

TTNCF3_5

TomTom Navigator SDK Wrapper for the Microsoft .NET Compact Framework

Version 5.0

*for TomTom Navigator SDK V 5.0
(for Windows Powered Pocket PCs)*

Navigator 3 plus 5 Support

***Using the TomTom SDK5 with the
Microsoft® .NET Compact Framework***

Copyright by

PPP – Pitsch & Pohler Professional KEG
Rohrauerweg 10
4030 Linz
Austria

Fax: +43 732 315935 40

Web: <http://www.pp-p.com> (only german)

Product Web: <http://ttncf.pp-p.com>

Customer Web: <http://ttncf.pp-p.net>

Contact: TTNCF@pp-p.com

1.Introduction

1.1. Components of this software

This software consists of:

- native (C++) DLLs **PPPTTHelper.dll** (SDK 3 helper – not included in the TTNCF3 owner version)
- a Microsoft® .NET CF Component in **PPPTTCF3_5.dll**
- **TTNCom.dll** TomTom SDK 3 library

1.2. Prerequisites

You need TTNCF5 (and its prerequisites) which includes:

- native (C++) DLLs **PPPTT5Helper.dll** (SDK 5 helper)

1.3. Installation

After unzipping the files **PPPTTCF3_5.dll**, **PPPTTHelper.dll** (not for TTNCF3 owners), **TTNCom.dll** and **PPPTT5Helper.dll** can be found in your installation directory.

1.4. The “TTNCF3 owner version”

If you already own TTNCF3 and bought the update version **PPPTTHelper.dll** is not included. Take the DLL from your TTNCF3 installation.

1.5. Program with the component

1.5.1. Add the component to your project

On the project view right click references.

Choose "Add reference".

Under .Net choose browse and select the **PPPTTCF3_5.dll**.

On the project view right click the project name and choose "Add existing element".

Add the **PPPTT5Helper.dll**. (From your TTNCFGen Working directory)

1.5.1.1. With the dynamic link version also

On the project view right click the project name and choose "Add existing element".

Add the **TTSDK5Link.dll** from the TomTom SDK.

On the project view right click the project name and choose "Add existing element".

Add the **PPPTTHelper.dll** (either from this software or from TTNCF3)

On the project view right click the project name and choose "Add existing element".

Add the **TTNCom.dll** (either from this software or from TomTom SDK 3)

1.5.2. Tip for faster deployment while developing

To speed up the deployment, run your program once on the device.

This will copy the DLLs to the device.

Next mark the **PPPTT5Helper.dll** (and if working with the dynamic link version

TTSDK5Link.dll), **PPPTTHelper.dll** and **TTNCom.dll**.

Right click and choose "exclude from project"

This will prevent from copying the files to the device on every build.

Don't forget to "reinclude" those DLLs before you build the CAB!!

1.5.3. Use the Component in your program

You can now call the function

- a.) With the full namespace
PPPTTCF3_5.TTN.GetSdkVersion(....)
- b.) Include the namespace in your project
C#: `using PPPTTCF3_5;`
VB: `import PPPTTCF3_5`
Call: `TTN.GedSdkVersion(.....)`

b.) is better if you use the component more often in your source, since you don't have to write always PPPTTCF3_5.TTN.....

And because the calls are compatible to TTNCF5 you have (if you choose to use native TTNCF5) only to replace once PPPTTCF3_5 with PPPTTCF.

1.6. Changes from the TomTom SDK

We changed the **long** values to **int**.

In some cases the coordinates (Longitude, Latitude) have been changed (additional functions) to double. The integer part of the number represents the degrees!

So you can easier write coordinates: **Instead of 4820400 you can use 48.204.**

We also built in double properties for longitude and latitude in the structures.

WARNING: TomTom changed WGS84 format from a millionth of a degree to a 100,000ths (onehundredthousandths). Take car of this when you use the integer versions of the functions.

Numeric structure members allowing enumerated values are replaced by the appropriate enumeration types.

The TomTom SDK uses C++ structures with char or TCHAR arrays.

Since there is no need for this in the CF we changed those arrays to strings.

Attention: this does not eliminate the presences of the TomTom SDK structures.

Therefore in input parameters strings are cut to the length of the SDK arrays which means that a string transferred to a TCHAR [64] array must only be 63 (one for the terminating zero) characters long!

The rest will be cut off!!

1.7. Conventions of this Wrapper

The calls in this wrapper have the same name and functionality as those in the TomTom SDK. One big difference is that the names of the structure members change.

Instead of always using iXXXX we use the following notation for better readability:

- **strXXnn** for strings (**nn** stands for the length in the TomTom SDK)
- **nXXX** for integers
- **lXXX** for long values
- **dXXX** for doubles
- **eXXX** for enumerations

1.8. Conventions of this document

The samples in this documentation are in C# - we chose this language, since the TomTom SDK comes with C++ DLL and therefore we think (hope) that most of you can understand C#.

By the nature of VB.NET versus C# it should be easy to translate the code to VB.NET!

If something is not very clear, please refer to the TomTom SDK documentation.

We kept the names and number the same, so this task should be pretty easy!

The function **5.13** for an example can be found in the TomTom SDK documentation as **2.13**.

1.8.1. Version information

Every entry has information in the Form TTN: x.xx TTNCF y.y.

This tells you about the navigator version needed for the call and the TTNCF version implementing this element.

1.9. Return values

As you can see in the TomTom SDK documentation every call has a structure **TError** as parameter. To provide higher compatibility with older versions of TTNCf we have always two versions of our functions differing only in the presence (or absence) of this structure. There are two special return values:

1.9.1.return -5000

If the component can't determine your Navigator Version it will return this value.

1.9.2.return -2000

If an exception occurs in the C++ dll we return this code.

1.10. Functions from SDK3 with differences

This component is based on TTN SDK 5. We implemented only the functions which are available in both (TTN3 and TTN5) Navigator SDK versions.

Some of the functions changed but are that similar that we made some "tricks" to have it on both versions.

These functions may provide different results when called against Navigator 3. We provide detailed information about the differences in this documentation.

1.10.1. Coordinates

Since SDK 5 uses coordinates in degrees / 100,000 while SDK 3 uses degrees / 1000,000 we built compensation for this.

No matter which TTN version is running use degrees / 100,000 like used to in SDK 5.

