

# ***TomTom Navigator SDK***

## ***Version 2.0 build 162***

Copyright 2002 Palmtop B.V. All rights reserved.

# Table of Contents

<b>1. Installation and Getting Started</b>	1
<b>1.1. Introduction</b>	1
<b>1.2. System requirements</b>	1
<b>1.3. Installation</b>	1
<b>1.3.1. ActiveX and DLL components</b>	1
<b>1.3.2. Installing the Visual Basic–specific parts</b>	1
<b>1.3.3. Installing the Visual C++–specific parts</b>	1
<b>1.3.4. Points of interests</b>	1
<b>1.3.5. Desktop Computer</b>	1
<b>1.4. System requirements</b>	2
<b>1.5. License</b>	2
<b>1.5.1. ATTENTION</b>	2
<b>1.5.2. LICENSE AGREEMENT</b>	2
<b>1.5.3. MANUFACTURER’S WARRANTY AND LIMITATION OF LIABILITY</b>	3
<b>1.6. Technical Support</b>	3
<b>2. Providing your own Points of Interest to TomTom Navigator</b>	4
<b>2.1. Installing a POI database</b>	4
<b>2.2. How to make POI databases</b>	5
<b>2.3. Converting to Navigator format</b>	5
<b>2.4. OV2 File Structure</b>	6
<b>3. Communicating with TomTom Navigator from Visual Basic</b>	8
<b>3.1. Introduction</b>	8
<b>3.2. Supported API calls</b>	8
<b>3.2.1. Function GetSdkVersionInfo( ) As String</b>	8
<b>3.2.2. Function GetNavigatorVersionInfo( ) As String</b>	8
<b>3.2.3. Sub BringNavigatorToForeground( )</b>	8
<b>3.2.4. Sub SwitchToNavigatorView( )</b>	8
<b>3.2.5. Sub TryCloseCurrentOpenedDialogs( )</b>	9
<b>3.2.6. Sub NavigateToCoordinate( aLongitude As Long, aLatitude As Long, aName As String)</b>	9
<b>3.2.7. Sub ClearFavorite( aFavorite As Long)</b>	9
<b>3.2.8. Sub SetFavorite( aFavorite As Long, aName As String, aDescription As String, aLongitude As Long, aLatitude As Long)</b>	9
<b>3.2.9. Function GetFavoriteName( aFavorite As Long) As String</b>	10
<b>3.2.10. Function GetFavoriteDescription( aFavorite As Long) As String</b>	10
<b>3.2.11. Function GetFavoriteLongitude( aFavorite As Long) As Long</b>	10
<b>3.2.12. Function GetFavoriteLatitude( aFavorite As Long) As Long</b>	10
<b>3.2.13. Sub NavigateToFavorite( aFavorite As Long)</b>	11
<b>3.2.14. Sub ShowCoordinateOnMap( aLongitude As Long, aLatitude As Long)</b>	11
<b>3.2.15. Sub SendDirectCommand( aCommand As Long)</b>	11
<b>3.2.16. Sub GetCurrentPosition( )</b>	12
<b>3.2.17. Sub MakePoiVisible( aPoiTypeID As Long, aVisibility As Boolean)</b>	13
<b>3.2.18. Sub AddPoi( aFilename As String, aLongitude As Long, aLatitude As Long, aName As String, anId As String)</b>	14

3.2.19. Sub DeleteClosestPoi( aFilename As String, aLongitude As Long, aLatitude As Long)	15
3.2.20. Sub DeleteAllPoi( aFilename As String)	15
3.2.21. Sub MakeUserPoiVisible( aFilename As String, aVisibility As Boolean)	15
3.2.22. Sub MoveClosestPoi( aFilename As String, aLongitude As Long, aLatitude As Long, aNewLongitude As Long, aNewLatitude As Long)	16
3.2.23. Sub Geocode( aCity As String, aStreet As String, aHouseNr As String)	16
3.2.24. Sub GetDataSetInfo( )	17
3.2.25. Sub GetRouteInfo( )	17
3.2.26. Sub UseGFFile( aFilename As String)	18
3.2.27. Sub EnableGFRecord( aRecordID As Long, aState As Boolean)	18
4. Communicating with TomTom Navigator from C++	19
4.1. Introduction	19
4.2. Supported API calls	19
4.2.1. INT GetSdkVersionInfoV01( TNavVersionInfoV01& aVersionInfo)	19
4.2.2. INT GetNavigatorVersionInfoV01( TNavVersionInfoV01& aVersionInfo)	19
4.2.3. INT BringNavigatorToForeground( )	19
4.2.4. INT SwitchToNavigatorView( )	19
4.2.5. INT TryCloseCurrentOpenedDialogs( )	19
4.2.6. INT NavigateToCoordinate( long aLongitude, long aLatitude, LPCTSTR aName)	20
4.2.7. INT ClearFavorite( INT aIndex)	20
4.2.8. INT SetFavoriteV01( INT aIndex, TFavouriteRecV01 & aFavorite)	20
4.2.9. INT GetFavoriteV01( INT aIndex, TFavouriteRecV01& aFavorite)	20
4.2.10. INT NavigateToFavorite( INT aIndex)	21
4.2.11. INT ShowCoordinateOnMap( long aLongitude, long aLatitude)	21
4.2.12. INT SendDirectCommand( EDirectCommands aCommand)	21
4.2.13. INT GetCurrentPositionV01( EGpsStatus & aStatus, TTTnLocationV01 & aLocation)	22
4.2.14. INT MakePoiVisible( EPoiTypeCode aPoiID, INT aVisibility)	22
4.2.15. INT AddPoi( LPCTSTR aFilename, long aLongitude, long aLatitude, LPCTSTR aName, LPCTSTR aId)	22
4.2.16. INT DeleteClosestPoi( LPCTSTR aFilename, long aLongitude, long aLatitude)	23
4.2.17. INT DeleteAllPoi( LPCTSTR aFilename)	23
4.2.18. INT MakeUserPoiVisible( LPCTSTR aFilename, INT aVisibility)	23
4.2.19. INT MoveClosestPoi( LPCTSTR aFilename, long aLongitude, long aLatitude, long aNewLongitude, long aNewLatitude)	24
4.2.20. INT GeocodeV01( LPCTSTR aCity, LPCTSTR aStreet, LPCTSTR aHouseNr, TGeocodeInfoV01& aGeocodeInfo)	24
4.2.21. INT GetDataSetInfoV01( TDataSetInfoV01& aDataSetInfo)	25
4.2.22. INT GetRouteInfoV01( TRouteInfoV01& aRouteInfo)	25
4.2.23. INT UseGFFile( LPCTSTR aFilename)	25
4.2.24. INT EnableGFRecord( long aRecordID, INT aState)	25
4.3. Structures and enumerations from the API	26

4.3.1. <i>TNavVersionInfoV01</i> structure .....	26
4.3.2. <i>TFavouriteRecV01</i> structure .....	26
4.3.3. <i>EDirectCommands</i> enumeration .....	26
4.3.4. <i>TTnLocationV01</i> structure .....	26
4.3.5. <i>EGpsStatus</i> enumeration .....	27
4.3.6. <i>TGeocodeInfoV01</i> structure .....	27
4.3.7. <i>EPoiTypeCode</i> enumeration .....	27
4.3.8. <i>TDataSetInfoV01</i> structure .....	29
4.3.9. <i>TRouteInfoV01</i> structure .....	29
5. <b>Graphic Information Files</b> .....	30
5.1. <b>About GF Files</b> .....	30
5.2. <b>GF File Format</b> .....	30
5.2.1. <b>Basics of the File Format</b> .....	30
5.2.2. <b>Supported Record Types</b> .....	31
6. <b>Deployment</b> .....	34
7. <b>External GPS Drivers</b> .....	35
7.1. <b>About External GPS Drivers</b> .....	35
7.2. <b>Providing GPS information</b> .....	35
7.2.1. <b>Configuring TomTom Navigator</b> .....	35
7.2.2. <b>Providing your own feed</b> .....	35
7.2.3. <b>DWORD AttachToTomTomGPSEngine( )</b> .....	36
7.2.4. <b>VOID DetachFromTomTomGPSEngine( DWORD aHandle)</b> .....	36
7.2.5. <b>BOOL CopyDataToTomTomGPSEngine( DWORD aHandle, DWORD aNumberOfBytes, LPBYTE aBuffer)</b> .....	36

# 1. Installation and Getting Started

## 1.1. Introduction

This manual explains how to install and use the software developer's kit (SDK) of TomTom Navigator. The SDK offers access to functions to add your own points of interest, and to send commands to TomTom Navigator through a dynamic link library in C++ or an ActiveX control in Visual Basic.

## 1.2. System requirements

To use the SDK, TomTom Navigator version 1.5 or higher should be installed on a PDA with Microsoft Windows CE 3.0 or higher (Pocket PC or Pocket PC 2002). Please check the read-me file for version compatibility issues.

The PC that is used for system development should have Microsoft Windows NT, Windows 2000 or Windows XP, Microsoft eMbedded Visual Tools version 3.0 or higher, and Microsoft ActiveSync 3.1 or higher installed. A relationship between the PDA and the PC should have been established.

## 1.3. Installation

### 1.3.1. ActiveX and DLL components

The ActiveX component and DLL are required to make function calls from C++ or Visual Basic. The `\Sdk\TTNCom\install\` contains SETUP.EXE, which will install the DLL and ActiveX components on your PDA.

### 1.3.2. Installing the Visual Basic-specific parts

The SETUP.EXE program will install the components required for your Pocket PC, but for development, you need to install additional components on your desktop computer. See the section "Desktop Computer" below. If you are upgrading from a previous version of the Navigator SDK, your old '.VB' files in the Pocket PC will stop working. In that case you need to open the corresponding Visual Basic project files on your desktop computer and regenerate the '.VB' files in the Pocket PC.

### 1.3.3. Installing the Visual C++-specific parts

The `\Sdk\TTNCom\lib\` directory contains the '.lib' files that an application needs to link against in order to use the SDK, and the directory `\Sdk\TTNCom\include\` contains any header file that a program needs to include to be able to use the SDK DLL. No additional installation but unpacking is required.

