OmTapi 1.2.5.0

Manual

Contents

Disclaimer:

THIS SOFTWARE IS NOT DESIGNED FOR THE ILLEGAL AND UNAUTHORIZED DUPLICATION OF COPYRIGHTED MATERIALS. THE AUTHOR REMINDS YOU THAT UNAUTHORIZED DUPLICATION OF COPYRIGHTED MATERIAL IS LIABLE TO CRIMINAL PROSECUTION. IF YOU ARE NOT SURE OF YOUR RIGHTS, PLEASE CONTACT YOUR LOCAL LEGAL ADVISOR.

THIS SOFTWARE IS PROVIDED "AS IS". THE USER IS STRICTLY LIABLE FOR USING IT AND SHALL FACE ALL CONSEQUENCES THAT MAY RESULT OF AUTHORIZED OR UNAUTHORIZED USE.

All trademarks, product names and company names or logos cited or used herein are the property of their respective owners.

OmTapi License Agreement

IMPORTANT - READ CAREFULLY:

This End-User License Agreement is a legal agreement between you (either an individual or a single entity) and SEO Softworks for the software product identified above, which includes computer software and may include associated media, printed materials, and "online" or electronic documentation ("SOFTWARE PRODUCT"). By installing, copying, or otherwise using the SOFTWARE PRODUCT, you agree to be bound by the terms of this LICENSE AGREEMENT. If you do not agree to the terms of this LICENSE AGREEMENT, do not install or use the SOFTWARE PRODUCT. Please do NOT order a registration key, if you do not agree to this license agreement.

License conditions

The SOFTWARE PRODUCT is protected by copyright laws and international copyright agreements, as well as other intellectual property laws and agreements. No part of the software or the manual may be multiplied, disseminated or processed in any way without the written consent of SEO Softworks. Violations of these conditions will be prosecuted in every case.

The use of the software is done at your own risk. The manufacturer and developer accepts no liability for any damages, either as direct or indirect consequence of the use or abuse of this product or software by the end-user. The SOFTWARE PRODUCT is being licensed and not sold.

All rights reserved. Software Copyright © 2003 - 2017 SEO Softworks, Germany.

SOFTWARE PRODUCT LICENSE

GRANT OF LICENSE. This LICENSE AGREEMENT grants you the following simple non-exclusive rights:

Applications Software / Development Version

For each development version license you purchase, you may install and use one copy of the SOFTWARE PRODUCT, or any prior version for the same operating system, on a single computer.

Applications Software / Runtime Version

Runtime licenses have to be purchased separately. For each copy of the complete application, consisting of an Omnis Runtime, at least one Omnis library and the runtime version oft the SOFTWARE PRODUCT a separate Runtime license is necessary, except you purchased an Application or Enterprise Runtime license.

If the complete application is running on a web server or server serving multiple users, either a separate Runtime license must be purchased for each user accessing the application or you purchase an Application Runtime License

Application Runtime License

If you purchase an Application Runtime License, you are allowed to distribute the SOFTWARE PRODUCT together with each copy of the complete application.

Enterprise Runtime License

If you purchase an Enterprise Runtime License, you are allowed to distribute the SOFTWARE PRODUCT together with each copy of all complete applications, you are distributing.

Storage/Network Use

You may also store or install a copy of the SOFTWARE PRODUCT on a storage device, such as a network server, used only to install or run the SOFTWARE PRODUCT on your other computers over an internal network; however, you must acquire and dedicate a license for each separate computer on which the SOFTWARE PRODUCT is installed or run from the storage device. A license for the SOFTWARE PRODUCT may not be shared or used concurrently on different computers.

OEM License

If you obtained this license agreement together with a hardware product, you are entitled to use this SOFTWARE PRODUCT as a part of the Hard- and Software package according to this license agreement.

DESCRIPTION OF OTHER RIGHTS AND LIMITATIONS

SOFTWARE PRODUCT as Demo Version

Even if the SOFTWARE PRODUCT is a demonstration version, it is protected by copyright laws and international copyright agreements, as well as other intellectual property laws and agreements.

Limitations on Reverse Engineering, Decompilation, and Disassembly
You may not reverse engineer, decompile, or disassemble the SOFTWARE PRODUCT, except and only to the extent that such activity is expressly permitted by applicable law notwithstanding this limitation. This also applies to Demo versions.

Separation of Components
The SOFTWARE PRODUCT is licensed as a single product. Its component parts may not be separated for use on more than one computer. This also applies to demo versions.

Rental
You may not rent, lease, or lend the SOFTWARE PRODUCT. This also applies to demo versions.

Software Transfer
You may not transfer the rights of this end-user LICENSE AGREEMENT.

Data Transmission
You may not transmit the SOFTWARE PRODUCT, or parts of it, without the prior permission of SEO Softworks.

Upgr**a**des
If the SOFTWARE PRODUCT is labeled as an upgrade, you must be properly licensed to use the SOFTWARE PRODUCT identified by SEO Softworks as being eligible for the upgrade in order to use the SOFTWARE PRODUCT. A SOFTWARE PRODUCT labeled as an upgrade replaces and/or supplements the product that forms the basis for your eligibility for the upgrade. You may use the resulting upgraded product only in accordance with the terms of this LICENSE AGREEMENT. If the SOFTWARE PRODUCT is an upgrade of a component of a package of software programs that you licensed as a single product, the SOFTWARE PRODUCT may be used and transferred only as part of that single product package and may not be separated for use on more than one computer.

What is TAPI?

The so called TAPI is an abbreviation for Telephone Application Programming Interface or Telephone API which was developed together by Intel an Microsoft.

It allows applications to access telephone lines, to listen to incoming calls and to make telephone calls from the computer.

The greatest advantage of TAPI is, that it abstracts from the particular hardware of the modem, telephone, ISDN card or PBX (Private Branch eXchange) you are using.

To be able to use TAPI you must have a TAPI driver installed, which is compatible with your telephone device.

TAPI provides a complete set of commands and methods which can be used to access the particular telephone device.

And not only this. The current version 3.0 of TAPI also supports IP telephony, standards-based H.323 conferencing and IP multicast conferencing.

Why an Omnis external for TAPI

Like a developer who's name I unfortunately don't remember, said: "TAPI is a great example, how a simple thing can be made incredibly complicated." This is true especially seen from a developer's point of view.

