

TAPIMaster® SDK programming manual

© 2020 Tino Kasubke

Contents

Contents	1
1 General	2
2 Client interface	3
Command reference	4
.NET	5
CtiConvert	6
ConvertToCanonical	7
ExpandInternNumber	7
GetDateTime	8
GetFormatDateTime	8
CtiLogin	10
AssistedTelephony	12
AutoAnswer	12
AutoConnect	12
AutoLogin	13
BeforeClose	13
CheckConnection	14
ClientActive	14
ConfigFileName	15
HotKey	15
Login	16
Logoff	16
NetworkConfigDialog	17
OnClientClose	17
OnConnectionState	18
OnLicenseInfo	19
OnLocaleInfo	19
OnLogin	20
OnMenuCommand	20
OnRaiseClient	21
OnShowPhoneList	21
OwnName	22
OwnNumber	22
PhoneIcon	23
Port	23
PrepareRestart	23
QuickMouseDial	24
SaveSettings	25
ServerName	25
SetConnectionIconOptions	26
SetMainWindow	26
ShowIcon	27
ShowInfoDialog	27
ShowMenu	28
StartClient	29
Messages	29
ChatEnd	31

ChatMessage	31
ChatStart	32
OnChatMessage.....	32
OnChatStart	33
OnChatStop	34
OnDatabaseToUser.....	34
OnServerMessage.....	35
OnUserToUser	35
SendUserToUser.....	36
UserSendToDataBase.....	37
SQLTool	37
Begin	39
End	39
GetDbBook	40
Login	41
Next	41
OnDbBookRecord.....	42
OnSQLData	43
OnSQLEnd	43
OnSQLLogin	44
Telephony	45
AddCallToConference.....	47
AddNumberToConference.....	47
Alternate	48
Answer	48
DialKey	49
DialSpecialKey	49
DropKey	50
DropSpecialKey	50
ForwardingDest.....	50
ForwardingState.....	51
Hangup	51
Hold	52
LastDialedNumber.....	52
LineReset	53
MakeCall	53
OnAcidCancelClient.....	54
OnCallNumber	54
OnCallReminder	55
OnCallState	56
OnConfNotify	56
OnForwardingState.....	57
OnGroupMember.....	57
OnIdentity	58
OnObjectOfRedirect.....	59
OnProtocolData	60
OnSendAddress.....	61
OnSendUserState.....	61
OnTransferPendInfo.....	62
Pickup	63
PrivateState	63
ProxyMakeCall	64
ReDial	64
Redirect	65

RemovePhoneIcon.....	65
SendCallReminder.....	66
SetAcidReady	67
SetForwarding	67
Transfer	68
UnHold	68
UserCallInfo	69
ActiveX	69
Network	71
NClientActive	72
NIsNetworkVersion.....	72
NLogin	72
NLogoff	73
NNetworkConfigDialog.....	73
NPortNumber	74
NPrepareRestart.....	74
NServerName	75
NStartClient	75
Options	77
OAssistedTelephony	79
OAutoAnswer	79
OAutoConnect	79
OAutoLogin	80
OCheckConnection	80
OConnectionIconOptions.....	81
ODialKey	82
ODialSpecialKey.....	82
ODropKey	82
ODropSpecialKey.....	83
OHotKey	83
OIniFileName	84
OOptions	84
OOwnerName	85
OOwnerNumber.....	85
OPhoneIcon	86
OShowIcon	86
OShowMenu	87
OUserCallInfo	87
Telephony functions.....	89
TAddCallToConference.....	91
TAddNumberToConference.....	91
TCallAnswer	92
TCallDrop	92
TCallHold	93
TCallRedirect	93
TCallToggle	94
TCallTransfer	94
TCallUnHold	95
TForwardingDest.....	95
TForwardingState.....	95
TLineReset	96
TMakeCall	96
TPickup	97
TProxyMakeCall	97

TReDial	98
TRemovePhoneIcon	98
TSendCallReminder	99
TSetForwarding	99
SQL	101
SQLBegin	102
SQLEnd	102
SQLGetDbBook	103
SQLLogin	103
SQLNext	104
Additional functions	106
EChatEnd	107
EChatMessage	107
EChatStart	107
EConvertToCanonical	108
EExpandInternNumber	109
EGetDateTime	109
EGetFormatDateTime	110
ESendUserToUser	111
ESetAcidReady	111
ESetMainWindow	112
EShowInfoDialog	112
EUserSendToDataBase	113
Events	114
OnCallNumber	116
OnCallReminder	117
OnCallState	117
OnChatMessage	118
OnChatStart	118
OnChatStop	119
OnClientClose	119
OnConfNotify	120
OnConnectionState	120
OnDatabaseToUser	121
OnDbBookRecord	121
OnForwardingState	122
OnGroupMember	122
OnIdentity	123
OnInvalidStringFormat	124
OnLicenseInfo	124
OnLocaleInfo	125
OnLogin	125
OnMenuCommand	126
OnObjectOfRedirect	126
OnProtocolData	127
OnRaiseClient	128
OnSendAddress	128
OnSendUserState	129
OnServerMessage	129
OnShowPhoneList	130
OnSQLData	130
OnSQLEnd	131
OnSQLLogin	132
OnUserToUser	132

C++ CTI API.....	134
Network.....	135
CTI_NClientActive.....	135
CTI_NClientRestart.....	136
CTI_NClientStop.....	136
CTI_NGetPortNumber.....	137
CTI_NGetServerName.....	137
CTI_NIsNetworkVersion.....	138
CTI_NLogin.....	138
CTI_NLogoff.....	138
CTI_NNetworkConfigDialog.....	139
CTI_NPrepareRestart.....	140
CTI_NSetNetworkOptions.....	140
CTI_NSetPortNumber.....	141
CTI_NSetServerName.....	142
CTI_NStartClientCALLBACK.....	142
CTI_NStartClientHWND.....	143
Options.....	144
CTI_OConnectionIconOptions.....	144
CTI_OGetDialKey.....	145
CTI_OGetDropKey.....	146
CTI_OGetIniFileName.....	146
CTI_OGetOptions.....	147
CTI_OGetOwnerName.....	148
CTI_OGetOwnerNumber.....	148
CTI_OGetUserCallInfo.....	149
CTI_OSetDialKey.....	149
CTI_OSetDropKey.....	150
CTI_OSetOptions.....	150
CTI_OSetOwnerName.....	151
CTI_OSetOwnerNumber.....	151
CTI_OSetUserCallInfo.....	152
Telephony functions.....	153
CTI_TAddCallToConference.....	154
CTI_TAddNumberToConference.....	154
CTI_TCallAnswer.....	155
CTI_TCallDrop.....	155
CTI_TCallHold.....	156
CTI_TCallRedirect.....	156
CTI_TCallToggle.....	157
CTI_TCallTransfer.....	157
CTI_TCallUnHold.....	158
CTI_TGetForwardingDest.....	158
CTI_TGetForwardingState.....	159
CTI_TLineReset.....	159
CTI_TMakeCall.....	160
CTI_TPickup.....	160
CTI_TProxyMakeCall.....	161
CTI_TReDial.....	161
CTI_TRemovePhoneIcon.....	162
CTI_TSendCallReminder.....	162
CTI_TSetForwarding.....	163
SQL.....	165
CTI_SQLBegin.....	166

CTI_SQLEnd	166
CTI_SQLGetDbBook.....	167
CTI_SQLLogin	168
CTI_SQLNext	168
Additional functions.....	170
CTI_EChatEnd	170
CTI_EChatMessage.....	171
CTI_EChatStart.....	171
CTI_EConvertToCanonical.....	172
CTI_EExpandInternNumber.....	173
CTI_EGetDateTime.....	173
CTI_EGetFormatDateTime.....	174
CTI_ESendUserToUser.....	175
CTI_ESetAcidReady.....	176
CTI_ESetMainWindow	176
CTI_EShow InfoDialog.....	177
CTI_EUserSendToDataBase.....	177
Events	179
WM_CTl_MENUCOMMAND.....	180
WM_CTl_SHOWPHONELIST.....	181
WM_CTl_NETWORKCLOSE.....	182
WM_CTl_NETWORKOPEN.....	183
WM_CTl_RAISECLIENT.....	184
WM_CTl_NETWORK_DISPATCH.....	185
CTI_CALLNUMBER.....	186
CTI_CALLREMINDER.....	187
CTI_CALLSTATE.....	188
CTI_CHATSTOP.....	188
CTI_CHATMESSAGE.....	189
CTI_CHATSTART.....	189
CTI_CLIENTCLOSE.....	190
CTI_CONFNOTIFY	190
CTI_DATABASETOUSER.....	191
CTI_FORWARDINGSTATE.....	191
CTI_GROUPMEMBER.....	192
CTI_IDENTITY	192
CTI_INVALIDSTRINGFORMAT.....	193
CTI_LICENSEINFO.....	193
CTI_LOCALEINFO.....	194
CTI_LOGIN	194
CTI_OBJECTOFREDIRECT.....	195
CTI_PROTOCOLDATA.....	195
CTI_SENDADDRESS.....	196
CTI_SENDUSERSTATE.....	197
CTI_SERVERMESSAGE.....	197
CTI_SERVICEPAUSE.....	198
CTI_USERTOUSER.....	198
SQL_DATA	199
SQL_DBBOOK.....	199
SQL_END	200
SQL_LOGIN	200
Structures	202
CTI_COMMAND_MSG.....	202
CTI_PHONE_DATA.....	203

Database queries.....	203
Constants	206
Call states	207
Connection options	208
CTICLIENT_ASSISTEDTELEPHONY	208
CTICLIENT_AUTOANSWER.....	209
CTICLIENT_AUTOCONNECT.....	209
CTICLIENT_AUTOLOGIN.....	209
CTICLIENT_CHECKCONNECTION.....	210
CTICLIENT_HOTKEY.....	210
CTICLIENT_PHONEICON.....	210
CTICLIENT_QUICKMOUSEDIAL.....	211
CTICLIENT_SHOWICON.....	211
CTICLIENT_SHOWMENU.....	212
Sample programs	213
Integration in VB.....	214
Simply CTI client C++.....	215
Simply CTI client Delphi.....	216
SQL sample.....	217
3 Server interface	219
Command reference	220
.NET	221
ClientCom	222
OnUserToDatabase.....	223
SendAdminMessage.....	223
SendIdentity	225
SendUserData	225
CtiConvert	226
ConvertToCanonical.....	227
GetDateTime	228
GetFormatDateTime.....	229
PBXInfo	231
GetExternPrefix.....	232
GetInboundFilter.....	232
GetInternPrefix.....	233
GetNameFromExtension.....	234
GetOutboundFilter.....	235
LicenseCount	236
OnLoginData	237
TMServiceStatus.....	238
Telephony	238
Alternate	240
Answer	241
Conference	242
Forward	243
GetPBXMonitoredLines	244
Hangup	245
Init	245
MakeCall	246
OnCallNumber	247
OnCallState	248
OnForwardingState.....	249
OnIdentity	250
OnNew Call	251

OnProtocolData.....	252
Pickup	254
Redirect	255
SendAllUserCallStates.....	256
SendLastCallRecords	257
SendPBXLoginLines.....	258
SetPhoneDisplay.....	259
Transfer	260
ActiveX	261
Functions	262
Alternate	263
Answer	263
Conference	264
ConvertToCanonical.....	264
Drop	265
GetAllUserCallStates.....	265
GetDateTime	266
GetExtension	266
GetExternPrefix.....	266
GetFilterCalibrationState.....	267
GetFormatDateTime.....	267
GetInboundFilter.....	268
GetInternPrefix.....	269
GetLastCallRecords.....	269
GetOutboundFilter.....	270
GetPBXFiltersAndPrefixes.....	270
GetPBXMonitoredLines	271
InitCOM	272
LicenseCount	272
MakeCall	272
Pickup	273
Redirect	273
ResetCalibration.....	274
SendIdentity	274
SendUserData	275
SetCalibrationParams.....	275
SetPhoneDisplay.....	276
TMServiceStatus.....	276
Transfer	276
UserCount	278
UserNameFromNumber.....	278
Events	278
OnCallNumber	279
OnCallState	279
OnForwardingState.....	280
OnGetIdentity	281
OnLoginData	281
OnNew Call	282
OnProtocolData.....	283
OnUserToDataBase.....	284
C++ CTI API.....	285
Functions	285
CTI_Answer	286
CTI_Conference.....	286

CTI_ConvertToCanonical.....	287
CTI_Drop	287
CTI_GetDateTime.....	288
CTI_GetFormatDateTime.....	288
CTI_GetLicenseCount.....	289
CTI_Init	290
CTI_MakeCall	290
CTI_Pickup	292
CTI_Redirect	292
CTI_SendIdentity.....	293
CTI_SendUserData.....	293
CTI_Toggle	294
CTI_Transfer	294
Events	296
CTI_CALLNUMBER.....	296
CTI_CALLSTATE.....	297
CTI_FORWARDINGSTATE.....	298
CTI_IDENTITY	298
CTI_NEWCALL.....	299
CTI_PROTOCOLDATA.....	300
CTI_USERTODATABASE.....	300
Sample programs	302
Dialing via the IP interface.....	303
Server interface.....	304
4 Install images	305
Using clients without installing	305
Embedded client install image	306
Embedded server install image	306
Customised server install image	307
Installation configuration	307
Deinstallation	309
5 Tools	310
CTI Browser	311
Line Watcher	314
Setup Configurator	315
Used files.....	316
Client install image.....	317
Server install image.....	318
6 Customization	322
Installation	322
Client interface	323
CTI Server	324
Server control panel	325
Line Watcher	326
Languages	326
Required resources	327

1 Contents

TAPIMaster® SDK programming manual

The TAPIMaster® allows you quickly to extend your software with telephony functions. TAPIMaster® replaces the complicated TAPI interface with simpler functions, which are geared towards user tasks. [Examples](#) provided for various programming languages, allowing you to become productive quickly and a range of [installation](#) and [program test tools](#) are provided as development aids.

[General](#)

[Client interface](#)

[Server interface](#)

[Install images](#)

[Tools](#)

[Customization](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.1 General

Function set

In addition to telephony functions, TAPIMaster® also offers numerous ways to enhance your programs with functions useful when working in groups. A few basic functions are all that is needed for the first steps.

Interfaces

TAPIMaster® has [client](#)-side and [server](#)-side interfaces. One DLL is provided for ActiveX and one for normal API access. Programming with ActiveX is simpler; with the API functions, it is possible to start applications without registering components or, where applicable, installing the client. TAPIMaster® relieves you of network programming tasks necessary for a third party telephony solution. Only the client-side programming interface is present in the single-workstation version. The function set is in this case limited according to the system.

Licences

TAPIMaster® allows you to use up to four of the standard clients delivered in the installation package. One of these five clients may also be a connection to a program developed by someone else. If you wish to connect more than one of your own software clients, you will require a licence. No licence is required for the single-workstation version.