### 1.3.4. Points of interests

The SDK contains a set of executables that facilitate creation of customized POI data sets that can be added to the map. These executables are installed simply by unpacking the complete SDK. The executables are stored in subdirectory `\SDK\POI\`, the chapter 'Providing your own points of interest to TomTom Navigator' explains how to use the executables.

### 1.3.5. Desktop Computer

In the desktop computer it is required to install a design mode version of TTNControl.ocx so that you are able to add the TomTom Navigator SDK functionality to your eMbedded Visual Basic project. Note that this is needed only during the development phase of your application. The design mode version of TTNControl.ocx can be found in the directory `\sdk\TTNCom\OcxDesktop\`. To install it, just copy it to any suitable place in your hard disk and, if you are installing a new version of the Navigator SDK over an old

## 1. Installation and Getting Started

one, register it with the Windows registry. If you are installing the Navigator SDK for the first time, you needn't register the control explicitly. Assuming your current directory is the directory where you copied TTNControl.ocx, the command line to perform the registration would be:

```
regsvr32.exe TTNControl.ocx
```

When you start a new eMbedded Visual Basic project, you need to add the TomTom Navigator SDK ActiveX component to it. To do this, select "Components..." from the "Project" menu and, in the "Controls" tab, check the box "TomTom Navigator SDK 1.X." If that box is not present you will need to click on the "Browse..." button and search for TTNControl.ocx on your hard disk.

### 1.4. System requirements

In addition to the requirements for TomTom Navigator, the SDK further requires "Microsoft eMbedded Visual Tools," that is, "eMbedded Visual C++ 3.0" or "eMbedded Visual Basic 3.0."

### 1.5. License

#### 1.5.1. ATTENTION

Use of our software is subject to the PALMTOP BV License and Limited Warranty terms set forth below. If you do not accept these terms, you cannot install our software and you are not entitled to use our software.

#### 1.5.2. LICENSE AGREEMENT

This is a legal agreement between you, the end user, and PALMTOP BV, ('Palmtop').

IF YOU DO NOT AGREE TO THE TERMS OF THIS AGREEMENT, PROMPTLY RETURN THE CD-ROM/FLOPPY DISK/MEMORY CARD/DOWNLOAD PACKAGE AND THE ACCOMPANYING ITEMS TO THE PLACE YOU OBTAINED THEM FOR A FULL REFUND. BY BREAKING THE SEAL OF THE CD-ROM/FLOPPY DISK/MEMORY CARD, PRESSING THE "I AGREE" BUTTON FOR A DOWNLOAD, OR BY USING THE SOFTWARE YOU ARE AGREEING TO BE BOUND BY THE TERMS OF THIS AGREEMENT.

The license agreement is subject to Dutch Law and the District Court of Amsterdam is the only competent court for disputes based on the license agreement or the use of the (licensed) software.

1) GRANT OF LICENSE: This Palmtop License Agreement ('License') permits you to use one of the Palmtop computer programs and the digital (map) data included in the accompanying package acquired with this license ('SOFTWARE') on any single computer, provided the SOFTWARE is installed on only one computer at any time and provided the software is combined only with one or more navigation systems.

2) COPYRIGHT: The SOFTWARE is owned by Palmtop or its suppliers and is protected by Netherlands Copyright Laws, international treaty provisions, and all other applicable national laws. Therefore, you must treat the SOFTWARE like any other copyrighted material (e.g., a book) except that if the software is not copy protected you may either (a) make one copy of the SOFTWARE solely for backup or archival purposes, or (b) transfer the SOFTWARE to a single medium provided you keep the original solely for backup or archival purposes. You may not copy the Product manual(s) or written materials accompanying the SOFTWARE. You only become the owner of the material data carrier and you do not acquire the ownership of the SOFTWARE.

3) DEPLOYMENT CONDITIONS: Any other product that is developed using the SOFTWARE may include(s) (a) software component(s) of Palmtop that were shipped as part of the SOFTWARE. This/These software component(s) are freely distributable components as long as it/they are only shipped as part of an

## *1. Installation and Getting Started*

installation package of the product that was developed. Copyright and intellectual property rights of the SOFTWARE and the included component(s) remain with Palmtop at all times. This should be indicated in the end user license agreement of the product that was developed using the SOFTWARE together with a statement that the software can only be used in combination with a valid TomTom Navigator license.

4) OTHER RESTRICTIONS: Unauthorised hiring, lending, public performance and broadcasting is prohibited, but you may transfer your rights under this Palmtop License Agreement on a permanent basis provided you transfer all copies of the SOFTWARE and all written materials and the recipient agrees to the terms of this License. You are not permitted to fully or partly modify the SOFTWARE, to analyse it by means of reverse-engineering, to decompile or disassemble the SOFTWARE, or to make products derived from it. You are explicitly prohibited from downloading the digital maps and programs contained in the SOFTWARE or from transferring these to another data carrier or computer.

### **1.5.3. MANUFACTURER'S WARRANTY AND LIMITATION OF LIABILITY**

1) Please read the instructions supplied with the software before you use it. If you have any difficulty using the software, consult the instructions to check you are using it correctly. The use of the SOFTWARE in a navigation system means that calculation errors can occur, caused by local environmental conditions and/or incomplete data.

2) For the above mentioned reasons Palmtop can not warrant that the SOFTWARE and data carrier operates error-free. During a period of three months after the date of purchase Palmtop or its dealer will replace the defective software. This warranty covers the prompt replacement of the data carrier only. Where the error is the result of misuse, unusual external effects, accidental damage, normal wear and tear, unauthorised repair, or any other cause Palmtop cannot be blamed for, Palmtop or the relevant dealer (as appropriate) may charge you for the repair or the replacement.

3) For any claim arising out of Palmtop's limited warranty, or for any other claim whatsoever related to the subject matter of these terms, Palmtop's liability for actual damages, regardless of the form of action, shall be limited to the greater of \$ 5.000 or the money paid to Palmtop or its dealer for the license of the software that caused the damages. The limitation will not apply to claims of personal injury or damages to tangible personal property caused by Palmtop's willful intent or gross negligence.

4) In no event will Palmtop be liable for any loss of data, lost profits, lost savings, or any incidental damages or other consequential damages, even if Palmtop or its dealers have been advised of the possibility of such damages, or for any claim by you based on a third party claim.

5) Palmtop makes no warranty or representation that its software products will work in combination with any hardware or software products provided by third parties.

6) Palmtop does not accept responsibility if the SOFTWARE does not function at all, does not function completely or does not function fully, if and for as far as this results from, or is caused by, using the SOFTWARE in combination with the hardware, the operating system or any other external aid, whether this aid is of a technical or any other nature.

7) IMPORTANT: PALMTOPO BV AND ITS SUPPLIERS DISCLAIM ALL LIABILITY FOR ANY USE OF THIS PRODUCT IN A WAY THAT MAY CAUSE ACCIDENTS, DAMAGE OR THAT MAY VIOLATE THE LAW.

## **1.6. Technical Support**

For technical support please refer to [www.tomtom.com](http://www.tomtom.com). There you will find the latest versions and documentation of the SDK, Questions & Answers, tips & tricks, add-ons and tools.

## 2. Providing your own Points of Interest to TomTom Navigator

It is possible to add new Points of Interest (POI) to TomTom Navigator. This section explains how.

### Warning:

- (1) No information stored in OV2 format, or extracted from files in OV2 format (using whatever means, tools or techniques, based upon either published or self-discovered knowledge about the OV2 format), nor any knowledge about the OV2 format itself, may be
  - sold in any form, unless with written permission from the owner of the original (raw) information,
  - used from within any application other than a TomTom product, commercial or otherwise, without explicit written permission from a director of either Palmtop BV or TomTom Inc, unless
    - (a) that application has the sole purpose of providing additional functionality to TomTom Navigator or TomTom Citymaps, and
    - (b) that application only functions on devices on which TomTom Navigator or TomTom Citymaps is installed, and
    - (c) that application is accompanied by a warning similar to this warning.
- (2) Furthermore,
  - we will not provide support about the format, or any related tools or documentation,
  - we do not guarantee correctness or completeness of the either the format, the tools or the documentation,
  - we hold the right to change or extend the format without notice,
  - we do not guarantee that details of future, changed or extended formats will be made available,
  - we do not guarantee that future, changed or extended formats will be compatible with the current format,
  - we do not promise that we will allow the same things for future, changed or extended formats.
- (3) Finally, please note that
  - the POI data that is distributed as part of TomTom products is
    - (a) not in OV2 format, (b) expressly not allowed to be accessed in any way whatsoever, and (c) protected by international copyright laws.

### 2.1. Installing a POI database

A POI database consists of a file with extension OV2 (e.g. Hilton\_Hotels.OV2) and, optionally, a companion file with extension BMP (e.g. Hilton\_Hotels.BMP) To make such databases visible on a TomTom Navigator map, make sure that: The file(s) are stored in the same directory on your PocketPC where the map data are located. TomTom Navigator (re)opened the map after the OV2 file was put there (if necessary, restart TomTom Navigator). In the POI tab of your TomTom Navigator preferences, make sure "Display POI" option is checkmarked, as well as the POI themselves. Example: to use the database Hilton\_Hotels.OV2 in the Germany map, you might do the following: Connect your PocketPC to the PC and start Microsoft ActiveSync. Use the menu option File->Explore to explore the directories on your

## 2. Providing your own Points of Interest to TomTom Navigator

PocketPC. Browse to the directory

```
\My Documents\Germany
```

or to

```
\Storage Card\Germany
```

if the map was installed on a storage card. (Usually, the Explore option will start in "My Documents" anyway, so you will immediately see the "Germany" directory.) Copy the file Hilton\_Hotels.OV2 (and optionally the file Hilton\_Hotels.BMP) to this directory.

Next, on your PocketPC, re-open the Germany map in TomTom Navigator. If necessary, re-start the whole TomTom Navigator application (using the TomTom Navigator menu option "Exit").

