlay's unique identifier.
`Poi`	Description of the POI to be shown on the map. The geometry must be of the type `GSVCPOINT`. This value cannot be `NULL`.
`pwszLabel`	Overlay label that replaces the POI name. If this value is `NULL`, the returned map will contain hotspot information for the POI, but no label will be displayed on the map.

See Also

- `GSVCPOI`

GSVCPoint

Description

This structure is used to represent a location on the Earth's surface as a latitude and longitude (lat/lon) coordinate pair.

Definition

```
typedef struct _GSVCPoint
  {
    int32   nLat;
    int32   nLon;

  }GSVCPoint;
```

Members

Members of `GSVCPoint`

`nLat`	Latitude, 180/2^25 degrees, WGS-84 ellipsoid
`nLon`	Longitude, 360/2^26 degrees, WGS-84 ellipsoid

Comments

This is the same format as that returned by gpsOne.

See Also

- `AEEGPSInfo`

GSVCPointArray

Description

This structure is used to hold an array of points.

Definition

```
typedef struct _GSVCPointArray
   {
      GSVCPoint*  pPoints;
      size_t  nPoints;

   } GSVCPointArray;
```

Members

Members of `GSVCPointArray`

`pPoints`	Array of points (`GSVCPoint`).
`nPoints`	Number of elements in the point array.

See Also

- `GSVCPoint`

GSVCPointOverlay

Description

Represents a map overlay (`GSVCOverlay`) defined by a point (`GSVCPoint`).

Definition

```
typedef struct _GSVCPointOverlay
   {
      GSVCOverlay  OverlayBase;
      GSVCPoint  Point;

   }GSVCPointOverlay;
```

Members

Members of `GSVCPointOverlay`

`OverlayBase`	Overlay base class.
`Point`	Point to be overlaid on the map.

Comments

By default, a symbol will be overlaid on the map at the location of the point. This style must be a valid symbol name associated with the map and cannot be `NULL`.

See Also

- `GSVCPoint`

GSVCPOIProperty

Description

Represents a property of a point of interest (POI) as a name-value pair. POI properties are used as filters in `FindFeatureRequest` searches.

Definition

```
typedef struct _GSVCPOIProperty
   {
      GSVCPropertyType  eType;
      GSVCPropertyName  pwszName;
      void*  pValue;

   }GSVCPOIProperty;
```

Members

Members of `GSVCPOIProperty`

`eType`	Enumeration that specifies the POI property type.
`pwszName`	POI property name (must be valid string defined by `GSVCPropertyName`). Cannot be `NULL`.
`pValue`	Value.

See Also

- `GSVCPropertyType`
- `GSVCPropertyName`

GSVCPolygon

Description

This structure represents an area of interest (`GSVCAreaOfInterest`) defined by a polygon. A polygon is a planar surface defined by one exterior boundary and zero or more interior boundaries. Each interior boundary defines a hole in the polygon. Boundaries are defined by a set of linear rings, which are closed, simple, piecewise linear paths that cannot self-touch and are specified with a list of points (`GSVCPoint`) connected by straight line segments. To be closed, a linear ring's first and last points must be coincident.

Definition

```
typedef struct _GSVCPolygon
   {
      GSVCAreaOfInterest  AOIBase;
      GSVCPoint*  pExteriorRing;
      size_t  nExteriorRing
      GSVCPointArray*  pInteriorRings;
      size_t  nInteriorRings;

   }GSVCPolygon;
```

Members

Members of `GSVCPolygon`

`AOIBase`	AOI base class. Must point to `GSVC_POLYGON`.
`pExteriorRing`	Pointer to point array that defines the exterior boundary of the polygon. The exterior ring must contain four or more points. In order to complete the ring, the last and first points must coincide. The points must be supplied in a counter-clockwise direction.
`nExteriorRing`	Number of `GSVCPoints` in the exterior ring.
`pInteriorRings`	Array of `GSVCPointarray` structures that is used to hold the polygon's interior rings. Each interior ring must have at least four points. In order to complete the ring, the first and last coordinates must be coincident. The points must be supplied in a clockwise direction. If the polygon contains no interior boundaries, set to `NULL` or a zero length array.
`nInteriorRings`	Number of elements in the `GSVCPointArray`.

Comments

The linear rings of the interior boundary cannot cross one another and cannot be contained within one another.

See Also

- `GSVCPoint`

GSVCPortrayMapRequest

Description

Represents a request to generate a map image to display. You can define a map's logical name, dimensions, image format, and geographical area. You can also overlay a map with specific locations or routes. These overlays can have associated hotspots. The response is an `GSVCPortrayMapResponse` instance.

