



TeleType GPS Software Developer's Kit (SDK)

SDK Release

WorldNavGPSAPI Library

This section describes the WorldNavGPSAPI Library in the header file `WorldNavGPSAPI.h`.

WorldNavGPSAPI Library Data Structures:

eTTTRUCKSETTINGS_HAZMAT:

`eTTTRUCKSETTINGS_HAZMAT` defines hazardous material information in trucks. Each class belongs to different types of hazardous materials. The truck route considers these classes to create a route to specified destination.

```
enum eTTTRUCKSETTINGS_HAZMAT {
    TRUCKSETTINGS_HAZMATNONE = 0, // none
    TRUCKSETTINGS_HAZMATCLASS1, // explosives
    TRUCKSETTINGS_HAZMATCLASS2, // compressed gasses
    TRUCKSETTINGS_HAZMATCLASS3, // flammable liquids
    TRUCKSETTINGS_HAZMATCLASS4, // flammable solids
    TRUCKSETTINGS_HAZMATCLASS5, // oxidizers
    TRUCKSETTINGS_HAZMATCLASS6, // poisons
    TRUCKSETTINGS_HAZMATCLASS7, // radioactive materials
    TRUCKSETTINGS_HAZMATCLASS8, // corrosive liquids
    TRUCKSETTINGS_HAZMATCLASS9, // miscellaneous
};
```

_TTFIND_QUERY:

`_TTFIND_QUERY` contains find information to find a specific address on a map. This structure contains country, state, city, street, and house or building number to find location onto the map.

```
typedef struct _TTFIND_QUERY {
    WCHAR szCountry[TTSTRLEN]; // Country
    WCHAR szState[TTSTRLEN]; // aka Province/Region
    WCHAR szCity[TTSTRLEN]; // aka Place
    WCHAR szStreet[TTSTRLEN]; // Street Name
    ULONG ulNumber; // House/Building number
} TTFIND_QUERY, *PTTFIND_QUERY;
```

_TTFIND_ANSWERS:

`_TTFIND_ANSWERS` defines a structure to store the result of find query. This structure contains information like result count, the location information, and the point belonging to a particular location.

TeleType GPS Software Developer's Kit (SDK)

```
typedef struct _TTFIND_ANSWERS {
    DWORD dwNumResults; // number of results
    WCHAR** pszName; // strings will be of TTSTRLEN2 length, pszName[0] ...
    pszName[dwNumResults - 1]
    POINT* pptPos; // pptPos[0] ... pptPos[dwNumResults - 1], these are GPS
    Coordinates
} TTFIND_ANSWERS, *PTTFIND_ANSWERS;
```

_TTFINDPOI_QUERY:

`_TTFINDPOI_QUERY` defines a structure to contain the find of any Point of Interest (POI) on the map. This find could be by location or by phone. Each find option has different structure to deal with like if the find is by phone then user should provide phone area code and phone number to be searched.

```
typedef struct _TTFINDPOI_QUERY {
    int nPOISearchType; // 0: By Location, 1: By Phone
    WCHAR szCountry[TTSTRLEN];

    // Find by Location/Place
    typedef struct _TTPOISEARCHTYPE_LOCATION {
        UINT uiPOICategory;

        // Optional (partial) name of POI
        WCHAR szName[TTSTRLEN]; // POI name

        // If these are set, search will be set to "Anywhere" as in the
        UI
        WCHAR szState[TTSTRLEN]; // aka Province/Region
        WCHAR szCity[TTSTRLEN]; // aka Place
    } TTPOISEARCHTYPE_LOCATION;

    // Find by Phone Number
    typedef struct _TTPOISEARCHTYPE_PHONE {
        ULONG ulPhoneAreaCode; // aka Dialing Code
        ULONG ulPhoneNumber;
    } TTPOISEARCHTYPE_PHONE;

    // Populate only one of the following search types
    union {
        TTPOISEARCHTYPE_LOCATION byLocation; // see above internal struct
        for data members
        TTPOISEARCHTYPE_PHONE byPhone; // see above internal struct for
        data members
    };
} TTFINDPOI_QUERY, *PTTFINDPOI_QUERY;
```

_TTFINDINTERSECTION_QUERY:

`_TTFIND_INTERSECTIONQUERY` defines a structure to find an intersection between two streets and its location information. This structure defines a format to the query which needs to be fired to find the actual intersection location and information such as two street names.

```
typedef struct _TTFINDINTERSECTION_QUERY {
    WCHAR szCountry[TTSTRLEN]; // Country
    WCHAR szState[TTSTRLEN]; // aka Province/Region
    WCHAR szCity[TTSTRLEN]; // aka Place
    WCHAR szStreet1[TTSTRLEN]; // Street
    WCHAR szStreet2[TTSTRLEN]; // Intersecting Street
} TTFINDINTERSECTION_QUERY, *PTTFINDINTERSECTION_QUERY;
```

eTTPOI_VISIBLE:

`eTTPOI_VISIBLE` is the enumeration to hide or show Points of Interest (POI) category on the map. The name of the enum type is sufficient to recognize the hiding or showing of the POI category.

```
enum eTTPOI_VISIBLE {
    eTTPOI_VISIBLE_HIDE = 0, // Hide the specified POI category icons
    on the map
    eTTPOI_VISIBLE_SHOW, // Show the specified POI category icons
    on the map
};
```

_TTGPSINFO:

`_TTGPSINFO` defines the structure format of the GPS information which contains data like GPS Time, Position (GPS coordinates), and Elevation in mm, heading in micro degrees, speed in mm/sec, and number of satellites.

```
typedef struct _TTGPSINFO {
    BOOL bValidSignal; // if FALSE, the rest of the data is irrelevant (GPS
was not locked)
    SYSTEMTIME tUTC; // GPS time
    POINT ptPos; // see documentation on the very top...
    int nElevation; // millimeters
    int nHeading; // micro degrees
    int nSpeed; // millimeters per second
    int nNumSats; // number of satellites
} TTGPSINFO, *PTTGPSINFO;
```

TeleType GPS Software Developer's Kit (SDK)

eTTROUTESETTINGSMODE:

eTTROUTESETTINGSMODE contains the information about the route settings mode which can be used to create a route. The route settings can be selected using this enum value.

```
enum eTTROUTESETTINGSMODE {
    // Car routing
    eTTROUTESETTINGSMODE_QUICKEST = 0,           // Car Quickest
    eTTROUTESETTINGSMODE_SHORTEST,              // Car Shortest
    eTTROUTESETTINGSMODE_AVOIDFREEWAYS,        // Car Avoid Freeways
    eTTROUTESETTINGSMODE_PREFERFREEWAYS,      // Car Prefer Freeways
    eTTROUTESETTINGSMODE_CARVIA,              // Car Via My Points

    // None vehicular routing
    eTTROUTESETTINGSMODE_WALKING,             // Walking Path

    // Truck routing
    eTTROUTESETTINGSMODE_TRUCKFREEWAY,        // Truck Prefer Freeways
    eTTROUTESETTINGSMODE_TRUCKQUICKEST,      // Truck Quickest
    eTTROUTESETTINGSMODE_TRUCKQUICKESTFERRY, // Truck Quickest with Ferry
    eTTROUTESETTINGSMODE_TRUCKSHORTEST,      // Truck Shortest Distance
    eTTROUTESETTINGSMODE_TRUCKEMERGENCY,     // Truck Emergency, ignores
non-physical restrictions
    eTTROUTESETTINGSMODE_TRUCKVIA,           // Truck Via My Points
};
```