Finally, in the POI tab of your TomTom Navigator Preferences, make sure "Hilton Hotels" has a checkmark – otherwise the hotels will not be displayed on your screen. Even if you do not display them on your screen, you can now "navigate to" a Hilton Hotel, find the Hilton Hotels near your location etc. If you experience problems, here are some possibilities: the OV2 file is not in the same directory as the map the OV2 file does not contain any POI, or at least no POI within the borders of the map you are using a TomTom Navigator version older than version 1.40

## 2.2. How to make POI databases

POI databases can be created in two ways: Manually, from within TomTom Navigator, by using the "add as Point of Interest" options (e.g. in the map view, tap & hold your pen on a particular place on the map). On a PC, by converting whole databases into TomTom Navigator Format (see below)

## 2.3. Converting to Navigator format

Palmtop provides a set of simple tools to generate POI databases. They are provided with the SDK in the directory `sdk\poi\`.

The first tool, MAKEOV2, can be used to convert a simple text file containing the coordinates and names of locations into a ready-to-use OV2 file. The second tool, DUMPOV2, can dump the content of POI databases into text format (yes, text that can be used by the MAKEOV2 tool).

The use of DUMPOV2.EXE is simple and straightforward:

```
DUMPOV2 inputfilename [outputfilename]
```

Where "inputfilename" must be a valid TomTom Navigator POI file (with extension OV2) or a valid Route Planner or Citymaps OVERLAY file (with extension OVR). It is recommended to provide an outputfilename with extension ASC. The use of MAKEOV2.EXE is different:

```
MAKEOV2 inputfilename [outputfilename]
```

The input file should be a text file (extension ASC is recommended) that should simply contain lines of text. Any line starting with a semi-colon will be ignored. Empty lines will also be ignored. All other lines are expected to represent points of interest. Such a line should specify a longitude, a latitude and a name, separated by commas. It is recommended that the name is put between double quotes. A name should not contain double quotes. Longitudes and latitudes may be specified either as degrees and fractions of degrees, or in degrees, minutes and seconds. Both colons and single-quote/double-quote notation may be used for minutes and seconds. So, all the following lines are all equivalent:

```
53.5 , 4 , "Truckers Restaurant La Bamba"  
53.5000000 , 4.00000000 , "Truckers Restaurant La Bamba"  
53'30"00 , 4'00"00 , "Truckers Restaurant La Bamba"  
53'30 , 4'0 , "Truckers Restaurant La Bamba"
```

## 2. Providing your own Points of Interest to TomTom Navigator

53:30:0 , 4 , "Truckers Restaurant La Bamba"

### 2.4. OV2 File Structure

TomTom Navigator POI files are stored in so-called OV2-format, and as a rule have the filename-extension "ov2". This document provides some information about the content of such files. Please note the following legal conditions:

- (1) No information stored in OV2 format, or extracted from files in OV2 format (using whatever means, tools or techniques, based upon either published or self-discovered knowledge about the OV2 format), nor any knowledge about the OV2 format itself, may be
  - sold in any form, unless with written permission from the owner of the original (raw) information,
  - used from within any application other than a TomTom product, commercial or otherwise, without explicit written permission from a director of either Palmtop BV or TomTom Inc, unless
    - (a) that application has the sole purpose of providing additional functionality to TomTom Navigator or TomTom Citymaps, and
    - (b) that application only functions on devices on which TomTom Navigator or TomTom Citymaps is installed, and
    - (c) that application is accompanied by a warning similar to this warning.
- (2) Furthermore,
  - we will not provide support about the format, or any related tools or documentation,
  - we do not guarantee correctness or completeness of the either the format, the tools or the documentation,
  - we hold the right to change or extend the format without notice,
  - we do not guarantee that details of future, changed or extended formats will be made available,
  - we do not guarantee that future, changed or extended formats will be compatible with the current format,
  - we do not promise that we will allow the same things for future, changed or extended formats.
- (3) Finally, please note that
  - the POI data that is distributed as part of TomTom products is
    - (a) not in OV2 format, (b) expressly not allowed to be accessed in any way whatsoever, and (c) protected by international copyright laws.

An OV2 file consists of a sequence of variable-length records. Every record starts with a one-byte "type". This type tells you how to process the rest of the record.

You should not encounter types other than 0, 1, 2 and 3 – if you do, the file is either corrupt or in a different (e.g. a higher-version) format.

Coordinates are stored as 4-byte integers representing a WGS84 longitude or latitude, multiplied by 100.000 and rounded to the nearest integer. As such, an X-coordinate should always be a value between -18.000.000 and +18.000.000, and a Y-coordinate should be a value between -9.000.000 and +9.000.000.

#### DELETED RECORD:

1 byte      T: type (always 0)

## 2. Providing your own Points of Interest to TomTom Navigator

4 bytes L: length of this record in bytes (including the T and L fields)  
L-5 bytes bytes to ignore (content undefined)

### **TYPE-1 RECORD:**

1 byte T: type (always 1)  
20 bytes bytes to ignore (proprietary content)

### **SIMPLE POI RECORD:**

1 byte T: type (always 2)  
4 bytes L: length of this record in bytes (including the T and L fields)  
4 bytes X: longitude coordinate of the POI  
4 bytes Y: latitude coordinate of the POI  
L-13 bytes Name: zero-terminated ASCII string specifying the name of the POI

### **EXTENDED POI RECORD:**

1 byte T: type (always 3)  
4 bytes L: length of this record in bytes (including the T and L fields)  
4 bytes X: longitude coordinate of the POI  
4 bytes Y: latitude coordinate of the POI  
P bytes Name: zero-terminated ASCII string specifying the name of the POI  
L-P-13 bytes bytes to ignore (proprietary content)

If you encounter any other type, either the file is corrupt, or the file contains extra (proprietary) records. In either case, you should stop processing immediately. Since there is always the danger that the file is corrupt, you should in fact wonder whether the preceding records read so far were in fact valid.

# 3. Communicating with TomTom Navigator from Visual Basic

## 3.1. Introduction

The TTNControl.ocx ActiveX control allows Visual Basic developers to create applications that can send commands to and request information from TomTom Navigator. To do this you simply have to place an instance of the TTNControl.ocx ActiveX control in the forms of your Visual Basic application. The control should be made invisible, since it has no interest for the user interface: all the work is carried out programmatically through OLE automation commands sent to it.

Please note that Pocket PC emulators do not always emulate real devices. Also, compatibility conflicts can arise with emulators that do not occur when running an application directly on a Pocket PC device. We therefore recommend to run applications directly on the device.

TTNControl.ocx defines a class called "TomTomNavigator", which is the ActiveX control that you have to instantiate. Besides the properties common to all ActiveX controls, the "TomTomNavigator" class defines a number of functions to enable communication with the TomTom Navigator application. You can view these functions using the Object Browser from within the eMbedded Visual Basic development environment. All these functions can raise errors at run time that need to be handled using the standard Visual Basic error handling mechanism.

For example, if the name given to the instance of TTNControl.ocx in your form is "TTNav", the following lines of code will request the version number of the TomTom Navigator SDK:

```
Dim sdkVersion As String
sdkVersion = TTNav.GetSdkVersionInfo
```

## 3.2. Supported API calls

The function calls currently supported are:

### 3.2.1. Function *GetSdkVersionInfo( ) As String*

**Returns:** SDK version as a string

**Supported Since:** TTN 1.42

Returns the SDK version number information with the format "version (build number)."

### 3.2.2. Function *GetNavigatorVersionInfo( ) As String*

**Returns:** Navigator version as a string

**Supported Since:** TTN 1.42

Returns the version number information of TomTom Navigator with the format "version (build number)."

### 3.2.3. Sub *BringNavigatorToForeground( )*

**Supported Since:** TTN 1.42

Puts the TomTom Navigator application in the foreground.

### 3.2.4. Sub *SwitchToNavigatorView( )*

**Supported Since:** TTN 1.42

### 3. Communicating with TomTom Navigator from Visual Basic

Switches TomTom Navigator to the navigation view. Useful because various API calls only work in the navigation view.

#### 3.2.5. Sub TryCloseCurrentOpenedDialogs( )

**Supported Since:** TTN 1.42

Tries to close any opened dialogs on top. Useful for API calls like navigating to a location, which won't work when a dialog is on screen. Some dialogs cannot be closed, so this operation might fail.

#### 3.2.6. Sub NavigateToCoordinate( aLongitude As Long, aLatitude As Long, aName As String)

**Command Parameters:**

- **aLongitude As Long:** longitude specified in WGS84 format (in millionths of degrees)
- **aLatitude As Long:** latitude specified in WGS84 format (in millionths of degrees)
- **aName As String:** name of the location

**Supported Since:** TTN 1.42

Navigates to a location specified in WGS84 format (longitude and latitude in millionths of degrees).

**Example:**

The following example call instructs TomTom Navigator to plan a route to a point somewhere in Amsterdam, the Netherlands.

```
Sub NavigateToAmsterdam
    On Error GoTo NavigateToAmsterdam_Error
    Call TTNNav.SwitchToNavigatorView
    Call TTNNav.NavigateToCoordinate(4860000, 52380000, "Amsterdam")
    Call TTNNav.BringNavigatorToForeground
NavigateToAmsterdam_Exit:
    Exit Sub
NavigateToAmsterdam_Error:
    MsgBox "Navigation to Amsterdam failed"
    Resume NavigateToAmsterdam_Exit
End Sub
```

**See Also:** NavigateToFavorite, ShowCoordinateOnMap

#### 3.2.7. Sub ClearFavorite( aFavorite As Long)

**Command Parameters:**

- **aFavorite As Long:** index of favorite (base zero)

**Supported Since:** TTN 1.42

Clears a favorite location.

**See Also:** SetFavorite, GetFavoriteName, GetFavoriteDescription, GetFavoriteLongitude, GetFavoriteLatitude, NavigateToFavorite

#### 3.2.8. Sub SetFavorite( aFavorite As Long, aName As String, aDescription As String, aLongitude As Long, aLatitude As Long)

**Command Parameters:**

- **aFavorite As Long:** index of favorite (base zero)
- **aName As String:** name of favorite

### 3. Communicating with TomTom Navigator from Visual Basic

- **aDescription As String:** description of favorite
- **aLongitude As Long:** longitude of favorite
- **aLatitude As Long:** latitude of favorite

**Supported Since:** TTN 1.42

Sets favorite location information.

**See Also:** ClearFavorite, GetFavoriteName, GetFavoriteDescription, GetFavoriteLongitude, GetFavoriteLatitude, NavigateToFavorite

#### **3.2.9. Function GetFavoriteName( aFavorite As Long) As String**

**Command Parameters:**

- **aFavorite As Long:** index of favorite (base zero)

**Returns:** name of favorite

**Supported Since:** TTN 1.42

Returns the name of a favorite location.