Even in C/C++ it's not quite easy to deal with TAPI. But in Omnis Studio and much more Omnis 7.x it's more or less impossible to get TAPI working.

The reason must be seen in the way of interface TAPI provides to the user. It is a huge set of Functions and methods which are encapsulated within a Windows DLL called tapi32.dll.

Although most DLLs and their functions (the easy ones) can be called from Omnis, the TAPI DLL is an exception because it uses input/output parameters and data structures which can't be accessed from within Omnis because the use C-type variable structures or need callback functions in the application.

So we decided to give Omnis developers easy and uncomplicated to the telephony API by building OmTAPI, a classic style Omnis External which can be used with Omnis Classic and Omnis Studio.

It’s a wrapper for all that complicated stuff within TAPI and gives you a set of 14 external commands, which should be sufficient, to do all the common stuff, your customers probably ask you to implement in your application.

What OmTAPI does

In short terms, the external OmTAPI commands allow you to:

- Initialize the TAPI interface

- Get a list of all TAPI devices installed on a particular system
- Retrieve information about each device/line
- Select a device/line
- Configure a device/line
- Configure the system dialing parameters
- Make a call on the device/line selected
- Monitor the TAPI status messages
- Listen to incoming calls on the selected device/line
- Retrieve the telephone number of the caller, to identify him by a database search
  (if the device/line supports this feature and if the caller number is transferred to you)

Installation

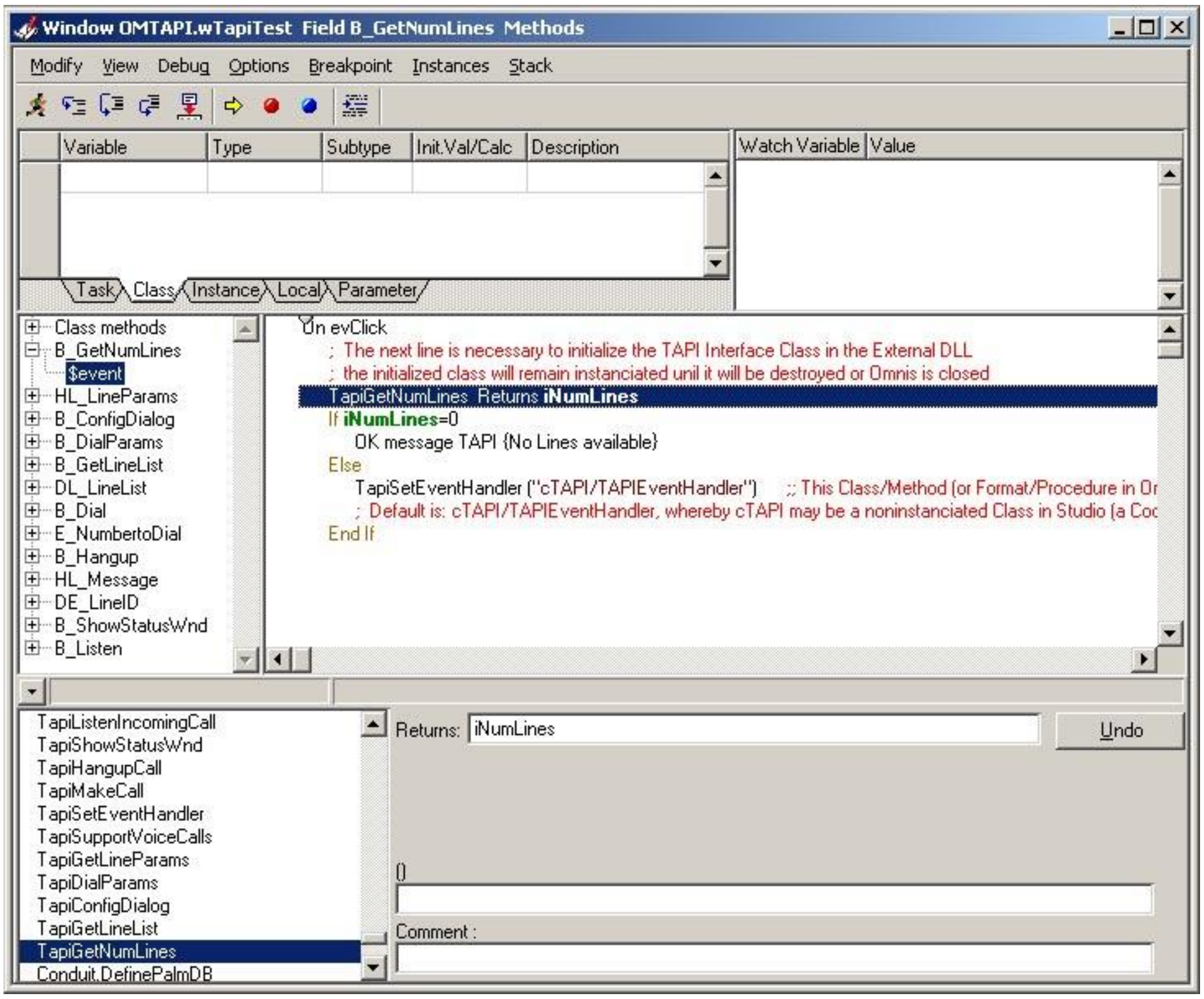
Just extract the ZIP archive provided an copy the two DLL's

- omtapi.dll
- evtapi.dll

in the EXTERNAL folder located in the program directory of your Omnis installation.

The license number must be saved to a file called omtapi.txt which resides directly in the Omnis application directory where omnis.exe is located also.

After restarting Omnis, the TAPI…-commands should show up in the External commands group.