_TTGPSROUTESETTINGS:

TTGPSROUTESETTINGS defines a structure to hold route settings information such as Allow/disallow Ferry or County Roads and allow/disallow toll roads while creating a route.

```
typedef struct _TTGPSROUTESETTINGS {
    int nMode; // eTTMODE_QUERY or eTTROUTESETTINGSMODE
    BOOL bFerry; // Allow ferries (for Cars) if TRUE OR Allow County Roads
                (For Trucks) if TRUE
    BOOL bToll; // Allow tolls if TRUE
} TTGPSROUTESETTINGS, *PTTGPSROUTESETTINGS;
```

eTTROUTEDETAILSTYPE:

eTTROUTEDETAILSTYPE defines an enumeration for the turn types a route can contain. The field pnType in structure uses this enumeration to fill up the turn type in the route details.

```
enum eTTROUTEDETAILSTYPE {
    eTTROUTEDETAILSTYPE_INFO = 0,
    eTTROUTEDETAILSTYPE_TURNLEFT,
```

```
eTTROUTEDetailSTYPE_BEARLEFT,  
eTTROUTEDetailSTYPE_TURNRIGHT,  
eTTROUTEDetailSTYPE_BEARRIGHT,  
eTTROUTEDetailSTYPE_EXIT,  
eTTROUTEDetailSTYPE_MERGE,  
eTTROUTEDetailSTYPE_CONTINUE,  
eTTROUTEDetailSTYPE_START,  
eTTROUTEDetailSTYPE_END,  
eTTROUTEDetailSTYPE_UTURNLEFT,  
eTTROUTEDetailSTYPE_UTURNRIGHT,  
eTTROUTEDetailSTYPE_VIA,  
};
```

_TTGPSROUTEDETAILS:

`_TTGPSROUTEDETAILS` defines a structure to hold the information about the route details of a particular route. This structure contains step by step information about the route such as each step in route to be followed along with distance from previous point for each step, GPS Co-ordinate for each position, type of turn, total route time and total route distance.

```
typedef struct _TTGPSROUTEDETAILS {  
    DWORD dwNumInstr; // Number of turn instructions  
    WCHAR** pszInstr; // Instruction strings will be of TTSTRLEN2 length,  
    pszInstr[0] ... pszInstr[dwNumInstr - 1]  
    POINT* pptPos; // GPS coordinate of turn, pptPos[0] ...  
    pptPos[dwNumInstr - 1]  
    DWORD* pdwDist; // Distance to turn from previous turn in  
    meters, pdwDist[0] .. pdwDist[dwNumInstr - 1]  
    int* pnType; // Type of turn (eTTROUTEDetailSTYPE),  
    pnType[0] ... pnType[dwNumInstr - 1]  
    DWORD dwTotalTime; // Total Time of Route, in minutes  
    DWORD dwTotalDist; // Total Distance of Route, in meters  
} TTGPSROUTEDETAILS, *PTTGPSROUTEDETAILS;
```

eTTSIMMODE:

`eTTSIMMODE` defines values of Simulator mode in Teletype GPS Application. Value 0 i.e. `eTTSIMMODE_OFF` indicates the simulator mode off and `eTTSIMMODE_ON` indicates the Simulator ON mode.

```
enum eTTSIMMODE {  
    eTTSIMMODE_OFF = 0, // Simulator OFF  
    eTTSIMMODE_ON // Simulator ON  
};
```

_TTGPSTRUCKSETTINGS:

`_TTGPSTRUCKSETTINGS` defines a structure to hold the information related to the truck such as its height, length, weight. While creating the truck route, the Software takes care of these settings to create a specific route accordingly.

```
typedef struct _TTGPSTRUCKSETTINGS {
    int nMode; // eTTMODE_QUERY or eTTTRUCKSETTINGS_HAZMAT
    int nTruckHeight; // stored in Feet (ft) * 100 (implied 2 decimal
places), ex. 1650 => 16.50 feet
    int nTruckLength; // stored in Feet (ft) * 100 (implied 2 decimal
places), ex. 5300 => 53.00 feet
    int nTruckWeight; // stored in Pounds (lb) * 100 (implied 2 decimal
places), ex. 8000000 => 80,000.00 lbs
} TTGPSTRUCKSETTINGS, *PTTGPSTRUCKSETTINGS;
```

eTTVIEWMODE:

`eTTVIEWMODE` contains information about the view mode in Teletype GPS Application. There are 4 kinds of modes available such as 2D Auto Zoom Mode, 3D Auto Zoom Mode, 2D No Auto Zoom Mode, and 3D No Auto Zoom Mode.

```
enum eTTVIEWMODE {
    eTTVIEWMODE_2DAUTOZOOM = 0, // 2D view mode with Auto-Zoom
    eTTVIEWMODE_3DAUTOZOOM, // 3D view mode with Auto-Zoom
    eTTVIEWMODE_2DNOAUTOZOOM, // 2D view mode without Auto-Zoom
    eTTVIEWMODE_3DNOAUTOZOOM // 3D view mode without Auto-Zoom
};
```

eTTDAYNIGHTMODE:

`eTTDAYNIGHTMODE` contains information about the Day or Night mode in Teletype GPS Application. There are 3 kinds of modes available such as Day Mode, Night Mode or Auto Mode.

```
enum eTTDAYNIGHTMODE {
    eTTDAYNIGHTMODE_DAY = 0, // Day colors
    eTTDAYNIGHTMODE_NIGHT, // Night colors
    eTTDAYNIGHTMODE_AUTO // Automatically switches between Day and
Night according to Sunrise/Sunset
};
```

eTTSYSUNITMODE:

eTTSYSUNITMODE contains information about the unit used to measure length, weight etc in Teletype GPS Application. There are 2 kinds of modes available such as Metric Units or English Units.

```
enum eTTSYSUNITMODE {
    eTTSYSUNITMODE_METRIC = 0,    // Metric units
    eTTSYSUNITMODE_ENGLISH,      // English units (Imperial)
};
```

eTTMAPMODE_TYPE:

eTTMAPMODE_TYPE contains information about the map mode type used in Teletype GPS Application. There are 2 kinds of map modes are available such as Map Mode in which cursor is a square cursor and the nav mode in which cursor becomes triangular and GPS gets locked.

```
enum eTTMAPMODE_TYPE {
    eTTMAPMODE_TYPE_MAP = 0, // Map mode (Square Cursor)
    eTTMAPMODE_TYPE_NAV      // Nav mode (Triangle Cursor, GPS is
locked)
};
```

_TTTRIPSTAT:

_TTTRIPSTAT defines a structure to hold the information about the trip statistics. This statistics include information such as current speed, distance to go to destination, estimated time of arrival, current time, elapsed distance, and elapsed time.

```
typedef struct _TTTRIPSTAT {
    BOOL bValidSignal; // if FALSE, only nOdometer and lTripTimer will
contain valid data (GPS was not locked)
    BOOL bValidRoute; // if FALSE, nDistToGo and lETA will contain
irrelevant data.
    int nSpeed; // millimeters per second
    int nDistToGo; // distance left in millimeters
    LONGLONG lETA; // estimated time of arrival in seconds
    SYSTEMTIME tTime; // current UTC time
    int nOdometer; // elapsed distance in millimeters
    LONGLONG lTripTimer; // elapsed time in seconds
} TTTRIPSTAT, *PTTTRIPSTAT;
```

Teletype GPS Software Developer's Kit (SDK)

eTTNAVPANEL_TYPE:

eTTNAVPANEL_TYPE contains information about the Navigation Panel settings which are displayed during the navigation mode. There are 5 kinds of navigation panel types are available in Teletype GPS Application such as Big Panels, Big Map Small Panels, Hide Panels, See Through big Panels, See Through Panels.

```
enum eTTNAVPANEL_TYPE {
    eTTNAVPANEL_TYPE_BIGPANELS = 0,        // Use Big Panels
    eTTNAVPANEL_TYPE_BIGMAP,              // Use Big Map (Small Panels)
    eTTNAVPANEL_TYPE_HIDEPANELS,         // Hide all panels
    eTTNAVPANEL_TYPE_SEETHRUBIGPANELS,   // See Thru Big Panels (Transparent
Big Panels)
    eTTNAVPANEL_TYPE_SEETHRUPANELS,      // See Thru Panels
};
```

eTTNAVPANEL_INFO:

eTTNAVPANEL_INFO contains the panel detailed information. In Teletype GPS Application Nav Mode, user can set atmost 3 panel info in a selected nav panel type. There are 18 types of panel info types are available such as Estimated Arrival Time, Estimated Arrival Time next Distance, distance remaining to destination, distance to next destination via, heading in degrees (0, 90, 180, 270), heading in Text (N, S, NE etc), distance travelled, Satellite signal strength, current speed, current time, time to destination, time to next destination via, time to next turn, current street, next sunrise time, next sunset time, battery charge, cycle all information.

```
enum eTTNAVPANEL_INFO {
    eTTNAVPANEL_INFO_ETA = 0,             // Estimated Arrival Time, time
when destination is reached
    eTTNAVPANEL_INFO_ETA2,              // estimated time arrival next dest
(via)
    eTTNAVPANEL_INFO_DIST2GO,           // Distance to Go, distance
remaining to destination
    eTTNAVPANEL_INFO_DIST2GO2,         // dist to next dest (via)
    eTTNAVPANEL_INFO_HEADINGDEGREE,    // Heading in Degrees (0°, 90°,
180°, 270°, etc.)
    eTTNAVPANEL_INFO_HEADINGTEXT,      // Heading in Text (N, NE, E, SE, S, SW,
W, NW)
    eTTNAVPANEL_INFO_ODOMETER,          // Odometer, distance traveled
    eTTNAVPANEL_INFO_SATINFO,          // Satellite Signal Strength
    eTTNAVPANEL_INFO_SPEED,            // Current Speed
    eTTNAVPANEL_INFO_TIME,             // Current Time
    eTTNAVPANEL_INFO_TIME2DEST,        // Time to Destination, time
remaining to destination
    eTTNAVPANEL_INFO_TIME2DEST2,       // time to next dest (via)
    eTTNAVPANEL_INFO_TIME2NEXTTURN,    // time to next turn
    eTTNAVPANEL_INFO_CURSTREET,        // Current Street
};
```

```
eTTNAVPANEL_INFO_SUNRISE,           // next sunrise time
eTTNAVPANEL_INFO_SUNSET,           // next sunset time
eTTNAVPANEL_INFO_BATTERY,         // battery charge
eTTNAVPANEL_INFO_CYCLEALL,        // cycle all info
};
```

_TTGPSFINDMYPOINT_QUERY:

`__TTGPSFINDMYPOINT_QUERY` defines a structure to hold mypoint query information related to a particular mypoint such as its search type for e.g. by name, by position, name of the my point to be searched, point to be searched.

```
typedef struct _TTGPSFINDMYPOINT_QUERY {
    int nMyPointSearchType; // 0: By Name, 1: By Position

    // Find by Name
    typedef struct _TTMYPOINTSEARCHTYPE_NAME {
        WCHAR szName[TTSTRLEN]; // Name of MyPoint to search for, can be
a partial name
    } TTMYPPOINTSEARCHTYPE_NAME;

    // Find by Position
    typedef struct _TTMYPOINTSEARCHTYPE_POS {
        POINT pt; // GPS coordinate of MyPoint to search for
    } TTMYPPOINTSEARCHTYPE_POS;

    // Populate only one of the following search types
    union {
        TTMYPPOINTSEARCHTYPE_NAME byName; // see above internal struct for
data members
        TTMYPPOINTSEARCHTYPE_POS byPos; // see above internal struct for
data members
    };
} TTGPSFINDMYPOINT_QUERY, *PTTGPSFINDMYPOINT_QUERY;
```

eTTMYPOINT_RESULT:

`__eTTMYPOINT_RESULT` defines an enumeration related to My Point operation. `eTTMYPOINT_RESULT_SUCCESS` indicates successful operation for e.g. Successful creation of the mypoint. `eTTMYPOINT_RESULT_DUPLICATENAME` indicates unsuccessful operation because of the duplicate name and `eTTMYPOINT_RESULT_DUPLICATEPOINT` indicates unsuccessful operation because of my point to the specified co-ordinate is already present.

TeleType GPS Software Developer's Kit (SDK)

```
enum eTTMYPOINT_RESULT {
    eTTMYPOINT_RESULT_SUCCESS = 0,           // Successful operation
    eTTMYPOINT_RESULT_DUPLICATENAME,       // MyPoint name exists already,
failed operation
    eTTMYPOINT_RESULT_DUPLICATEPOINT,     // MyPoint position exists already,
failed operation
};
```

eTTRESTRICTIONINFO_PREFERRED:

eTTRESTRICTIONINFO_PREFERRED defines an enumeration for road preference at a particular location.

```
enum eTTRESTRICTIONINFO_PREFERRED {
    RESTRICTIONINFO_PREFERREDU = -1, // Unknown
    RESTRICTIONINFO_PREFERRED0 = 0, // No info, can be ignored
    RESTRICTIONINFO_PREFERRED1 = 1, // Designated route for semi-trailer(s)
    RESTRICTIONINFO_PREFERRED2 = 2, // Designated access road for semi-
trailer(s)
    RESTRICTIONINFO_PREFERRED3 = 3, // Designated truck route
    RESTRICTIONINFO_PREFERRED9 = 9, // Other designated route
};
```

eTTRESTRICTIONINFO_RESTRICTED:

eTTRESTRICTIONINFO_RESTRICTED defines an enumeration for road restriction at a particular location.

```
enum eTTRESTRICTIONINFO_RESTRICTED {
    RESTRICTIONINFO_RESTRICTEDU = -1, // Unknown
    RESTRICTIONINFO_RESTRICTED0 = 0, // No restriction, can be ignored
    RESTRICTIONINFO_RESTRICTED1 = 1, // No commercial vehicles
    RESTRICTIONINFO_RESTRICTED2 = 2, // Immediate access only
    RESTRICTIONINFO_RESTRICTED3 = 3, // No trucks
    RESTRICTIONINFO_RESTRICTED9 = 9, // Other restricted route
};
```