**See Also:** ClearFavorite, GetFavoriteDescription, GetFavoriteLongitude, GetFavoriteLatitude, SetFavorite, NavigateToFavorite

#### **3.2.10. Function GetFavoriteDescription( aFavorite As Long) As String**

**Command Parameters:**

- **aFavorite As Long:** index of favorite (base zero)

**Returns:** description of favorite

**Supported Since:** TTN 1.42

Returns the description of a favorite location.

**See Also:** ClearFavorite, GetFavoriteName, GetFavoriteLongitude, GetFavoriteLatitude, SetFavorite, NavigateToFavorite

#### **3.2.11. Function GetFavoriteLongitude( aFavorite As Long) As Long**

**Command Parameters:**

- **aFavorite As Long:** index of favorite (base zero)

**Returns:** longitude of favorite

**Supported Since:** TTN 1.42

Returns the longitude of a favorite location.

**See Also:** ClearFavorite, GetFavoriteName, GetFavoriteDescription, GetFavoriteLatitude, SetFavorite, NavigateToFavorite

#### **3.2.12. Function GetFavoriteLatitude( aFavorite As Long) As Long**

**Command Parameters:**

- **aFavorite As Long:** index of favorite (base zero)

**Returns:** latitude of favorite

**Supported Since:** TTN 1.42

Returns the latitude of a favorite location.

**See Also:** ClearFavorite, GetFavoriteName, GetFavoriteDescription, GetFavoriteLongitude, SetFavorite,

### 3. Communicating with TomTom Navigator from Visual Basic

NavigateToFavorite

#### 3.2.13. Sub *NavigateToFavorite(aFavorite As Long)*

**Command Parameters:**

- **aFavorite As Long:** index of favorite (base zero)

**Supported Since:** TTN 1.42

Navigates to a specific favorite location.

**See Also:** NavigateToCoordinate, ClearFavorite, SetFavorite, GetFavoriteName, GetFavoriteDescription, GetFavoriteLongitude, GetFavoriteLatitude

#### 3.2.14. Sub *ShowCoordinateOnMap(aLongitude As Long, aLatitude As Long)*

**Command Parameters:**

- **aLongitude As Long:** longitude specified in WGS84 format (in millionths of degrees).
- **aLatitude As Long:** latitude specified in WGS84 format (in millionths of degrees).

**Supported Since:** TTN 1.42

Centers the map around the given coordinate (zoomed in), specified in WGS84 format (longitude and latitude in millionths of degrees).

**Example:**

```
Sub ShowAmsterdam
    On Error GoTo ShowAmsterdam_Error
    Call TTNNav.ShowCoordinateOnMap(4860000, 52380000)
ShowAmsterdam_Exit:
    Exit Sub
ShowAmsterdam_Error:
    MsgBox "Failed to center around a point in Amsterdam"
    Resume ShowAmsterdam_Exit
End Sub
```

**See Also:** NavigateToCoordinate

#### 3.2.15. Sub *SendDirectCommand(aCommand As Long)*

**Command Parameters:**

- **aCommand As Long:** code of the command to execute

**Supported Since:** TTN 1.42

Performs one of various commands to change a setting or instruct TomTom Navigator to perform a certain task. The command codes are:

- 1 – SoundOn: Turn spoken voice instructions on (works only when in navigation view)
- 2 – SoundOff: Turn spoken voice instructions off (works only when in navigation view)
- 3 – NightColors: Show map in night colors (works only when in navigation view)
- 4 – DayColors: Show map in day colors (works only when in navigation view)
- 5 – ShowMap: Show map and instructions while driving (works only when in navigation view)
- 6 – HideMap: Only show instructions while driving (works only when in navigation view)
- 7 – Recalculate: Recalculate a route (works only when in navigation view)
- 8 – ShowPOI: Show points of interest on the map
- 9 – HidePOI: Hide points of interest on the map

### 3. Communicating with TomTom Navigator from Visual Basic

- 10 – ShowExits: Show the next highway exit sign on screen (works only when in navigation view)
- 11 – HideExits: Hide the next highway exit sign on screen (works only when in navigation view)
- 12 – DisableGuidance: Disable active guidance support (works only when in navigation view)
- 13 – EnableGuidance: Enable active guidance support (works only when in navigation view)
- 14 – ShowCompass: Show compass on screen (works only when in navigation view)
- 15 – HideCompass: Hide compass (works only when in navigation view)
- 16 – GoHome: Navigate to the "Home" favorite (works only when in navigation view)
- 17 – ShowFavorites: Show favorite locations on the map
- 18 – HideFavorites: Hide favorite locations on the map

Versions 1.51 and later of TomTom Navigator support also the following direct commands:

- 19 – UnitsKm: set distance units to kilometers
- 20 – UnitsM: set distance units to miles
- 21 – AutoRecalcOn: turn on automatic recalculation of the route
- 22 – AutoRecalcOff: turn off automatic recalculation of the route
- 23 – AutoZoomOn: turn on automatic zooming
- 24 – AutoZoomOff: turn off automatic zooming

#### Example:

The following example invocation call demonstrates how to switch the map display to night colors:

```
Sub SwitchToNightView
    On Error GoTo SwitchToNightView_Error
    Call TTNav.SendDirectCommand(3)
SwitchToNightView_Exit:
    Exit Sub
SwitchToNightView_Error:
    MsgBox "Failed to change to night view mode"
    Resume SwitchToNightView_Exit
End Sub
```

#### 3.2.16. Sub GetCurrentPosition( )

**Supported Since:** TTN 1.42

Obtains the current position information from GPS. The GPS data is available after this call in the read-only properties GpsStatus, Longitude, Latitude, Direction and Speed:

**Property GpsStatus As Long:** Indication of the quality of the GPS data (if < 0 indicates an error)

**Property Longitude As Long:** Longitude in millionths of degrees (WGS84 format)

**Property Latitude As Long:** Latitude in millionths of degrees (WGS84 format)

**Property Direction As Long:** Direction in degrees: zero is north, clockwise

**Property Speed As Long:** Speed in kilometers per hour

If GetCurrentPosition has never been called, GpsStatus will be -1, indicating that these properties don't contain valid GPS information.

#### Example:

The shows how the GetCurrentPosition call can be used to warn when the driver is driving too fast.

```
Sub CheckSpeed
    On Error GoTo CheckSpeed_Error
    Call TTNav.GetCurrentPosition
```

### 3. Communicating with TomTom Navigator from Visual Basic

```
If TTNNav.GpsStatus >= 0 And TTNNav.Speed > 120 Then
    MsgBox "You are driving too fast!"
End If
CheckSpeed_Exit:
    Exit Sub
CheckSpeed_Error:
    MsgBox "Failed to get the current location"
    Resume CheckSpeed_Exit
End Sub
```

#### **3.2.17. Sub MakePoiVisible( aPoiTypeID As Long, aVisibility As Boolean)**

##### **Command Parameters:**

- **aPoiTypeID As Long:** POI unique type identifier
- **aVisibility As Boolean:** whether to show or hide this POI type on the map

##### **Supported Since:** TTN 1.42

Enables or disables displaying a specific POI type.

The POI type identifier can be one of the following:

- Government Office – 7367
- Mountain Peak – 9364
- Open Parking – 7369
- Parking Garage – 7313
- Petrol Station – 7311
- Railway Station – 7380
- Rest Area – 7395
- Airport – 7383
- Car Dealer – 9910
- Casino – 7341
- Church – 9906
- Cinema – 7342
- City Center – 7379
- Company – 9352
- Concert Hall – 9367
- Courthouse – 9363
- Cultural Center – 7319
- Exhibition Center – 7385
- Ferry Terminal – 7352
- Frontier Crossing – 7366
- Golf Course – 9911
- Hospital Polyclinic – 7321
- Hotel/Motel – 7314
- Tourist Attraction – 7376
- Mountain Pass – 9935
- Museum – 7317
- Opera – 9365
- Place of Worship – 7339
- Post Office – 7324
- Rent Car Facility – 7312

### 3. Communicating with TomTom Navigator from Visual Basic

- Rent Car Parking – 9930
- Restaurant – 7315
- Shop – 9361
- Shopping Center – 7373
- Stadium – 7374
- Theatre – 7318
- Tourist Information Office – 7316
- Zoo – 9927
- Sports Centre – 7320
- Police Station – 7322
- Embassy – 7365
- College University – 7377
- Cash Dispenser – 7397
- Beach – 9357
- Ice Skating Ring – 9360
- Tennis Court – 9369
- Water Sport – 9371
- Doctor – 9373
- Dentist – 9374
- Veterinarian – 9375
- Nightlife – 9379
- Amusement Park – 9902
- Library – 9913

Versions 2.0 and later of TomTom Navigator also support the following POI types:

- Car Repair Facility – 7310
- Pharmacy – 7326
- Scenic/Panoramic View – 7337
- Swimming Pool – 7338
- Winery – 7349
- Camping Ground – 7360
- Park and Recreation Area – 9362
- Convention Centre – 9377
- Leisure Centre – 9378
- Yacht Basin – 9380

#### **3.2.18. Sub AddPoi( aFilename As String, aLongitude As Long, aLatitude As Long, aName As String, anId As String)**

##### **Command Parameters:**

- **aFilename As String:** name of the POI category.
- **aLongitude As Long:** longitude specified in WGS84 format (in millionth of degrees).
- **aLatitude As Long:** latitude specified in WGS84 format (in millionth of degrees).
- **aName As String:** name of the POI.
- **anId As String:** optional external identifier of the POI.

**Supported Since:** TTN 1.51

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.

### 3. Communicating with TomTom Navigator from Visual Basic

#### Example:

The following example invocation call demonstrates how to create a new point of interest:

```
Sub CreateNewPoi
    On Error GoTo CreateNewPoi_Error
    Call TTNNav.AddPoi("ATM", 4749620, 52374810, "Postbank", "")
CreateNewPoi_Exit:
    Exit Sub
CreateNewPoi_Error:
    MsgBox "Failed to create the point of interest"
    Resume CreateNewPoi_Exit
End Sub
```

**See Also:** DeleteClosestPoi, DeleteAllPoi, MakeUserPoiVisible, MoveClosestPoi

#### **3.2.19. Sub DeleteClosestPoi( aFilename As String, aLongitude As Long, aLatitude As Long)**

##### Command Parameters:

- **aFilename As String:** name of the POI category.
- **aLongitude As Long:** longitude specified in WGS84 format (in millionth of degrees).
- **aLatitude As Long:** latitude specified in WGS84 format (in millionth of degrees).