Definition

```
typedef struct _GSVCPortrayMapRequest
   {
      GSVCRequest  RequestBase;
      AECHAR  pwszMapname;
      uint16  unImageWidth;
      uint16  unImageHeight;
      GSVCImageType  eImageType ;
      boolean  bGeoRegContext;
      GSVCAreaOfInterest*  pMapContext;
      GSVCOverlay**  ppOverlays;
      size_t  nOverlaysLength;
      boolean  bHotSpotRequest;

   }GSVCPortrayMapRequest;
```

Members

Members of `GSVCPortrayMapRequest` (1 of 2)

`RequestBase`	Base class for the request
`pwszMapname`	Name of the map.
`unImageWidth`	Width of the image. Set this to a value between 16–2048.
`unImageHeight`	Height of the image. Set this to a value between 16–2048.
`eImageType`	Type of the image.
`bGeoRegContext`	Boolean that specifies whether `GeoRegContext` information is needed.

Members of `GSVCPortrayMapRequest` (2 of 2)

`pMapContext`	`MapContext` type, either `BoundingBox` or `Circle`.
`ppOverlays`	Array of pointers to different overlays.
`nOverlaysLength`	Length of `GSVCOverlay` array.
`bHotSpotRequest`	Hotspot request.

GSVCPortrayMapResponse

Description

Represents a response to a `GSVCPortrayMapRequest` for a map image to be displayed.

Definition

```
typedef struct _GSVCPortrayMapResponse
   {
      GSVCResponse  ResponseBase;
      uint32  unImageSizeInBytes;
      byte*  pMapImage;
      GSVCAreaOfInterest* pMapContext;
      uint16  unImageWidth;
      uint16  unImageHeight;
      GSVCImageType  eImageType;
      GSVCHotSpot*  pHotSpot;
      size_t  nHotSpots;
      GSVCGeoRegContext  GeoRegContext;

   }GSVCPortrayMapResponse;
```

Members

Members of `GSVCPortrayMapResponse` **(1 of 2)**

`ResponseBase`	Base class for the response.
`unImageSizeInBytes`	Number of bytes of image data.
`pMapImage`	Map image data.
`pMapContext`	Map context.
`unImageWidth`	Width of the image. Set this to a value between 16–2048.
`unImageHeight`	Height of the image. Set this to a value between 16–2048.
`eImageType`	Type of the image.

Members of `GSVCPortrayMapResponse` (2 of 2)

`pHotSpot`	Array of hotspots.
`nHotSpots`	Number of elements in hotspot array.
`GeoRegContext`	`GeoRegContext` data.

GSVCPropertyName

Description

The name of a property for a Point of Interest (POI).

Definition

```
typedef AECHAR* GSVCPropertyName;
```

Comments

The value of `GSVCPropertyName` cannot be `NULL`. Common examples of valid strings are:

`ID`—Unique ID for POI

`NAME`—Unique name for POI

`GEOM`—Geometric representation for POI (which may be `GSVCPoint`, `GSVCLineString`, or `GSVCPolygon`)

`PHONENUM`—Unique phone number for POI

`DESCRIPTION`—Descripton of POI

`_DISTANCE`—Distance of POI from search point, returned only when `GSVCNearestFilter` or `GSVCWithinDistanceFilter` is used in `GSVCFindFeatureRequest`

`_ADDRESS`—Street address of POI

Note These strings must map to an actual field in the POI database.

GSVCPropertyType

Description

Enumerates the types that a point of interest (POI) property (GSVCPOIProperty) can take.

Definition

```
typedef enum _GSVCPropertyType
   {
      GSVC_PROPERTYTYPE_ADDRESS = 0,
      GSVC_PROPERTYTYPE_INT = 1,
      GSVC_PROPERTYTYPE_LINESTRING = 2,
      GSVC_PROPERTYTYPE_POINT = 3,
      GSVC_PROPERTYTYPE_POLYGON = 4,
      GSVC_PROPERTYTYPE_STRING = 5,
      GSVC_PROPERTYTYPE_UNKNOWN = 6

   }GSVCPropertyType;
```

Members

Members of `GSVCPropertyType`

`GSVC_PROPERTYTYPE_ADDRESS`	POI property is a postal address (see `GSVCPoint`).
`GSVC_PROPERTYTYPE_INT`	POI property is an integer.
`GSVC_PROPERTYTYPE_LINESTRING`	POI property is a linestring (see `GSVCLineString`).
`GSVC_PROPERTYTYPE_POINT`	POI Property is a point (see `GSVCPoint`).
`GSVC_PROPERTYTYPE_POLYGON`	POI Property is a polygon (see `GSVCPolygon`).
`GSVC_PROPERTYTYPE_STRING`	POI Property is a string.
`GSVC_PROPERTYTYPE_UNKNOWN`	POI property is of an unknown type.