eTTRESTRICTIONINFO_HAZMATPRS:

eTTRESTRICTIONINFO_HAZMATPRS defines an enumeration for Hazmat class information at a particular location.

```
enum eTTRESTRICTIONINFO_HAZMATPRS {
    RESTRICTIONINFO_HAZMATU = -1, // Unknown
    RESTRICTIONINFO_HAZMATNONE = 0, // No Info, can be ignored
    RESTRICTIONINFO_HAZMATP = 1, // Permitted
    RESTRICTIONINFO_HAZMATTS = 2, // Soft Restrictited
    RESTRICTIONINFO_HAZMATR = 3, // Restricted};
```

eTTRESTRICTIONINFO_HAZMATCLASS:

eTTRESTRICTIONINFO_HAZMATCLASS defines an enumeration for Hazmat Class restriction info for a particular position. It details the list of information about the hazardous material such as explosives, compressed gases etc and class to which it belongs to. This enumeration info along with eTTRESTRICTIONINFO_HAZMATPRS can give information whether either the particular hazardous material is permitted or restricted at a particular point.

```
enum eTTRESTRICTIONINFO_HAZMATCLASS {
    RESTRICTIONINFO_HAZMATCLASS1 = 0, // explosives
    RESTRICTIONINFO_HAZMATCLASS2, // compressed gasses
    RESTRICTIONINFO_HAZMATCLASS3, // flammable liquids
    RESTRICTIONINFO_HAZMATCLASS4, // flammable solids
    RESTRICTIONINFO_HAZMATCLASS5, // oxidizers
    RESTRICTIONINFO_HAZMATCLASS6, // poisons
    RESTRICTIONINFO_HAZMATCLASS7, // radioactive materials
    RESTRICTIONINFO_HAZMATCLASS8, // corrosive liquids
    RESTRICTIONINFO_HAZMATCLASS9, // miscellaneous
    RESTRICTIONINFO_HAZMATCLASSTOTAL, // not used
};
```

_TTRESTRICTIONINFO:

_TTRESTRICTIONINFO defines a structure containing the restriction info for a particular position. It contains values for preferred, restricted streets, their respective Hazmat level, limit to the truck height, truck length and truck weight. The value 0 to truck height, weight and length indicates that there are no truck restriction for that particular point.

```
typedef struct _TTRESTRICTIONINFO
{
    int nPreferred; // a value from eTTRESTRICTIONINFO_PREFERRED
    int nRestricted; // a value from eTTRESTRICTIONINFO_RESTRICTED
    int nHazMat[RESTRICTIONINFO_HAZMATCLASSTOTAL]; // Indexed by using
    eTTRESTRICTIONINFO_HAZMATCLASS,
    // each index contain a value from
    eTTRESTRICTIONINFO_HAZMATPRS
    // ex. uiHazMat[RESTRICTIONINFO_HAZMATCLASS6] =
    RESTRICTIONINFO_HAZMATR, means Poisons are [R]estricted
    // ex. uiHazMat[RESTRICTIONINFO_HAZMATCLASS5] =
    RESTRICTIONINFO_HAZMATP, means Oxidizers are [P]ermitted
    // For the following, 0 means no limit, otherwise the value is the
    maximum limit for the given dimension
    int nTruckHeight; // stored in Feet (ft) * 100 (implied 2 decimal
    places), ex. 1650 => 16.50 feet
    int nTruckLength; // stored in Feet (ft) * 100 (implied 2 decimal
    places), ex. 5300 => 53.00 feet
};
```

TeleType GPS Software Developer's Kit (SDK)

```
int nTruckWeight; // stored in Pounds (lb) * 100 (implied 2 decimal
places), ex. 8000000 => 80,000.00 lbs} TTRESTRICTIONINFO, *
PTTRESTRICTIONINFO;
```

POI Categories:

There are different kinds of POI Categories available in the Teletype GPS Application. Below is the list of POI categories and its related code.

0 All

```
100 Food
    101 Restaurant
    102 Drink
    103 Other
200 Transportation
    201 Gas Station
    202 Truck Stop
    203 Rest Area
    204 Distribution Center
    205 Weigh Station
    206 Parking
    207 Surveillance Camera
    208 Auto Service
    209 Car Rental
    210 Car Wash
    211 Airport
    212 Other
    213 Truck Terminal
300 Other
    301 Shopping Mall
    302 Tourist
    303 Bank/ATM
    304 Hotel/Motel
    305 Hospital
    306 Post Office
    307 Police
    308 Religious Structure
    309 Library
    310 Other
```

TeleType GPS Software Developer's Kit (SDK)

ICON IDs:

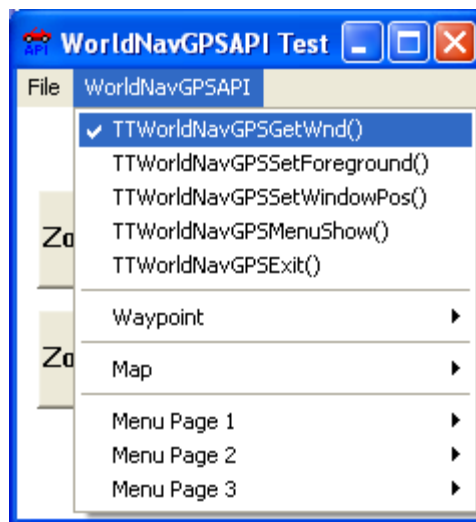
0: Medical Red Cross		
1: Airplane	2: Flag (Question Mark)	3: Heart (Red)
4: Camera	5: Bullseye	6: Soccer Ball
7: Mail Letter	8: Book (Blue)	9: Shopping Cart
10: Check Mark	11: Book (Light Blue)	12: House
13: Gift Box	14: Heart (Blue)	15: Apple
16: Star (Yellow)	17: Orchid (Purple)	18: Tall Building
19: Clover	20: Pad Lock	21: Star (Red)
22: Two Person Avatar	23: Cup of Coffee	24: Briefcase
25: No Entry Sign	26: Bell	27: Bag of Money
28: Burger	29: Flag (United Kingdom)	30: Flag(United States)
31: Smiley	32: Secret	33: Red X
34: Diamond (Red)	35: Car	36: Flag (Canada)
37: Question Mark Sphere	38: Bullet Point	39: Life Preserver

Teletype GPS Software Developer's Kit (SDK)

WorldNavGPSAPI Test

To test the Teletype GPS API on a Windows PC, make sure that you have the files WorldNavGPSAPI.dll and Test.exe together in same directory. Launch Test.exe along with Teletype GPS Application (WorldNavigator.exe).