Window OMTAPI.wTapiTest Field B_GetNumLines Methods
Modify View Debug Options Breakpoint Instances Stack
Variable Type Subtype Init.Val/Calc Description
Watch Variable Value
Task Class Instance Local Parameter
Returns: iNumLines
Undo
()
Comment :

Class methods
B_GetNumLines
$event
HL_LineParams
B_ConfigDialog
B_DialParams
B_GetLineList
DL_LineList
B_Dial
E_NumbertoDial
B_Hangup
HL_Message
DE_LineID
B_ShowStatusWnd
B_Listen

```
On evClick
  ;  The next line is necessary to initialize the TAPI Interface Class in the External DLL
  ;  the initialized class will remain instanciated unil it will be destroyed or Omnis is closed
  TapiGetNumLines  Returns iNumLines
  If iNumLines=0
     OK message TAPI {No Lines available}
  Else
     TapiSetEventHandler ("cTAPI/TAPIEventHandler")     ;; This Class/Method (or Format/Procedure in Or
     ;  Default is: cTAPI/TAPIEventHandler, whereby cTAPI may be a noninstanciated Class in Studio (a Coc
  End If
```

TapiListenIncomingCall
TapiShowStatusWnd
TapiHangupCall
TapiMakeCall
TapiSetEventHandler
TapiSupportVoiceCalls
TapiGetLineParams
TapiDialParams
TapiConfigDialog
TapiGetLineList
TapiGetNumLines
Conduit.DefinePalmDB

OmTAPI Commands

The OmTAPI external consists of a set of 14 external command, which give you access to the most common functions of the TAPI interface. The sample application OMTAPI.LBS provided with OmTapi is fully documented shows the use of each command.

Important:
OmTAPI uses the hash variables #60 and #S5 for internal purposes. So don't use #60 and #S5 for storing permanent values in your application. They will be changed.

TapiGetNumLines()

Parameters:	None
Return Value:	Integer, Number of Lines useable for TAPI communication

Description
TapiGetNumLines returns the number of TAPI devices installed in the users system in an integer value. If the return code is zero, there are no TAPI devices installed.

Important:
TapiGetNumLines also initializes OmTAPI, so it should **ALWAYS** be the first command, when working with OmTAPI.

Example:
```
;  The next line is necessary to initialize the
;  TAPI Interface class in the External DLL.
;  The initialized class will remain instanciated until it
;  will be destroyed or Omnis is closed
TapiGetNumLines  Returns iNumLines
If iNumLines=0
     OK message TAPI {No Lines available}
Else
     TapiSetEventHandler ("cTAPI/TAPIEventHandler")
     ; This Class/Method (or Format/Procedure in Omnis Classic)
     ; will be called by OmTAPI to signal certain events
     ; Default is: cTAPI/TAPIEventHandler, whereby cTAPI may
     ; be a noninstanciated Class in Studio (a Code Class)
     ; or a Format (Window or Menu) in Omnis Classic
End If
```

TapiGetLineList(lList)

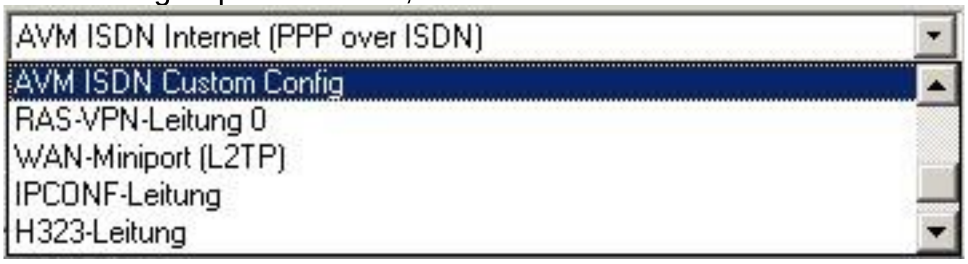
Parameters:	List variable
Return Value:	Integer

Description
TapiGetLineList() returns an Omnis list which contains the number and the name of each usable TAPI device. When TapiGetLineList() succeeds, it returns kTrue / 1. In case of an error, zero / kFalse is returned.
The list variable is to be defined before calling TapiGetLineList. The first column is an integer, the second a character field with a length of at least 20 characters. If the line name retuned by OmTapi is longer, than the limit defined, the name will be truncated, no error occurs.

Important:
Numbering in the list starts with one to correspond with the numbering of the current line (#L resp. mylist.$line) in Omnis. Internally TAPI starts counting with zero.

Example:
```
Do iTapiLineList.$define(lLineNr,lLineName)
TapiGetLineList (iTapiLineList) Returns lRet
If lRet>0
     Do iTapiLineList.$line.$assign(1)
     Do $cinst.$objs.DL_LineList.$redraw()
End If
```

After calling TapiGetLineList, the list of TAPI devices could look like this:



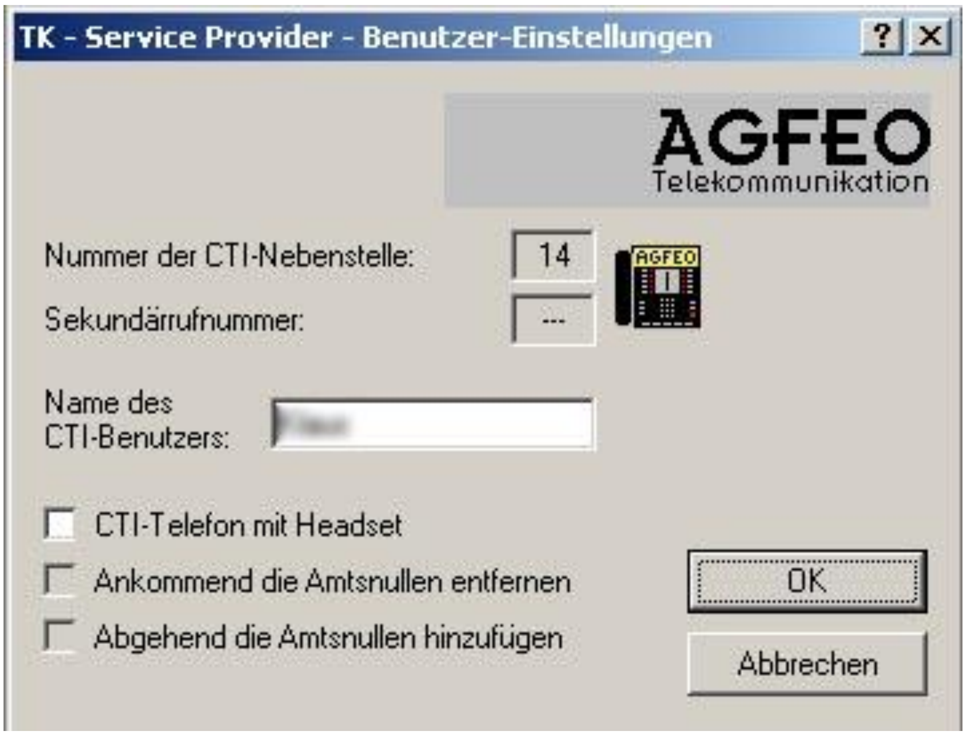
```
AVM ISDN Internet (PPP over ISDN)
AVM ISDN Custom Config
RAS-VPN-Leitung 0
WAN-Miniport (L2TP)
IPCONF-Leitung
H323-Leitung
```

TapiConfigDialog(iNumLine)

Parameters:	Number of line, integer
Return Value:	Integer

Description
TapiConfigDialog opens the configuration dialog of the selected TAPI line, if available. No each TAPI device – like the RAS VPN line - is configurable and so not each TAPI has a configuration dialog. This configuration dialog is "built in" in the particular TAPI driver. Its not an Omnis dialog.
TapiConfigDialog returns kTrue if successful, also if there is no configuration dialog. If no configuration dialog is available, an OK Message Box appears.