See Also

- `GSVCPOIProperty`

GSVCRequest

Description

This structure is used as the base class for all requests.

Definition

```
typedef struct _GSVCRequest
  {
    uint16 unReqId;

  }GSVCRequest;
```

Members

Members of `GSVCRequest`

`nReqId`	Unique number for each request instance (reserved)

GSVCResponse

Description

This structure is used as the base class for all responses.

Definition

```
typedef struct _GSVCResponse
{
  GSVCError* pErrors;
  size_t    nErrors;
  GSVCRequest* pRequest;

}GSVCResponse;
```

Members

Members of `GSVCResponse`

`pErrors`	Errors returned from the server
`nErrors`	Number of GSVCError objects in `pErrors` array
`pRequest`	Pointer to the request

See Also

- `GSVCError`

GSVCReverseGeocodeRequest

Description

This structure represents a request to reverse geocode a geographical point.

Definition

```
typedef struct _GSVCReverseGeocodeRequest
  {
    GSVCRequest RequestBase;
    GSVCPoint point;

  }GSVCReverseGeocodeRequest;
```

Members

Members of `GSVCReverseGeocodeRequest`

`RequestBase`	Holds Request ID
`point`	GSVCPoint (lat/lon) to be reverse geocoded

See Also

- `GSVCPoint`

GSVCReverseGeocodeResponse

Description

This structure represents a response to a `GSVCReverseGeocodeRequest`.

Definition

```
typedef struct _GSVCReverseGeocodeResponse
  {
    GSVCResponse ResponseBase;
    GSVCGeocodeMatch* pGeocodeMatches;
    size_t  nGeocodeMatches;

  }GSVCReverseGeocodeResponse;
```

Members

Members of `GSVCReverseGeocodeResponse`

`ResponseBase`	Holds an error response (if any) from the server
`pGeocodeMatches`	Holds the geocode matches in an array

Comments

In case of an error, `pGeocodeMatches` is set to `NULL` and `ResponseBase` contains error information.

See Also

- `GSVCResponse`
- `GSVCGeocodeMatch`

GSVCRouteInstruction

Description

Represents single route instruction in a `GSVCDetermineRouteResponse`.

Definition

```
typedef struct _GSVCRouteInstruction
   {
      AECHAR*  pwszInstruction;
      uint32  unDistance;
      int32  nDuration;

   }GSVCRouteInstruction;
```

Members

Members of `RouteInstruction`

`pwszInstruction`	Human-readable navigation instruction.
`unDistance`	Distance covered by this route instruction, expressed in the distance units of the request.
`nduration`	Time to complete the instruction, in seconds.

Comments

`GSVCDetermineRouteResponse` contains an array of `GSVCRouteInstruction`.

See Also

- `GSVCDetermineRouteResponse`

GSVCRouteOverlay

Description

Represents a route overlay defined by a linestring.

Definition

```
typedef struct _GSVCRouteOverlay
   {
      GSVCLineString pLineString;

   }GSVCRouteOverlay;
```

Members

Members of `GSVCRouteOverlay`

`pLineString`	Pointer to the Linestring which contain the set of points in the Line

Comments

The route overlay style must be `1`.

See Also

- `GSVCLineString`

GSVCRoutePreference

Description

This enumeration indicates the preference to use when calculating a route.

Definition

```
typedef enum _GSVCRoutepreference
   {
          GSVC_ROUTEPREFERENCE_FASTEST =1,
          GSVC_ROUTEPREFERENCE_SHORTEST=2,
          GSVC_ROUTEPREFERENCE_PEDESTRIAN =3

   }GSVCRoutepreference;
```

Members

Members of `GSVCRoutepreference`

`GSVC_ROUTEPREFERENCE_FASTEST`	Indicates preference for fastest driving time
`GSVC_ROUTEPREFERENCE_SHORTEST`	Indicates preference for shortest driving distance
`GSVC_ROUTEPREFERENCE_PEDESTRIAN`	Indicates preference for fastest walking time

GSVCRoutePlan

Description

Represents a route plan, which is the criteria upon which a route is determined. It contains the start point, the end point, and any waypoints (intermediate locations) for the route, travel preferences, locations and features to avoid, and the travel start time.