There will be 2 menus : File, WorldNavGPSAPI.
It will appear like so:



Following table maps each menu item to its corresponding API function. A brief description is also given:

Sr. No.	Function	Description
1	TTWorldNavGPSGetWnd()	Get the main window Handle
2	TTWorldNavGPSSetForeground()	Makes the Teletype GPS application in focus
3	TTWorldNavGPSSetWindowPos()	Resizes the Teletype GPS Application
4	TTWorldNavGPSMenuShow()	Shows/Hides Teletype GPS Application Menu bar.
5	TTWorldNavGPSExit()	Quits Teletype GPS Application
6	TTWorldNavGPSCreateMyPoint()	Adds mypoint to Teletype GPS Application at user specified location with user specified name and icon
7	TTWorldNavGPSDeleteMyPoint()	Removes a mypoint from user specified GPS Co-ordinate (if any)

TeleType GPS Software Developer's Kit (SDK)

8	TTWorldNavGPSFindMyPoints()	Gets a mypoint from user specified GPS Co-ordinate or user specified name (if any)
9	TTWorldNavGPSMapUp()	Re-center map above current location
10	TTWorldNavGPSMapDown()	Re-center map below current location
11	TTWorldNavGPSMapLeft()	Re-center map left of current location
12	TTWorldNavGPSMapRight()	Re-center map right of current location
13	TTWorldNavGPSZoomIn()	Zooms into the map from current location
14	TTWorldNavGPSZoomOut()	Zooms out of map from current location
15	TTWorldNavGPSZoomToPoint()	Re-centers map to a specified GPS co-ordinate position
16	TTWorldNavGPSZoomToRect()	Re-centers map to a given rectangle
17	TTWorldNavGPSZoomCtrlShow()	Show/Hide Zoom Controls in Teletype GPS Application
18	TTWorldNavGPSRefreshMap()	Refreshes map in Teletype GPS Application
19	TTWorldNavGPSMapMode()	Gets or Sets the Map Mode in Teletype GPS Application
20	TTWorldNavGPSCurPos()	Gets a current cursor's GPS Co-ordinates
21	TTWorldNavGPSLocaleInfo()	Gets locale information at user-specified position
22	TTWorldNavGPSRouteTo()	Creates a route from one Point to another Point.
23	TTWorldNavGPSRouteVia()	Creates a route from one Point to another Point via user-specified list of points
24	TTWorldNavGPSViewRouteDetail()	Views a Route step by step along with other route details.
25	TTWorldNavGPSDeleteRoute()	Deletes a current route.
26	TTWorldNavGPSRouteToDestName()	Creates a route from one point to another and voice reads the destination name specified.
27	TTWorldNavGPSRouteSettings()	Queries or sets the Route Settings in Teletype GPS Application
28	TTWorldNavGPSSimulator()	Queries or sets GPS Simulator mode in Teletype GPS Application
29	TTWorldNavGPSTruckSettings()	Queries or sets Truck Settings in Teletype GPS Application
30	TTWorldNavGPSFind()	Finds a specified address

Teletype GPS Software Developer's Kit (SDK)

31	TTWorldNavGPSFindPOI()	Finds a point of Interest
32	TTWorldNavGPSFindIntersection()	Finds an intersection between two roads (if any)
33	TTWorldNavGPSPosInfo()	Get position info at user-specified position
34	TTWorldNavGPSUnitMode()	Queries or Sets the Teletype GPS Application System Unit
35	TTWorldNavGPSViewMode()	Queries or Sets the Teletype GPS Application View Mode
36	TTWorldNavGPSPOIMode()	Shows or Hides Points of Interest (POI) in Teletype GPS Application
37	TTWorldNavGPSDayNightMode()	Queries or sets the Day/Night mode in Teletype GPS Application
38	TTWorldNavGPSGPSInfo()	Retrieves the GPS information
39	TTWorldNavGPSTripStatReset()	Resets the trip statics
40	TTWorldNavGPSTripStat()	Retrieves the Trip Statistics information
41	TTWorldNavGPSNavPanel()	Queries or sets the Navigation Panel Settings in Teletype GPS Application
42	TTWorldNavGPSFree()	Frees allocated memory from Certain API Calls.
43	TTWorldNavGPSGetFullPath()	Gets the location of currently running WorldNav executable.
44	TTWorldNavGPSRestrictionInfo()	Get Truck Restriction info at a particular location.
45	TTWorldNavEnableBlockingRouteTo()	Enable/Disable Blocking call to wait for route to finish.

Following is a detail of each of the Functions available in the Teletype GPS API:

1) void WINAPI TTGPSGetWnd(HWND &g_hWorldNav);

Behavior:

Gets the main window handle of the Teletype GPS Application.

Expected Arguments:

A NULL Window Handle object which gets filled with Teletype GPS Application as HWND if it finds one or it remains as NULL.

Return Value:

Function is Void

TeleType GPS Software Developer's Kit (SDK)

2) void WINAPI TTWorldNavGPSSetForeground(HWND hTTWorldNavGPS)

Behavior:

Makes the Teletype GPS Application in Focus.

Expected Arguments:

HWND hTTWorldNavGPS is the window handle of the TeleType Application.

Return Value:

Function is Void

3) void WINAPI TTWorldNavGPSSetWindowPos (HWND hTTWorldNavGPS,const int x,const int y,const int nWidth, const int nHeight)

Behavior:

Resizes the Teletype GPS Application.

Expected Arguments:

HWND hTTWorldNavGPS is the window handle of the TeleType Application.

int x is the x co-ordinate in pixels to position the map.

int y is the y co-ordinate in pixels to position the map.

int nWidth is map width in pixels.

int nHeight is map height in pixels.

Return Value:

Function is Void

4) void WINAPI TTWorldNavGPSMenuShow (HWND hTTWorldNavGPS,bool bShow)

Behavior:

Show/Hide Worldnav menu bar.

Expected Arguments:

HWND hTTWorldNavGPS is the window handle of the TeleType Application.

Bool bShow true for showing the menu bar and false for hiding the menu bar.

TeleType GPS Software Developer's Kit (SDK)

Return Value:

Function is Void

5) void WINAPI TTWorldNavGPSExit (HWND hTTWorldNavGPS)

Behavior:

Quits Teletype GPS Application.

Expected Arguments:

HWND hTTWorldNavGPS is the window handle of the TeleType Application.

Return Value:

Function is Void

6) void WINAPI TTWorldNavGPSCreateMyPoint (HWND hTTWorldNavGPS, const WCHAR*
pszName, const POINT &pt, const int nIconId,
int &nResult)

Behavior:

Adds mypoint Teletype GPS Application at user specified location with user specified name and icon ID.

Expected Arguments:

HWND hTTWorldNavGPS is the window handle of the TeleType Application.

WCHAR* pszName is the name for the mypoint to be added.

POINT pt is the GPS Co-ordinates for the mypoint to be added.

int nIconId is the Icon ID for the mypoint to be added. (List of ICON ID's are specified in above section)

int nResult is the result for operation. 0 if Success, 1 if duplicate name and 2 if duplicate coordinate occurred.

Return Value:

Function is Void

TeleType GPS Software Developer's Kit (SDK)

7) void WINAPI TTWorldNavGPSDeleteMyPoint (HWND hTTWorldNavGPS, const POINT& pt)

Behavior:

Removes a mypoint from user specified GPS Co-ordinate.