**Supported Since:** TTN 1.51

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.

**See Also:** AddPoi, DeleteAllPoi, MakeUserPoiVisible, MoveClosestPoi

#### **3.2.20. Sub DeleteAllPoi( aFilename As String)**

##### Command Parameters:

- **aFilename As String:** name of the POI category.

**Supported Since:** TTN 1.51

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

**See Also:** AddPoi, DeleteClosestPoi, MakeUserPoiVisible, MoveClosestPoi

#### **3.2.21. Sub MakeUserPoiVisible( aFilename As String, aVisibility As Boolean)**

##### Command Parameters:

- **aFilename As String:** name of the POI category.
- **aVisibility As Boolean:** whether to show or hide this POI type on the map.

**Supported Since:** TTN 2.0

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

**See Also:** AddPoi, DeleteClosestPoi, DeleteAllPoi, MoveClosestPoi

### 3.2.22. Sub *MoveClosestPoi*( *aFilename As String*, *aLongitude As Long*, *aLatitude As Long*, *aNewLongitude As Long*, *aNewLatitude As Long*)

#### Command Parameters:

- **aFilename As String:** name of the POI category.
- **aLongitude As Long:** old longitude specified in WGS84 format (in millionth of degrees).
- **aLatitude As Long:** old latitude specified in WGS84 format (in millionth of degrees).
- **aNewLongitude As Long:** new longitude specified in WGS84 format (in millionth of degrees).
- **aNewLatitude As Long:** new latitude specified in WGS84 format (in millionth of degrees).

**Supported Since:** TTN 1.51

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.

**See Also:** AddPoi, DeleteClosestPoi, DeleteAllPoi, MakeUserPoiVisible

### 3.2.23. Sub *Geocode*( *aCity As String*, *aStreet As String*, *aHouseNr As String*)

#### Command Parameters:

- **aCity As String:** city name.
- **aStreet As String:** street name.
- **aHouseNr As String:** house number.

**Supported Since:** TTN 1.51

Find the location specified by an address, given in the city name, street name and house number arguments, and return the co-ordinates of the found location and its complete city and street names in the corresponding read-only properties of the control. If the street name argument is empty, the function will try to find the center of the city specified by the city name argument. The house number is always ignored in versions of TomTom Navigator prior to 2.0. 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 geocode status property of the control.

**Property GeocodeStatus As Long:** Validity of the rest of the geocode properties, as explained below

**Property GeocodeLongitude As Long:** Longitude in millionth of degrees (WGS84 format)

**Property GeocodeLatitude As Long:** Latitude in millionth of degrees (WGS84 format)

**Property GeocodeCity As String:** Complete city name

**Property GeocodeStreet As String:** Complete street name

The geocode status property can contain the following flags:

- 8 – no location found at all, the rest of the geocode properties are invalid
- 4 – the given city name was ambiguous, random (possibly wrong) selection made
- 2 – the given street name was ambiguous, tried to make smart selection
- 1 – the house number was ignored, either because there was no house number data available or because the given house number was not found

If Geocode has never been called, GeocodeStatus will be 8, indicating that the rest of these properties don't contain valid information.

### 3. Communicating with TomTom Navigator from Visual Basic

#### Example:

The following example shows how to instruct TomTom Navigator to plan a route to a given address.

```
Sub NavigateToAddress
    On Error GoTo NavigateToAddress_Error
    Call TTNNav.Geocode("Zwanenburg", "Julianalaan", "12")
    If TTNNav.GeocodeStatus <> 8 Then
        Call TTNNav.NavigateToCoordinate(TTNNav.GeocodeLongitude, ↵
            TTNNav.GeocodeLatitude, ↵
            "Julianalaan, 12")
    Else
        MsgBox "Address not found"
    End If
NavigateToAddress_Exit:
    Exit Sub
NavigateToAddress_Error:
    MsgBox "Failed to navigate to address"
    Resume NavigateToAddress_Exit
End Sub
```

#### 3.2.24. Sub *GetDataSetInfo()*

**Supported Since:** TTN 2.0

Obtains information about the currently selected data set. The information is available after this call in the following read-only properties of the control:

**Property DataSetName As String:** name of the data set

**Property DataSetFlags As Long:** explained below

The data set flags property can contain the following flags:

- 1 – indicates that the house numbers are written before the street name in addresses
- 2 – indicates left-sided driving

#### 3.2.25. Sub *GetRouteInfo()*

**Supported Since:** TTN 2.0

Obtains information about the current route. The information is available after this call in the following read-only properties of the control:

**Property RouteStatus As Long:** validity of the rest of the fields, as explained below

**Property RouteDepartureLongitude As Long:** endpoint co-ordinates in millionth of degrees (WGS84 format)

**Property RouteDepartureLatitude As Long:** –

**Property RouteDestinationLongitude As Long:** –

**Property RouteDestinationLatitude As Long:** –

**Property RouteEnclosingLongitudeW As Long:** rectangle enclosing the route (co-ordinates in millionth of degrees, WGS84 format)

**Property RouteEnclosingLatitudeS As Long:** –

**Property RouteEnclosingLongitudeE As Long:** –

**Property RouteEnclosingLatitudeN As Long:** –

### 3. Communicating with TomTom Navigator from Visual Basic

**Property RouteLength As Long:** length of the route in meters

**Property RouteDuration As Long:** duration of the route in seconds

**Property RouteDistanceToDestination As Long:** distance from current position to destination in meters

**Property RouteTimeToDestination As Long:** time from current position to destination in seconds

The route status property can contain the following flags:

- 1 – the departure point has been set
- 2 – the destination point has been set
- 4 – in the process of calculating a route
- 8 – there is a valid route planned
- 16 – time and distance to destination are available

If GetRouteInfo has never been called, RouteStatus will be 0, indicating that the rest of these properties don't contain valid information.

#### **3.2.26. Sub UseGFFile( aFilename As String)**

**Command Parameters:**

- **aFilename As String:** full name of the GF file

**Supported Since:** TTN 2.0

Instruct Navigator to display a GF file, discarding the previous GF file (if any).

**See Also:** EnableGFRecord

#### **3.2.27. Sub EnableGFRecord( aRecordID As Long, aState As Boolean)**

**Command Parameters:**

- **aRecordID As Long:** unique ID of the record to be enabled or disabled
- **aState As Boolean:** whether to enable or disable the record

**Supported Since:** TTN 2.0

Enable or disable a record in the current GF file.

**See Also:** UseGFFile

# 4. Communicating with TomTom Navigator from C++

## 4.1. Introduction

This SDK allows third-party application developers to communicate with TomTom Navigator.

The SDK comes with a library that allows external applications to communicate with TomTom Navigator. The first version of TomTom Navigator to fully support this library is 1.42.

To use the library, link the executable C++ application with the library "TTNCom.lib" (this library is provided for ARM, MIPS and SH3), and include the header file TTNCom.h.

The library contains a class CTomTomNavigatorCom with static methods which can perform the communication request.

## 4.2. Supported API calls

The function calls currently supported are:

### 4.2.1. INT GetSdkVersionInfoV01( TNavVersionInfoV01& aVersionInfo)

**Command Parameters:**

- **TNavVersionInfoV01& aVersionInfo:** version struct to fill

**Returns:** 0 if successful, non-zero value otherwise.

**Supported Since:** TTN 1.42

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

### 4.2.2. INT GetNavigatorVersionInfoV01( TNavVersionInfoV01& aVersionInfo)

**Command Parameters:**

- **TNavVersionInfoV01& aVersionInfo:** version struct to fill

**Returns:** 0 if successful, non-zero value otherwise.

**Supported Since:** TTN 1.42

Return the version number information of TomTom Navigator (version string and build number).

### 4.2.3. INT BringNavigatorToForeground( )

**Returns:** 0 if successful, non-zero value otherwise.

**Supported Since:** TTN 1.42

Puts the application in the foreground

### 4.2.4. INT SwitchToNavigatorView( )

**Returns:** 0 if successful, non-zero value otherwise.

**Supported Since:** TTN 1.42

Switches to the navigation view. Useful because various API calls only work in Navigation view.

### 4.2.5. INT TryCloseCurrentOpenedDialogs( )

## 4. Communicating with TomTom Navigator from C++

**Returns:** 0 if successful, non-zero value otherwise.

**Supported Since:** TTN 1.42

Tries to close any opened dialogs on top. Useful for API calls like navigating to a location, which won't work when a dialog is on screen. Some dialogs cannot be closed, so this operation might fail.

### 4.2.6. *INT NavigateToCoordinate( long aLongitude, long aLatitude, LPCTSTR aName)*

**Command Parameters:**

- **long aLongitude:** longitude specified in WGS84 format (in millionth of degrees)
- **long aLatitude:** latitude specified in WGS84 format (in millionth of degrees)
- **LPCTSTR aName:** the name of the location being navigated to (shown in the route description)

**Returns:** 0 if successful, non-zero value otherwise.

**Supported Since:** TTN 1.42

Navigate to a location specified in WGS84 format (longitude and latitude in millionth of degrees).

**Example:**

The following example call instructs TomTom Navigator to plan a route to a point somewhere in Amsterdam, the Netherlands.

```
INT error = CTomTomNavigatorCom::NavigateToCoordinate(4860000, 52380000,
                                                    _T("Amsterdam"));

if (error)
{
    ::AfxMessageBox(TEXT("Navigation to Amsterdam failed"));
}
```

**See Also:** NavigateToFavorite, ShowCoordinateOnMap

### 4.2.7. *INT ClearFavorite( INT aIndex)*

**Command Parameters:**

- **INT aIndex:** index of favorite (base zero)

**Returns:** 0 if successful, non-zero value otherwise.

**Supported Since:** TTN 1.42

Clear a favorite.