1.10.2. A last few words

Like the TomTom SDK this wrapper uses static methods. The only things you will have to instantiate are the structures.

All error codes are directly passed from the TomTom SDK, so you will get exactly the same results, as if you would use C++ to call the SDK functions.

2. Asynchronous calls

With SDK5 TomTom introduces asynchronous calls.

Since those calls are not available in TTN3 we removed them in this wrapper.

3.Supporting functions

To keep the following section equal with the TomTom documentation numbering we add this section for functions not available in the C++ API of the SDK 5.
We added functions needed for the use with SDK 5 as well as some functions to overcome limitations of the CF. These functions are listed below.

3.1. *GetInternalVersions*

3.1.1.Version

TTN: **NotNeeded**

TTNCF: **3.0** (differs by new structure and return)

3.1.2.Declaration(s)

int GetInternalVersions(**ref** **TInternalVersion** tIV)

3.1.2.1. TTN3

No differences.

3.1.3.Parameter

- **TInternalVersion** **tIV** structure to fill with the information

3.1.4.Returns

1 if OK
-2000 if a problem occurs in PPPTTHelper.dll.
-1 if an error in occurred PPPTTCF.dll.
Values of zero in the versions indicate also an error.

3.1.5.Action

Get version information of TTNCF.

3.2. *SetForegroundWindow*

3.2.1.Version

TTN: **NotNeeded**

TTNCF: **1.0**

3.2.2.Declaration(s)

bool SetForegroundWindow(**System.Windows.Forms.Form** theForm)

3.2.2.1. TTN3

No differences.

3.2.3.Parameters

- **System.Windows.Forms.Form** **theForm** the form you want to show

3.2.4.Returns

Success of setting the foreground window.

3.2.5.Action

Since the CF does not offer a method to bring your application to the foreground, we built this function. You must use a "topmost" form that is already open and can be made your applications view.

3.2.5.1. Hint

With this function you can (for an example) switch to your application when some event occurs while running the navigator.
Normally you would use your applications main form or the current active dialog in your application!

3.3. *SendNMEAData*

Removed in this version because the timing in TTN5 makes problems.

3.4. *SendNMEASentenceBuffered*

3.4.1. Version

TTN: 5.0

TTNCF: 5.0

3.4.2. Declaration(s)

int SendNMEASentenceBuffered(**TExternalGPSResult** tGPS, **string** strNMEASentence)

3.4.2.1. TTN3

In TTN3 no timing problems exist. Therefore sentences are neither filtered nor buffered.

3.4.3. Parameters

- **TExternalGPSResult** tGPS the value returned from EnableExternalGPSInput
- **string** strNMEASentence a single NMEA sentence

3.4.4. Returns

0 if successful.

-100 signals that you did not transfer a valid GPRMC or GPGGA sentence.

1000 signals the buffering (nothing was sent to TTN)

Otherwise a negative value.

-1000 nResult in tGPS was zero

-2000 a catchable error occurred

-3000 the string was too long

3.4.5. Action

Collects NMEA sentences. Every time a GPRMC sentence is found the buffered sentences are transferred to TTN as a whole. If a sentence does not end with “\r\n” this sequence will be added to it.

If no GPRMC sentence arrives within 2200 characters – the buffer is transferred to TTN.

Because of timing problems in TTN5 only GPRMC and GPGGA sentences are transferred to TTN5. With TTN3 all sentences (without buffering) are transferred.

3.4.5.1. Hint

We think the easiest way to transfer NMEA to TTN while the “timing problem” exists. This function is (like all the others) thread safe! BUT: it is achieved by locking – so take care if you use different threads calling this function.

With TTN3 you must set the GPS driver selection to “External driver”.

3.5. *DetectedMainVersion*

This integer tells you which Navigator version (3 or 5) was detected.

A value of 0 signals that detection was not possible.

4. Supported API calls

4.1. *GetSdkVersion3 / GetSdkVersion5*

4.1.1. Version

TTN: 1.42

TTNCF: 1.0 (formerly called GetSdkVersionInfoV01)

4.1.2. Declaration(s)

int GetSdkVersion3(**ref** TVersion tnV)

int GetSdkVersion4(**ref** TVersion tnV)

4.1.2.1. TTN3

No differences.

4.1.3. Parameters

- TVersion **tnV** version structure to fill

4.1.4. Returns

0 if successful, non-zero value otherwise.

-2000 a catchable error occurred

4.1.5. Action

Returns the SDK version information (version string and build number).

Since this wrapper uses 2 SDKs you have 2 different functions.

4.1.5.1. Hint

Use this call to determine the presence of the SDK (if using DLL version) and Helper DLL.

The call will catch the Error returning -1 and the Error Message in the structure.

4.2. *GetApplicationVersion*

4.2.1. Version

TTN: 1.42

TTNCF: 1.0 (formerly called GetNavigatorVersionInfoV01)

4.2.2. Declaration(s)

int GetApplicationVersion(**ref** TVersion tnV)

4.2.2.1. TTN3

No differences

4.2.3. Parameters

- TVersion **tnV** version structure to fill

4.2.4. Returns

0 if successful, non-zero value otherwise.

4.2.5. Action

Returns the TomTom Navigator version information (version string and build number)

4.2.5.1. Hint

Use this call to determine the presence of SDK, Helper DLL, and the Navigator.

The call will catch the Error returning -1 and the Error Message in the structure.

4.3. *StartApplication*

4.3.1.Version

TTN: 5.0

TTNCF: 5.0

4.3.2.Declaration(s)

int StartApplication()

4.3.2.1. TTNCF3

Since this function is not available in SDK 3 – we use “BringNavigatorToForeground with TTN3.

4.3.3.Parameter

None

4.3.4.Returns

0 if successful, non-zero value otherwise.

4.3.5.Action

Starts TomTom Navigator. If TTN is already running this function returns immediately with 0.

4.3.5.1. Hint

Since this call would bring TomTom Navigator to foreground in most cases be prepared that your application is no longer visible after that call.

We built [SetForegroundWindow](#) to bring your application back to foreground.

Since the call returns before the Navigator has finished starting – use a

System.Threading.Thread.Sleep(3000) to ensure that the start process has finished before you bring your app to foreground.

If you miss the sleep – your app comes to foreground – but finishing the start of Navigator results in a “switch back” to the Navigator.