Expected Arguments:

HWND hTTWorldNavGPS is the window handle of the TeleType Application.

POINT pt is the GPS Co-ordinate of the mypoint to be removed.

Return Value:

Function is Void

8) void WINAPI TTWorldNavGPSFindMyPoints (HWND hTTWorldNavGPS, const TTGPSFINDMYPOINT_QUERY &TTFindMyPoint_Query, PTTFIND_ANSWERS &pTTFind_Answers)

Behavior:

Finds all mypoints available in Teletype GPS Application according input structured query.

Expected Arguments:

HWND hTTWorldNavGPS is the window handle of the TeleType Application.

TTGPSFINDMYPOINT_QUERY &TTFindMyPoint_Query is structure object which has necessary information related to search of the mypoint. The structure should be filled with information like search type e.g. by name or by point and relative information such as name of the mypoint or GPS co-ordinate of the mypoint.

PTTFIND_ANSWERS pTTFind_Answers is the void pre-allocated structure which gets an output as the search results or it remains NULL if no match address found.

Return Value:

Function is Void

9) void WINAPI TTWorldNavGPSMapUp (HWND hTTWorldNavGPS)

Behavior:

Re-center map above current location.

Expected Arguments:

HWND hTTWorldNavGPS is the window handle of the TeleType Application.

Return Value:

Function is Void

10) void WINAPI TTWorldNavGPSMapDown (HWND hTTWorldNavGPS)

Behavior:

Re-center map below current location.

Expected Arguments:

HWND hTTWorldNavGPS is the window handle of the TeleType Application.

Return Value:

Function is Void

11) void WINAPI TTWorldNavGPSMapLeft (HWND hTTWorldNavGPS)

Behavior:

Re-center map left of current location.

Expected Arguments:

HWND hTTWorldNavGPS is the window handle of the TeleType Application.

Return Value:

Function is Void

12) void WINAPI TTWorldNavGPSMapRight (HWND hTTWorldNavGPS)

Behavior:

Re-center map right of current location.

Expected Arguments:

HWND hTTWorldNavGPS is the window handle of the TeleType Application.

Return Value:

Function is Void

13) void WINAPI TTWorldNavGPSZoomIn (HWND hTTWorldNavGPS)

Behavior:

Zooms into the map from current location.

Expected Arguments:

HWND hTTWorldNavGPS is the window handle of the TeleType Application.

Return Value:

Function is Void

14) void WINAPI TTWorldNavGPSZoomOut (HWND hTTWorldNavGPS)

Behavior:

Zooms out of map from current location.

Expected Arguments:

HWND hTTWorldNavGPS is the window handle of the TeleType Application.

Return Value:

Function is Void

TeleType GPS Software Developer's Kit (SDK)

15) void WINAPI TTWorldNavGPSZoomToPoint (HWND hTTWorldNavGPS,const POINT &pt)

Behavior:

Re-centers map to a specified GPS co-ordinate position.

Expected Arguments:

HWND hTTWorldNavGPS is the window handle of the TeleType Application.

POINT pt is the GPS Co-ordinate where a map needs to be re-centered.

Return Value:

Function is Void

16) void WINAPI TTWorldNavGPSZoomToRect (HWND hTTWorldNavGPS,const RECT &r)

Behavior:

Re-centers map to a given rectangle.

Expected Arguments:

HWND hTTWorldNavGPS is the window handle of the TeleType Application.

RECT r is the rectangle where a map needs to be re-centered in which Top-Left and Bottom-Right are GPS Co-ordinates forming a bounding box.

Return Value:

Function is Void

17) void WINAPI TTWorldNavGPSZoomCtrlShow (HWND hTTWorldNavGPS,BOOL bShow)

Behavior:

Show/Hide Zoom Controls in Teletype GPS Application.

Expected Arguments:

HWND hTTWorldNavGPS is the window handle of the TeleType Application.

BOOL bShow if True shows the Zoom Controls or if false hides the Zoom Controls.

Return Value:

Function is Void

TeleType GPS Software Developer's Kit (SDK)

18) void WINAPI TTWorldNavGPSRefreshMap (HWND hTTWorldNavGPS)

Behavior:

Refreshes map in Teletype GPS Application.

Expected Arguments:

HWND hTTWorldNavGPS is the window handle of the TeleType Application.

Return Value:

Function is Void

19) void WINAPI TTWorldNavGPSMapMode (HWND hTTWorldNavGPS,int &nMode)

Behavior:

Gets or Sets the Map Mode in Teletype GPS Application.

Expected Arguments:

HWND hTTWorldNavGPS is the window handle of the TeleType Application.

int nMode – when inputting it could be eTTMAPMODE_TYPE to set the value or eTTMODE_QUERY to query current Settings. Output will be in nMode having either eTTMAPMODE_TYPE (query result) or eTTMODE_ERROR if an error encounters.

Return Value:

Function is Void

20) void WINAPI TTWorldNavGPSCurPos (HWND hTTWorldNavGPS,POINT& pt)

Behavior:

Gets a current cursor's GPS Co-ordinates.

Expected Arguments:

HWND hTTWorldNavGPS is the window handle of the TeleType Application.

POINT pt is the GPS Co-ordinate which gets value as current cursor GPS co-ordinates.

Return Value:

Function is Void

TeleType GPS Software Developer's Kit (SDK)

21) void WINAPI TTWorldNavGPSLocaleInfo (HWND hTTWorldNavGPS,const POINT& pt, WCHAR* pszInfo, const DWORD dwCount)

Behavior:

Gets locale information at user-specified position. (Provided the Zoom Level is greater than or equal to 9)

Expected Arguments:

HWND hTTWorldNavGPS is the window handle of the TeleType Application.

POINT pt is the GPS Co-ordinate whose locale information needs to be retrieved.

WCHAR* pszInfo is pre-allocated string with dwCount number of characters in it.

DWORD dwCount is the length of pszInfo.

Output is pszInfo with Locale Info (City/ State/Province) of the given input or empty if nothing was found.

Output will be empty if zoom is too high.

Return Value:

Function is Void

22) void WINAPI TTWorldNavGPSRouteTo (HWND hTTWorldNavGPS,const POINT& ptStart, POINT &ptEnd)

Behavior:

Creates a route from one Point to another Point.

Expected Arguments:

HWND hTTWorldNavGPS is the window handle of the TeleType Application.

POINT ptStart is the starting GPS co-ordinate of the route to be created.

POINT ptEnd is the ending GPS co-ordinate of the route to be created.

Return Value:

Function is Void

TeleType GPS Software Developer's Kit (SDK)

23) void WINAPI TTWorldNavGPSRouteVia (HWND hTTWorldNavGPS,const POINT& ptStart, POINT
&ptEnd, const int nVia, const POINT* pptVia)

Behavior:

Creates a route from one Point to another Point via user-specified list of points. The sort order of via points is the order in which the array of viaPoints is filled in.

Expected Arguments:

HWND hTTWorldNavGPS is the window handle of the TeleType Application.

POINT ptStart is the starting GPS co-ordinate of the route to be created.

POINT ptEnd is the ending GPS co-ordinate of the route to be created.

int nVia is the number of via points.

POINT* viaPoints is the list of via Points to be considered while creating route.