**See Also:** SetFavoriteV01, GetFavoriteV01, NavigateToFavorite

### 4.2.8. *INT SetFavoriteV01( INT aIndex, TFavouriteRecV01 & aFavorite)*

**Command Parameters:**

- **INT aIndex:** index of favorite (base zero)
- **TFavouriteRecV01 & aFavorite:** favorite information

**Returns:** 0 if successful, non-zero value otherwise.

**Supported Since:** TTN 1.42

Set favorite information.

**See Also:** ClearFavorite, GetFavoriteV01, NavigateToFavorite

### 4.2.9. *INT GetFavoriteV01( INT aIndex, TFavouriteRecV01& aFavorite)*

## 4. Communicating with TomTom Navigator from C++

### Command Parameters:

- **INT aIndex:** index of favorite (base zero)
- **TFavouriteRecV01& aFavorite:** favorite information

**Returns:** 0 if successful, non-zero value otherwise.

**Supported Since:** TTN 1.42

Get favorite information.

**See Also:** ClearFavorite, SetFavoriteV01, NavigateToFavorite

### 4.2.10. INT NavigateToFavorite( INT aIndex)

#### Command Parameters:

- **INT aIndex:** index of favorite (base zero)

**Returns:** 0 if successful, non-zero value otherwise.

**Supported Since:** TTN 1.42

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

**See Also:** NavigateToCoordinate, ClearFavorite, SetFavoriteV01, GetFavoriteV01

### 4.2.11. INT ShowCoordinateOnMap( long aLongitude, long aLatitude)

#### Command Parameters:

- **long aLongitude:** longitude specified in WGS84 format (in millionth of degrees)
- **long aLatitude:** latitude specified in WGS84 format (in millionth of degrees)

**Returns:** 0 if successful, non-zero value otherwise.

**Supported Since:** TTN 1.42

Centers the map around the given coordinate (zoomed in), specified in WGS84 format (longitude and latitude in millionth of degrees).

#### Example:

```
INT error = CTomTomNavigatorCom::ShowCoordinateOnMap(4860000, 52380000);
if (error)
{
    ::AfxMessageBox(TEXT("Failed to center around a point in Amsterdam"));
}
```

**See Also:** NavigateToCoordinate

### 4.2.12. INT SendDirectCommand( EDirectCommands aCommand)

#### Command Parameters:

- **EDirectCommands aCommand:** command to execute

**Returns:** 0 if successful, non-zero value otherwise.

**Supported Since:** TTN 1.42

Various commands to change a setting or instruct TomTom Navigator to perform a certain task.

#### Example:

The following example invocation call demonstrates how to switch the map display to night colors:

```
INT error = CTomTomNavigatorCom::SendDirectCommand(KCommandNightColors);
if (error)
```

## 4. Communicating with TomTom Navigator from C++

```
{
    ::AfxMessageBox(TEXT("Failed to change to night view mode"));
}
```

### 4.2.13. INT GetCurrentPositionV01( EGpsStatus & aStatus, TTTnLocationV01 & aLocation)

#### Command Parameters:

- **EGpsStatus & aStatus:** at return, if no error occurred, contains an indication of the quality of the data received.
- **TTTnLocationV01 & aLocation:** struct that will be filled with the location information

**Returns:** 0 if successful, non-zero value otherwise.

**Supported Since:** TTN 1.42

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

#### Example:

It shows how the GetCurrentPosition call can be used to warn when the driver is driving too fast.

```
EGpsStatus status;
TTTnLocationV01 loc;
INT error = CTomTomNavigatorCom::GetCurrentPositionV01(status, loc);
if (error)
{
    ::AfxMessageBox(TEXT("Failed to get the current location"));
}
else
{
    if (status >= 0 && loc.iSpeedKmh > 120)
    {
        ::AfxMessageBox(TEXT("You are driving too fast!"));
    }
}
```

### 4.2.14. INT MakePoiVisible( EPoiTypeCode aPoiID, INT aVisibility)

#### Command Parameters:

- **EPoiTypeCode aPoiID:** POI unique type identifier
- **INT aVisibility:** whether to show or hide this POI type on the map

**Returns:** 0 if successful, non-zero value otherwise.

**Supported Since:** TTN 1.42

Enable or disable displaying a specific POI type.

### 4.2.15. INT AddPoi( LPCTSTR aFilename, long aLongitude, long aLatitude, LPCTSTR aName, LPCTSTR anId)

#### Command Parameters:

- **LPCTSTR aFilename:** name of the POI category
- **long aLongitude:** longitude specified in WGS84 format (in millionth of degrees)
- **long aLatitude:** latitude specified in WGS84 format (in millionth of degrees)
- **LPCTSTR aName:** name of the POI
- **LPCTSTR anId:** optional external identifier of the POI

## 4. Communicating with TomTom Navigator from C++

**Returns:** 0 if successful, non-zero value otherwise.

**Supported Since:** TTN 1.51

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.

### Example:

The following example invocation call demonstrates how to create a new point of interest:

```
INT error = CTomTomNavigatorCom::AddPoi(_T("ATM"), 4749620, 52374810,
                                         _T("Postbank"), _T(""));
if (error)
{
    ::AfxMessageBox(TEXT("Failed to create the point of interest"));
}
```

**See Also:** DeleteClosestPoi, DeleteAllPoi, MakeUserPoiVisible, MoveClosestPoi

### 4.2.16. INT DeleteClosestPoi( LPCTSTR aFilename, long aLongitude, long aLatitude)

#### Command Parameters:

- **LPCTSTR aFilename:** name of the POI category
- **long aLongitude:** longitude specified in WGS84 format (in millionth of degrees)
- **long aLatitude:** latitude specified in WGS84 format (in millionth of degrees)

**Returns:** 0 if successful, non-zero value otherwise.

**Supported Since:** TTN 1.51

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.

**See Also:** AddPoi, DeleteAllPoi, MakeUserPoiVisible, MoveClosestPoi

### 4.2.17. INT DeleteAllPoi( LPCTSTR aFilename)

#### Command Parameters:

- **LPCTSTR aFilename:** name of the POI category

**Returns:** 0 if successful, non-zero value otherwise.

**Supported Since:** TTN 1.51

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

**See Also:** AddPoi, DeleteClosestPoi, MakeUserPoiVisible, MoveClosestPoi

### 4.2.18. INT MakeUserPoiVisible( LPCTSTR aFilename, INT aVisibility)

#### Command Parameters:

- **LPCTSTR aFilename:** name of the POI category
- **INT aVisibility:** whether to show or hide this POI type on the map

**Returns:** 0 if successful, non-zero value otherwise.

**Supported Since:** TTN 2.0

## 4. Communicating with TomTom Navigator from C++

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

**See Also:** AddPoi, DeleteClosestPoi, DeleteAllPoi, MoveClosestPoi

### 4.2.19. *INT MoveClosestPoi( LPCTSTR aFilename, long aLongitude, long aLatitude, long aNewLongitude, long aNewLatitude)*

**Command Parameters:**

- **LPCTSTR aFilename:** name of the POI category
- **long aLongitude:** old longitude specified in WGS84 format (in millionth of degrees)
- **long aLatitude:** old latitude specified in WGS84 format (in millionth of degrees)
- **long aNewLongitude:** new longitude specified in WGS84 format (in millionth of degrees)
- **long aNewLatitude:** new latitude specified in WGS84 format (in millionth of degrees)

**Returns:** 0 if successful, non-zero value otherwise.

**Supported Since:** TTN 1.51

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.

**See Also:** AddPoi, DeleteClosestPoi, DeleteAllPoi, MakeUserPoiVisible

### 4.2.20. *INT GeocodeV01( LPCTSTR aCity, LPCTSTR aStreet, LPCTSTR aHouseNr, TGeocodeInfoV01& aGeocodeInfo)*

**Command Parameters:**

- **LPCTSTR aCity:** city name
- **LPCTSTR aStreet:** street name
- **LPCTSTR aHouseNr:** house number
- **TGeocodeInfoV01& aGeocodeInfo:** geocode information

**Returns:** 0 if successful, non-zero value otherwise.

**Supported Since:** TTN 1.51

Find the location specified by an address, given in the city name, street name and house number 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. The house number is always ignored in versions of TomTom Navigator prior to 2.0. 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.

**Example:**

The following example shows how to instruct TomTom Navigator to plan a route to a given address.

```
TGeocodeInfoV01 info;
INT error = CTomTomNavigatorCom::GeocodeV01(_T("Zwanenburg"),
                                             _T("Julianalaan"),
                                             _T("12"), info);

if (error)
{
    ::AfxMessageBox(TEXT("Geocode failed"));
}
else if (info.iStatus != CTomTomNavigatorCom::KGeocodeFailure)
```

#### 4. Communicating with TomTom Navigator from C++

```
{
    error = CTomTomNavigatorCom::NavigateToCoordinate(info.iLongitude,
                                                    info.iLatitude,
                                                    _T("Julianalaan, 12"));

    if (error)
    {
        ::AfxMessageBox(TEXT("Failed to navigate to address"));
    }
}
else
{
    ::AfxMessageBox(TEXT("Address not found"));
}
```

##### **4.2.21. INT GetDataSetInfoV01( TDataSetInfoV01& aDataSetInfo)**

###### **Command Parameters:**

- **TDataSetInfoV01& aDataSetInfo:** data set information

**Returns:** 0 if successful, non-zero value otherwise.

**Supported Since:** TTN 2.0

Get information about the currently selected data set.

##### **4.2.22. INT GetRouteInfoV01( TRouteInfoV01& aRouteInfo)**

###### **Command Parameters:**

- **TRouteInfoV01& aRouteInfo:** route information

**Returns:** 0 if successful, non-zero value otherwise.

**Supported Since:** TTN 2.0

Get information about the current route.

##### **4.2.23. INT UseGFFile( LPCTSTR aFilename)**

###### **Command Parameters:**

- **LPCTSTR aFilename:** full name of the GF file

**Returns:** 0 if successful, non-zero value otherwise.

**Supported Since:** TTN 2.0

Instruct Navigator to display a GF file, discarding the previous GF file (if any).

**See Also:** EnableGFRecord

##### **4.2.24. INT EnableGFRecord( long aRecordID, INT aState)**

###### **Command Parameters:**

- **long aRecordID:** unique ID of the record to be enabled or disabled
- **INT aState:** whether to enable or disable the record

**Returns:** 0 if successful, non-zero value otherwise.

**Supported Since:** TTN 2.0

Enable or disable a record in the current GF file.