4.4. *IsApplicationRunning*

4.4.1.Version

TTN: 5.0

TTNCF: 5.0

4.4.2.Declaration(s)

int IsApplicationRunning()

4.4.2.1. TTN3

Although not available in SDK 3 we provide this functionality also with TTN3.

4.4.3.Parameter

None.

4.4.4.Returns

0 if not running, 1 if running.

A negative value signals an error.

4.4.5.Action

Checks if TomTom Navigator is running

4.5. *BringNavigatorToForeground*

4.5.1.Version

TTN: 1.42

TTNCF: 1.0

4.5.2.Declaration(s)

int BringNavigatorToForeground ()

4.5.2.1. TTNCF3

No differences.

4.5.3.Parameter

None

4.5.4.Returns

0 if successful, non-zero value otherwise.

4.5.5.Action

This call brings the navigator into the foreground. If it is not running it starts the navigator.

4.5.5.1. Hint

There is no "BringNavigatorToBackground".

Since a call like this would bring "any window" to foreground just use a call that brings your application to foreground!!

We built [SetForegroundWindow](#) for this.

4.6. *TomTomApplicationToForeground*

4.6.1.Version

TTN: 5.0

TTNCF: 5.0

4.6.2.Declaration(s)

int TomTomApplicationToForeground()

4.6.2.1. TTN3

Makes a call to BringNavigatorToForegrund. So with TTN3 the application will be started no matter if you use this function or BringNavigatorToForeground.

4.6.3.Parameter

- None

4.6.4.Returns

0 if successful. A negative value signals an error.

-5 INavigator is not running

4.6.5.Action

Brings the TomTom Navigator to foreground if it is running.

4.6.5.1. Hint

This call is faster than [BringNavigatorToForeground](#) but it does not start the Navigator 5 if it is not running. **TTN3 will start if not running on this call.**

4.7. StopApplication

4.7.1.Version

TTN: 5.0

TTNCF: 5.0

4.7.2.Declaration(s)

int StopApplication()

4.7.2.1. TTN3

Although not available in SDK 3 we built the functionality to make this work.

4.7.3.Parameter

None

4.7.4.Returns

0 if successful, non-zero value otherwise.

4.7.5.Action

Stops the TomTom Navigator Application.

4.8. SwitchToNavigatorView

4.8.1.Version

TTN: 1.42

TTNCF: 1.0

4.8.2.Declaration(s)

int SwitchToNavigatorView()

4.8.2.1. TTN3

To provide the same functionality a call to "TryCloseOpenDialogs" is also called.

4.8.3.Parameter

None

4.8.4.Returns

0 if successful, non-zero value otherwise.

4.8.5.Action

Switches to the navigation view.

4.9. FlashMessage

Not available since there is no such function in SDK 3.

4.10. GetDataSetInfo

4.10.1. Version

TTN: 2.0

TTNCF: 1.0 (formerly GetDataSetInfoV01 TDataSetInfo extended)

4.10.2. Declaration(s)

int GetDataSetInfo(**ref** TDataSetInfo DSI)

4.10.2.1. TTN3

The field strDataSetPath128 is not filled since SDK 3 does not provide this information.

4.10.3. Parameters

- **TDataSetInfo** DSI data set information

4.10.4. Returns

0 if successful, non-zero value otherwise.

4.10.5. Action

Get information about the currently selected data set (Map).

4.10.5.1. Hint

Since this call also returns the path to the dataset it is useful to place certain files there.

Take care – whit TTN3 this path is not filled!!!

4.11. GetRouteInfo

4.11.1. Version

TTN: 2.0

TTNCF: 1.0 (formerly GetRouteInfoV01)

4.11.2. Declaration(s)

int GetRouteInfo(**ref** TRouteInfo RI)

4.11.2.1. TTN3

No differences.

4.11.3. Parameters

- **TRouteInfo** RI route information

4.11.4. Returns

0 if successful, non-zero value otherwise.

4.11.5. Action

Get information about the current route.

4.12. GetRouteCoordinates

Since the implementation of this function is totally different between SDK 3 and SDK 5 we do not provide it at the moment.

4.13. *GetLocationInfo*

4.13.1. Version

TTN: 2.25

TTNCF: 3.0 (formerly *GetLocationInfoV01*)

4.13.2. Declaration(s)

int *GetLocationInfo*(**int** nLongi, **int** nLati, **ref** *TLocationInfo* L)

int *GetLocationInfo*(**double** dLongi, **double** dLati, **ref** *TLocationInfo* LI)

4.13.2.1. TTN3

The field Longitude and Latitude in the returned structure are filled with the given values. This happens because SDK 3 does not provide those values on return.

4.13.3. Parameters

- **int** **nLongitude** longitude specified in WGS84 format (in 100,000ths of a degree)
- **int** **nLatitude** latitude specified in WGS84 format (in 100,000ths of a degree)
- *TLocationInfo* LI structure that will be filled with the location information

The double version makes it easier to write constant coordinates.

See: [Changes from the TomTom SDK](#)

4.13.4. Returns

0 if successful, non-zero value otherwise.

4.13.5. Action

Get information about the location closest to the given point. In other words, perform reverse geocoding, or 'geodecoding'.

4.14. *GetLocationInfoEx*

Since SDK3 does not provide this function it is not supported.

4.15. *GetUniqueDeviceId*

Since SDK3 does not provide this function it is not supported.

4.16. *NavigateToCoordinates*

4.16.1. Version

TTN: 1.42

TTNCF: 1.0 (formerly *NavigateToCoordinate* – “s” was missing)

4.16.2. Declaration(s)

int *NavigateToCoordinates*(**int** nLongitude, **int** nLatitude, **string** strName)

int *NavigateToCoordinates*(**double** dLongitude, **double** dLatitude, **string** strName)

4.16.2.1. TTN3

No differences.

4.16.3. Parameters

- **int** **nLongitude** longitude spec. in WGS84 format (in 100,000ths of a degree)
- **int** **nLatitude** latitude specified in WGS84 format (in 100,000ths of a degree)
- **string** **strName** name of the location (shown in the route description)

The double version makes it easier to write constant coordinates.

See: [Changes from the TomTom SDK](#)

4.16.4. Returns

0 if successful, non-zero value otherwise.

4.16.5. Action

Navigate to a location

4.16.5.1. Hint

If you use the long version be sure to have 5 digits after the degrees!!!