Effort

A basic connection to the CTI interface can be developed in a few hours. Almost all the [example programs](#) were developed within one working day. Use the [CTI Browser](#) to test functions without having to write any code.

See also

[Contents](#) | [Client interface](#) | [Server interface](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2 Client interface

The client interface comprises two separate interfaces in one DLL. You can use ActiveX or call DLL functions in the traditional way. In the latter case, there is no need to register the DLL. Use the standard client or the [CTI Browser](#) to test the interface before use.

The names of functions and events are roughly the same on both interfaces. The [example programs](#) show the use of the major functions.

Two versions of the client interface DLL are available: single-workstation version and network version. Several functions are missing from the single-workspace version.

[Command reference](#)

[Sample programs](#)

See also

[Contents](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1 Command reference

The command reference is split into two sections: one for ActiveX and the other for the C++ API. The C++ functions start with the prefix CTI_.

[.NET](#)

[ActiveX](#)

[C++ CTI API](#)

See also

[Client interface](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.1 .NET

The client interface is composed of the five classes. The class is mandatory CtiLogin used, as here takes place the initialization of the interface.

CtiConvert	The class converts numbers and date formats.
CtiLogin	The class contains functions for configuration, access to the client and general settings.
Messages	The class is used to send messages from server to Client.und vice versa.
SQLTool	The class contains commands and events for querying SQL data on the server.
Telephony	The class contains all functions and events which have to do directly with the calls.

See also

[Command reference - Client](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.1.1 CtiConvert

Class CtiConvert

The class converts numbers and date formats.

Functions

ConvertToCanonical	Convert telephone number to canonical form.
ExpandInternalNumber	Expand an extension.
GetDateTime	Converts a time_t var into a standard string.
GetFormatDateTime	Converts a time_t var into an individual formatted string.

See also

[.NET overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.1.1.1 ConvertToCanonical

Convert telephone number to canonical form.

```
String ConvertToCanonical  
(  
    String strNumber  
)
```

Parameters

strNumber
Number to convert to canonical form

Return value

Converted number

Remarks

You can use this function to convert a dataset containing telephone numbers to canonical form. The output string should be at least 32 long.

See also

[CtiConvert overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.1.1.2 ExpandInternNumber

Expand an extension.

```
String ExpandInternNumber  
(  
    String strNumber  
)
```

Parameters

strNumber
Internal telephone number to be expanded

Return value

Return value containing the expanded number.

Remarks

Extends a telephone number with the country code, prefix, and, if necessary, main number.

Example

Presuming the settings shown below, EExpandInternNumber("22") returns the value +49 (6033) 44422 in Germany.

Outside line	Internal	Filter inbound	Filter outbound	Area code	Main number
0		0	0	9549	444

See also

[CtiConvert overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.1.1.3 GetDateTime

Converts a time_t var into a standard string.

```
String GetDateTime
(
    uint dwTime
)
```

Parameters

dwTime

time_t var, e. g. delivered in the event [OnProtocolData](#).

Return value

Return value in a standard format: 2007-01-12T00:53:32.875

Remarks

Use this function to get a sortable date format string. Any programs like Excel use this format.

See also

[CtiConvert overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.1.1.4 GetFormatDateTime

Converts a time_t var into an individual formatted string.

```
String GetFormatDateTime
(
    uint dwTime,
    String strFormat
)
```

Parameters

dwTime

time_t var, e. g. delivered in the event [OnProtocolData](#).

strFormat

String with date format. This can be formed of the following format characters.

Token	Description
%a	Abbreviated weekday name
%A	Full weekday name
%b	Abbreviated month name
%B	Full month name
%c	Date and time representation appropriate for locale
%d	Day of month as decimal number (01 – 31)
%H	Hour in 24-hour format (00 – 23)
%I	Hour in 12-hour format (01 – 12)
%j	Day of year as decimal number (001 – 366)
%m	Month as decimal number (01 – 12)
%M	Minute as decimal number (00 – 59)
%p	Current locale's A.M./P.M. indicator for 12-hour clock
%S	Second as decimal number (00 – 59)
%U	Week of year as decimal number, with Sunday as first day of week (00 – 53)
%w	Weekday as decimal number (0 – 6; Sunday is 0)
%W	Week of year as decimal number, with Monday as first day of week (00 – 53)
%x	Date representation for current locale
%X	Time representation for current locale
%y	Year without century, as decimal number (00 – 99)
%Y	Year with century, as decimal number
%z,%Z	Either the time-zone name or time zone abbreviation, depending on registry settings; no characters if time zone is unknown
%%	Percent sign

Return value

Null-terminated string in a specific format.

Remarks

The format characters can be combined.

See also

[CtiConvert overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.1.2 CtiLogin

Class CtiLogin

The class contains functions for configuration, access to the client and general settings.

Functions

AssistedTelephony	Activate assisted telephony support.
AutoAnswer	Calls are answered automatically.
AutoConnect	The network connection is maintained automatically.
AutoLogin	Automatically log on to the CTI server.
BeforeClose	Will be called before the program is terminated.
CheckConnection	Connection parameters are checked at start-up.
ClientActive	Open or close the network connection.
ConfigFileName	Returns the name of the configuration file.
HotKey	Activate text-dial via key-stroke.
Login	Logs a client on to the CTI server.
Logoff	Log client off from server.
NetworkConfigDialog	Display a dialog containing the most important network and connection parameters.
OnClientClose	The client should be closed and prepared for restart.
OnConnectionState	The connection state has changed.
OnLicenseInfo	Delivers extended licence information.
OnLocaleInfo	Delivers switch and country settings.
OnLogin	Confirms log-on to CTI server.
OnMenuCommand	The menu of the connection icon was clicked.
OnRaiseClient	Tells the client it should raise itself into the foreground.
OnShowPhoneList	The user has double-clicked the yellow telephone symbol.
OwnName	Sets ones own login name.
OwnNumber	Sets ones own extension.
PhoneIcon	A telephone symbol is displayed when incoming calls arrive.
Port	Change the port number of the network connection.
PrepareRestart	Prepare client for restart.
QuickMouseDial	Activates text-dial via context menu.
SaveSettings	Will be called when the interface is to save your settings.
ServerName	Change the name or IP address of the server of the network connection.

SetConnectionIconOptions	Changes the appearance of the connection icons.
SetMainWindow	Passes the handle of the main window to the interface.
ShowIcon	Show a connection symbol.
ShowInfoDialog	Displays interface version information.
ShowMenu	The connection icon has a menu.
StartClient	Initialises the CTI interface.

See also

[.NET overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.1.2.1 AssistedTelephony

Activate assisted telephony support.

Property AssistedTelephony**Type**

bool, read and write

Remarks

Accepts command for Assisted Telephony The interface initiates a call when a third-party program calls the Windows TAPI function `tapiRequestMakeCall`.

Default value

true

See also

[CtiLogin overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.1.2.2 AutoAnswer

Calls are answered automatically.

Property AutoAnswer**Type**

bool, read and write

Remarks

Incoming calls are answered automatically. This option is of use in call centers. The telephone or headset must be capable of switching to hands-free mode and support must be present in the TAPI driver.

Default value

false

See also

[CtiLogin overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.1.2.3 AutoConnect

The network connection is maintained automatically.

Property AutoConnect

Type

bool, read and write

Remarks

At program start-up, the CTI interface automatically establishes the connection to the CTI-Server. If the connection is interrupted, the client tries automatically to re-establish it. There is no need to set [ClientActive](#) to true as well.

Default value

true

See also

[CtiLogin overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.1.2.4 AutoLogin

Automatically log on to the CTI server.

Property AutoLogin**Type**

bool, read and write

Remarks

After the network connection has been established, the interface logs on to the server automatically. There is no need to call, [Login](#) separately.

Default value

true

See also

[CtiLogin overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.1.2.5 BeforeClose

Will be called before the program is terminated.

void BeforeClose()**Parameters**

None

Return value

None

Remarks

The function starts cleaning up the client interface.

See also

[CtiLogin overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.1.2.6 CheckConnection

Connection parameters are checked at start-up.

Property CheckConnection**Type**

bool, read and write

Remarks

At start-up, the interface checks the connection parameters – server name, port number and internal extension number – for completeness. If a parameter is missing, the interface raises the [login dialog](#) to the foreground.

Default value

true

See also

[CtiLogin overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.1.2.7 ClientActive

Open or close the network connection.

Property ClientActive**Type**

bool, read and write

Remarks

Set this property to False to terminate an existing connection to the CTI-Server. Set the value to True to establish the connection once more.

See also

[CtiLogin overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.1.2.8 ConfigFileName

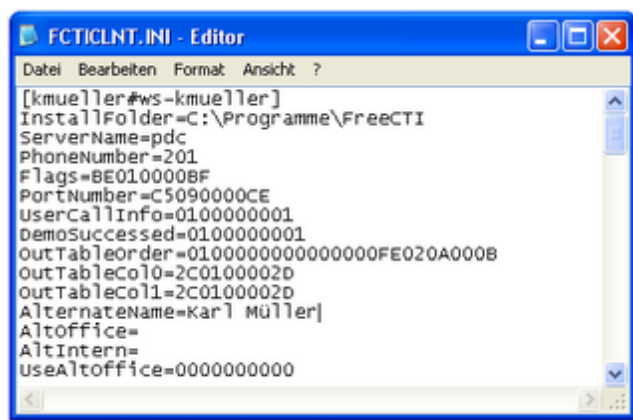
Returns the name of the configuration file.

Property ConfigFileName**Type**

String, read only

Remarks

The interface stores its settings in a configuration file. You may use this file yourself if you do not wish to program the creation of a separate file for storing the client settings. The file is in Windows INI format. The section names have the form: user name + # + computer name, for example, [smith#wk12]. Take a look at one of the INI files provided in order to avoid using existing key names.

**See also**

[CtlLogin overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.1.2.9 HotKey

Activate text-dial via key-stroke.

Property HotKey**Type**

bool, read and write

Remarks

Determines whether highlighted text can be dialed by key-stroke. The key will be in the range F1 to F12. Additionally, the shift and/or control key can be pressed. If the text contains a dialable number, the number will be called. The text-dial and additional key are set using [DialKey](#) and [DialSpecialKey](#).

Default value

false

See also

[CtiLogin overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.1.2.10 Login

Logs a client on to the CTI server.

void Login()

Parameters

None

Return value

None

Remarks

This function is performed automatically if [AutoLogin](#) is active and the network connection exists. **Login** activates the functionality for the client in the CTI-Server. The other members of the group then see the client as logged-on. When you use the interface from a program which is active all day long, it may make sense to turn off the CTI functionality occasionally, such as during lunch. The other group members then see that the user is not available. [Logoff](#) turns CTI off and **Login** turns it on again.

See also

[CtiLogin overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.1.2.11 Logoff

Log client off from server.

void Logoff()

Parameters

None

Return value

None

Remarks

Turns off the CTI function in the CTI-Server. The members of the group then see the client as logged-off. Alternatively, the whole network connection can also be torn down if [ClientActive](#) is set to false.

See also

[CtiLogin overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.1.2.12 NetworkConfigDialog

Display a dialog containing the most important network and connection parameters.

void NetworkConfigDialog()

Parameters

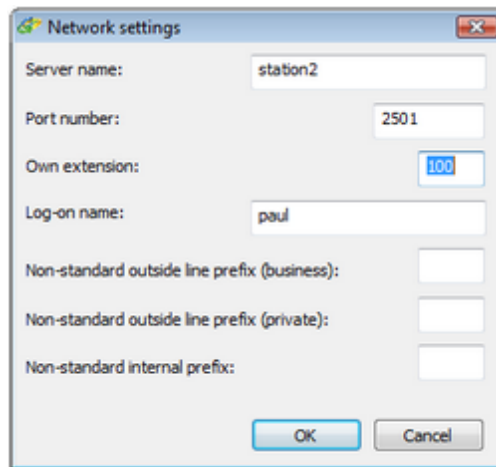
None

Return value

None

Remarks

A dialog for entering the connection parameters is opened. Alternatively you can use [ServerName](#), [Port](#) and [OwnNumber](#). If the first free parameters are not supplied and the property [CheckConnection](#) is set to true, then the dialog appears automatically when the program is started. When the dialog is displayed, the CTI server connection is interrupted. Pressing OK causes a new connection to be established. Use this dialog if you need to change the connection parameters and do not want to have to program a means of entering the parameters yourself.



See also

[CtiLogin overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.1.2.13 OnClientClose

The client should be closed and prepared for restart.

Event OnClientClose

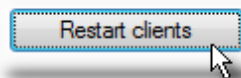
```
(  
    object sender,  
    int e.nMinutes  
)
```

Parameters*sender*object [CtiLogin](#)*e.nMinutes*

Length of time in minutes until the client should be restarted – if the value is non-negative.

Remarks

The administrator sends this message when he wants to replace important files. This is useful when the clients were started from a shared network directory. To replace the files, all clients must be shut down. If you wish to support this feature, call [PrepareRestart](#) upon receiving the event, passing *nMinutes* as a parameter. If *nMinutes* is negative, the client should not be restarted. Then close your application. **OnClientClose** is triggered from the server control panel (“User administration” page).

**See also**[CtiLogin overview](#)[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.1.2.14 OnConnectionState

The connection state has changed.

Event OnConnectionState

```
(
    object sender,
    int e.nState
)
```

Parameters*sender*object [CtiLogin](#)*e.nState*

State of the server connection.

Remarks

The state of the server connection has changed.

Possible state values are:

- 0: Unknown
- 1: No network connection
- 2: Connection up, server inactive.
- 3: Connection up, CTI-Server active.

The connection state changes when the client connects to the server or when the server is shut down or when the server takes a break.

See also[CtiLogin overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.1.2.15 OnLicenseInfo

Delivers extended licence information.

Event OnLicenseInfo

```
(  
    object sender,  
    String e.strMessage  
)
```

Parameters

sender

object [CtiLogin](#)

e.strMessage

Extended licence information. The data are enclosed in curly brackets and separated by semi-colons. Example format:

{License for: Anyfirm;Street: High street 1;City: 12345 Anytown;Phone: +49 (5555) 664488;Web: http://www.anyfirm.de}

Remarks

This message is sent after log-on.

See also

[CtiLogin overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.1.2.16 OnLocaleInfo

Delivers switch and country settings.

Event OnLocaleInfo

```
(  
    object sender,  
    int e.nInternLength,  
    int e.nCountry,  
    String e.strCity,  
    String e.strPBX  
)
```

Parameters

sender

object [CtiLogin](#)

e.nInternLength

Maximum length of internal extensions – value must be in the range 2 - 4.

e.nCountry

Country code, e.g, 49 for Germany

e.strCity

Prefix without the leading null

e.strPBX

Switch main number. Useful for switches connected to the public network. In the number +49 (69) 123456-789, 123456 is the main number.

Remarks

Upon log-on this function delivers the local parameters for use by the program.

See also

[CtiLogin overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.1.2.17 OnLogin

Confirms log-on to CTI server.