**See Also:** UseGFFile

## 4.3. Structures and enumerations from the API

### 4.3.1. TNavVersionInfoV01 structure

The struct TNavVersionInfoV01 has the following fields:

```
char iVersion[16];  
INT iBuildNumber;
```

### 4.3.2. TFavouriteRecV01 structure

The favorites record has the following fields:

```
TCHAR iName[64];  
TCHAR iDescription[128];  
long iLongitude;  
long iLatitude;
```

### 4.3.3. EDirectCommands enumeration

EDirectCommands is an enumeration with the following items:

- KCommandSoundOn – turn spoken voice instructions on (works only when in navigation view)
- KCommandSoundOff – turn spoken voice instructions off (works only when in navigation view)
- KCommandNightColors – show map in night colors (works only when in navigation view)
- KCommandDayColors – show map in day colors (works only when in navigation view)
- KCommandShowMap – show map and instructions while driving (works only when in navigation view)
- KCommandHideMap – show "instructions only" view while driving (works only when in navigation view)
- KCommandRecalculate – recalculate a route (works only when in navigation view)
- KCommandShowPOI – show "points of interest" on the map
- KCommandHidePOI – hide "points of interest" on the map
- KCommandShowExits – show the next highway exit sign on screen (works only when in navigation view)
- KCommandHideExits – hide the next highway exit sign on screen (works only when in navigation view)
- KCommandDisableGuidance – disable active guidance support (works only when in navigation view)
- KCommandEnableGuidance – enable active guidance support (works only when in navigation view)
- KCommandShowCompass – show compass on screen (works only when in navigation view)
- KCommandHideCompass – hide compass (works only when in navigation view)
- KCommandGoHome – navigate to the "Home" favorite (works only when in navigation view)
- KCommandShowFavorites – show "Favorites" on the map
- KCommandHideFavorites – hide "Favorites" on the map

Versions 1.51 and later of TomTom Navigator support also the following direct commands:

- KCommandUnitsKm – set distance units to kilometers
- KCommandUnitsM – set distance units to miles
- KCommandAutoRecalcOn – turn on automatic recalculation of the route
- KCommandAutoRecalcOff – turn off automatic recalculation of the route
- KCommandAutoZoomOn – turn on automatic zooming
- KCommandAutoZoomOff – turn off automatic zooming

### 4.3.4. TTTnLocationV01 structure

## 4. Communicating with TomTom Navigator from C++

The TTTnLocationV01 struct contains the following fields:

- long iLongitude – Longitude and latitude in millionth of degrees (WGS84 format)
- long iLatitude –
- int iDirection – Direction in degrees: zero is north, continuing in clockwise direction
- int iSpeedKmh – current driving speed in kilometers per hour

### 4.3.5. EGpsStatus enumeration

Status specifies if the information in this record is recent or not. Status<0 means some error occurred (-1 is general), Status>=0 means the data is valid.

Currently allowed values are:

```
KErrGeneral  
KErrReceiving
```

### 4.3.6. TGeocodeInfoV01 structure

The TGeocodeInfoV01 struct contains the following fields:

- int iStatus – Validity of the rest of the fields, as explained below
- long iLongitude – Longitude and latitude in millionth of degrees (WGS84 format)
- long iLatitude –
- TCHAR iCityName[128] – Complete city name
- TCHAR iStreetName[128] – Complete street name

The status field can contain the following flags:

- KGeocodeFailure – no location found at all, the rest of the fields are invalid
- KGeocodeCityAmbiguous – the given city name was ambiguous, random (possibly wrong) selection made
- KGeocodeStreetAmbiguous – the given street name was ambiguous, tried to make smart selection
- KGeocodeHouseNrIgnored – the house number was ignored, either because there was no house number data available or because the given house number was not found

### 4.3.7. EPoiTypeCode enumeration

The Point of Interest types that are supported are:

- KPoiTypeGovernmentOffice
- KPoiTypeMountainPeak
- KPoiTypeOpenParking
- KPoiTypeParkingGarage
- KPoiTypePetrolStation
- KPoiTypeRailwayStation
- KPoiTypeRestArea
- KPoiTypeAirport
- KPoiTypeCarDealer
- KPoiTypeCasino
- KPoiTypeChurch
- KPoiTypeCinema
- KPoiTypeCityCenter
- KPoiTypeCompany
- KPoiTypeConcertHall
- KPoiTypeCourthouse
- KPoiTypeCulturalCenter

#### 4. Communicating with TomTom Navigator from C++

- KPoiTypeExhibitionCenter
- KPoiTypeFerryTerminal
- KPoiTypeFrontierCrossing
- KPoiTypeGolfCourse
- KPoiTypeHospitalPolyclinic
- KPoiTypeHotelMotel
- KPoiTypeTouristAttraction
- KPoiTypeMountainPass
- KPoiTypeMuseum
- KPoiTypeOpera
- KPoiTypePlaceofWorship
- KPoiTypePostOffice
- KPoiTypeRentCarFacility
- KPoiTypeRentCarParking
- KPoiTypeRestaurant
- KPoiTypeShop
- KPoiTypeShoppingCenter
- KPoiTypeStadium
- KPoiTypeTheatre
- KPoiTypeTouristInformationOffice
- KPoiTypeZoo
- KPoiTypeSportsCentre
- KPoiTypePoliceStation
- KPoiTypeEmbassy
- KPoiTypeCollegeUniversity
- KPoiTypeCashDispenser
- KPoiTypeBeach
- KPoiTypeIceSkatingRing
- KPoiTypeTennisCourt
- KPoiTypeWaterSport
- KPoiTypeDoctor
- KPoiTypeDentist
- KPoiTypeVeterinarian
- KPoiTypeNightlife
- KPoiTypeAmusementPark
- KPoiTypeLibrary

Versions 2.0 and later of TomTom Navigator also support the following POI types:

- KPoiTypeCarRepairFacility
- KPoiTypePharmacy
- KPoiTypeScenicPanoramicView
- KPoiTypeSwimmingPool
- KPoiTypeWinery
- KPoiTypeCampingGround
- KPoiTypeParkRecreationArea
- KPoiTypeConventionCentre
- KPoiTypeLeisureCentre
- KPoiTypeYachtBasin

#### 4.3.8. TDataSetInfoV01 structure

The TDataSetInfoV01 struct contains the following fields:

- TCHAR iDataSetName[128] – name of the data set
- unsigned long iFlags – explained below

The flags field can contain the following flags:

- KHouseNrPrefixed – indicates that the house numbers are written before the street name in addresses
- KLeftSideDriving – indicates left-sided driving

#### 4.3.9. TRouteInfoV01 structure

The TRouteInfoV01 struct contains the following fields:

- int iStatus – validity of the rest of the fields, as explained below
- long iDeparture.iLongitude – endpoint co-ordinates in millionth of degrees (WGS84 format)
- long iDeparture.iLatitude –
- long iDestination.iLongitude –
- long iDestination.iLatitude –
- long iEnclosingLongitudeW – rectangle enclosing the route (co-ordinates in millionth of degrees, WGS84 format)
- long iEnclosingLatitudeS –
- long iEnclosingLongitudeE –
- long iEnclosingLatitudeN –
- long iLength – length of the route in meters
- long iDuration – duration of the route in seconds
- long iDistanceToDestination – distance from current position to destination in meters
- long iTimeToDestination – time from current position to destination in seconds

The status field can contain the following flags:

- KDepartureSet – the departure point has been set
- KDestinationSet – the destination point has been set
- KCalculatingRoute – in the process of calculating a route
- KRoutePlanned – there is a valid route planned
- KLeftToDestinationAvailable – time and distance to destination are available

# 5. Graphic Information Files

This section explains how to create graphic information files, also called GF files.

## 5.1. About GF Files

You can provide "graphic information" to TomTom Navigator in the form of a file consisting of variable-length records describing graphic shapes and elements (e.g. representing dynamic situations such as traffic conditions).

TomTom Navigator will load and maintain such a file as soon as you pass it the path name of the file, using the routine UseGFFile. When this routine returns, TomTom Navigator will have discarded any previous files, and will have loaded the new file into memory. The caller is thus free to delete or change the file after having called this routine.

Calling UseGFFile with an empty name, or with the name of a non-existent or empty file, in effect "clears" any previously loaded data.

For any change in the situation (e.g. the traffic situation) either call EnableGFRecord with the unique id of some record you want enabled or disabled, or call UseGFFile with the name of a new or adjusted file (or with the empty string as filename, which effectively deletes all the current records).

### Caveats:

On general principles, it is recommended that you "refresh" files relating to traffic information every 15 minutes even if there are no changes. On the other hand, there are obvious disadvantages to refreshing files too often, e.g. every few seconds, in terms of power drain, application speed degradation etc.

### Tips:

For some applications, it may be useful to switch between a limited set of pre-defined, fixed files. For some applications, it may be useful to use a single pre-defined file, of which records are "enabled" or "disabled" by the simple action of stamping them with an "end-of-validity" in the past resp. the future.

## 5.2. GF File Format

### 5.2.1. Basics of the File Format

The following rules apply to all the records of a GF file.

- Multi-byte values are stored least-significant-byte first.
- Timestamps are specified as (4-byte) unsigned longs, representing seconds since midnight 1/1/1970.
- Coordinates are stored as (4-byte) signed longs, representing WGS84-coordinates in degrees multiplied by 100.000. The coordinates of a point are stored in the order: x, y.
- Strings (including filenames) are stored in ASCII format, are zero-terminated, and contain at most 254 non-zero characters.
- Rectangles must be "normalized", i.e. the four co-ordinates are in the following sequence: minimum-x, minimum-y, maximum-x, maximum-y.
- Every record must have the same 8-byte "header" and must be a multiple of 4 bytes in length.