When using OmTAPI together with a PBX, which probably is the most frequent use of this command, this is the place where you tell OmTAPI to use the telephone next to you on your desk.


TK - Service Provider - Benutzer-Einstellungen

Nummer der CTI-Nebenstelle: 14

Sekundärrufnummer: ---

Name des CTI-Benutzers:

☐ CTI-Telefon mit Headset

☐ Ankommend die Amtsnullen entfernen

☐ Abgehend die Amtsnullen hinzufügen

The screenshot above shows the configuration dialog of an AGFEO AS 40 ISDN PBX, which is very common in Europe. Here the internal number (14) is combined with a nickname for each user connected to the PBX.

Example:
```
TapiConfigDialog (iLineNum) Returns lRet
```

TapiDialParams(iNumLine, [cExampleNumber])

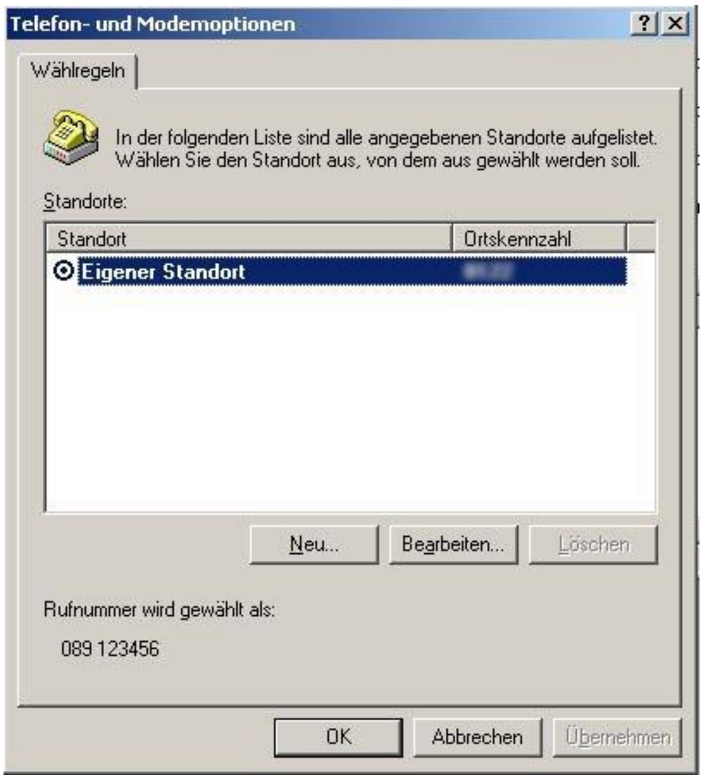
Parameters:	Number of line, integer, [Number to test, character [30]]
	Number to dial, character [32]
Return Value:	Integer

Description

This command opens the windows built in dialog for telephone and modem options. for the selected line. It returns kTrue if successful, otherwise kFalse.
Here you can determine the dialing rules, for example the digit to prefix, to get a public or long distance line.
As an optional second parameter, you can pass a string containing a telephone number, to show how TAPI will dial it (with leading zeros etc.) The number passed will be displayed left at the bottom of the configuration dialog.

TapiDialParams returns kTrue if successful, otherwise zero.

Example:
```
TapiDialParams (iLineNum,cNumbertoTest) Returns lRet
```


Telefon- und Modemoptionen

Wählregeln

In der folgenden Liste sind alle angegebenen Standorte aufgelistet. Wählen Sie den Standort aus, von dem aus gewählt werden soll.

Standorte:

Standort	Ortskennzahl
⦿ Eigener Standort	[redacted]

Rufnummer wird gewählt als:

089 123456

TapiGetLineParams(iLineNum, lList)

Parameters:	Number of Line, List with results
Return Value:	Integer

Description
TapiGetLineParams returns a list given as the second parameter, which contains a number of selected properties for the selected line. The list has to be defined before calling TapiGetLineParams. The first column is character value with a length of 40 characters, the second one a character value with a length of 100 characters.

Important:
TapiGetLineParams also selects the line for some commands i.e. *TapiSupportVoiceCalls,* which do not require a line number as parameter.

The TAPI line number shown in the screen shot below is always the line number used in Omnis minus one because TAPI internally starts counting with zero.

TapiGetLineParams returns true if successful, otherwise zero.