Event OnLogin

```
(
    object sender,
    bool e.bSuccess
)
```

Parameters

sender

object [CtiLogin](#)

e.bSuccess

Success or failure of log-on: True means the user logged on successfully.

Remarks

The user must be logged on before the majority of commands can be issued and events can be received..

See also

[CtiLogin overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.1.2.18 OnMenuCommand

The menu of the connection icon was clicked.

Event OnMenuCommand

```
(  
    object sender,  
    uint e.dwCommand  
)
```

Parameters

sender
object [CtiLogin](#)

e.dwCommand
Command ID

Remarks

The user has clicked on the menu of the connection symbol. The command triggered is conveyed in *dwCommand*.

See also

[CtiLogin overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.1.2.19 OnRaiseClient

Tells the client it should raise itself into the foreground.

```
Event OnRaiseClient  
(  
    object sender  
)
```

Parameters

sender
object [CtiLogin](#)

Remarks

Double-clicking on the network symbol triggers this event. The user would like to see the telephony application in the foreground.

See also

[CtiLogin overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.1.2.20 OnShow PhoneList

The user has double-clicked the yellow telephone symbol.

```
Event OnShowPhoneList  
(  
    object sender
```

)**Parameters**

sender
object [CtiLogin](#)

Remarks

A telephone symbol may be displayed in the task bar when a call arrives. Double-clicking the icon leads to this event. The program should react by showing the caller list, if there is one.

See also

[CtiLogin overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.1.2.21 OwnName

Sets ones own login name.

Property OwnName**Typ**

String, read and write

Remarks

Sets ones own login name. The value will be saved by the CTI interface. The interface will be reconnected.

See also

[CtiLogin overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.1.2.22 OwnNumber

Sets ones own extension.

Property OwnNumber**Typ**

String, read and write

Remarks

Own extension, which the server requires to assign line.

See also

[CtiLogin overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.1.2.23 PhoneIcon

A telephone symbol is displayed when incoming calls arrive.

Property PhoneIcon**Type**

bool, read and write

Remarks

A small telephone symbol can be displayed in the task bar when a call arrives. This function parallels the display of a letter by MS Outlook. The telephone number, if present, is shown in the symbol's tool-tip. The icon has a short menu, which allows the call to be returned. [RemovePhoneIcon](#) removes the icon. The [OnShowPhoneList](#) event is sent when the icon is double-clicked and you should then display the caller list, if present.

**Default value**

true

See also

[CtiLogin overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.1.2.24 Port

Change the port number of the network connection.

Property Port**Type**

int32, read and write

Remarks

The Port number of the connection. The connection is not re-established with the new parameters if the value is changed. In order to start a new connection, turn the option [ClientActive](#) off and on again.

See also

[CtiLogin overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.1.2.25 PrepareRestart

Prepare client for restart.

```
void PrepareRestart  
(  
    int nMinutes  
)
```

Parameters*nMinutes*

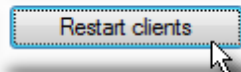
Time in minutes which should elapse before the restart. When *nMinutes* = 0, a restart occurs immediately. If the value is negative, the client is not restarted.

Return value

None

Remarks

The CTI-Server can shut down and restart the standard client. This can be useful if the clients were started from the network and the files have to be changed. The restart can be triggered from the server console "Users" page.



You can also support these functions. When an [OnClientClose](#) event arrives, the *nMinutes* tells you the time until the client should restart. Call [PrepareRestart](#) passing this time as a parameter, and end the application. The application will be restarted after the time has elapsed.

See also[CtiLogin overview](#)

[Send feedback to TAPIMaster®](#)[© 2020 Tino Kasubke. All rights reserved.](#)

1.2.1.1.2.26 QuickMouseDial

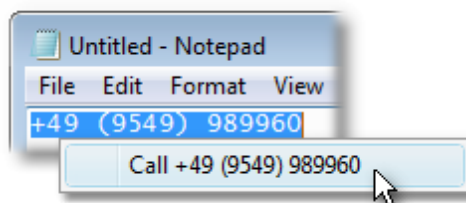
Activates text-dial via context menu.

Property QuickMouseDial**Type**

bool, read and write

Remarks

Using this option, a context menu can be used to dial highlighted text. Such a context menu is displayed if the text contains a dialable number:



Default value

false

See also

[CtiLogin overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.1.2.27 SaveSettings

Will be called when the interface is to save your settings.

void SaveSettings()**Parameters**

None

Return value

None

Remarks

Normally, the settings are saved when you exit the program. With this function, you can do it in between. The settings are written to a file that can be interrogated with [ConfigFileName](#).

See also

[CtiLogin overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.1.2.28 ServerName

Change the name or IP address of the server of the network connection.

Property ServerName**Type**

String, read and write

Remarks

The name of the CTI server. An IP address or "localhost" is also acceptable. The connection is not re-established with the new parameters if the value is changed. In order to start a new connection, turn the option [ClientActive](#) off and on again.

See also

[CtiLogin overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.1.2.29 SetConnectionIconOptions

Changes the appearance of the connection icons.

```
void ConnectionIconOptions
(
    uint hlconOn,
    uint hlconOff,
    uint hMenu,
    String strText
);
```

Parameters*hlconOn*

Handle of icon which is displayed when a connection exists to the CTI server. The value 0 can be specified if the standard icon should be used.

hlconOff

Handle of icon which is displayed when the connection to the CTI server is interrupted. The value 0 can be specified if the standard icon should be used.

hMenu

Handle of menu which is displayed over the connection icon if this is clicked with the right mouse button. The value 0 can be specified if no menu should be used.

strText

Tool-tip which should be shown over the connection icon. Maximum length: 63 characters. The value 0 can be specified if the standard text should be displayed. Specify an empty string if no tip should be shown..

Return value

None

Remarks

Tool-tip which should be shown over the connection icon. Maximum length: 63 characters. The value 0 can be specified if the standard text should be displayed. Specify an empty string if no tip should be shown. It is possible to alter the appearance, menu and tool-tips of the connection icons. There is also a standard text for the tool-tip. The [ShowIcon](#) option must be active to allow the use of the connection icon. For menus, [ShowMenu](#) is also necessary.

See also

[CtlLogin overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.1.2.30 SetMainWindow

Passes the handle of the main window to the interface.

```
void SetMainWindow
(
```

```
uint hWnd
```

```
)
```

Parameters

hWnd

Application's window handle

Return value

None

Remarks

Passes the handle of the application's main window to the interface. This is necessary so that the interface can display child dialogs in the correct position.

See also

[CtiLogin overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.1.2.31 Show Icon

Show a connection symbol.

Property ShowIcon**Type**

bool, read and write

Remarks

A connection icon is shown in the task bar reflecting the state of the network connection. The icon can also have a user-defined menu and tool-tip. You can also supply your own icons for active and interrupted connections instead of using the standard icons.

**Default value**

true

See also

[CtiLogin overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.1.2.32 Show InfoDialog

Displays interface version information.

```
void ShowInfoDialog()
```

Parameters

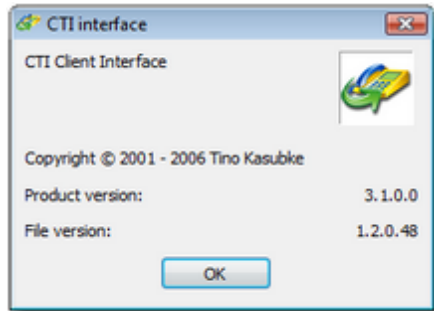
None

Return value

None

Remarks

Displays a small dialog with information about the interface. Manufacturer and product and file version are displayed.

**See also**[CtiLogin overview](#)

[Send feedback to TAPIMaster®](#)[© 2020 Tino Kasubke. All rights reserved.](#)

1.2.1.1.2.33 Show Menu

The connection icon has a menu.

Property ShowMenu**Type**

bool, read and write

Remarks

A menu can be assigned to the connection item in the task bar. The menu appears when the user right-clicks the icon with the mouse. [SetConnectionIconOptions](#) is used to specify the menu handle. Menu events are delivered by [OnMenuCommand](#). [ShowIcon](#) must be set to true.

Default value

false

See also[CtiLogin overview](#)

[Send feedback to TAPIMaster®](#)[© 2020 Tino Kasubke. All rights reserved.](#)

1.2.1.1.2.34 StartClient

Initialises the CTI interface.

void StartClient()**Parameters**

None

Return value

None

Remarks

This function initialises the [client interface](#). From this point on, events can be received.

See also

[CtiLogin overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.1.3 Messages

Class Messages

The class is used to send messages from server to Client.und vice versa.

Functions

ChatEnd	End the active chat session. Only available in the network version.
ChatMessage	Send chat message. Only available in the network version.
ChatStart	Starts a chat session.
OnChatMessage	The chat partner is sending a chat message.
OnChatStart	A chat session is starting.
OnChatStop	The chat session should be terminated.
OnDatabaseToUser	A user-defined message was sent from the server.
OnServerMessage	The administrator sends a message to the user.
OnUserToUser	Another user sends a message.
SendUserToUser	Send a short message. Only available in the network version.
UserSendToDatabase	Send user-defined messages.

See also

[.NET overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.1.3.1 ChatEnd

End the active chat session. Only available in the network version.

void ChatEnd()

Parameters

None

Return value

None

Remarks

Both of the parties chatting may call this function. After the call is made, both parties receive an event of type [OnChatStop](#). The application can then close or deactivate the chat window.

See also

[Messages overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.1.3.2 ChatMessage

Send chat message. Only available in the network version.

```
void ChatMessage  
(  
    String strMessage  
)
```

Parameters

strMessage

Message text from chat box

Return value

None

Remarks

The whole text from the chat box should be sent. The function should be called as soon as something changes in the entry field. The function leads to event [OnChatMessage](#) at the other chatting party.

See also

[Messages overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.1.3.3 ChatStart

Starts a chat session.

```
void ChatStart  
(  
    String strDest  
)
```

Parameters

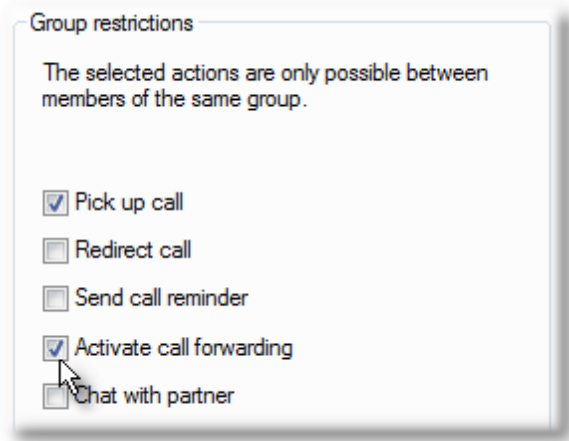
strDest
Phone extension of the chat partner

Return value

None

Remarks

The person with whom chatting is to start must be logged on and chat permission must be granted. The person must not be in a chat session. If the chat initiation was successful, both chat parties receive an event of type [OnChatStart](#). The input and output fields for the session may then be displayed. If a chat session cannot be raised, this may be due to chat being restricted to members of the same group. You can remove this restriction from the “Groups” page of the server control panel.

**See also**

[Messages overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.1.3.4 OnChatMessage

The chat partner is sending a chat message.

```
Event OnChatMessage  
(  
    object sender,
```

```
String e.strMessage
```

```
)
```

Parameters

sender

object [Messages](#)

e.strMessage

Message, up to 255 characters

Remarks

A chat session is in progress with another user, and the user has sent us a message. Display the message in the output window for these messages. The whole text is sent, not just the last character.

See also

[Messages overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.1.3.5 OnChatStart

A chat session is starting.

Event OnChatStart

```
(
```

```
    object sender,  
    bool e.bStartOK,  
    String e.strUser
```

```
)
```

Parameters

sender

object [Messages](#)

e.bStartOK

Shows whether the session was successfully opened.

e.strUser

Extension of chat partner

Remarks

There are two reasons why this event might be received:

1. You have used [ChatStart](#) to request a chat session with another user. If relevant permissions are not set, the session may be refused and so *bStartOK* will be False. Otherwise, it will be True and you can display the relevant input and output windows for chatting..
2. Someone is trying to start a chat session with you. You can see from *strUser* who that is. If you would like to refuse to chat, call [ChatEnd](#) Otherwise, display the windows for input and output for chatting.

See also

[Messages overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.1.3.6 OnChatStop

The chat session should be terminated.

Event OnChatStop

```
(  
    object sender  
)
```

Parameters*sender*object [Messages](#)**Remarks**

The message is sent to both chat partners when one of them closes the chat session. The chat windows should then be closed.

See also[Messages overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.1.3.7 OnDatabaseToUser

A user-defined message was sent from the server.

Event OnDatabaseToUser

```
(  
    object sender,  
    uint e.dwCommand,  
    String e.strMessage  
)
```

Parameters*sender*object [Messages](#)*e.dwCommand*

Type of user-defined command

e.strMessage

Parameter sent

Remarks

TAPIMaster® has a second interface on the server side. Database queries are normally made via the ODBC server interface. Alternatively, the server interface can be linked to a database or other application. [UserSendToDataBase](#) can then be used to exchange user-defined data.

See also

[Messages overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.1.3.8 OnServerMessage

The administrator sends a message to the user.

Event OnServerMessage

```
(  
    object sender,  
    String e.strMessage  
)
```

Parameters

sender

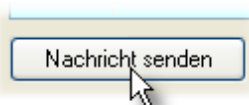
object [Messages](#)

e.strMessage

Message text

Remarks

The administrator sends a message to the user which is contained in *strMessage*. Up to 255 characters can be sent. The messages are sent from the server control panel "User administration" page.

**See also**

[Messages overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.1.3.9 OnUserToUser

Another user sends a message.

Event OnUserToUser

```
(  
    object sender,  
    String e.strNumber,  
    String e.strMessage  
)
```

Parameters*sender*object [Messages](#)*e.strNumber*

Extension of the other user

e.strMessage

Text message the other user has sent, maximum 255 characters.

Remarks

Users can send each other messages using [SendUserToUser](#). Implementing this function may allow some workplaces to do without a mail client.

See also[Messages overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.1.3.10 SendUserToUser

Send a short message. Only available in the network version.

```
void SendUserToUser  
(  
    String strOtherUser,  
    String strMessage  
)
```

Parameters*strOtherUser*

Extension of the other party

strMessage

Short message (max 220 characters) to send to other party

Return value

None

Remarks

Using this function, various users can send short messages to each other without having to use a mail client. [OnUserToUser](#) is triggered in the receiving client.

See also[Messages overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.1.3.11 UserSendToDataBase

Send user-defined messages.

```
void UserSendToDataBase  
(  
    uint dwCommand,  
    String strText  
)
```

Parameters

dwCommand
Command number

strText
Text to send

Return value

keiner

Remarks

This function sends user-defined message to the server interface, where they cause the [OnUserToDataBase](#) event to be generated. *dwCommand* is used to differentiate commands arriving there. The server interface can send an answer to the message, leading to the [OnDatabaseToUser](#) event on the client side.