4.16.6. See also

[NavigateToFavorite](#), [ShowCoordinatesOnMap](#)

4.17. *NavigateToFavorite*

4.17.1. Version

TTN: 1.42

TTNCF: 1.0

4.17.2. Declaration(s)

[int](#) NavigateToFavorite([int](#) nIndex)

4.17.2.1. TTN3

No differences-

4.17.3. Parameter

- [int](#) **nIndex** index of favorite (base zero)

4.17.4. Returns

0 if successful, non-zero value otherwise.

4.17.5. Action

Navigates to a specific favorite with the index nIndex (base zero).

4.17.6. See also

[SetFavorite](#), [GetFavorite](#), [ClearFavorite](#)

4.18. *ShowCoordinatesOnMap*

4.18.1. Version

TTN: 1.42

TTNCF: 1.0 (formerly ShowCoordinateOnMap – “s” missing)

4.18.2. Declaration(s)

[int](#) ShowCoordinatesOnMap([int](#) nLongitude, [int](#) nLatitude)

[int](#) ShowCoordinatesOnMap([double](#) dLongitude, [double](#) dLatitude)

4.18.2.1. TTN3

No differences.

4.18.3. Parameters

- [int](#) **nLongitude** longitude spec. in WGS84 format (in 100,000ths of a degree)
- [int](#) **nLatitude** latitude specified in WGS84 format (in 100,000ths of a degree)

The double version makes it easier to write constant coordinates.

See: [Changes from the TomTom SDK](#)

4.18.4. Returns

0 if successful, non-zero value otherwise.

4.18.5. Action

Centers the map on the given coordinate (zoomed in).

4.18.6. See also

[NavigateToCoordinate](#), [ShowRectangleOnMap](#)

4.19. ShowRectangleOnMap

4.19.1. Version

TTN: 3.0

TTNCF: 3.0

4.19.2. Declaration(s)

```
int ShowRectangleOnMap(int nLongiW, int nLatiS, int nLongiE, int nLatiN)
int ShowRectangleOnMap(double dLongiW, double dLatiS, double dLongiE, double dLatiN)
```

4.19.2.1. TTN3

No differences.

4.19.3. Parameters

- o **int** **nLongiW** western longitude spec.in WGS84 format (in 100,000ths of a degree)
- o **int** **nLatiS** southern latitude spec.in WGS84 format (in 100,000ths of a degree)
- o **int** **nLongiE** eastern longitude spec.in WGS84 format (in 100,000ths of a degree)
- o **int** **nLatiN** northern latitude spec.in WGS84 format (in 100,000ths of a degree)

The double version makes it easier to write constant coordinates.

See: [Changes from the TomTom SDK](#)

4.19.4. Returns

0 if successful, non-zero value otherwise.

4.19.5. Action

Show the given rectangle of the map.

4.19.6. See also

[ShowCoordinatesOnMap](#)

4.20. ChangeSetting

4.20.1. Version

TTN: 5.0

TTNCF: 5.0

4.20.2. Declaration(s)

```
int ChangeSetting(ESettingAndValue eSV)
```

4.20.2.1. TTN3

Although not available in SDK 3 we built the functionality to make this work.

4.20.3. Parameters

- o [ESettingAndValue](#) **eSV** the structur to fill

4.20.4. Returns

0 if successful, non-zero value otherwise.

4.20.5. Action

Changes one of the supported settings in SDK 5, like sound on, show map, etc.

4.21. SwitchMap

4.21.1. Version

TTN: 3.0

TTNCF: 3.0

4.21.2. Declaration(s)

```
int SwitchMap(string strPath)
```

4.21.2.1. TTN3

No differences.

4.21.3. Parameters

- o **string** **strPath** Full path and Name of the PNA file

4.21.4. Returns

0 if successful, non-zero value otherwise.

4.21.5. Action

Switch to the map specified by the full name and path of the .PNA file.

4.22. SetNavigatorProperty

4.22.1. Version

TTN: 3.0 TTNCF: 3.0 (indicators, Menu and HWButton removed)

4.22.2. Declaration(s)

```
int SetNavigatorProperty(EPProperty eProp, int nValue)
int SetNavigatorProperty(EPProperty eProp, EPVColor ePVC)
```

4.22.2.1. TTN3

We made value corrections and removed 4 Color values not available with SDK 3.

4.22.3. Parameters

- [EPProperty](#) eProp property to set
- [int](#) nValue pure numeric value of the property
- [EPVColor](#) ePVC enumeration for color values

4.22.4. Returns

0 if successful, non-zero value otherwise.

4.22.5. Action

Sets the value for one of the properties (user settings), like current color schemes, voice volume, etc.

4.22.5.1. Hint

We built an extra functions with [EPVColor](#) as parameter to make the use more save.

Be sure to use this function when setting colors since the values between TTN3 and TTN5 differ. We compensate this in the Color parameter handling.

4.23. ShowAddressOnMap

Since SDK3 does not provide this function it is not supported.

But you can easily “rebuild it” by a call to [Geocode](#) followed by a call to [ShowCoordinatesOnMap](#).

4.24. NavigateToAddress

Since SDK3 does not provide this function it is not supported.

But you can easily “rebuild it” by a call to [Geocode](#) followed by a call to [NavigateToCoordinates](#).

4.25. GetCurrentPosition

4.25.1. Version

TTN: 1.42 TTNCF: 1.0 (formerly GetCurrentPositionV01 – EGPSStatus moved to struct)

4.25.2. Declaration(s)

```
int GetCurrentPosition(ref TCurrentPositionInfo tCP)
```

4.25.2.1. TTN3

No differences.

4.25.3. Parameters

- [TCurrentPositionInfo](#) tCP structure to fill with the values of the current position

4.25.4. Returns

0 if successful, non-zero value otherwise.

4.25.5. Action

Fills the structure passed in with current position information from GPS.

4.26. Geocode

4.26.1. Version

TTN: 1.51

TTNCF: 1.0 (formerly GeocodeV01)

4.26.2. Declaration(s)

int Geocode(**string** strCity, **string** strStreet, **string** strHouseNr, **ref** TGeocodeResult tGR)

4.26.2.1. TTN3

No differences.

4.26.3. Parameters

- | | | |
|-------------------------|-------------------|--------------------------------------|
| ○ string | strCity | city name |
| ○ string | strStreet | street name |
| ○ string | strHouseNr | house number |
| ○ <u>TGeocodeResult</u> | tGR | filled with the results of geocoding |

4.26.4. Returns

0 if successful, non-zero value otherwise.