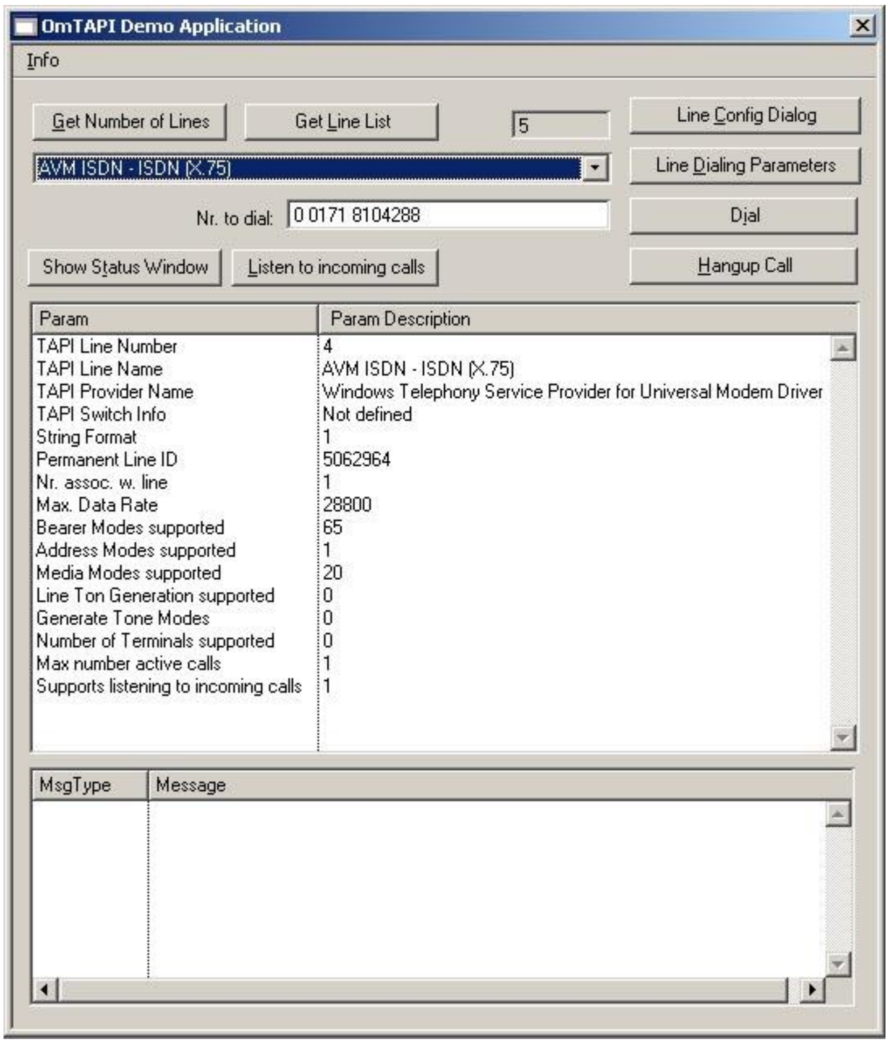
Most of the the names of the parameters and values returned in the list are self-explanatory. One of the most interesting parameters is the last one. It indicates, if the line is able to listen to incoming calls.

If you're interested in the meaning of the remaining parameters, please visit http://msdn.microsoft.com/library/default.asp?url=/library/en-us/tapi/tapi2/linedevcaps_str.asp

It explains the contents of a LINEDEVCAPS C-Structure which is returned by the genuine TAPI function.

Example:

```
Do iTapiLineParamList.$define(iParamName,lParamValue)
TapiGetLineParams (pLineNum,iTapiLineParamList) Returns lRet
If lRet
     Do $cinst.$objs.HL_LineParams.$redraw()
     TapiSupportVoiceCalls  Returns lRet
     Do $cinst.$objs.B_Dial.$enabled.$assign(lRet)
     If int(iTapiLineParamList.16.lParamValue)=kTrue
          Do $cinst.$objs.B_Listen.$enabled.$assign(kTrue)
     Else
          Do $cinst.$objs.B_Listen.$enabled.$assign(kFalse)
     End If
End If
```



Nr. to dial: 0 0171 8104288

Param	Param Description
TAPI Line Number	4
TAPI Line Name	AVM ISDN - ISDN (X.75)
TAPI Provider Name	Windows Telephony Service Provider for Universal Modem Driver
TAPI Switch Info	Not defined
String Format	1
Permanent Line ID	5062964
Nr. assoc. w. line	1
Max. Data Rate	28800
Bearer Modes supported	65
Address Modes supported	1
Media Modes supported	20
Line Ton Generation supported	0
Generate Tone Modes	0
Number of Terminals supported	0
Max number active calls	1
Supports listening to incoming calls	1

TapiSupportVoiceCalls()

Parameters:	None
Return Value:	Integer

Description
TapiSupportVoiceCalls indicates, if the line selected with TapiGetLineParams does support voice calls. The return value is kTrue if yes, otherwise zero.

Example:
See TapiGetLineParams

TapiSetEventHandler(cEventHandler [ClassName/MethodName])

Parameters:	Character, optional [Class or Format name/Method name]
Return Value:	Integer

Description
TapiSetEventHandler defines a callback method (or procedure) in Omnis which is called by the OmTAPI external to post back status messages, information or error codes into Omnis studio.
The optional parameter is a character string containing the name of the class resp. format containing the method/procedure and the name of the method/procedure.

If omitted the default value is:
```
cTAPI/TAPIEventHandler
```

In Omnis Classic the procedure can reside within a window or menu format. In Omnis Studio it should reside within a code Class.

Why do we have to set an event handler for TAPI? TAPI is able to informs the user about a lot of things i.e. the status of call being processed etc. BUT most of the TAPI functions are asynchronous and also work with C-type callbacks. This means, that after calling a TAPI function, i.e. dialing a phone number, the control is returned to the calling application (Omnis) immediately. But nevertheless, TAPI does it's job in an own thread. To get informed about the calling status, you must tell TAPI, respectively OmTAPI the name/address of a function to call, if there is some information for the user.

Example:
```
TapiSetEventHandler ("cTAPI/TAPIEventHandler") Returns lRet
```

The method *TAPIEventHandler* could look like this:

```
Do $iwindows.wTapiTest.$addLintoActionList(#60,#S5)
```

I the Method *$addLintoActionList* in the Window instance wTapiTest, the information contained in #60 and #S5 are added to a list and a window redraw is done.
The storing of the values in #60 and #S5 is done directly by the OmTAPI external, so please don't use #60 and #S5 for storing any other values, they will be overwritten whenever an information is passed over to Omnis.

Values:
#60 can contain the values from 1 to 3

1	Status information
2	Error
3	Incoming call

#S5 contains an explanation or information, or in case of an incoming call, the caller's telephone number, if the PPBX supports this and if the caller doesn't suppress his number being transferred.

TapiMakeCall(cNumberToDial, iLineNum [, iPrivilege])

Parameters:	Character, phone number to call [32]
	Line number
	Integer, optional, line privilege
Return Value:	Integer

Description
TapiMakeCall starts dialing a call to the TAPI line number given as second parameter. The number (maximum 32 characters) is the first parameter.
Second mandatory parameter is the line number to use for making the call.

The third parameter is optional and was introduced with OmTapi 1.2.2.0

The values are the same as with Listening to a line. If the parameter is omitted, Lineprivilege NONE will be used, what used to work, but as far as the MSDN documentation says, with Lineprivilege Owner (+ Monitor) you might get more status messages.

TapiMakeCall returns kTrue if successful, otherwise it returns kFalse.