If current route settings is not set to use Via, default Via Route Settings will be set automatically.

If currently set to Truck; default will be truck via else default will be Car via.

If nVia is 0, then the function will behave like TTWorldNavGPSRouteTo()

Return Value:

Function is Void

24) void WINAPI TTWorldNavGPSRouteDetails (HWND hTTWorldNavGPS,
PTTGPSROUTEDETAILS &pTTGPSRouteDetails)

Behavior:

Gets a Route detail for a particular route.

Expected Arguments:

HWND hTTWorldNavGPS is the window handle of the TeleType Application.

PTTGPSROUTEDETAILS &pTTGPSRouteDetails is non-allocated pointer to the structure defined by TTTGPSROUTEDETAILS which gets filled with Route's detailed step by step approach with each step's distance, each step's GPS Co-ordinate, each step's Turn type, total route time and total route distance.

Return Value:

Function is Void

TeleType GPS Software Developer's Kit (SDK)

25) void WINAPI TTWorldNavGPSDeleteRoute (HWND hTTWorldNavGPS)

Behavior:

Deletes a current route.

Expected Arguments:

HWND hTTWorldNavGPS is the window handle of the TeleType Application.

Return Value:

Function is Void

26) void WINAPI TTWorldNavGPSRouteToDestName(HWND hTTWorldNavGPS, const POINT &ptStart,
const POINT &ptEnd, const WCHAR*
pszDestName)

Behavior:

Creates a route between ptStart and ptEnd and voice reads the specified destination name.

Expected Arguments:

HWND hTTWorldNavGPS is the window handle of the TeleType Application.

POINT ptStart is the starting GPS co-ordinate of the route to be created.

POINT ptEnd is the ending GPS co-ordinate of the route to be created.

WCHAR* pszDestName is a NULL terminated string to be read by voice.

Return Value:

Function is Void

27) void WINAPI TTWorldNavGPSRouteSettings(HWND hTTWorldNavGPS, TTGPSROUTESETTINGS
&TTRouteSettings)

Behavior:

Queries or sets the Route Settings in Teletype GPS Application.

Expected Arguments:

HWND hTTWorldNavGPS is the window handle of the TeleType Application.

TTGPSROUTESETTINGS TTRouteSettings is the pre-allocated structure containing route settings.

If TTGPSROUTESETTINGS::nMode = eTTMODE_QUERY will get the current route settings or else

TTGPSROUTESETTINGS::nMode = an enum of type of eTTROUTESETTINGSMODE to change the current

TeleType GPS Software Developer's Kit (SDK)

route settings. If queried, output will be in TTRouteSettings or else TTGPSROUTESETTINGS::nMode = eTTMODE_ERROR if any error gets occurred.

Return Value:

Function is Void

28) void WINAPI TTWorldNavGPSSimulator (HWND hTTWorldNavGPS, int &nMode)

Behavior:

Queries or sets GPS Simulator mode in Teletype GPS Application.

Expected Arguments:

HWND hTTWorldNavGPS is the window handle of the TeleType Application.

int nMode – when inputting it could be eTTSIMMODE to set the Simulator Mode or eTTMODE_QUERY to query current Settings. Output will be in nMode having either eTTSIMMODE (query result) or eTTMODE_ERROR if an error encounters.

Return Value:

Function is Void

29) void WINAPI TTWorldNavGPSTruckSettings (HWND hTTWorldNavGPS, TTGPSTRUCKSETTINGS &TTTruckSettings)

Behavior:

Queries or sets Truck Settings in Teletype GPS Application.

Expected Arguments:

HWND hTTWorldNavGPS is the window handle of the TeleType Application.

TTGPSTRUCKSETTINGS TTTruckSettings is the pre-allocated structure containing Truck settings.

If TTGPSTRUCKSETTINGS::nMode = eTTMODE_QUERY will get the current Truck settings or else TTGPSROUTESETTINGS::nMode = an enum of type of eTTTRUCKSETTINGS_HAZMAT to change the current truck settings. If queried, output will be in TTTruckSettings or else TTGPSTRUCKSETTINGS::nMode = eTTMODE_ERROR if any error gets occurred.

TeleType GPS Software Developer's Kit (SDK)

Return Value:

Function is Void

30) void WINAPI TTWorldNavGPSFind (HWND hTTWorldNavGPS, const TTFIND_QUERY &TTFind_Query, PTTFIND_ANSWERS &pTTFind_Answers)

Behavior:

Finds a specified address.

Expected Arguments:

HWND hTTWorldNavGPS is the window handle of the TeleType Application.

TTFIND_QUERY TTFind_Query is the pre-allocated structure containing the search parameters such as Country Name, State Name, City Name, Street Name, House/Building number.

PTTFIND_ANSWERS pTTFind_Answers is the void pre-allocated structure which gets an output as the search results or it remains NULL if no match address found.

Return Value:

Function is Void

31) void WINAPI TTWorldNavGPSFindPOI (HWND hTTWorldNavGPS, const TTFINDPOI_QUERY &TTFindPOI_Query, PTTFIND_ANSWERS &pTTFind_Answers)

Behavior:

Finds a point of Interest.

Expected Arguments:

HWND hTTWorldNavGPS is the window handle of the TeleType Application.

TTFINDPOI_QUERY TTFindPOI_Query is the pre-allocated structure containing the search parameters defined by TTFINDPOI_QUERY structure such as find POI by location or place or find POI by phone number.

PTTFIND_ANSWERS pTTFind_Answers is the void pre-allocated structure which gets an output as the search results or it remains NULL if no results found.

Return Value:

Function is Void

TeleType GPS Software Developer's Kit (SDK)

32) void WINAPI TTWorldNavGPSFindIntersection (HWND hTTWorldNavGPS,
const TTFINDINTERSECTION_QUERY &TTFindInterSection_Query,
PTTFIND_ANSWERS &pTTFind_Answers)

Behavior:

Finds an intersection information between two roads (if any)

Expected Arguments:

HWND hTTWorldNavGPS is the window handle of the TeleType Application.

TTFINDINTERSECTION_QUERY &TTFindInterSection_Query is the pre-allocated structure containing the search parameters defined by TTFINDINTERSECTION_QUERY structure such as Country Name, State Name, City Name, and Street Names whose intersections whose intersection has to be found.

PTTFIND_ANSWERS pTTFind_Answers is the void pre-allocated structure which gets an output as the search results or it remains NULL if no results found.

Return Value:

Function is Void

33) void WINAPI TTWorldNavGPSPosInfo (HWND hTTWorldNavGPS,
const POINT& pt, WCHAR* pszInfo, const DWORD dwCount)

Behavior:

Get info at user-specified position. (Provided the Zoom Level is greater than or equal to 9)

Expected Arguments:

HWND hTTWorldNavGPS is the window handle of the TeleType Application.

POINT pt is GPS Co-ordinate whose information needs to be retrieved.

WCHAR* pszInfo is the void string which gets an output as the search results or it remains NULL if no results found.

DWORD dwCount is the length of pszInfo string.

Building number (0 if none), Street, City/Place, State/Province, Country, Zip/Postal Code.

Output will be empty if zoom level is too high.