4.26.5. Action

Returns geocode (latitude and longitude) for the given address, specified by the city name, street name and house number arguments, and returns the co-ordinates of the found location and its complete city and street names in the geocode information argument. If the street name argument is empty, the function will try to find the center of the city specified by the city name argument. If no location is found with city and street names exactly matching the arguments, then a city and a street will be chosen with names as similar as possible to the given ones. This will be indicated in the status field of the geocode information argument.

4.27. GeocodeExtended

4.27.1. Version

TTN: 2.22

TTNCF: 3.0 (formerly GeocodeExV01)

4.27.2. Declaration(s)

int GeocodeExtended(**string** strCity, **string** strStreet, **string** strHouseNr, **string** strPostcode, **ref** TGeocodeResult tGR)

4.27.2.1. TTN3

No differences.

4.27.3. Parameters

- | | | |
|-------------------------|-------------------|--------------------------------------|
| ○ string | strCity | city name |
| ○ string | strStreet | street name |
| ○ string | strHouseNr | house number |
| ○ <u>TGeocodeResult</u> | tGR | filled with the results of geocoding |

4.27.4. Returns

0 if successful, non-zero value otherwise.

4.27.5. Action

Returns geocode (latitude and longitude) for the given address, specified by the city name, street name, house number and postal code arguments, and return the co-ordinates of the found location and its complete city and street names in the geocode information argument. If the street name argument is empty, the function will try to find the center of the city specified by the city name argument. . If no location is found with city and street names exactly matching the arguments, then a city and a street will be chosen with names as similar as possible to the given ones. This will be indicated in the status field of the geocode information argument.

4.27.5.1. Hint

This extended version accepts also the postcode in geocoding. This brings better results as Geocode (especially with map which have a postcode file included – for an example UK).

With TTN3 there are no such enhancements – it takes the postcode only as hint!

4.28. ClearFavorite

4.28.1. Version

TTN: 1.42

TTNCF: 1.0

4.28.2. Declaration(s)

int ClearFavorite(**int** nIDX)

4.28.2.1. TTN3

No differences.

4.28.3. Parameters

- **int** nIDX index of favorite (base zero)

4.28.4. Returns

0 if successful, non-zero value otherwise.

4.28.5. Action

Clears a favorite.

4.28.6. See also

[SetFavorite](#), [GetFavorite](#), [NavigateToFavorite](#)

4.29. SetFavorite

4.29.1. Version

TTN: 1.42

TTNCF: 1.0 (formerly SetFavoriteV01)

4.29.2. Declaration(s)

int SetFavorite(**int** nIDX, **string** strName, **string** strDescription, **int** nLongitude, **int** nLatitude)

int SetFavorite(**int** nIDX, **string** strName, **string** strDescription, **double** dLongitude, **double** dLatitude)

int SetFavorite(**int** nIDX, **TFavoriteInfo** fi)

4.29.2.1. TTN3

No differences.

4.29.3. Parameters

- **int** nIndex index of favorite (base zero)
- **string** strName Name the favorite should get
- **string** strDescription Description of the favorite
- **int** nLongitude longitude specified in WGS84 format (in 100,000ths of a degree)
- **int** nLatitude latitude specified in WGS84 format (in 100,000ths of a degree)
- **TFavoriteInfo** fi structure containing the information

The double version makes it easier to write constant coordinates.

See: [Changes from the TomTom SDK](#)

4.29.4. Returns

0 if successful, non-zero value otherwise.

4.29.5. Action

Assigns the data to the element from the favourites array with the given index (zero-based).

4.29.5.1. Hint

For your convenience we added a version where you can pass the values in a [TFavoriteInfo](#) structure. This makes it easy to use the result of [GetFavorite](#).

4.29.6. See also

[GetFavorite](#), [ClearFavorite](#), [NavigateToFavorite](#)

4.30. *GetFavorite*

4.30.1. Version

TTN: 1.42

TTNCF: 1.0 (formerly GetFavoriteV01)

4.30.2. Declaration(s)

int GetFavorite(**int** nIDX, **ref** **TFavoriteInfo** tFI)

4.30.2.1. TTN3

No differences.

4.30.3. Parameters

- **int** nIDX index of favorite (base zero)
- **TFavoriteInfo** tFI struture to retrieve the information

4.30.4. Returns

0 if successful, non-zero value otherwise.

4.30.5. Action

Get favorite information.

4.30.6. See also

[SetFavorite](#), [ClearFavorite](#), [NavigateToFavorite](#)

4.31. *MakePoiVisible*

4.31.1. Version

TTN: 1.42

TTNCF: 1.0

4.31.2. Declaration(s)

int MakePoiVisible(**EPoiTypeCode** eTypeCode, **int** nVisibility)

4.31.2.1. TTN3

We changed the POI Type codes to fit SDK 3.

4.31.3. Parameters

- **EPoiTypeCode** ePoiID POI unique type identifier
- **int** nVisibility show or hide this POI type on the map

4.31.4. Returns

0 if successful, non-zero value otherwise.

4.31.5. Action

Enable or disable displaying a specific POI type.

4.31.6. See also

[AddPoi](#), [DeleteAllPoi](#), [DeleteClosestPoi](#), [MakeUserPoiVisible](#), [MoveClosestPoi](#)

4.32. AddPoi

4.32.1. Version

TTN: 1.51

TTNCF: 1.0

4.32.2. Declaration(s)

```
int AddPoi(string strFName, int nLongi, int nLati, string strName, string strID)
int AddPoi(string strFName, int nLongi, int nLati, string strName)
int AddPoi(string strFName, double dLongi, double dLati, string strName, string strID)
int AddPoi(string strFName, double dLongi, double dLati, string strName)
```

4.32.2.1. TTN3

No differences.

4.32.3. Parameters:

- **string strFName** name of the POI category
- **int nLongi** longitude specified in WGS84 format (in 100,000ths of a degree)
- **int nLati** latitude specified in WGS84 format (in 100,000ths of a degree)
- **string strName** name of the POI
- **string strID** **optional** external identifier of the POI

The double version makes it easier to write constant coordinates.

See: [Changes from the TomTom SDK](#)

4.32.4. Returns

0 if successful, non-zero value otherwise.