See also

[Messages overview](#)

[Send feedback to TAPIMaster@](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.1.4 SQLTool

Class SQLTool

The class contains commands and events for querying SQL data on the server.

Functions

Begin	Start an SQL query.
End	End an SQL query.
GetDbBook	Search for telephone number records from name or part of name.
Login	Log on to the database.
Next	Continue SQL query.
OnDbBookRecord	Delivers a record from a telephone number search.
OnSQLData	Answers an SQL query.
OnSQLEnd	An SQL query was ended.

OnSQLLogin	Confirms log-on to the database.
----------------------------	----------------------------------

See also[.NET overview](#)

[Send feedback to TAPIMaster®](#)[© 2020 Tino Kasubke. All rights reserved.](#)

1.2.1.1.4.1 Begin

Start an SQL query.

```
uint Begin  
(  
    String strCommand,  
    uint dwCount  
)
```

Parameters

strCommand

SQL command to send to the database.

dwCount

Maximum number of records to query.

Return value

Query ID as return value

Remarks

The function starts a database query. Any SQL string can be passed in, so long as the administrator has not made any restrictions. Since a large number of records may be discovered, *dwCount* should be used to limit the number. The query can be continued with [Next](#) and ended with [End](#). Queries started with **Begin** are terminated in the server automatically after a few minutes. A Query ID is returned from each query and is required in subsequent calls to [Next](#) and [End](#). A return value of 0 means the query could not be initiated. This occurs when the user is not logged onto the database.

See also

[SQLTool overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.1.4.2 End

End an SQL query.

```
void End  
(  
    uint dwRequest  
)
```

Parameters

dwRequest

Query ID

Return value

None

Remarks

End a query. This leads to event [OnSQLEnd](#) being sent to the client. Queries which are no longer required should be terminated as quickly as possible because a statement handle is kept in the server for each active query. When calling **End** you should pass as a parameter the query ID returned from [Begin](#). Queries which have not been terminated are terminated automatically after a few minutes.

See also

[SQLTool overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.1.4.3 GetDbBook

Search for telephone number records from name or part of name.

```
void GetDbBook  
(  
    String strText  
)
```

Parameters

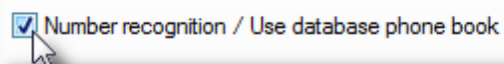
strText
Search term

Return value

None

Remarks

The function searches the calling number identification database for a number. A name or part of a name is supplied and records with names and telephone numbers are returned. Supplying "Wil", for example, would search for names like Williamson and Wilson. If an empty string is supplied, the database creates a complete telephone book and returns it to the client. The returned records lead to event of type [OnDbBookRecord](#) in the client. The database interface of the CTI server must be configured before this function can be used. Caller number identification should be activated and configured on the "Database interface" page of the server control panel.

**See also**

[SQLTool overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.1.4.4 Login

Log on to the database.

```
void Login  
(  
    String strPwd  
)
```

Parameters

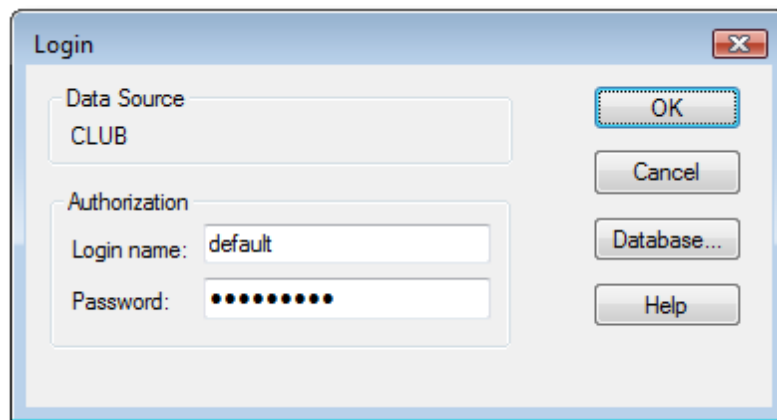
strPwd
Database password

Return value

keiner

Remarks

The client logs on to the database with a password in order to use the SQL interface. The administrator sets the password when configuring the ODBC data source.



An empty string can be supplied in *strPwd* if no password has been set. In response, the client receives an event of type [OnSQLLogin](#).

See also

[SQLTool overview](#)

[Send feedback to TAPIMaster@](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.1.4.5 Next

Continue SQL query.

```
bool Next  
(  
    uint dwRequest,  
    uint dwCount  
)
```

Parameters

dwRequest
Query ID

dwCount
Maximum number of data records

Return value

True on success

Remarks

Continue a query which was started using [Begin](#), supply the query ID returned from **Begin** as the parameter. The maximum number of data records should be limited to the number which can be processed in a reasonable time frame. Each call to **Next** extends the query lifetime. If the client is not logged on to the database, the function fails and returns False.

See also

[SQLTool overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.1.4.6 OnDbBookRecord

Delivers a record from a telephone number search.

Event OnDbBookRecord

```
(  
    object sender,  
    String e.strName,  
    String e.strNumber,  
    String e.strExtra  
)
```

Parameters

sender
object [SQLTool](#)

e.strName
Name of party

e.strNumber
Telephone number of party

e.strExtra
Additional information

Remarks

The user has started a database query of type [GetDbBook](#). The records are returned one by one. *strExtra* may be empty. Extent and format of the data arriving can be configured in the CTI-Servers database settings.

See also

[SQLTool overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.1.4.7 OnSQLData

Answers an SQL query.

Event OnSQLData

```
(  
    object sender,  
    uint e.dwRequest,  
    int e.nCount,  
    String e.strData  
)
```

Parameters

sender

object [SQLTool](#)

e.dwRequest

Number of the request, which was initiated with [Begin](#).

e.nCount

Number of data records returned. 0 means no records were found. -1 means the query failed.

e.strData

Text with the queried data.

Remarks

This event is always sent when data are queried using [Begin](#) or [Next](#). The data may arrive in XML format or as text. The query may fail when the query has already been terminated or when all the data have already been read.

See also

[SQLTool overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.1.4.8 OnSQLEnd

An SQL query was ended.

Event OnSQLEnd

```
(  
    object sender,  
    uint e.dwRequest,  
    Bool e.bSuccess  
)
```

Parameters*sender*object [SQLTool](#)*e.dwRequest*Number of the database query, as assigned during [Begin](#)*e.bSuccess*

True when the query was successfully ended.

Remarks

The event is sent as a consequence of the command [End](#). The event comes of its own accord if all the data have been read using [Begin](#) or [Next](#). If a query does not get terminated, the server terminates it itself after a number of minutes and sends this event.

See also[SQLTool overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.1.4.9 OnSQLLogin

Confirms log-on to the database.

Event OnSQLLogin

```
(
    object sender,
    Bool e.bSuccess
)
```

Parameters*sender*object [SQLTool](#)*e.bSuccess*

True when log-on succeeded.

Remarks

Before a user can use the database, he must log-on with [Login](#). As a result of this, the server sends this event. The log-on is rejected if the wrong password is supplied or if the database connection is not active. The database connection can be established on the “Database interface” page of the server control panel.

**See also**[SQLTool overview](#)

[Send feedback to TAPIMaster®](#)
 © 2020 Tino Kasubke. All rights reserved.

1.2.1.1.5 Telephony

Class Telephony

The class contains all functions and events which have to do directly with the calls.

Functions

AddCallToConference	Adds a call to a conference.
AddNumberToConference	Initiate a call and add it to the conference.
Alternate	Switch back and forth between two calls.
Answer	Answer a call.
DialKey	Sets the key for dialing highlighted text.
DialSpecialKey	Sets the additional key for dialing highlighted text.
DropKey	Sets the key for hanging up calls.
DropSpecialKey	Sets the additional key for dialing highlighted text.
ForwardingDest	Sets the destination device for call forwarding.
ForwardingState	Sets the call forwarding status.
Hangup	Hang up a call.
Hold	Put a call on hold.
LastDialedNumber	The last called number.
LineReset	Resets the TAPI line.
MakeCall	Initiate a call.
OnAcdCancelClient	For a client, the ACD login mode is reset.
OnCallNumber	Signals telephone number.
OnCallReminder	A call reminder has been sent to the user.
OnCallState	Signals the state of one's own calls.
OnConfNotify	Conference member status has changed.
OnForwardingState	Delivers ones own call forwarding settings.
OnGroupMember	Informs clients on start-up of the members of their group and their respective states.
OnIdentity	The caller number identification arrives.
OnObjectOfRedirect	Call forwarding has been activated to this user.

OnProtocolData	Delivers call data records.
OnSendAddress	Sends the address for which the call arrived.
OnSendUserState	The status of a group member has changed.
OnTransferPendInfo	The internal call partner has a waiting caller on the line.
Pickup	Pick up an incoming call which has not yet been answered.
PrivateState	Enable or disable the private call mode.
ProxyMakeCall	Initiate a call from a proxy telephone. Useful for some types of cordless (DECT) telephone.
ReDial	Start redial.
Redirect	Redirect a call.
RemovePhonelcon	Remove the telephone symbol.
SendCallReminder	Send a call reminder.
SetAcdReady	Der Client of a service (ACD) group member informs the server he is ready to receive calls.
SetForwarding	Activate or deactivate call forwarding.
Transfer	Connect two calls.
UnHold	Retrieve held call.
UserCallInfo	Determines the method of notification of incoming calls.

See also

[.NET overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.1.5.1 AddCallToConference

Adds a call to a conference.

```
void AddCallToConference  
(  
    uint dwCallID  
)
```

Parameters

dwCallID

Anruf-ID of the call which should be added to the conference. If a value of 0 is supplied, the CTI function tries either to establish a conference automatically or to add the most likely participant.

Return value

None

Remarks

This function allows existing calls to be added to a conference. In most cases a held call is brought into conference with an active consultation call. Some switches do not support this function. In particular, many manufacturers do not support conferences with more than three participants.

See also

[Telephony overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.1.5.2 AddNumberToConference

Initiate a call and add it to the conference.

```
void AddNumberToConference  
(  
    String strNumber  
)
```

Parameters

strNumber

Telephone number of the party who should be called and added to the conference

Return value

None

Remarks

The function calls *strNumber*. If this called party answers, the party is added to the conference / the active call is extended to a three-way conference. Some switches do not support this function. In particular, many manufacturers do not support conferences with more than three participants.

See also

[Telephony overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.1.5.3 Alternate

Switch back and forth between two calls.

```
void Alternate
(
    uint dwCall1,
    uint dwCall2
);
```

Parameters*dwCall1*

Anruf-ID of the call which is currently active and should be put on hold.

dwCall2

Anruf-ID of the call which is currently on hold and should be made active.

Return value

None

Remarks

The function alternates back and forth between two calls. One call gets connected and the other held. The function will also work if the parameters are supplied the wrong way around. If 0 is supplied for the parameters, the CTI function attempts automatically to find the two calls which should be switched. Some switches do not support this function.

See also[Telephony overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.1.5.4 Answer

Answer a call.

```
void Answer
(
    uint dwCallID
)
```

Parameters*dwCallID*

Anruf-ID Call ID of the call to answer. Supplying 0 attempts to answer the first call in [OFFERING](#) state in one's own queue.

Return value

None

Remarks

Answers an incoming call. The function works only if the telephone can switch to hands-free mode, a feature supported by most telephones.

See also

[Telephony overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.1.5.5 DialKey

Sets the key for dialing highlighted text.

Property DialKey**Type**

int32, read and write

Remarks

If [HotKey](#) is set active, highlighted text can be dialed using a function key. The value corresponds to the virtual key-code of the key and should be in the range 0x70 (F1) to 0x7B (F12). The function key can be combined with a control or shift key.

See also

[Telephony overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.1.5.6 DialSpecialKey

Sets the additional key for dialing highlighted text.

Property DialSpecialKey**Type**

int32, read and write

Remarks

The key for dialing highlighted text can also be combined with special keys, which are set using this property. Possible values are:

0 = No additional key

1 = Shift key

2 = Control key

3 = Shift and control keys

See also

[Telephony overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.1.5.7 DropKey

Sets the key for hanging up calls.

Property DropKey

Type

int32, read and write

Remarks

If [HotKey](#) is set active, a call can be hung up using a function key. The value corresponds to the virtual key-code of the key and should be in the range 0x70 (F1) to 0x7B (F12). The function key can be combined with a control or shift key. The values should be different from [DialKey](#).

See also

[Telephony overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.1.5.8 DropSpecialKey

Sets the additional key for dialing highlighted text.

Property DropSpecialKey

Type

int32, read and write

Remarks

The key for dialing highlighted text can also be combined with special keys, which are set using this property. Possible values are:

0 = No additional key

1 = Shift key

2 = Control key

3 = Shift and control keys

See also

[Telephony overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.1.5.9 ForwardingDest

Sets the destination device for call forwarding.

Property ForwardingDest

Type

String, read and write

Remarks

Shows the call forwarding destination or activates call forwarding with a new destination.

See also

[Telephony overview](#)

[Send feedback to TAPIMaster@](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.1.5.10 ForwardingState

Sets the call forwarding status.

Property ForwardingState**Type**

bool, read and write

Remarks

Activates or deactivates call forwarding without changing the forwarding destination device, or returns the forwarding status.

See also

[Telephony overview](#)

[Send feedback to TAPIMaster@](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.1.5.11 Hangup

Hang up a call.

```
void Hangup  
(  
    uint dwCallID  
);
```

Parameters

dwCallID

Call ID of the call to be hung up. Supply 0 to hang up the first call on the line.

Return value

None

Remarks

Hangs up a call. Almost all drivers support this function.

See also

[Telephony overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.1.5.12 Hold

Put a call on hold.

```
void Hold  
(  
    uint dwCallID  
)
```

Parameters

dwCallID

Call ID of the call to put on hold. If you supply a value of 0, the most likely candidate call is held.

Return value

None

Remarks

Puts an active call on hold. Der caller hears on-hold music. [UnHold](#) retrieves the call. Some switches do not support this function.

See also

[Telephony overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.1.5.13 LastDialedNumber

The last called number.

```
Property LastDialedNumber
```

Type

String, read only

Remarks

Delivers the last called number of the local client.

See also

[Telephony overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.1.5.14 LineReset

Resets the TAPI line.

```
void LineReset()
```

Parameters

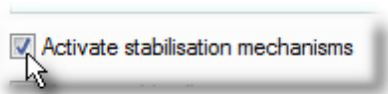
None

Return value

None

Remarks

Occasionally the TAPI driver may fail to respond. This function resets the line, which usually corrects the problem. It is also possible to activate auto-correction in the CTI server, a mechanism which tries to resolve such problems automatically. This can be activated from the server control panel on the "TAPI special treatment" page.

**See also**

[Telephony overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.1.5.15 MakeCall

Initiate a call.

```
void MakeCall  
(  
    String strNumber  
);
```

Parameters

strNumber

Telephone number of the device to call

Return value

None

Remarks

If an active call already exists, an attempt will be made to put this call on hold and to initiate a new call. You can also use this function for consultation calls.