Return Value:

Function is Void

34) void WINAPI TTWorldNavGPSUnitMode (HWND hTTWorldNavGPS, int &nMode)

Behavior:

Queries or Sets the Teletype GPS Application System Unit.

Expected Arguments:

HWND hTTWorldNavGPS is the window handle of the TeleType Application.

int nMode – when inputting it could be eTTSYSUNITMODE to set the System Unit or eTTMODE_QUERY to query current Settings. Output will be in nMode having either eTTSYSUNITMODE (query result) or eTTMODE_ERROR if an error encounters.

Return Value:

Function is Void

35) void WINAPI TTWorldNavGPSViewMode (HWND hTTWorldNavGPS, int &nMode)

Behavior:

Queries or Sets the Teletype GPS Application View Mode.

Expected Arguments:

HWND hTTWorldNavGPS is the window handle of the TeleType Application.

int nMode – when inputting it could be eTTVIEWMODE to set the View Mode or eTTMODE_QUERY to query current Settings. Output will be in nMode having either eTTVIEWMODE (query result) or eTTMODE_ERROR if an error encounters.

Return Value:

Function is Void

TeleType GPS Software Developer's Kit (SDK)

36) void WINAPI TTWorldNavGPSPOIMode (HWND hTTWorldNavGPS, int &nMode,const unsigned int uiCategory)

Behavior:

Shows or Hides Points of Interest (POI) in Teletype GPS Application.

Expected Arguments:

HWND hTTWorldNavGPS is the window handle of the TeleType Application.

int nMode – when inputting it could be eTTPOI_VISIBLE to set the POI Mode or eTTMODE_QUERY to query current Settings. Output will be in nMode having either eTTPOI_VISIBLE (query result) or eTTMODE_ERROR if an error encounters.

int uiCategory is the POI category to perform the mode change.

Return Value:

Function is Void

37) void WINAPI TTWorldNavGPSDayNightMode (HWND hTTWorldNavGPS, int &nMode)

Behavior:

Queries or sets the Day/Night Mode in Teletype GPS Application

Expected Arguments:

HWND hTTWorldNavGPS is the window handle of the TeleType Application.

int nMode – when inputting it could be eTTDAYNIGHMODE to set the Day or Night Mode Mode or eTTMODE_QUERY to query current Settings. Output will be in nMode having either eTTDAYNIGHMODE_VISIBLE (query result) or eTTMODE_ERROR if an error encounters.

Return Value:

Function is Void

TeleType GPS Software Developer's Kit (SDK)

38) void WINAPI TTWorldNavGPSGPSInfo (HWND hTTWorldNavGPS, PTTGPSINFO &pTTGpsInfo)

Behavior:

Retrieves the GPS information.

Expected Arguments:

HWND hTTWorldNavGPS is the window handle of the TeleType Application.

PTTGPSINFO pTTGpsInfo is void pre-allocated pointer to structure and will gets filled as an output if the proper GPS information available or NULL if it encounters any error.

Return Value:

Function is Void

39) void WINAPI TTWorldNavGPSTripStatReset (HWND hTTWorldNavGPS)

Behavior:

Resets the trip statics.

Expected Arguments:

HWND hTTWorldNavGPS is the window handle of the TeleType Application.

Return Value:

Function is Void

40) void WINAPI TTWorldNavGPSTripStat (HWND hTTWorldNavGPS,PTTRIPSTAT &pTTripStat)

Behavior:

Retrieves the Trip Statistics information.

Expected Arguments:

HWND hTTWorldNavGPS is the window handle of the TeleType Application.

PTTRIPSTAT pTTripStat is void pre-allocated pointer to the structure where actual Trip Statistics gets filled in or NULL if there is no-data or error gets occurred while retrieving.

Return Value:

Function is Void

TeleType GPS Software Developer's Kit (SDK)

41) void WINAPI TTWorldNavGPSNavPanel (HWND hTTWorldNavGPS, int &nMode, int &nUpper, int &nLower)

Behavior:

Queries or sets the Navigation Panel Settings in Teletype GPS Application.

Expected Arguments:

HWND hTTWorldNavGPS is the window handle of the TeleType Application.

int nMode – when inputting it could be eTTNAVPANEL_TYPE to set the Navigation Panel Settings or eTTMODE_QUERY to query current Settings. Output will be in nMode having either eTTNAVPANEL_TYPE (query result) or eTTMODE_ERROR if an error encounters.

int nUpper – when inputting it could be eTTNAVPANEL_INFO to set the Navigation Panel Settings or eTTMODE_QUERY to query current Settings. Output will be in nMode having either eTTNAVPANEL_INFO (query result) or eTTMODE_ERROR if an error encounters.

int nLower – when inputting it could be eTTNAVPANEL_INFO to set the Navigation Panel Settings or eTTMODE_QUERY to query current Settings. Output will be in nMode having either eTTNAVPANEL_INFO (query result) or eTTMODE_ERROR if an error encounters.

Return Value:

Function is Void

42) void WINAPI TTWorldNavGPSFree(void *lpData)

Behavior:

Frees Allocated memory for certain API Calls.

Expected Arguments:

Void *lpData is the input data to be freed.

Return Value:

Function is Void

TeleType GPS Software Developer's Kit (SDK)

43) void WINAPI TTWorldNavGPSGetFullPath(HWND hTTWorldNavGPS, WCHAR* pszPath, const
DWORD dwCount)

Behavior:

Gets the location of currently running WorldNav Executable.

Expected Arguments:

HWND hTTWorldNavGPS is the window handle of the TeleType Application.

WCHAR* pszPath is the pre-allocated string consisting of dwCount characters of type WCHAR.

DWORD dwCount is the length of the pszPath.

Output path to the exe of WorldNav gets copied into pszPath.

Return Value:

Function is Void

44) void WINAPI TTWorldNavGPSRestrictionInfo(HWND hTTWorldNavGPS, const POINT &pt,
PTRESTRICTIONINFO &pTTRestrictionInfo)

Behavior:

Gets the truck restriction information at a particular location.

Expected Arguments:

HWND hTTWorldNavGPS is the window handle of the TeleType Application.

POINT &pt is a GPS Co-ordinate whose restriction info needs to be retrieved.

PTRESTRICTIONINFO &pTTRestrictionInfo is a non-allocated pointer to the structure
TTRESTRCITIONINFO.

Output will be an API allocated pointer of restriction info (TTRESTRICTIONINFO) which gets
filled with Restriction Information of a particular location and NULL if nothing was found.

Return Value:

Function is Void

TeleType GPS Software Developer's Kit (SDK)

45) void WINAPI TTWorldNavGPSEnableBlockingRouteTo(HWND hTTWorldNavGPS, const BOOL bWait)

Behavior:

Enables/Disables blocking call to wait for route to finish.

Expected Arguments:

HWND hTTWorldNavGPS is the window handle of the TeleType Application.

BOOL bWait is the flag to enable or disable blocking call. If true it enable blocking until route is finished else returns without waiting for route to finish.

This must be called before TTWorldNavGPSRouteTo() or TTWorldNavGPSRouteToDestName().

Whenever WorldNav restarts it will default to non-blocking routing.

Return Value:

Function is Void