4.32.5. Action

Add a point of interest to the .OV2 file specified by the POI category argument. The point of interest will have the name and co-ordinates given by the corresponding arguments. An external identifier string can also be attached to the point of interest.

When the .OV2 file does not yet exist, it is created.

4.32.6. See also

[DeleteAllPoi](#), [DeleteClosestPoi](#), [MakePoiVisible](#), [MakeUserPoiVisible](#), [MoveClosestPoi](#)

4.33. DeleteClosestPoi

4.33.1. Version

TTN: 1.51

TTNCF: 1.0

4.33.2. Declaration(s)

`int DeleteClosestPoi(string strFName, int nLongi, int nLati)`

`int DeleteClosestPoi(string strFName, double dLongi, double dLati)`

4.33.2.1. TTN3

No differences.

4.33.3. Parameters

- `string` `strFName` name of the POI category
- `int` `nLongi` longitude spec. in WGS84 format (in 100,000ths of a degree)
- `int` `nLati` latitude specified in WGS84 format (in 100,000ths of a degree)

The double version makes it easier to write constant coordinates.

See: [Changes from the TomTom SDK](#)

4.33.4. Returns

0 if successful, non-zero value otherwise.

4.33.5. Action

Delete a point of interest from the .OV2 file specified by the POI category argument. The point of interest to be deleted is the closest one to the given co-ordinates among the points of interest that are within a distance of 100 meters from there.

4.33.6. See also

[AddPoi](#), [DeleteAllPoi](#), [MakePoiVisible](#), [MakeUserPoiVisible](#), [MoveClosestPoi](#)

4.34. DeleteAllPoi

4.34.1. Version

TTN: 1.51

TTNCF: 1.0

4.34.2. Declaration(s)

`int DeleteAllPoi(string strFName)`

4.34.2.1. TTN3

No differences.

4.34.3. Parameter

- `string` `strFName` name of the POI category

4.34.4. Returns

0 if successful, non-zero value otherwise.

4.34.5. Action

Delete all points of interest from the .OV2 file specified by the POI category argument. The POI category itself is also removed after that.

4.34.6. See also

[AddPoi](#), [DeleteClosestPoi](#), [MakePoiVisible](#), [MakeUserPoiVisible](#), [MoveClosestPoi](#)

4.35. *MakeUserPoiVisible*

4.35.1. Version

TTN: 2.0

TTNCF: 1.0

4.35.2. Declaration(s)

int MakeUserPoiVisible(**string** strFName, **int** nVisibility)

4.35.2.1. TTN3

No differences.

4.35.3. Parameters:

- **string** **strFName** name of the POI category
- **int** **nVisibility** whether to show or hide this POI type on the map

4.35.4. Returns

0 if successful, non-zero value otherwise.

4.35.5. Action

Enable or disable displaying a specific user-created POI type.

4.35.6. See also

[AddPoi](#), [DeleteAllPoi](#), [DeleteClosestPoi](#), [MakePoiVisible](#), [MoveClosestPoi](#)

4.36. MoveClosestPoi

4.36.1. Version

TTN: 1.51

TTNCF: 1.0

4.36.2. Declaration(s)

```
int MoveClosestPoi(string strFName, int nLongi, int nLati, int nNewLongi, int nNewLati)
int MoveClosestPoi(string strFName, double dLongi, double dLati, double dNewLongi,
double dNewLati)
```

4.36.2.1. TTN3

No differences.

4.36.3. Parameters

- **string strFName** name of the POI category
- **int nLongi** old longitude specified in WGS84 format (in 100,000ths of a degree)
- **int nLati** old latitude specified in WGS84 format (in 100,000ths of a degree)
- **int nNewLongi** new longitude specified in WGS84 format (in 100,000ths of a degree)
- **int nNewLati** new latitude specified in WGS84 format (in 100,000ths of a degree)

The double version makes it easier to write constant coordinates.

See: [Changes from the TomTom SDK](#)

4.36.4. Returns

0 if successful, non-zero value otherwise.

4.36.5. Action

Move a point of interest from the .OV2 file specified by the POI category argument to a new position. The point of interest to be moved is the closest one to the given old co-ordinates among the points of interest that are within a distance of 100 meters from there.

4.36.6. See also

[AddPoi](#), [DeleteAllPoi](#), [DeleteClosestPoi](#), [MakePoiVisible](#), [MakeUserPoiVisible](#)

4.37. EnableExternalGPSInput

4.37.1. Version

TTN: 5.0

TTNCF: 5.0

4.37.2. Declaration(s)

int EnableExternalGPSInput(**ref** TExternalGPSResult tGPS)

4.37.2.1. TTN3

Take care for the – 10 return. It indicates that TTN3 is not configured to use external driver.

4.37.3. Parameters

- o TExternalGPSResult tGPS holds the “access handle” later used in SendNMEAData

4.37.4. Returns

0 if successful, non-zero value otherwise.

-10 The Function failed with **TTN3**.

4.37.5. Action

Enables external GPS input coming from the user instead of GPS input supplied via the Navigator itself.

4.37.5.1. Hint

After a call to this function the navigator will close the used port.

Add a short delay before you open the port with your application.

The delay ensures ensures that the resource is freed by the navigator.

With TTN3 you must set the GPS driver selection to “External driver”.

4.38. DisableExternalGPSInput

4.38.1. Version

TTN: 5.0

TTNCF: 5.0

4.38.2. Declaration(s)

int DisableExternalGPSInput()

4.38.2.1. TTN3

No differences.

4.38.3. Parameter

None

4.38.4. Returns

0 if successful, non-zero value otherwise.

4.38.5. Action

Disables external GPS input from the user and reverts to GPS input from the Navigator.

4.38.5.1. Hint

With TTN5 close the port before calling this function. So the Navigator can reopen it for internal use.

Adding a short delay after closing the port ensures that the resource is freed.

With TTN3 you must change the GPS driver selection to enable direct GPS input.