Definition

```
typedef struct _GSVCRoutePlan
   {
      GSVCLocation**  ppLocations;
      size_t  nLocations;
      GSVCRoutepreference eRoutepreference;
      GSVCAvoid**  ppAvoid;
      size_t  nAvoid;
      uint32  unStartTime;

   }GSVCRoutePlan;
```

Members

Members of `GSVCRoutePlan` **(1 of 2)**

`ppLocations`	An array of ordered `GSVCLocation` instances (point and/or address).
`nLocations`	Number of locations in this route plan.
`eRoutepreference`	Route preference to use when calculating route: • Fastest • Shortest • Pedestrian • Features, areas and locations to avoid when calculating route

Members of `GSVCRoutePlan` (2 of 2)

`ppAvoid`	An array of `GSVCAvoid` instances that indicates which street addresses, areas of interest, features, and points to avoid when calculating the route. Set this value is `NULL` to calculate a route that does not avoid any particular geographic areas or features.
`nAvoid`	Number of `Avoid` types
`unStartTime`	Specifies the date and time at which travel is expected to begin (in milliseconds since January 6, 1980 00:00:00 GMT). This value will be NULL when a route is calculated that is not time-dependent.

GSVCSortCriteria

Description

This enumeration defines the sort order of Points of Interest (POIs) in the response of `GSVCFindFeatureRequest`.

Definition

```
typedef enum _GSVCSortCriteria
        {
          GSVC_SORTCRITERIA_NAME =1,
          GSVC_SORTCRITERIA_DISTANCE =2

        }GSVCSortCriteria;
```

Members

Members of `GSVCSortCriteria`

`GSVC_SORTCRITERIA_NAME`	Indicates sort order by POI name in the response of `GSVCFindFeatureRequest`
`GSVC_SORTCRITERIA_DISTANCE`	Indicates sort order by distance in the response of `GSVCFindFeatureRequest`

GSVCSortDirection

Description

This enumeration defines the sort direction of Points of Interest (POIs) in the response of `GSVCFindFeatureRequest`.

Definition

```
typedef enum _GSVCSortDirection
     {
        GSVC_SORTDIRECTION_ASCENDING=1,
        GSVC_SORTDIRECTION_DESCENDING =2

     }GSVCSortDirection;
```

Members

Members of `GSVCSortDirection`

`GSVC_SORTDIRECTION_ASCENDING`	Indicates an ascending sort direction for POIs in the response of `GSVCFindFeatureRequest`
`GSVC_SORTDIRECTION_DESCENDING`	Indicates a descending sort direction for POIs in the response of `GSVCFindFeatureRequest`

GSVCWithinBoundaryFilter

Description

Represents a geographic filter that finds points of interest (POIs) within a specified area of interest (`GSVCAreaOfInterest`) in a `GSVCFindFeaturesRequest` search.

Definition

```
typedef struct _GSVCWithinBoundaryFilter
   {
      GSVCAreaOfInterest*  pAreaOfInterest;

   }GSVCWithinBoundaryFilter;
```

Members

Members of `GSVCWithinBoundaryFilter`

`pAreaOfInterest`	The area of interest to search. Cannot be `NULL`. Points to `SVCBoundingBox`, `GSVCCircleByCenterPoint`, or `GSVCPolygon`.

Comments

For example:

```
pAreaOfInterest = (GSVCAreaOfInterest*)pCircleByCenterPoint;
```

Where

- `pCircleByCenterPoint` points to the structure `GSVCCircleByCenterPoint`.

In this case, `pAreaOfInterest->eContextType` should have the value `GSVC_CIRCLE_BY_CENTER_PT`.

See Also

- `GSVCAreaOfInterest`
- `GSVCBoundingBox`
- `GSVCCircleByCenterPoint`
- `GSVCPolygon`

GSVCWithinDistanceFilter

Description

Represents a geographic filter that finds points of interest (POIs) within a specified distance of a location or along a route in a `GSVCFindFeaturesRequest` search.

Definition

```
typedef struct _GSVCWithinDistanceFilter
   {
      GSVCLocation*  pLocation;
      GSVCLineString*  pLinestring;
      uint32  unMaxDistance;
      uint32  unMinDistance;

   }GSVCWithinDistanceFilter;
```

Members

Members of `GSVCWithinDistanceFilter`

`pLocation`	Location search centroid around which to search. Set to `NULL` if this filter was constructed with a route geometry rather than a location.
`pLinestring`	Linestring along which to search. Set to `NULL` if this filter was constructed with a location rather than a route geometry.
`unMaxDistance`	Maximum distance from search centroid (in meters).
`unMinDistance`	Minimum distance from search centroid (in meters).

See Also

- `GSVCLocation`
- `GSVCLineString`