See also

[Telephony overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.1.5.16 OnAcidCancelClient

For a client, the ACD login mode is reset.

```
Event OnAcidCancelClient
(
    object sender
)
```

Parameters

sender

object [Telephony](#)

Remarks

If a client does not answer incoming calls, it is removed on the server from the list of agents present.

See also

[Telephony overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.1.5.17 OnCallNumber

Signals telephone number..

```
Event OnCallNumber
(
    object sender,
    int e.nType,
    zunt e.dwCallID
    String e.strNumber
)
```

Parameters

sender

object [Telephony](#)

e.nType

Type of call, taking one of the following values:

- 1: Primary incoming call
- 2: Call-waiting call when another call is already in progress
- 3: Outbound call

e.dwCallID

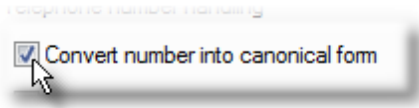
Call ID

e.strNumber

Telephone number signaled, maximum length 31.

Remarks

Signals a telephone number, if one has been received. No number is signaled for calls from the analog network or for anonymous calls. This event is also triggered for outgoing calls. The server can be configured to send the number in canonical format ("PBX settings" page).



A new call is normally signalled via event [OnCallState](#) with a status of [LINECALLSTATE_OFFERING](#) (eingehend) or [LINECALLSTATE_DIALTONE](#) (outgoing). OnCallNumber can, however, also be sent before all other events. The telephone number can change throughout the course of the call, for example, when the call is transferred to the user.

See also

[Telephony overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.1.5.18 OnCallReminder

A call reminder has been sent to the user.

Event OnCallReminder

```
(  
    object sender,  
    String e.bstrUser,  
    String e.strNumber,  
    String e.strMessage  
)
```

Parameters

sender

object [Telephony](#)

e.strUser

Extension of user who sent the call reminder.

e.strNumber

Number of caller whose call is to be returned.

e.strMessage

Additional message about the problem. The sender might describe in this field why *bstrNumber* needs calling back.

Remarks

Users can send call reminders to each other, simplifying the management of calls which need returning. Call reminders are sent using [SendCallReminder](#).

See also

[Telephony overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.1.5.19 OnCallState

Signals the state of one's own calls.

```
Event OnCallState
(  
    object sender,  
    uint e.dwState,  
    uint e.dwCallID  
)
```

Parameters

sender

object [Telephony](#)

e.dwState

The call's [call state](#).

e.dwCallID

Call ID of call whose state has changed.

Remarks

A call has changed state. You should keep a list of active calls in the program. Each time **OnCallState** is received, check if the Call ID is in the list and add it to the list if necessary. If a state of [LINECALLSTATE_IDLE](#) is sent, remove the call from the list. The list will generally only contain one or two calls.

States of group members are sent on event [OnSendUserState](#).

See also

[Telephony overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.1.5.20 OnConfNotify

Conference member status has changed.

```
Event OnConfNotify
(  
    object sender,  
    bool e.bConf  
)
```

Parameters

sender

object [Telephony](#)

e.bConf

True, if the user has entered a conference. False, if the user leaves the conference.

Remarks

The message is sent upon entering and leaving a conference.

See also

[Telephony overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.1.5.21 OnForwardingState

Delivers ones own call forwarding settings.

```
Event OnForwardingState
(  
    object sender,  
    bool e.bSet,  
    String e.strNumber  
)
```

Parameters

sender

object [Telephony](#)

e.bSet

True if forwarding is active.

e.strNumber

Destination telephone number for forwarding.

Remarks

The forwarding settings are stored centrally on the CTI server. These messages are therefore sent after log-on. **OnForwardingState** is also called when forwarding settings are changed.

See also

[Telephony overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.1.5.22 OnGroupMember

Informs clients on start-up of the members of their group and their respective states.

```
Event OnGroupMember
(  
    object sender,  
    String e.strNumber,  
    String e.strName,  
    bool e.bOnline,  
    bool e.bActiveCall,
```

```

    bool e.bRinging,
    bool e.bRedirection
)

```

Parameters

sender

object [Telephony](#)

e.strNumber

Extension of group member, 2 – 4 characters in length.

e.strName

Name of the group member. This is the log-on name. Since this name is determined automatically, it is only available when that group member has already logged on at least once.

e.bOnline

True when the group member is currently logged on.

e.bActiveCall

True when the group member has an active call.

e.bRinging

True when unanswered calls are waiting for the group member in the queue. Many switches allow these calls to be picked up.

e.bRedirection

True when call forwarding is active for the group member.

Remarks**Remarks**

The group member data are sent after each log-on, one message per member. An empty string for *strName* means that the transmission is complete. These events ensure that the program knows the call states of the group members after starting up. [OnSendUserState](#) keeps the program informed of subsequent changes.

See also

[Telephony overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.1.5.23 OnIdentity

The caller number identification arrives.

```

Event OnIdentity
(
    object sender,
    uint e.dwCallID,
    String e.strName
)

```

Parameters

sender

object [Telephony](#)

e.dwCallID

Call ID of the call

e.strName

Caller name, up to 255 characters

Remarks

Caller number identification has been performed on the server. This can come from the ODBC interface or the server programming interface. In addition to the name, *strName* may contain other data.

See also

[Telephony overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.1.5.24 OnObjectOfRedirect

Call forwarding has been activated to this user.

Event OnObjectOfRedirect

```
(  
    object sender,  
    bool e.bSet,  
    String e.strNumber  
)
```

Parameters

sender

object [Telephony](#)

e.bSet

True when forwarding has been activated to this user. False when the forwarding has been deactivated again.

e.strNumber

Extension of user who has activated forwarding to this device.

Remarks

This event is sent when another user activates or deactivates forwarding to this device. The event is also sent after log-on if any users have active forwarding to this device.

See also

[Telephony overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.1.5.25 OnProtocolData

Delivers call data records.

```
Event OnProtocolData
(  
    object sender,  
    String e.strOther,  
    String e.strOwn,  
    uint e.dwStart,  
    uint e.dwEnd,  
    bool e.bSuccess,  
    bool e.bInbound  
)
```

Parameters

sender

object [Telephony](#)

e.strOther

Telephone number of the other party in the call

e.strOwn

Own extension

e.dwStart

Call start time, format time_t

e.dwEnd

Call end time, format time_t

e.bSuccess

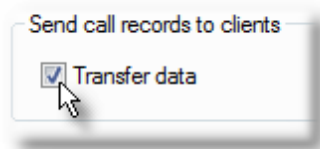
True when the call was established successfully and the parties were connected with each other.

e.bInbound

True for incoming call, false for outgoing.

Remarks

Such a call data record is delivered at the end of each call. If the user is not logged on, the data are buffered on the CTI server sent after the next log-on. The data are only sent when the option is enabled on the server. Delivery can be activated on the “Database interface” page of the server control panel.

**See also**

[Telephony overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.1.5.26 OnSendAddress

Sends the address for which the call arrived.

```
Event OnSendAddress  
(  
    object sender,  
    bool e.blsAddress,  
    uint e.dwCallID,  
    String e.strAddress  
)
```

Parameters

sender

object [Telephony](#)

e.blsAddress

If True, *bstrAddress* contains the telephone address. If False, *bstrAddress* contains the telephone number.

e.dwCallID

Call ID of the call

e.strAddress

Address or number called.

Remarks

This message is sent when a call arrives. An ISDN telephone may answer to several MSN's.

Consider, for example, a home office work place. Here there will be one MSN for private calls and one for business calls. At the end of the month, we need to work out what proportion of the calls were for business. The calls can be assigned correctly using the called MSN in *bstrAddress*.

Another possibility exists in a switch context. A single line may have several internal extensions. 210 may be for normal internal calls and 310 is a hotline connection. *bstrAddress* would contain either 210 or 310.

TAPIMaster® attempts to discover and signal one of these two values. However, most drivers do not support addresses. What can be signaled depends on what the switch, telephone and drivers support. It is also possible for this event not to be sent at all, or alternatively, to be sent several times during a call.

See also

[Telephony overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.1.5.27 OnSendUserState

The status of a group member has changed.

```
Event OnSendUserState
```

```
(  
    object sender,  
    String e.strNumber,  
    bool e.bLogin,  
    bool e.bActiveCall,  
    bool e.bRinging,  
    bool e.bRedirection  
)
```

Parameters*sender*object [Telephony](#)*e.strNumber*

Extension of group member

e.bLogin

True when the group member is logged on.

e.bActiveCall

True when the group member is currently in a call.

e.bRinging

True when the group member has been called and he has not yet answered.

e.bRedirection

True when forwarding is active for the group member.

Remarks

The message is sent when the state of a group member changes. It allows the user to take action when a group member fails to answer a call or when he receives a waiting call whilst he is already on the phone. His availability can also always be seen.

See also[Telephony overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.1.5.28 OnTransferPendInfo

The internal call partner has a waiting caller on the line.

```
Event OnTransferPendInfo  
(  
    object sender,  
    String e.strMember,  
    String e.strHeldParty,
```

Parameters*sender*object [Telephony](#)

e.strMember

Extension of the call partner

e.bLogin

Phone number of the waiting party

Remarks

Please activate the option

TransferPendInfo=1

in the file FCTIINI.TXT (server)..

See also

[Telephony overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.1.5.29 Pickup

Pick up an incoming call which has not yet been answered.

```
void Pickup
(
    String strNumber
)
```

Parameters

strNumber

Extension of the device from which the a call should be picked up.

Return value

None

Remarks

This function allows calls to be picked up from a different device. TAPIMaster® employs to some extent substitute functions with respect to the TAPI, allowing cross-switch pick-up. If this function should fail none the less TAPIMaster® can issue direct hardware commands to pick the call up. On some switches it is possible to pick up waiting calls.

See also

[Telephony overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.1.5.30 PrivateState

Enable or disable the private call mode.

Property PrivateState**Type**

bool, read and write

Remarks

For the private mode other outbound prefixes can be defined. These are then used.

See also

[Telephony overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.1.5.31 ProxyMakeCall

Initiate a call from a proxy telephone. Useful for some types of cordless (DECT) telephone.

```
void ProxyMakeCall  
(  
    String strNumber,  
    String strProxy,  
);
```

Parameters

strNumber

Number of the device to call

strProxy

Extension of the mediating device. Must be an internal phone belonging to the switch.

Return value

None

Remarks

A user device cannot use TAPI to make a call. A proxy device calls the user. The user answers, whereupon the proxy device establishes a consultation call to the actual target device. The call is transferred to the user after dialing has started.

See also

[Telephony overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.1.5.32 ReDial

Start redial.

```
void ReDial()
```

Parameters

None

Return value

None

Remarks

Initiates redial to the number last called.

See also

[Telephony overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.1.5.33 Redirect

Redirect a call.

```
void CallRedirect  
(  
    uint dwCallID  
    String strNumber  
)
```

Parameters

dwCallID

Call ID of the call to redirect.

strNumber

Destination telephone number to which the call should be redirected.

Return value

None

Remarks

The function redirects a call to a different telephone. It is not important whether the call is already connected or not. Good switches also allow redirection of waiting calls. Some switches do not support this function. A value of 0 for *dwCallID* causes the CTI function to redirect the most likely call in OFFERING or ACCEPTED state. If no call is in such a state, the most likely connected call is redirected.

See also

[Telephony overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.1.5.34 RemovePhoneIcon

Remove the telephone symbol.

```
void RemovePhoneIcon()
```

Parameters

None

Return value

None

Remarks

By setting the [PhoneIcon](#) option, a small telephone is displayed when a call arrives. This function allows the symbol to be removed again when the call has been processed.

**See also**

[Telephony overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.1.5.35 SendCallReminder

Send a call reminder.

```
void SendCallReminder  
(  
    String strOther  
    String strNumber  
    String strInfo  
)
```

Parameters

strOther

Durchwahl Extension of the party to inform.

strNumber

Number of caller whose call should be returned.

strInfo

Short message (up to 255 characters) describing the caller's problem.

Return value

None

Remarks

A call arrives which turns out to be for a colleague who is currently not available. This function provides a quick and simple way to notify the colleague about the call. It leads to event [OnCallReminder](#) in the client of the receiving colleague. If the colleague is not logged on, the event will be delivered when he does log on. This method assures less information loss and, because the data sent can be used in the colleague's client as a template for returning the call, the process is faster than using e-mail.

See also

[Telephony overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.1.5.36 SetAcidReady

Der Client of a service (ACD) group member informs the server he is ready to receive calls.

```
void SetAcidReady  
(  
    bool bReady  
)
```

Parameters

bReady

True if the client is available for the hotline

Return value

None

Remarks

The client must be a member of the service group.

See also

[Telephony overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.1.5.37 SetForwarding

Activate or deactivate call forwarding.

```
void SetForwarding  
(  
    bool bSet,  
    String strNumber  
)
```

Parameters

bSet

Activate or deactivate forwarding

strNumber

Destination number to which to forward.

Return value

None

Remarks

Activates or deactivates call forwarding. In response, the user receives event [OnForwardingState](#).

The function may fail if the number already has forwarding active, permission for forwarding has not been granted, or the switch's TAPI function does not support this feature. In the case that the switch

does not support the feature, it is possible to control forwarding via Telefoncodes, which can be set in the Serverkonsole page "TAPI special treatment".

See also

[Telephony overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.1.5.38 Transfer

Connect two calls.

```
void Transfer
(
    uint dwCall1,
    uint dwCall2
)
```

Parameters

dwCall1

Call ID of the first call

dwCall2

Call ID of the second call

Return value

None

Remarks

Connect two calls together. In most cases one call is active and the other on hold (hearing music). The parameter order is unimportant. Some switches do not support this function. If one or both of the parameters are 0, the CTI function connects the two most likely calls.

See also

[Telephony overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.1.5.39 UnHold

Retrieve held call.

```
void CallUnHold
(
    uint dwCallID
)
```

Parameters

dwCallID

Call ID the call to retrieve. If a value of 0 is supplied, the last held call is retrieved.

Return value

None

Remarks

The function can retrieve a call held by [Hold](#) so that the call is active once more. Some switches do not support this function.

See also

[Telephony overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.1.5.40 UserCallInfo

Determines the method of notification of incoming calls.

Property OUserCallInfo**Type**

int32, read and write

Remarks

Determines how the interface client should be informed of incoming calls. Possible values are:

- | | |
|---|---|
| 0 | No notification |
| 1 | The telephone client receives an event, upon which it should raise itself to the foreground |
| 2 | Customer-specific display |

The standard client uses the customer-specific display.

See also

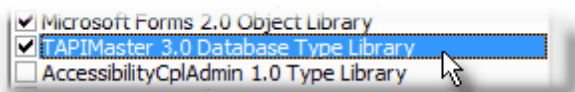
[Telephony overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.2 ActiveX

The ActiveX interface can be found in the file FCTICLNT.DLL. Perform a [client setup](#) on the target computer so that all necessary files are present and registered there.



The functions are ordered according to category. The examples for [VB](#) and [Delphi](#) show the major functions in use.