5. Structures and enumerations from the API

All structures differ from the TomTom SDK by the use of str for string, n for Integers, l for longs and finally a length name suffix at some string values! **The sub-numbering is the same as in the SDK 5 documentation.**

5.1. TVersion structure

```
public struct TVersion {  
    public int nBuildNumber;  
    public string strVersion16;  
}
```

5.2. TError structure

```
public struct TError {  
    public int nError;  
}
```

5.3. TDataSetInfo structure

```
public struct TDataSetInfo {  
    public string strDataSetName128;    // name of the dataset  
    public string strDataSetPath128;    // path to the dataset  
    public EDatasetFlags eFlags;  
    public int nBuildNumber;  
}
```

5.3.1. The EDatasetFlags enumeration

Possible values in eFlags of the [TDataSetInfo](#).

```
public enum EDatasetFlags {  
    KHouseNrPrefixed=1, // house numbers are written before street name  
    KLeftSideDriving=2  // indicates left-sided driving  
}
```

5.4. TLocationInfo structure

```
public struct TLocationInfo {  
    public long lLongitude;  
    public long lLatitude;  
    public double dLongitude;    // double accessor for lLongitude  
    public double dLatitude;    // double accessor for lLatitude  
    public ELocDataType eType;  
    public string strStreetName128;  
    public int lnHouseNr1;  
    public int lnHouseNr2;  
}
```

5.5. TLocationInfoEx structure

```
public struct TLocationInfoEx {  
    public long lLongitude;  
    public long lLatitude;  
    public double dLongitude;    // double accessor for lLongitude  
    public double dLatitude;    // double accessor for lLatitude  
    public ELocDataType eType;  
    public string strStreetName128;  
    public int lnHouseNr1;  
    public int lnHouseNr2;  
    public string strCityName128;  
}
```

5.6. *ELocDataType* enumeration

Possible values in eType in [TLocationInfo](#) and in [TLocationInfoEx](#)

```
public enum ELocDataType {
    KMapPosition = 0,      // the location is a position on the map
    KNode = 1              // the location is a singled node (e.g. city center or a pos. selected on map)
    KCrossing = 2           // the location is a crossing
    KStreetPos = 3          // the location is somewhere on a street
    KStreet = 4,            // the location is a street
    KBackToRoute = 5        // found in header file
}
```

5.7. *TRouteInfo* structure

```
public struct TRouteInfo {
    public ERouteInfoStatus eStatus;
    public TLocation Departure; // coordinates for departure and destination
    public TLocation Destination;
                                // rectangle enclosing the route
    public long lEnclosingLongitudeW;
    public double dEnclosingLongitudeW; // double accessor
    public long lEnclosingLatitudeS;
    public double dEnclosingLatitudeS; // double accessor
    public long lEnclosingLongitudeE;
    public double dEnclosingLongitudeE; // double accessor
    public long lEnclosingLatitudeN;
    public double dEnclosingLatitudeN; // double accessor
    public long lLength; // length of route in meters
    public long lDuration; // duration of route in seconds
    public long lDistanceToDestination; // distance current pos to dest in meters
    public long lTimeToDestination; // time from current pos to dest in seconds
}
```

5.7.1. The *ERouteInfoStatus* enumeration

Possible values in eStatus of the [TRouteInfo](#).

```
public enum ERouteInfoStatus { //combined values are possible
    KDepartureSet = 1,          //the departure point has been set
    KDestinationSet = 2,        //the destination point has been set
    KCalculatingRoute = 4,      //in the process of calculating a route
    KRoutePlanned = 8,          //there is a valid route planned
    KLeftToDestinationAvailable = 16 //time + distance to destination available
}
```

5.8. *TLocation* structure

Used in [TRouteInfo](#)

```
public struct TLocation {
    public long lLongitude;
    public long lLatitude;
    public double dLongitude; // double accessor for lLongitude
    public double dLatitude;  // double accessor for lLatitude
}
```

5.9. *TDeviceID* structure

```
public struct TDeviceID {
    public string strDeviceId32;
    public int nDeviceIDLength; // length of the string in strDeviceID32
}
```

5.10. TCurrentPositionInfo structure

This structure differs from the TomTom SDK structure by the double accessors.

```
public struct TCurrentPositionInfo {
    public EGpsStatus eGpsStat;
    public long lLongitude;
    public long lLatitude;
    public double dLongitude;    // double accessor for lLongitude
    public double dLatitude;    // double accessor for lLatitude
    public int nDirection;    // Direction: zero is north, clockwise in degrees
    public int nSpeedKMh;
}
```

5.11. TGeocodeResult structure

```
public struct TGeocodeResult {
    public EGeoCodeStatus eStatus;
    public long lLongitude;
    public long lLatitude;
    public double dLongitude    // double accessor for lLongitude
    public double dLatitude;    // double accessor for lLatitude
    public string strCityName128;
    public string strStreetName128;
}
```

5.11.1. EGeoCodeStatus enumeration

Possible values in eStatus in [TGeocodeResult](#)

```
public enum EGeoCodeStatus {
    KGeocodeHouseNrIgnored=1,    // the house number was ignored, either because there
                                // was no house number data available or because the given
                                // house number was not found
    KGeocodeStreetAmbiguous = 2, // g. street name ambiguous, tried to make smart selection
    KGeocodeCityAmbiguous = 4,  // the given city name was ambiguous, random
                                // (possibly wrong) selection made
    KGeocodeFailure = 8         // no location found at all, the rest of the fields are invalid
}
```

5.12. TFavoriteInfo structure

This structure differs from the TomTom SDK structure by the double access and the limited (programmatically cut of) strings!

```
public struct TFavoriteInfo {
    public string strName64;    // limited to 63 characters text
    public string strDescription128; // limited to 128 characters text
    public long lLongitude;
    public long lLatitude;
    public double dLongitude;    // double accessor for lLongitude
    public double dLatitude;    // double accessor for lLatitude
}
```

5.13. TRouteCoordinates structure

```
public struct TRouteCoordinates {
    public int nNrCoordinates;    // number of coordinates in the route
    public TLocation [] aCoordinates; // array with coordinates
}
```

5.14. TExternalGPSResult structure

```
public struct TExternalGPSResult {  
    public int nResult;    // "handle" for GPS calls  
}
```

5.15. ESettingAndValue enumeration

```
public enum ESettingAndValue {  
    KCommandSoundOn= 0,    // turn spoken voice instructions on  
    KCommandSoundOff= 1,   // turn spoken voice instructions off  
    KCommandNightColors=2,  //- show map in night colors  
    KCommandDayColors=3,    // show map in day colors  
    KCommandShowMap=4,     // show map and instructions while driving  
    KCommandHideMap=5      // show "instructions only" view while driving  
}
```