TapiMakeCall is a typical example for an asynchronous TAPI function. Control is returned immediately to Omnis. Status messages are signaled to Omnis by using the method given with *TapiSetEventHandler*.

Read more about this in the description to *TapiGetMessages()*.

Example:
```
Do iMsgList.$clear()
TapiMakeCall (iNumbertoDial,iLineNum) Returns lRet
If lRet
     Do $cobj.$enabled.$assign(kFalse)
     Do $cinst.$objs.B_Hangup.$enabled.$assign(kTrue)
End If
```

TapiHangupCall()

Parameters:	None
Return Value:	Integer

Description
Any outgoing call will be terminated. Returns kTrue if successful, otherwise it returns kFalse.

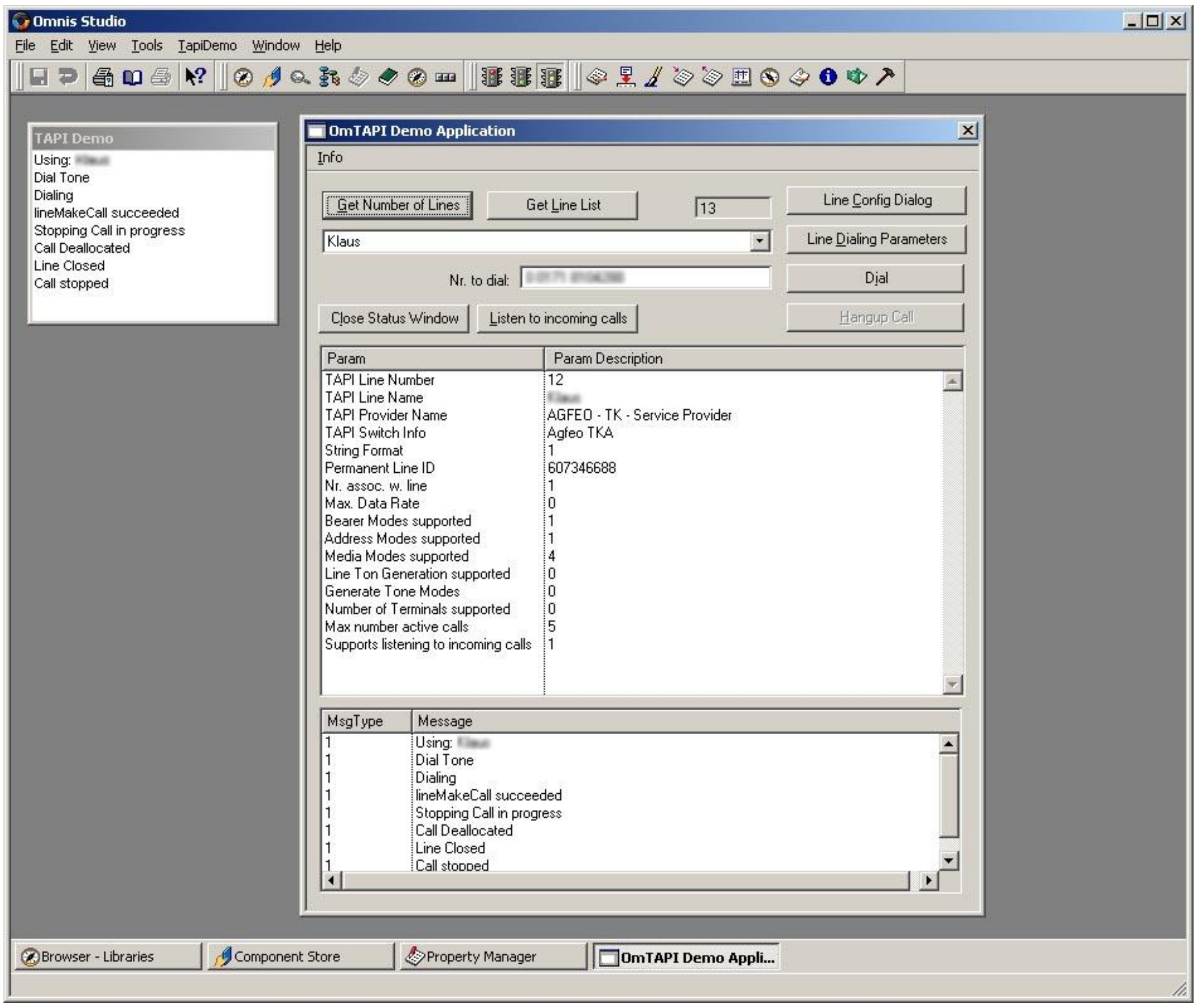
TapiShowStatusWnd(bShow,[iXPos,iYPos,cTitle,iFontSize])

Parameters:	Boolean bShow, [Integer x-position, Integer y-position, character Title, Integer fontsize]
Return Value:	Integer

Description
This command is very useful for developing a TAPI application but maybe also for the end user.
With TapiShowStatusWnd, a status windows will be opened, which displays all TAPI messages and Status events. The status window also will be in front of all Omnis windows.
The window opened is not an Omnis window, its generated by the external and so it cannot be modified.
The first parameter is boolean and results the window to be shown (kTrue) or to be closed (kFalse), if its open.

The second and all following parameters are optional. With iXPos and iYPos you set the window position in pixels relative to the upper left corner of the whole screen, not the Omnis main window.
The fourth parameter sets the status window title.
The fifth parameter sets the font size of the messages displayed. If omitted default values will be used.

Returns kTrue if successful, otherwise it returns kFalse.
Note: If the status window is closed successfully, TapiShowStatusWnd returns kTrue also.



TAPI Demo
Using: [redacted]
Dial Tone
Dialing
lineMakeCall succeeded
Stopping Call in progress
Call Deallocated
Line Closed
Call stopped

Nr. to dial: [redacted]

Param	Param Description
TAPI Line Number	12
TAPI Line Name	[redacted]
TAPI Provider Name	AGFEO - TK - Service Provider
TAPI Switch Info	Agfeo TKA
String Format	1
Permanent Line ID	607346688
Nr. assoc. w. line	1
Max. Data Rate	0
Bearer Modes supported	1
Address Modes supported	1
Media Modes supported	4
Line Ton Generation supported	0
Generate Tone Modes	0
Number of Terminals supported	0
Max number active calls	5
Supports listening to incoming calls	1

MsgType	Message
1	Using: [redacted]
1	Dial Tone
1	Dialing
1	lineMakeCall succeeded
1	Stopping Call in progress
1	Call Deallocated
1	Line Closed
1	Call stopped

TapiListenIncomingCall(iNumLine)

Parameters:	Integer, number of line
	Integer, LinePrivileges
Return Value:	Integer

Description
This command switches OmTAPI into the listening mode. Incoming calls on the line selected by the integer parameter iNumLine will be signaled and the caller's number – if available – will be passed over to Omnis with a status value of 3.
The first parameter ist the number of the line to use.