[Network](#)

[Options](#)

[Telephony functions](#)

[SQL](#)

[Additional functions](#)

[Events](#)

See also

[Command reference - Client](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.2.1 Network

Use of network functions

The network functions allow you to control the connection to the CTI-Server.

Prefix

The network functions start with the prefix "N".

Server identification

NServerName	Name or IP address of the CTI server.
NPortNumber	Change the port number of the network connection.
NNetworkConfigDialog	Display a dialog containing the most important network and connection parameters.

Controlling the network connection

NClientActive	Open or close the network connection.
NIsNetworkVersion	Information about the type of FCTICLNT.DLL.
NLogin	Log a client on to the CTI server.
NLogoff	Log client off from server.
NPrepareRestart	Prepare client for restart.
NStartClient	Initialise the interface.

See also

[ActiveX overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.2.1.1 NClientActive

Open or close the network connection.

Property NClientActive**Type**

Bool, read and write

Remarks

Set this property to False to terminate an existing connection to the CTI-Server. Set the value to True to establish the connection once more.

Example

[Integration in Excel and VB](#)

See also

[Network functions overview](#) | [NStartClient](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.2.1.2 NIsNetworkVersion

Information about the type of FCTICLNT.DLL.

Property NIsNetworkVersion**Type**

Bool, read only

Remarks

Shows whether it is the network version of FCTICLNT.DLL.

See also

[Network functions overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.2.1.3 NLogin

Logs a client on to the CTI server.

Function NLogin()**Parameters**

None

Remarks

This function is performed automatically if [OAutoLogin](#) is active and the network connection exists. **NLogin** activates the functionality for the client in the CTI-Server. The other members of the group then see the client as logged-on. When you use the interface from a program which is active all day long, it may make sense to turn off the CTI functionality occasionally, such as during lunch. The other group members then see that the user is not available. [NLogoff](#) turns CTI off and **NLogin** turns it on again.

See also

[Network functions overview](#) | [OAutoLogin](#) | [NLogoff](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.2.1.4 NLogoff

Log client off from server.

Function NLogoff()**Parameters**

None

Remarks

Turns off the CTI function in the CTI server. The members of the group then see the client as logged-off. Alternatively, the whole network connection can also be torn down if [NClientActive](#) is set to False.

See also

[Network functions overview](#) | [NLogin](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.2.1.5 NNetworkConfigDialog

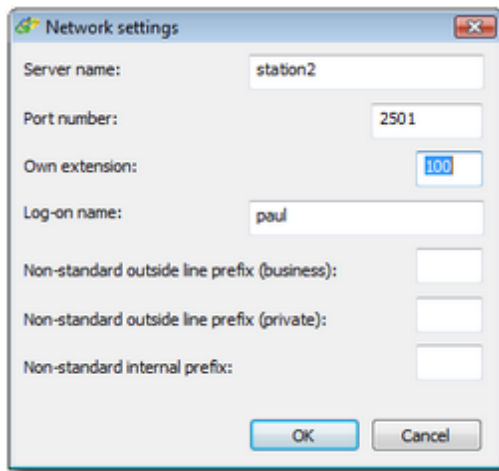
Display a dialog containing the most important network and connection parameters.

Function NNetworkConfigDialog()**Parameters**

None

Remarks

A dialog for entering the connection parameters is opened. Alternatively you can use [NServerName](#), [NPortNumber](#) and [OOwnerNumber](#). If the first free parameters are not supplied and the property [OCheckConnection](#) is set to True, then the dialog appears automatically when the program is started. When the dialog is displayed, the CTI server connection is interrupted. Pressing OK causes a new connection to be established. Use this dialog if you need to change the connection parameters and do not want to have to program a means of entering the parameters yourself.

**Example**

[Simply CTI client \(Delphi\)](#)
[Integration in Excel and VB](#)

See also

[Network functions overview](#) | [NServerName](#) | [NPortNumber](#) | [OOwnerNumber](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.2.1.6 NPortNumber

Change the port number of the network connection.

Property NPortNumber**Type**

Long, read and write

Remarks

The port number of the connection. The connection is not re-established with the new parameters if the value is changed. In order to start a new connection, turn the option [NClientActive](#) off and on again.

See also

[Network functions overview](#) | [NNetworkConfigDialog](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.2.1.7 NPrepareRestart

Prepare client for restart.

Function NPrepareRestart(

Long IMinutes

)

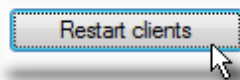
Parameters

IMinutes

Time in minutes which should elapse before the restart. When *IMinutes* = 0, a restart occurs immediately. If the value is negative, the client is not restarted.

Remarks

The CTI server can shut down and restart the standard clients. This can be useful if the clients were started from the network and the files have to be changed. The restart can be triggered from the server control panel, "Users" page.



You can also support these functions. When an [OnClientClose](#) event arrives, the *IPParam* tells you the time until the client should restart. Call [NPrepareRestart](#) passing this time as a parameter, and end the application. The application will be restarted after the time has elapsed.

See also

[Network functions overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.2.1.8 NServerName

Change the name or IP address of the server of the network connection.

Property NServerName

Typ

String, read and write

Remarks

The name of the CTI server. An IP address or "localhost" is also acceptable. The connection is not re-established with the new parameters if the value is changed. In order to start a new connection, turn the option [NClientActive](#) off and on again.

See also

[Network functions overview](#) | [NNetworkConfigDialog](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.2.1.9 NStartClient

Initialises the CTI interface.

Function NStartClient()

Parameters

None

Remarks

This function initialises the [interface](#). From this point on, events can be received.

Example

[Simply CTI client \(Delphi\)](#)

[Integration in Excel and VB](#)

See also

[Network functions overview](#) | [NClientActive](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.2.2 Options

Use of options

Options determine the behaviour of the [CTI interface](#). Options are activated before initialising the interface with [NStartClient](#).

Program options are stored internally by the interface, there is no need for you to save these values yourself. Upon [installation](#), the interface retrieves the initial values from the file FCTIINST.INI. If this file is not present, default settings are used. You do not normally require the options unless your program allows them to be changed. If several CTI programs use the interface on the same computer, each program should make its standard settings explicitly. Most options, with the exception of the internal extension, can also be set using the [Setup Configurator](#), which writes the FCTIINST.INI file.

Prefix

The options functions start with the prefix "O".

Switches

OOptions	Combines the other options as bit flags in a single value.
OAssistedTelephony	Client provides assisted telephony (reacts to calls to <code>tapiRequestMakeCall</code>)
OAutoAnswer	Incoming calls are answered automatically.
OAutoConnect	The network connection is automatically established during program start-up.
OAutoLogin	The client logs on to the server automatically during start-up.
OCheckConnection	Puts the log-on dialog in the foreground when the connection parameters are incomplete.
OHotKey	Use of text-dial keys.
OPhoneIcon	Show call symbol in taskbar.
OShowIcon	Show symbol for network connection.
OShowMenu	Show user-defined menu

Parameter

ODialKey	Function key for dialing
ODialSpecialKey	Additional key for dialing
ODropKey	Function key for hanging up
ODropSpecialKey	Additional key for hanging up
OIniFileName	File in which the CTI interface saves its settings
OOwnerName	Sets ones own login name.
OOwnerNumber	Own extension
OUserCallInfo	Client display upon incoming calls.

Functions

OConnectionIconOptions	User-defined connection symbol, tool-tip and menu
--	---

See also[ActiveX overview](#)

[Send feedback to TAPIMaster®](#)[© 2020 Tino Kasubke. All rights reserved.](#)

1.2.1.2.2.1 OAssistedTelephony

Activate assisted telephony support.

Property OAssistedTelephony

Type

Bool, read and write

Remarks

Accepts command for assisted telephony. The interface initiates a call when a third-party program calls the Windows TAPI function `tapiRequestMakeCall`.

Default value

True

See also

[Options overview](#) | [OOptions](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.2.2.2 OAutoAnswer

Calls are answered automatically.

Property OAutoAnswer

Type

Bool, read and write

Remarks

Incoming calls are answered automatically. This option is of use in call centers. The telephone or headset must be capable of switching to hands-free mode and support must be present in the TAPI driver.

Default value

False

See also

[Options overview](#) | [OOptions](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.2.2.3 OAutoConnect

The network connection is maintained automatically.

Property OAutoConnect

Type

Bool, read and write

Remarks

At program start-up, the CTI interface automatically establishes the connection to the CTI server. If the connection is interrupted, the client tries automatically to re-establish it. There is no need to set [NClientActive](#) to True as well.

Default value

True

Example

[Simply CTI client \(Delphi\)](#)

See also

[Options overview](#) | [OOptions](#) | [NNetworkConfigDialog](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.2.2.4 OAutoLogin

Automatically log on to the CTI server.

Property OAutoLogin**Type**

Bool, read and write

Remarks

After the network connection has been established, the interface logs on to the server automatically. There is no need to call [NLogin](#) separately.

Beispiel

[Simply CTI client \(Delphi\)](#)

Default value

True

See also

[Options overview](#) | [OOptions](#) | [NLogin](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.2.2.5 OCheckConnection

Connection parameters are checked at start-up.

Property OCheckConnection**Type**

Bool, read and write

Remarks

At start-up, the interface checks the connection parameters – server name, port number and internal extension number – for completeness. If a parameter is missing, the interface raises the [logon dialog](#) to the foreground.

Default value

True

Example

[Simply CTI client \(Delphi\)](#)

See also

[Options overview](#) | [OOptions](#) | [NNetworkConfigDialog](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.2.2.6 OConnectionIconOptions

Changes the appearance of the connection icons.

```
Function OConnectionIconOptions(  
    ULong hlconOn,  
    ULong hlconOff,  
    ULong hMenu,  
    String bstrText  
);
```

Parameters

hlconOn

Handle of icon which is displayed when a connection exists to the CTI server. The value 0 can be specified if the standard icon should be used.

hlconOff

Handle of icon which is displayed when the connection to the CTI server is interrupted. The value 0 can be specified if the standard icon should be used.

hMenu

Handle of menu which is displayed over the connection icon if this is clicked with the right mouse button. The value 0 can be specified if no menu should be used.

bstrText

Tool-tip which should be shown over the connection icon. Maximum length: 63 characters. The value 0 can be specified if the standard text should be displayed. Specify an empty string if no tip should be shown.

Remarks

It is possible to alter the appearance, menu and tool-tips of the connection icons. There is also a standard text for the tool-tip. The [OShowIcon](#) option must be active to allow the use of the connection icon. For menus, [OShowMenu](#) is also necessary.

See also

[Options overview](#) | [OShowIcon](#) | [OShowMenu](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.2.2.7 ODialKey

Sets the key for dialing highlighted text.

Property ODialKey

Type

Long, read and write

Remarks

If [OHotKey](#) is set active, highlighted text can be dialed using a function key. The value corresponds to the virtual key-code of the key and should be in the range 0x70 (F1) to 0x7B (F12). The function key can be combined with a control or shift key.

See also

[Options overview](#) | [OOptions](#) | [ODialSpecialKey](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.2.2.8 ODialSpecialKey

Sets the additional key for dialing highlighted text.

Property ODialSpecialKey

Type

Long, read and write

Remarks

The key for dialing highlighted text can also be combined with special keys, which are set using this property. Possible values are:

- 0 = No additional key
- 1 = Shift key
- 2 = Control key
- 3 = Shift and control keys

See also

[Options overview](#) | [ODialKey](#) | [OHotKey](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.2.2.9 ODropKey

Sets the key for hanging up calls.

Property ODropKey

Type

Long, read and write

Remarks

If [OHotKey](#) is set active, a call can be hung up using a function key. The value corresponds to the virtual key-code of the key and should be in the range 0x70 (F1) to 0x7B (F12). The function key can be combined with a control or shift key. The values should be different from [ODialKey](#).

See also

[Options overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.2.2.10 ODropSpecialKey

Sets the additional key for dialing highlighted text.

Property ODropSpecialKey**Type**

Long, read and write

Remarks

The key for dialing highlighted text can also be combined with special keys, which are set using this property. Possible values are:

- 0 = No additional key
- 1 = Shift key
- 2 = Control key
- 3 = Shift and control keys

See also

[Options overview](#) | [ODropKey](#) | [OHotKey](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.2.2.11 OHotKey

Activate text-dial via key-stroke.

Property OHotKey**Type**

Bool, read and write

Remarks

Determines whether highlighted text can be dialed by key-stroke. The key will be in the range F1 to F12. Additionally, the shift and/or control key can be pressed. If the text contains a dialable number, the number will be called. The text-dial and additional key are set using [ODialKey](#) and [ODialSpecialKey](#).

Default value

False

See also

[Options overview](#) | [OOptions](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.2.2.12 OIniFileName

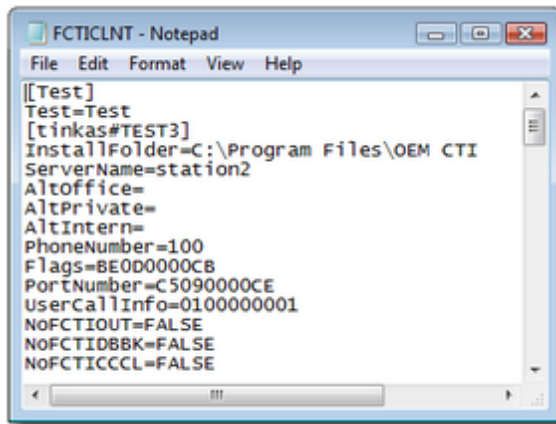
Returns the name of the configuration file.

Property OIniFileName**Type**

String, read only

Remarks

The interface stores its settings in a configuration file. You may use this file yourself if you do not wish to program the creation of a separate file for storing the client settings. The file is in Windows INI format. The section names have the form: user name + # + computer name, for example, [smith#wk12]. Take a look at one of the INI files provided in order to avoid using existing key names.

**See also**

[Options overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.2.2.13 OOptions

The options are processed as a bitmap.

Property OOptions**Type**

Long, read and write

Remarks

The [connectoin options](#) can also be read and written in one go as a bitmap.

Default value

0x0000019E

Example

[Integration in Excel and VB](#)

See also

[Options overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.2.2.14 OOwnerName

Sets ones own login name.

Property OOwnerName**Type**

String, read and write

Remarks

Sets ones own login name. The value will be saved by the CTI interface. The interface will be reconnected.

See also

[Options overview](#) | [NNetworkConfigDialog](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.2.2.15 OOwnerNumber

Sets ones own extension.

Property OOwnerNumber**Type**

String, read and write

Remarks

Own extension, which the server requires to assign lines.

See also

[Options overview](#) | [NNetworkConfigDialog](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.2.2.16 OPhoneIcon

A telephone symbol is displayed when incoming calls arrive.

Property OPhoneIcon**Type**

Bool, read and write

Remarks

A small telephone symbol can be displayed in the task bar when a call arrives. This function parallels the display of a letter by MS Outlook. The telephone number, if present, is shown in the symbol's tool-tip. The icon has a short menu, which allows the call to be returned. [TRemovePhoneIcon](#) removes the icon. The [OnShowPhoneList](#) event is sent when the icon is double-clicked and you should then display the caller list, if present.