5.16. EProperty enumeration

```
public enum EProperty {  
    KPropertyColorStyle= 0,    // colour style used in daylight mode -- see EPVColor  
    KPropertyNightColorStyle= 1, // the colour style used in night mode -- see EPVColor  
    KPropertyVoiceVolume=2,    //- the volume of the voice prompts  
    KPropertySafetySpeed=3,    // show map in day colors  
    KPropertyMaxPoiOnMap=4     // found in header file  
}
```

5.16.1. The EPVColor enumeration

```
public enum EPVColor {    // map color styles  
    KColorStyleBelgica=1027,  
    KColorStyleBrittanica=1028,  
    KColorStyleGermanica=1029,  
    // KColorStyleAustralia=1030,    REMOVED – not available in SDK 3  
    // KColorStyleDeuteranopia=1031,  REMOVED – not available in SDK 3  
    KColorStyleGreys=1032,  
    KColorStyleAmerica=1033,  
    KColorStyleAstra=1034,  
    KColorStyleAfrica=1035,  
    KColorStyleAntarctica=1036,  
    // KColorStyleTraffic=1037,      REMOVED – not available in SDK 3  
    // KColorStyleCustom=1100        REMOVED – not available in SDK 3  
}
```

5.17. EGpsStatus enumeration

Possible values for eGpsStat in [TCurrentPositionInfo](#)

```
public enum EGpsStatus {  
    KGpsStateGeneralError = -1,    // error == position invalid  
    KGpsStateGeneral = 0           // position valid (documentation)  
    KGpsStateReceiving = 1         // position valid  
}
```

5.18. EPoiTypeCode enumeration

```
public enum EPoiTypeCode {
    KPoiTypeGovernmentOffice = 0 ,
    KPoiTypeOpenParking = 2 ,
    KPoiTypePetrolStation = 4 ,
    KPoiTypeRestArea = 6 ,
    KPoiTypeCarDealer = 8 ,
    KPoiTypeCasino = 9 ,
    KPoiTypeChurch = 10 ,
    KPoiTypeCinema = 11 ,
    KPoiTypeCityCenter = 12 ,
    KPoiTypeCompany = 13 ,
    KPoiTypeConcertHall = 14 ,
    KPoiTypeCourthouse = 15 ,
    KPoiTypeCulturalCenter = 16 ,
    KPoiTypeExhibitionCenter = 17 ,
    KPoiTypeFerryTerminal = 18 ,
    KPoiTypeFrontierCrossing = 19 ,
    KPoiTypeGolfCourse = 20 ,
    KPoiTypeHospitalPolyclinic = 21 ,
    KPoiTypeHotelMotel = 22 ,
    KPoiTypeTouristAttraction = 23 ,
    KPoiTypeMountainPass = 24 ,
    KPoiTypeMuseum = 25 ,
    KPoiTypeOpera = 26 ,
    KPoiTypePlaceofWorship = 27 ,
    KPoiTypePostOffice = 28 ,
    KPoiTypeRentCarFacility = 29 ,
    KPoiTypeRentCarParking = 30 ,
    KPoiTypeRestaurant = 31 ,
    KPoiTypeShop = 32 ,
    KPoiTypeShoppingCenter = 33 ,
    KPoiTypeStadium = 34 ,
    KPoiTypeTheatre = 35 ,
    KPoiTypeTouristInformationOffice = 36 ,
    KPoiTypeZoo = 37 ,
    KPoiTypeSportsCentre = 38 ,
    KPoiTypePoliceStation = 39 ,
    KPoiTypeEmbassy = 40 ,
    KPoiTypeCollegeUniversity = 41 ,
    KPoiTypeCashDispenser = 42 ,
    KPoiTypeBeach = 43 ,
    KPoiTypeIceSkatingRing = 44 ,
    KPoiTypeTennisCourt = 45 ,
    KPoiTypeWaterSport = 46 ,
    KPoiTypeDoctor = 47 ,
    KPoiTypeDentist = 48 ,
    KPoiTypeVeterinarian = 49 ,
    KPoiTypeNightlife = 50 ,
    KPoiTypeAmusementPark = 51 ,
    KPoiTypeLibrary = 52 ,
    KPoiTypeCarRepairFacility = 53 ,
    KPoiTypePharmacy = 54 ,
    KPoiTypeScenicPanoramicView = 55 ,
    KPoiTypeSwimmingPool = 56 ,
    KPoiTypeWinery = 57 ,
    KPoiTypeCampingGround = 58 ,
    KPoiTypeParkRecreationArea = 59 ,
    KPoiTypeConventionCentre = 60 ,
    KPoiTypeLeisureCentre = 61 ,
    KPoiTypeYachtBasin = 62
};
```

```
    KPoiTypeMountainPeak = 1 ,
    KPoiTypeParkingGarage = 3 ,
    KPoiTypeRailwayStation = 5 ,
    KPoiTypeAirport = 7 ,
```

5.19. TInternalVersion structure

This structure is use by the [GetInternalVersions](#) function.

```
public struct TInternalVersion {  
    public int nMajor;           //major version of helper Dll  
    public int nMinor;          //minor version of helper Dll  
    public int nRelease;        //release of helper Dll  
    public string strBrand;      //brand of helper Dll  
    public System.Version vPPPTTCFVersion; //version of PPTTCF.dll (can be null)  
}
```

6. Error codes

The following error codes are currently defined in TomTom API:

6.1. *ESDKErrors enumeration*

```
public enum ESDKErrors {  
    KErrSDKNone= 0,      //returned if the command was processed successfully  
    KErrSDKNoSession= -1, //no session was created  
    KErrSDKNoResponse= -2, //no response from the server to the client  
    KErrSDKNoMethod= -3,  //command not supported by TomTomNavigationServer  
    KErrSDKBadParams= -4, //number of params for the command does not match  
    KErrSDKMethodFailed= -5, //internal error during processing of the command  
    KErrSDKEngineBusy= -6, //server busy, command was ignored  
    KErrSDKNoMapOpen= -7, //no map was currently open (could happen when the map  
                          //was not activated)  
    KErrSDKUnknownError= -100, //unspecified error  
}
```

Note that these are the error codes returned from the navigation application as stored in TError. The API calls themselves return a subset of the above as a result:

- KErrSDKNone - if the response for a given request has been received;
- KErrSDKNoSession - if the client session could not be created;
- KErrSDKNoResponse - if the response was never received or did not have a valid format.