The second parameter specifies the privilege the application wants for the calls it is notified for. This parameter can be a combination of the LINECALLPRIVILEGE_* constants.
Value Description
0 - The application can make only outgoing calls.
1 - The application can monitor only incoming and outgoing calls.
2 - The application can own only incoming calls of certin modes.

This value must be given because not each line/device can be switched to listening mode with the same values. Modems i.e. need a value of 1, ISDN PBX devices a value of 2.

Returns kTrue if successful, otherwise it returns kFalse.

Note: While listening, you also can make outgoing calls. Both functions will not interfere.

TapiAnswerCall()

Parameters:	None
Return Value:	kTrue (1) when answering succeeds, kFalse (0) if answering fails

Description
Will answer an incoming call (new in V 1.21).
Preconditions:
a) The line has to be set to listening status before. To be successful, ownership of the line is taken automatically before issuing the answer function.
b) The device connected to line must support answering a call.

Returns kTrue if successful, otherwise it returns kFalse.

TapiCloseListenLine()

Parameters:	None
Return Value:	Integer

Description
OmTAPI will stop listening to incoming calls on the line selected with TapiListenIncomingCall.
Returns kTrue if successful, otherwise it returns kFalse.

TapiGetMessages()

Parameters:	None
Return Value:	None

Description
This command is a little hard to explain because it needs some understanding of the internal functions of the TAPI interface.
Like mentioned above, the TAPI interface has some asynchronous functions.
But that's not all. To make it even more complicated, some TAPI events and status messages appear a (relatively) long time after returning control back to Omnis. And due to the architecture of the Omnis' external API, an external cannot make a

callback to Omnis without having a valid callback handle. So, normally, a lot of things happening with TAPI, Omnis would never be informed about.

To avoid this, OmTAPI uses a trick and this is the reason why you have to copy a second DLL called evtapi.dll to the externals folder.

If OmTAPI, which is active all the time after you it was initialized with TapiGetNumLines(), detects an event and there is no valid callback handle to Omnis, the status information will be put on a stack and it generates an own user defined Windows event, addresses it to Omnis (because we have the handle to the main application window) and puts it into the Windows internal message queue. evtapi.dll is the event handler within Omnis which receives this event and has to be activated with the command

```
Load event handler EvTapi ("cTAPI/TAPIGetMsg")
```

In the example, the parameter *cTAPI/TAPIGetMsg* points to a procedure/method TAPIGetMsg in a menu format (Classic) or a code class (Studio).

This method only has one line an this is

```
TapiGetMessages
```
and causes OmTAPI now to call the method set with TAPISetEventHandler to post all status messages and information on the stack back to Omnis, because now we again have a valid callback handle to Omnis.
This sounds a little complicated and tricky and in fact, it is. But if you look at the example library provided with OmTAPI, it’s easy to use.

TapiGetCallInfo()

Parameters:	None
Return Value:	Integer, kTrue if successful, kFalse if not.

Description
TapiGetCallInfo returns information about a present call, usually the phone number calling or being called. Can also return error messages.

TapiSendEvents(bSend)

Parameters:	Bool, kTrue – OmTapi sends events via evTapi DLL, kFalse: No events are generated
Return Value:	Integer, kTrue if successful, kFalse if not.

Description
Introduces with version 1.2.4.0. TapiSendEvents() turns on/off sending of events from OmTapi/evTapi.dll to Omnis.
To maintain compatibility with existing implementations and code, the default setting is On, so sending events has to be turned off explicitly.

The reason for introducing this command was, that on some customer sites with heavy phone/call traffic it has been seen that Omnis 4.x was crashing after a while with no obvious reasons.
We never could reproduce this errors but they probably depend very much of the kind of code running in Omnis.
We suppose, that this had to do with OmTapi resp. evTapi.dll sending events into Omnis. Since Omnis (if not running as a multithreaded web application server) is single threaded, depending on the kind of code running while evTapi.dll signals an event/messages to be fetched, this can conflict with the internal Omnis event loop which also handles Omnis timers.
These were used heavily in the conflicting application and so may have caused Omnis crashing.

To be able to keep track of the TAPI/phone activity while sending events is disabled, there must be options to retrieve this information.
For this reason, we introduced the following commands

```
TapiGetMessageList (iEventList)
TapiGetNumMessages()
```

which are discussed in the next paragraphs.

TapiGetNumMessages()

Parameters:	None
Return Value:	Integer, number of messages waiting to be retrieved

Description
If sending events is turned of by issuing TapiSendEvents(kFalse), messages and information about incoming calls etc. are cached within OmTapi.

TapiGetNumMessages() returns the number of messages waiting to be retrieved. Sending events has to be turned off for this command to work properly. Otherwise it always will return zero.

TapiGetMessageList (&iEventList)

Parameters:	List
Return Value:	Integer, kTrue if successful

Description
The command returns the pending messages in a list. The latest events are on the bottom of the list.

TapiGetNumMessages and TapiGetMessageList only work, if sending events is turned off.

The list iEventList ist passed by reference and hast to be defined before calling TapiGetMessageList()

```
Do iEventList.$define(lCount,lMsgType,lMessage)
TapiGetMessageList (iEventList)
```

The fields in the list are:

lCount	integer	Consecuting numbers
lMsgType	short integer	Type of message
lMessage	Character(1000)	Message text

The meaning of the lMsgType field ist:
1 - Status message
2 – Error message
3 – Incoming call

In the demo library we inserted a checkbox. If you turn it off, after showing a message box with an explanation, a timer will be started which calls a method inside the demo window each 5 seconds (default), which checks, if there are pending messages, and if yes, they will be retrieved, which also resets the message/event cache to zero.

Please note, that synchronous messages from OmTapi still are sent directly to Omnis because in that case the Omnis Main Thread is exactly there and doesn’t have to be interrupted.