**Example**

[Simply CTI client \(Delphi\)](#)

Default value

True

See also

[Options overview](#) | [OOptions](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.2.2.17 OShowIcon

Show a connection symbol.

Property OShowIcon**Type**

Bool, read and write

Remarks

A connection icon is shown in the task bar reflecting the state of the network connection. The icon can also have a user-defined menu and tool-tip. You can also supply your own icons for active and interrupted connections instead of using the standard icons.

**Example**

[Simply CTI client \(Delphi\)](#)

See also

[Options overview](#) | [OOptions](#) | [OConnectionIconOptions](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.2.2.18 OShow Menu

The connection icon has a menu.

Property OShowMenu**Type**

Bool, read and write

Remarks

A menu can be assigned to the connection item in the task bar. The menu appears when the user right-clicks the icon with the mouse. [OConnectionIconOptions](#) is used to specify the menu handle. Menu events are delivered by OnMenuCommand. [OShowIcon](#) must be set to True.

Default value

False

See also

[Options overview](#) | [OOptions](#) | [OConnectionIconOptions](#) | [OShowIcon](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.2.2.19 OUserCallInfo

Determines the method of notification of incoming calls.

Property OUserCallInfo**Type**

Long, read and write

Remarks

Determines how the interface client should be informed of incoming calls. Possible values are:

- 0 No notification
- 1 The telephone client receives an event, upon which it should raise itself to the foreground
- 2 Customer-specific display

The standard client uses the customer-specific display.

Beispiel

[Integration in Excel and VB](#)

See also

[Options overview](#) | [OnCallNumber](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.2.3 Telephony functions

Use of the telephony functions

The TAPIMaster® interface encapsulates the TAPI interface, providing a substantial reduction in the number of interface commands. The functions listed below provide control of the telephone or similar functionality. Not all functions are possible with all switches – it depends on how much the driver supports.

To allow differentiation of calls, each call has a unique ID (not related to the telephone number) in the TAPIMaster® interface. You should store these numbers internally as they are required by many of the telephony commands. The numbers are known as Call IDs.

Prefix

The telephony functions start with the prefix "T".

Single-call functions

TMakeCall	Initiate a new call.
TCallAnswer	Answer a call.
TCallDrop	Hang up a call.

Multicall functions

TAddCallToConference	Add an existing call to a conference.
TAddNumberToConference	Call a new party and add the party to a conference.
TCallHold	Hold call.
TCallRedirect	Redirect call.
TCallToggle	Switch back and forth between active and held calls.
TCallTransfer	Connect two calls.
TCallUnHold	Retrieve a held call.
TPickup	Pick up a call from a third party.
TProxyMakeCall	A call is initiated from another telephone and when established is connected through to the target device.

Utility functions

TLineReset	Reset TAPI line.
TReDial	Start redial.
TRemovePhoneIcon	Remove call symbol.
TSendCallReminder	Send call reminder.
TSetForwarding	Activate or deactivate call forwarding.

Parameter

TForwardingState	Query or set call forwarding status.
----------------------------------	--------------------------------------

TForwardingDest	Query or set call forwarding destination.
---------------------------------	---

See also[ActiveX overview](#)

[Send feedback to TAPIMaster®](#)[© 2020 Tino Kasubke. All rights reserved.](#)

1.2.1.2.3.1 TAddCallToConference

Adds a call to a conference.

```
Function TAddCallToConference(  
    Long ICallID  
)
```

Parameters

ICallID

Anruf-ID of the call which should be added to the conference. If a value of 0 is supplied, the CTI function tries either to establish a conference automatically or to add the most likely participant.

Remarks

This function allows existing calls to be added to a conference. In most cases a held call is brought into conference with an active consultation call. Some switches do not support this function. In particular, many manufacturers do not support conferences with more than three participants.

See also

[Telephony functions overview](#) | [TAddNumberToConference](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.2.3.2 TAddNumberToConference

Initiate a call and add it to the conference.

```
Function TAddNumberToConference(  
    String bstrNumber  
)
```

Parameters

bstrNumber

Telephone number of the party who should be called and added to the conference

Remarks

The function calls *bstrNumber*. If this called party answers, the party is added to the conference / the active call is extended to a three-way conference. Some switches do not support this function. In particular, many manufacturers do not support conferences with more than three participants.

See also

[Telephony functions overview](#) | [TAddCallToConference](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.2.3.3 TCallAnswer

Answer a call.

```
Function TCallAnswer(  
    Long ICallID  
)
```

Parameters

ICallID

Anruf-ID Call ID of the call to answer. Supplying 0 attempts to answer the first call in [OFFERING](#) state in one's own queue.

Remarks

Answers an incoming call. The function works only if the telephone can switch to hands-free mode, a feature supported by most telephones.

Example

[Simply CTI client \(Delphi\)](#)

See also

[Telephony functions overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.2.3.4 TCallDrop

Hang up a call.

```
Function TCallDrop(  
    Long ICallID  
);
```

Parameters

ICallID

Anruf-ID Call ID of the call to be hung up. Supply 0 to hang up the first call on the line.

Remarks

Hangs up a call. Almost all drivers support this function.

Example

[Simply CTI client \(Delphi\)](#)

See also

[Telephony functions overview](#) | [TMakeCall](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.2.3.5 TCallHold

Put a call on hold.

```
Function TCallHold(  
    Long ICallID  
)
```

Parameters

ICallID

Anruf-ID of the call to put on hold. If you supply a value of 0, the most likely candidate call is held.

Remarks

Puts an active call on hold. Der caller hears on-hold music. [TCallUnHold](#) retrieves the call. Some switches do not support this function.

See also

[Telephony functions overview](#) | [TCallToggle](#)

[Send feedback to TAPIMaster@](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.2.3.6 TCallRedirect

Redirect a call.

```
Function TCallRedirect(  
    Long ICallID  
    String bstrNumber  
)
```

Parameters

ICallID

Anruf-ID of the call to redirect.

bstrNumber

Destination telephone number to which the call should be redirected.

Remarks

The function redirects a call to a different telephone. It is not important whether the call is already connected or not. Good switches also allow redirection of waiting calls. Some switches do not support this function. A value of 0 for *ICallID* causes the CTI function to redirect the most likely call in OFFERING or ACCEPTED state. If no call is in such a state, the most likely connected call is redirected.

See also

[Telephony functions overview](#)

[Send feedback to TAPIMaster@](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.2.3.7 TCallToggle

Switch back and forth between two calls.

```
Function TCallToggle(  
    Long IOldCall,  
    Long INewCall  
);
```

Parameters

IOldCall

Anruf-ID of the call which is currently active and should be put on hold.

INewCall

Anruf-ID of the call which is currently on hold and should be made active.

Remarks

The function alternates back and forth between two calls. One call gets connected and the other held. The function will also work if the parameters are supplied the wrong way around. If 0 is supplied for the parameters, the CTI function attempts automatically to find the two calls which should be switched. Some switches do not support this function.

See also

[Telephony functions overview](#) | [TCallHold](#) | [TCallUnHold](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.2.3.8 TCallTransfer

Connect two calls.

```
Function TCallTransfer(  
    Long ICall1,  
    Long ICall2  
)
```

Parameters

ICall1

Anruf-ID of the first call.

ICall2

Anruf-ID of the second call

Remarks

Connect two calls together. In most cases one call is active and the other on hold (hearing music). The parameter order is unimportant. Some switches do not support this function. If one or both of the parameters are 0, the CTI function connects the two most likely calls.

See also

[Telephony functions overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.2.3.9 TCallUnHold

Retrieve held call.

**Function TCallUnHold(
Long ICallID
)**

Parameters

ICallID

Anruf-ID the call to retrieve. If a value of 0 is supplied, the last held call is retrieved.

Remarks

The function can retrieve a call held by [TCallHold](#) so that the call is active once more. Some switches do not support this function

See also

[Telephony functions overview](#) | [TCallToggle](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.2.3.10 TForwardingDest

Sets the destination device for call forwarding.

Property TForwardingDest

Type

String, read and write

Remarks

Shows the call forwarding destination or activates call forwarding with a new destination.

See also

[Telephony functions overview](#) | [TForwardingState](#) | [TSetForwarding](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.2.3.11 TForwardingState

Sets the call forwarding status.

Property TForwardingState

Type

Bool, read and write

Remarks

Activates or deactivates call forwarding without changing the forwarding destination device, or returns the forwarding status.

See also

[Telephony functions overview](#) | [TForwardingDest](#) | [TSetForwarding](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.2.3.12 TLineReset

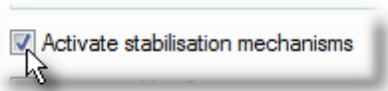
Resets the TAPI line.

Function TLineReset()**Parameters**

None

Remarks

Occasionally the TAPI driver may fail to respond. This function resets the line, which usually corrects the problem. It is also possible to activate auto-correction in the CTI server, a mechanism which tries to resolve such problems automatically. This can be activated from the server control panel on the "TAPI special treatment" page.

**See also**

[Telephony functions overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.2.3.13 TMakeCall

Initiate a call.

**Function TMakeCall(
 String bstrNumber
);****Parameters**

bstrNumber

Telephone number of the device to call.

Remarks

If an active call already exists, an attempt will be made to put this call on hold and to initiate a new call. You can also use this function for consultation calls.

Example

[Simply CTI client \(Delphi\)](#)

[Integration in Excel and VB](#)

See also

[Telephony functions overview](#) | [TCallDrop](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.2.3.14 T Pickup

Pick up an incoming call which has not yet been answered.

```
Function TPickup(  
    String bstrNumber  
)
```

Parameters

bstrNumber

Extension of the device from which the a call should be picked up.

Remarks

This function allows calls to be picked up from a different device. TAPIMaster® employs to some extent substitute functions with respect to the TAPI, thus allowing cross-switch pick-up. If this function should fail none the less TAPIMaster® can issue direct hardware commands to pick the call up. On some switches it is possible to pick up waiting calls.

Example

[Integration in Excel and VB](#)

See also

[Telephony functions overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.2.3.15 T ProxyMakeCall

Initiate a call from a proxy telephone. Useful for some types of cordless (DECT) telephone.

```
Function TMakeCall(  
    String bstrNumber,  
    String bstrProxy,  
)
```

Parameters

bstrNumber

Number of the device to call.

bstrProxy

Extension of the mediating device. Must be an internal phone belonging to the switch.

Remarks

A user device cannot use TAPI to make a call. A proxy device calls the user. The user answers, whereupon the proxy device establishes a consultation call to the actual target device. The call is transferred to the user after dialing has started.

See also

[Telephony functions overview](#) | [TCallDrop](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.2.3.16 TReDial

Start redial.

Function TReDial()**Parameters**

None

Remarks

Initiates redial to the number last called.

See also

[Telephony functions overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.2.3.17 TRemovePhoneIcon

Remove the telephone symbol.

Function TRemovePhoneIcon()**Parameters**

None

Remarks

By setting the [CTICLIENT_PHONEICON](#) option, a small telephone is displayed when a call arrives. This function allows the symbol to be removed again when the call has been processed.

**See also**

[Telephony functions overview](#) | [OOptions](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.2.3.18 TSendCallReminder

Send a call reminder.

```
Function TSendCallReminder(  
    String bstrOther  
    String bstrNumber  
    String bstrMessage  
)
```

Parameters

bstrOther

Extension of the party to inform.

bstrNumber

Number of caller whose call should be returned.

bstrMessage

Short message (up to 255 characters) describing the caller's problem.

Remarks

A call arrives which turns out to be for a colleague who is currently not available. This function provides a quick and simple way to notify the colleague about the call. It leads to event [OnCallReminder](#) in the client of the receiving colleague. If the colleague is not logged on, the event will be delivered when he does log on. This method assures less information loss and, because the data sent can be used in the colleague's client as a template for returning the call, the process is faster than using e-mail.

See also

[Telephony functions overview](#)

[Send feedback to TAPIMaster@](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.2.3.19 TSetForwarding

Activate or deactivate call forwarding.

```
Function TSetForwarding(  
    Bool bSet,  
    String bstrNumber  
)
```

Parameters

bSet

Activate or deactivate forwarding.

bstrNumber

Destination number to which to forward.

Remarks

Activates or deactivates call forwarding. In response, the user receives event [OnForwardingState](#). The function may fail if the number already has forwarding active, permission for forwarding has not been granted, or the switch's TAPI function does not support this feature. In the case that the switch does

not support the feature, it is possible to control forwarding via telephone codes, which can be set in the server control panel page “TAPI special treatment”.

Example

[Integration in Excel and VB](#)

See also

[Telephony functions overview](#) | [TForwardingState](#) | [TForwardingDest](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.2.4 SQL

Use of SQL functions

In telephony applications it is often necessary to access a database, in order, for example, to display further information beyond a telephone number: address, notes, etc. The CTI server already uses database connections in order to identify caller numbers and save call data records. You may use a third database connection to perform your own queries. First configure the data source on the CTI server. This can be set on the "Database interface" page of the server control panel.



It is then recommended to test the commands [SQLLogin](#), [SQLBegin](#), [SQLNext](#) and [SQLEnd](#) using the [CTI Browser](#).

Prefix

The SQL functions start with the prefix "SQL".

Retrieve addresses from the master database

The database for caller number identification can also be used in reverse to get a number from a name.

SQLGetDbBook	Retrieves particular records or a complete address list from the master database.
------------------------------	---

Queries with additional database connections

Use the following commands for your own database queries.

SQLBegin	Start a database query.
SQLEnd	End a query.
SQLLogin	Log client on to database.
SQLNext	Continue a query.

Steps in an example session

1. Log on to database with [SQLLogin](#)("Password");
2. [OnSQLLogin](#) returns True: logon succeeded
3. Request 50 records using [SQLBegin](#)("SELECT * FROM Members",50); Function returns Query ID of 1.
4. Data records arrive in event [OnSQLData](#) an.
5. A further 30 records are requested for query number 1: [SQLNext](#)(1,30);
6. Data records arrive in event [OnSQLData](#).
7. End query number 1 using [SQLEnd](#)(1);
8. Event [OnSQLEnd](#) confirms the end of the query.

See also

[ActiveX overview](#)

[Send feedback to TAPIMaster@](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.2.4.1 SQLBegin

Start an SQL query.

```
Function SQLBegin(  
    String bstrCommand,  
    ULONG dwCount,  
    ULONGPtr pdwRequest  
)
```

Parameters

bstrCommand

SQL command to send to the database

dwCount

Maximum number of records to query

pdwRequest

Query ID as return value

Remarks

The function starts a database query. Any SQL string can be passed in, so long as the administrator has not made any restrictions. Since a large number of records may be discovered, *dwMaxCount* should be used to limit the number. The query can be continued with [SQLNext](#) and ended with [SQLEnd](#). Queries started with **SQLBegin** are terminated in the server automatically after a few minutes. A Query ID is returned from each query and is required in subsequent calls to [SQLNext](#) and [SQLEnd](#). A return value of 0 means the query could not be initiated. This occurs when the user is not logged onto the database.