The "header" must be as follows:

- |         |   |
|---------|---|
| 1 byte  | Type of this record, which is a value between 0 and 127.<br>When the most significant bit of this byte is set ( $\geq 128$ ),<br>it indicates that the record is disabled |
| 3 bytes | Length of this record as a number of 4-byte words,  |

## 5. Graphic Information Files

EXCLUDING the 8-byte header  
4 bytes Record ID (for future unique manipulation of this record)

### 5.2.2. Supported Record Types

**TIMESTAMP (type 5):** Specifies an "end-of-validity" for all the records that follow it (i.e. until the next timestamp record)

8 BYTES HEADER  
4 bytes End-of-validity timestamp  
4 bytes Number of bytes (EXcluding this record itself) that can be skipped immediately when end-of-validity is in the past (note: this value can be set to 0 if you're lazy)

NOTE: Records NOT preceded by an end-of-validity are "permanently valid." Even so, it is highly recommended that you consider the issue of validity periods, and always make sure that the very first record of the file specifies the end-of-validity for the whole file.

**SKIPPER (type 6):** Specifies that the coming X bytes relate to a certain rectangle

8 BYTES HEADER  
16 bytes Normalized coordinate rectangle of the area  
4 bytes Number of bytes (EXcluding this record itself) that can be skipped if rectangle doesn't overlap the current screen

NOTE: It is highly recommended, in the interest of processing speed, that you always provide a skipper record that specifies the surrounding-rectangle for all the rectangle-dependent records in the whole file together.

**IGNORE (type 0):** Records with type 0 are completely ignored by the application.

8 BYTES HEADER  
X bytes Ignored content

NOTE: Uses of this type of record range from permanent deletion (rather than de-activation) of records to watermarking copyright information into your files. However, please be aware that these records still consume memory, and also cost a (very small) amount of time to ignore.

**LINE (type 1):**

8 BYTES HEADER  
8 bytes Start coordinates of the line  
8 bytes End coordinates of the line  
4 bytes Line type (0=default)  
4 bytes Red/Green/Blue color code of line

NOTE: The line type is the combination of a line thickness and a line pattern. The line thickness codes are:

- 0 – width of motorway 2
- 1 – width of motorway 1
- 2 – width of major road 4
- 3 – width of major road 2
- 4 – width of major road 1
- 5 – width of secondary road 4
- 6 – width of secondary road 2
- 7 – width of secondary road 1
- 8 – width of local road 4
- 9 – width of local road 3

## 5. Graphic Information Files

- 10 – width of local road 2
- 11 – width of local road 1
- 12 – width of other road 4
- 13 – width of other road 3
- 14 – width of other road 2
- 15 – width of other road 1

The line pattern codes are:

- 0 – solid
- 16 – dashed
- 32 – dotted
- 48 – dot dash
- 64 – dot dot dash

The line thickness and line pattern codes can be combined using arithmetic addition or bit-wise OR.

### **POLYLINE (type 2):**

8 BYTES	HEADER
16 bytes	Normalized coordinate rectangle of the area around polyline
4 bytes	Line type (0=default)
4 bytes	Red/Green/Blue color code of lines
4 bytes	N = number of co-ordinates (should be 2 or more)
N*8 bytes	x/y-coordinates of the poly-line

NOTE: The line type field can contain the same codes as in the LINE (type 1) records.

### **WARNING-ICON (type 3):**

8 BYTES	HEADER
16 bytes	Normalized rectangular area
4 bytes	Flags: 0x0001 – if set, icon is ALWAYS shown. If not set, icon is only shown if the current GPS position is inside the specified rectangle
4 bytes	First parameter of the message to be sent to the callback window
4 bytes	Second parameter of the message to be sent to the callback window
1 byte	W = length of the name of the callback window (INcluding zero-termination)
1 byte	M = length of the name of the message to be sent to the callback window (INcluding zero-termination)
1 byte	L = length of the full filename of the icon to display (INcluding zero-termination)
1 byte	K = length of the full filename of the bitmap mask for the icon (INcluding zero-termination)
W bytes	Zero-terminated name of the callback window
M bytes	Zero-terminated name of the message to be sent to the callback window
L bytes	Zero-terminated filename of a .BMP file (a 48x48, 16-color or 256-color bitmap to be used as icon)
K bytes	Zero-terminated filename of a .BMP file (a 48x48 monochrome bitmap to be used as mask)
? bytes	Zero-bytes, to make this record a multiple of 4 bytes long

## 5. *Graphic Information Files*

NOTE: When the user taps on a warning icon, a message will be sent to the specified callback window with the given parameters. The callback window is the top-level window with the given name. On the MS Pocket PC, the message sent is the registered message with the given name, obtained by using the Windows OS function "RegisterWindowMessage", and the two message parameters correspond to the WPARAM and LPARAM fields of the message structure, respectively.

NOTE: TomTom Navigator will never show more than 3 warning icons. No matter how many icons match the specified criteria, only the first three will be displayed and activated. Please note that each string needs to be less than 256 characters, and this INCLUDES the zero-terminator.

## 6. Deployment

To deploy a VB or C++ application that makes use of the TomTom Navigator SDK, the appropriate CAB file from the directory

```
sdk\TTNCom\install\sdk\pocketpc\dll
```

needs to be installed on the device. A CAB file for ARM, MIPS and SH3 devices is provided (note that all Pocket PC 2002 devices use the ARM processor).

The CAB files provided are:

- TTNControl.ARM.cab (for the ARM CPU)
- TTNControl.MIPS.cab (for the MIPS CPU)
- TTNControl.SH3.cab. (for the SH3 CPU)

The CAB file will install the components

- TTNCom.dll
- TTNControl.ocx

and set settings in the registry to configure the SDK components.

Please be warned that the installation must take place in the default install directory. If the file TTNCom.dll is not installed in the \Windows\ directory, it will not be accessible. Also, using the .CAB file will ensure that TTNControl.ocx is properly registered with the Windows CE registry.

# 7. External GPS Drivers

This section explains how to feed TomTom Navigator with your own GPS information (position, speed, direction).

## 7.1. About External GPS Drivers

You can provide your own GPS positional information to TomTom Navigator. Whilst the GPS support application for TomTom Navigator allows you to connect to a GPS feed through the built-in COM ports, certain applications require a feed which is delivered through a different source, such as a TCP/IP connection. The mechanism described here also allows multi-plexed GPS/GSM/Telemetry data delivered over a built-in COM port to be demulti-plexed by you – the GPS feed can then be passed on to TomTom Navigator. Many blackboxes apply this mechanism.

When providing your own GPS feed, you may supply NMEA 0183 V2 or SiRF binary data formats.

You must at least supply UTC, Longitude, Latitude, Direction, Speed and HDOP (horizontal dilution of precision).

For NMEA 0183 V2 the RMC, GLL, GGA, GSA, GSV and VTG verbs are recognised.

**Caveats:** The feed works in one direction only – you will never receive any commands (such as GPS reset) from TomTom Navigator. The "Reset GPS for this location" feature in TomTom Navigator will have no effect, and you will need to implement such features yourself.

## 7.2. Providing GPS information

### 7.2.1. Configuring TomTom Navigator

In TomTom Navigator, go to the GPS information screen and select the "GPS" tab. Choose either the "NMEA 0183v2 Driver" or the "SiRF Driver" as your GPS device. The usual COM port drop down list will only allow you to select "Ext Driver".

The driver may be selected programmatically by setting the following registry keys (TomTom Navigator and the GPS support application should not be running on the device):

For an NMEA 0183 V2 feed:

```
[HKEY_LOCAL_MACHINE\SOFTWARE\TomTom\GPS Engine]
"Enabled"=dword:00000001
"Device"="NMEA 0183v2 Driver"
"Option"="Ext Driver"
```

For a SiRF Binary feed:

```
[HKEY_LOCAL_MACHINE\SOFTWARE\TomTom\GPS Engine]
"Enabled"=dword:00000001
"Device"="SiRF Driver"
"Option"="Ext Driver"
```

**Caveats:** Changing any other registry keys may render TomTom Navigator unusable.

### 7.2.2. Providing your own feed

TomTom Navigator installs a Control Panel applet in the \Windows folder on the Pocket PC. It will show up in the System tab of the Settings option on the main Start menu and pops up the GPS support application.

## 7. External GPS Drivers

The applet also exports these three functions:

```
DWORD AttachToTomTomGPSEngine();
VOID DetachFromTomTomGPSEngine(DWORD aHandle);
BOOL CopyDataToTomTomGPSEngine(DWORD aHandle,
                                DWORD aNumberOfBytes,
                                LPBYTE aBuffer);
```

To call these functions, include "ttgpsextdriver.h" in your C++ project, provided in the \Sdk\TTNCom\include\ folder. This header file contains all the necessary declarations.

You will need to dynamically load the Control Panel applet, located in the \Windows\ttgpscpl.dll DLL on the device. After successfully loading the DLL, use the Windows API function GetProcAddress to find the proper function addresses.

An example is provided with the SDK in the \Sdk\TTNCom\examples\extdriver folder.

### 7.2.3. **DWORD AttachToTomTomGPSEngine( )**

**Returns:** An opaque handle is returned if successful, otherwise 0 (zero)

**Supported Since:** GPS 2.00

Creates a session between your application and the TomTom Navigator GPS support application. A session must exist. The returned handle must be passed in to the other functions. Warning: Although it is possible to create multiple simultaneous sessions, this is not recommended, as the receiver of the feed will only recognise one feed stream.

### 7.2.4. **VOID DetachFromTomTomGPSEngine( DWORD aHandle)**

**Command Parameters:**

- **DWORD aHandle:** the session handle returned by a successful call to AttachToTomTomGPSEngine().

**Supported Since:** GPS 2.00

Terminates a session between your application and the TomTom Navigator GPS support application.

### 7.2.5. **BOOL CopyDataToTomTomGPSEngine( DWORD aHandle, DWORD aNumberOfBytes, LPBYTE aBuffer)**

**Command Parameters:**

- **DWORD aHandle:** the session handle returned by a successful call to AttachToTomTomGPSEngine().
- **DWORD aNumberOfBytes:** the number of bytes that you intend to feed to the TomTom Navigator GPS support application.
- **LPBYTE aBuffer:** a pointer to a buffer of at least aNumberOfBytes bytes containing the data to be fed.

**Returns:** TRUE (non-zero) if successful, FALSE (zero) otherwise

**Supported Since:** GPS 2.00

Synchronously passes your GPS information. Pass the data exactly as it is received from the GPS receiver, including all field and record delimiters. It is not necessary to pass data on exact record boundaries.