TapiShutDown ()

Parameters:	None
Return Value:	Integer

Description
Shuts down the OmTAPI Interface completely. Listening and all outgoing calls will be terminated.
Returns kTrue if successful, otherwise kFalse. If you want to reuse OmTAPI after TapiShutdown, Omnis has to be restarted.
Normally you won't have to use this command, all TAPI activity is terminated also, when you close Omnis without using this command.

TapiSetStatusVars(cStringVar, cNumVar)

Parameters	cStringVar, Character (MaxLen 64), name of string variable
	cNumVar, Character (Maxlen 64), name of Integer variable
Return Value	int: 1 = Success, 0 = Error

New in Version 1.104
Sine using the hash vars #60 and #S5 is sometimes inconvenient or causes conflicts with existing applications, the method TapiSetStatusvars() allows to set user defined variables for handling and displaying status information given back from OmTAPI.

It’s recommended to use Omnis Studio task variables which are present and already initialized when the method is called. The names of the variables have to be passed as strings (case sensitive) and max not exceed a length of 64 characters.
Also setting the Status variables should be done as one of the first actions using OmTAPI.
Directly after passing the variable names to OmTAPI, it tries to get a reference to them, so that they can be used.

If getting the reference is successful, the return value will be 1, in case of an error 0 will be returned.

Important: TapiSetStatusVars() is **NOT** tested anymore with Omnis 7 (Omnis Classic). It may work when using Library variables.

TapiAbout()

Parameters:	None
Return Value:	None

Description
Displays a message box with Copyright and license information and the serial number.

TapiSetLogging(bMode,[cPath])

Parameters:	Bool, Starts/Stops Logging
Return Value:	String, Optional , Path of Logfile

Description
New in version 1.2.2.0. OmTapi now is capable of logging events coming in like incoming calls and user actions taken.

Starts/Stops Logging. The parameter cPath is optional. If Path and name is omitted, a logfile called Omatapilog.txt will be created in same folder as the library from which the command is issued.
Otherwise a complete path AND name of the logfile has to passed as a parameter.

The log messages are generated automatically within OmTapi. There is no way to write individual log messages.

Please consider only to create log files in directories where you can be sure Omnis has write access to.

Important: OmTapi automatically creates the logfile BUT NOT folders containing the log, they have to be created within Omnis.
Also “housekeeping” of the logfiles must be done from within Omnis to avoid the log directory running full of files.

The log files created look like this:

```
10.08.2018 - 20:22:59: Logging started
10.08.2018 - 20:22:59: OmTapi V 1.25
10.08.2018 - 20:23:04: Initializing TAPI
10.08.2018 - 20:23:04: TAPI initialized
10.08.2018 - 20:23:04: Number of lines: 9
10.08.2018 - 20:23:09: Getting line properties
10.08.2018 - 20:23:09: Reading line properties successful
10.08.2018 - 20:23:15: Getting line properties
10.08.2018 - 20:23:15: Reading line properties successful
10.08.2018 - 20:23:20: Sending events turned OFF
10.08.2018 - 20:23:23: Init listening on current line
10.08.2018 - 20:23:23: TAPIMSG: 1, Using: xyz
10.08.2018 - 20:23:28: TAPIMSG: 1, Listening on line 5
```

```
10.08.2018 - 20:23:28: Listening on current line successful
10.08.2018 - 20:24:01: TAPIMSG: 1, Incoming call
10.08.2018 - 20:24:01: TAPIMSG: Caching Msg. # 1
10.08.2018 - 20:24:01: TAPIMSG: 3, 017112345678
10.08.2018 - 20:24:01: TAPIMSG: Caching Msg. # 2
10.08.2018 - 20:24:01: TAPIMSG: 1, lineGetCallStatus succeeded
10.08.2018 - 20:24:01: TAPIMSG: Caching Msg. # 3
10.08.2018 - 20:24:01: TAPIMSG: 1, LINEPRIVILEGE: LINECALLPRIVILEGE_OWNER
10.08.2018 - 20:24:01: TAPIMSG: Caching Msg. # 4
10.08.2018 - 20:24:01: TAPIMSG: 1, lineGetCallStatus succeeded
10.08.2018 - 20:24:01: TAPIMSG: Caching Msg. # 5
10.08.2018 - 20:24:01: TAPIMSG: 1, LINEPRIVILEGE: LINECALLPRIVILEGE_OWNER
10.08.2018 - 20:24:01: TAPIMSG: Caching Msg. # 6
10.08.2018 - 20:24:07: TAPIMSG: 1, Line idle
10.08.2018 - 20:24:07: TAPIMSG: Caching Msg. # 1
10.08.2018 - 20:24:07: TAPIMSG: 1, Call Deallocated
10.08.2018 - 20:24:07: TAPIMSG: Caching Msg. # 2
10.08.2018 - 20:24:07: TAPIMSG: 1, lineGetCallStatus failed: LINEERR_INVALCALLHANDLE
10.08.2018 - 20:24:07: TAPIMSG: Caching Msg. # 3
10.08.2018 - 20:24:18: Stopping listening on line 5
10.08.2018 - 20:24:18: TAPIMSG: 1, Stop listening requested
10.08.2018 - 20:24:19: TAPIMSG: 1, Listening stopped
10.08.2018 - 20:24:19: Stopping listening on line successful.
10.08.2018 - 20:24:19: Logging stopped
```

Known Problems

OmTAPI is based upon the Microsoft TAPI Architecture and so all of it's drawbacks also apply to OmTAPI.
One oft them is using TAPI with Terminal Server. Please read the Microsoft KnowledgeBase article at http://support.microsoft.com/kb/308405/en-us
This means we do **NOT** recommend using OmTAPI on Windows Terminal Server.

FAQ

1. Will there be a version of OmTAPI for the Macintosh, Linux or Sun Solaris?

No, since TAPI is a windows specific technology, there will be only a windows version.

2. What are the limitations of the demo version?

There are four limitations in the demo version.

1. It runs for a trial period of 15 days.
2. It will on work in the development environment, not in the runtime or a webserver runtime
3. When an incoming call is signaled, the digits in the second half of the caller's number (if transferred) will be replaced by the letter 'x'.
4. There are only 5 outgoing call possible per Omnis session. The Omnis has to be restarted

These limitations allow full testing of OmTAPI but (that's the intention) not the practical use within a production environment.