Example

[SQL sample](#)

See also

[SQL overview](#) | [OnSQLData](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.2.4.2 SQLEnd

End an SQL query.

```
Function SQLEnd(  
    ULONG dwRequest  
)
```

Parameters

dwRequest

Query ID

Remarks

End a query. This leads to event [OnSQLEnd](#). being sent to the client. Queries which are no longer required should be terminated as quickly as possible because a statement handle is kept in the server for each active query. When calling **SQLEnd** you should pass as a parameter the query ID returned from [SQLBegin](#). Queries which have not been terminated are terminated automatically after a few minutes.

Example

[SQL sample](#)

See also

[SQL overview](#) | [OnSQLEnd](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.2.4.3 SQLGetDbBook

Search for telephone number records from name or part of name.

```
Function SQLGetDbBook(  
    String bstrText  
)
```

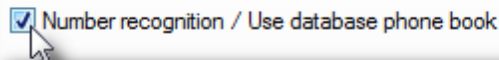
Parameters

bstrText

Search term

Remarks

The function searches the calling number identification database for a number. A name or part of a name is supplied and records with names and telephone numbers are returned. Supplying "Wil", for example, would search for names like Williamson and Wilson. If an empty string is supplied, the database creates a complete telephone book and returns it to the client. The returned records lead to event of type [OnDbBookRecord](#) in the client. The database interface of the CTI server must be configured before this function can be used. Caller number identification should be activated and configured on the "Database interface" page of the server control panel.

**See also**

[SQL overview](#) | [OnDbBookRecord](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.2.4.4 SQLLogin

Log on to the database.

```
Function SQLLogin(  
    String bstrText  
)
```

String *bstrPwd*

)

Parameters

bstrPwd

Database password

Remarks

The client logs on to the database with a password in order to use the SQL interface. The administrator sets the password when configuring the ODBC data source.



An empty string can be supplied in *bstrPwd* if no password has been set. In response, the client receives an event of type [OnSQLLogin](#).

Example

[SQL sample](#)

See also

[SQL overview](#) | [OnSQLLogin](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.2.4.5 SQLNext

Continue SQL query.

```
Function SQLNext(
    ULong dwRequest,
    ULong dwCount,
    BoolPtr pbSuccess
)
```

Parameters

dwRequest

Query ID

dwCount

Maximum number of data records

pbSuccess

Return value. True on success.

Remarks

Continue a query which was started using [SQLBegin](#), supply the query ID returned from [SQLBegin](#) as the parameter. The maximum number of data records should be limited to the number which can be processed in a reasonable time frame. Each call to **SQLNext** extends the query lifetime. If the client is not logged on to the database, the function fails and returns False.

Example

[SQL sample](#)

See also

[SQL overview](#) | [OnSQLData](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.2.5 Additional functions

Use of additional functions

The CTI interface also provides functions which, although largely unrelated to telephony, can be used to add useful program features. For example, chat and the sending of messages are supported.

Prefix

Die additional functions start with the prefix "E".

Sending messages

EChatStart	Start a chat session.
EChatMessage	Send a chat message.
EChatEnd	End an active chat session.
ESendUserToUser	Send a short message.
EUserSendToDataBase	Send user-defined messages.

Other

EExpandInternNumber	Expand an extension.
EGetDateTime	Converts a time_t var into a standard string.
EGetFormatDateTime	Converts a time_t var into an individual formatted string.
ESetAcidReady	Signal readiness to receive calls in an ACD group.
ESetMainWindow	Pass the handle of the main window to the interface.
EShowInfoDialog	Show interface version information.
EConvertToCanonical	Convert a telephone number to canonical form.

See also

[ActiveX overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.2.5.1 EChatEnd

End the active chat session. Only available in the network version.

Function EChatEnd()

Parameters

None

Remarks

Both of the parties chatting may call this function. After the call is made, both parties receive an event of type [OnChatStop](#). The application can then close or deactivate the chat window.

See also

[Additional functions overview](#) | [EChatStart](#) | [EChatMessage](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.2.5.2 EChatMessage

Send chat message. Only available in the network version.

Function EChatMessage(String bstrMessage)

Parameters

bstrMessage

Message text from chat box.

Remarks

The whole text from the chat box should be sent. The function should be called as soon as something changes in the entry field. The function leads to event [OnChatMessage](#) at the other chatting party.

See also

[Additional functions overview](#) | [EChatStart](#) | [EChatEnd](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.2.5.3 EChatStart

Start a chat session. Only available in the network version.

Function EChatStart(String bstrDest)

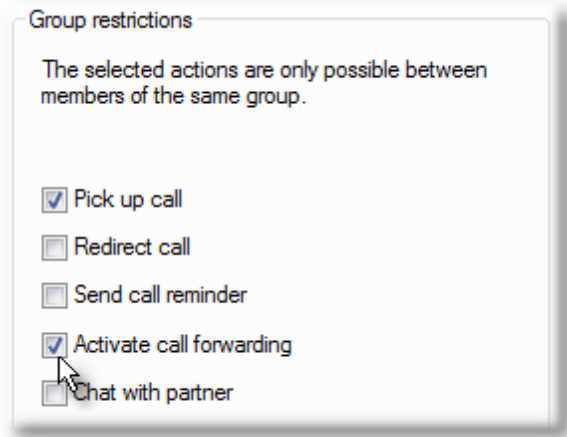
Parameters

bstrDest

Extension of person with whom to chat

Remarks

The person with whom chatting is to start must be logged on and chat permission must be granted. The person must not be in a chat session. If the chat initiation was successful, both chat parties receive an event of type [OnChatStart](#). The input and output fields for the session may then be displayed. If a chat session cannot be raised, this may be due to chat being restricted to members of the same group. You can remove this restriction from the “Groups” page of the server control panel.



See also

[Additional functions overview](#) | [EChatMessage](#) | [EChatEnd](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.2.5.4 EConvertToCanonical

Convert telephone number to canonical form.

```
Function EConvertToCanonical(  
    String bstrInput,  
    StringPtr pbstrOutput  
)
```

Parameters

bstrInput

Number to convert to canonical form

pbstrOutput

Converted number

Remarks

You can use this function to convert a dataset containing telephone numbers to canonical form. The output string should be at least 32 long.

See also

[Additional functions overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.2.5.5 EExpandInternNumber

Expand an extension.

```
Function EExpandInternNumber(  
    String bstrNumber,  
    StringPtr pbstrReturn  
)
```

Parameters

bstrNumber

Internal telephone number to be expanded

pbstrReturn

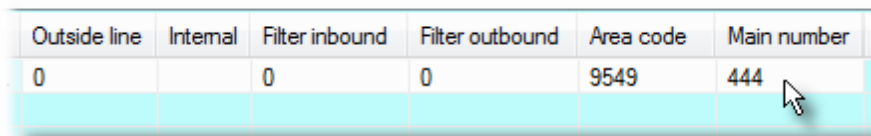
Return value containing the expanded number.

Remarks

Extends a telephone number with the country code, prefix, and, if necessary, main number.

Example

Presuming the settings shown below, EExpandInternNumber("22") returns the value +49 (6033) 44422 in Germany.



Outside line	Internal	Filter inbound	Filter outbound	Area code	Main number
0		0	0	9549	444

See also

[Additional functions overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.2.5.6 EGetDateTime

Converts a time_t var into a standard string.

```
Function EGetDateTime(  
    Long ITime,  
    StringPtr pbstrTimeString,  
)
```

Parameters

ITime

time_t var, e. g. delivered in the event [OnProtocolData](#).

pbstrTimeString

Return value in a standard format: 2007-01-12T00:53:32.875

Remarks

Use this function to get a sortable date format string. Any programs like Excel use this format.

See also

[Additional functions overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.2.5.7 EGetFormatDateTime

Converts a time_t var into an individual formatted string.

```
Function EGetFormatDateTime(
    Long lTime,
    String bstrFormat,
    StringPtr pbstrTimeString,
)
```

Parameter

lTime

time_t var, e. g. delivered in the event [OnProtocolData](#).

bstrFormat

String with date format. This can be formed of the following format characters.

Token	Description
%a	Abbreviated weekday name
%A	Full weekday name
%b	Abbreviated month name
%B	Full month name
%c	Date and time representation appropriate for locale
%d	Day of month as decimal number (01 – 31)
%H	Hour in 24-hour format (00 – 23)
%I	Hour in 12-hour format (01 – 12)
%j	Day of year as decimal number (001 – 366)
%m	Month as decimal number (01 – 12)
%M	Minute as decimal number (00 – 59)
%p	Current locale's A.M./P.M. indicator for 12-hour clock
%S	Second as decimal number (00 – 59)
%U	Week of year as decimal number, with Sunday as first day of week (00 – 53)
%w	Weekday as decimal number (0 – 6; Sunday is 0)
%W	Week of year as decimal number, with Monday as first day of week (00 – 53)

%x	Date representation for current locale
%X	Time representation for current locale
%y	Year without century, as decimal number (00 – 99)
%Y	Year with century, as decimal number
%z,%Z	Either the time-zone name or time zone abbreviation, depending on registry settings; no characters if time zone is unknown
%%	Percent sign

pbstrTimeString

Null-terminated string in a specific format.

Bemerkungen

The format characters can be combined.

Siehe auch

[Additional functions overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.2.5.8 ESendUserToUser

Send a short message. Only available in the network version.

```
Function ESendUserToUser(
    String bstrOther,
    String bstrMessage
)
```

Parameters

bstrOther

Extension of the other party

bstrMessage

Short message (max 220 characters) to send to other party

Remarks

Using this function, various users can send short messages to each other without having to use a mail client. [OnUserToUser](#) is triggered in the receiving client.

See also

[Additional functions overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.2.5.9 ESetAcidReady

Der Client of a service (ACD) group member informs the server he is ready to receive calls. Only available in the network version.

```
Function ESetAcidReady(  
    Bool bReady,  
)
```

Parameters*bReady*

True if the client is available for the hotline.

Remarks

The client must be a member of the service group.

See also

[Additional functions overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.2.5.10 ESetMainWindow

Passes the handle of the main window to the interface.

```
Function ESetMainWindow(  
    ULong hWnd  
)
```

Parameters*hWnd*

Application's window handle

Remarks

Passes the handle of the application's main window to the interface. This is necessary so that the interface can display child dialogs in the correct position.

Example

[Simply CTI client \(Delphi\)](#)

See also

[Additional functions overview](#) | [EShowInfoDialog](#) | [NNetworkConfigDialog](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.2.5.11 EShowInfoDialog

Displays interface version information.

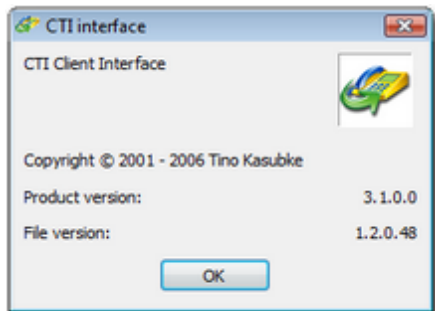
```
Function EShowInfoDialog()
```

Parameters

None

Remarks

Displays a small dialog with information about the interface. Manufacturer and product and file version are displayed.

**See also**

[Additional functions overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.2.5.12 EUserSendToDataBase

Send user-defined messages.

```
Function EUserSendToDataBase(  
    Long ICommand,  
    String bstrText  
)
```

Parameters

ICommand

Command number

bstrText

Text to send.

Remarks

This function sends user-defined message to the server interface, where they cause the [OnUserToDataBase](#) event to be generated. *ICommand* is used to differentiate commands arriving there. The server interface can send an answer to the message, leading to the [OnDatabaseToUser](#) event on the client side.

Example

[Excel client](#)

See also

[Additional functions overview](#) | [ActiveX \(Server\)](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.2.6 Events

Telephony

OnCallNumber	Signals telephone number.
OnCallState	Call state changes.
OnConfNotify	Entering and leaving conference
OnIdentity	Telephone number identification in server database or telephone book CD.
OnLocaleInfo	Switch and local settings
OnCallReminder	Call reminder received.
OnForwardingState	Call forwarding state
OnGroupMember	Members of own group (s)
OnProtocolData	Conveys call data record.
OnSendAddress	Address called
OnSendUserState	Information about the state of group members
OnObjectOfRedirect	Call forwarding set to forward to this user.

Program control

OnClientClose	Client should be closed and maybe restarted.
OnConnectionState	Change in state of network connection.
OnInvalidStringFormat	Client and server are from different products.
OnLogin	Successful log-on
OnMenuCommand	Click on menu of tray icon.
OnRaiseClient	Telephony application should be raised to the foreground.
OnShowPhoneList	Call list should be displayed.

Messages

OnChatStart	Chat session start.
OnChatMessage	Chat message sent.
OnChatStop	Chat session end.
OnLicenseInfo	Conveys license information.
OnServerMessage	Message from the Administrator
OnUserToUser	Messages between users

Database access

OnDatabaseToUser	User-defined command for server interface
OnDbBookRecord	Query telephone number by name

OnSQLData	SQL data from query arrive
OnSQLEnd	SQL query ended
OnSQLLogin	Successful database log-on

See also

[ActiveX overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.2.6.1 OnCallNumber

Signals telephone number.

```
Event OnCallNumber(  
    Long IType,  
    Long ICallID  
    String bstrNumber  
)
```

Parameters

IType

Type of call, taking one of the following values:

- 1: Primary incoming call
- 2: Call-waiting call when another call is already in progress
- 3: Outbound call

ICallID

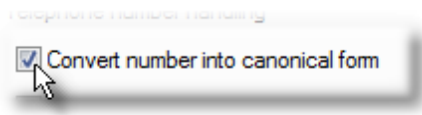
Call ID

bstrNumber

Telephone number signaled, maximum length 31.

Remarks

Signals a telephone number, if one has been received. No number is signaled for calls from the analog network or for anonymous calls. This event is also triggered for outgoing calls. The server can be configured to send the number in canonical format ("PBX settings" page).



A new call is normally signalled via event [OnCallState](#) with a status of [LINECALLSTATE_OFFERING](#) (incoming) or [LINECALLSTATE_DIALTONE](#) (outgoing). OnCallNumber can, however, also be sent before all other events. The telephone number can change throughout the course of the call, for example, when the call is transferred to the user.

Example

[Simply CTI client \(Delphi\)](#)
[Integration in Excel and VB](#)

See also

[Events overview](#) | [OnCallState](#) | [Call states](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.2.6.2 OnCallReminder

A call reminder has been sent to the user.

```
Event OnCallReminder(  
    String bstrUser,  
    String bstrNumber,  
    String bstrMessage  
)
```

Parameters

bstrUser

Extension of user who sent the call reminder.

bstrNumber

Number of caller whose call is to be returned.

bstrMessage

Additional message about the problem. The sender might describe in this field why *bstrNumber* needs calling back.

Remarks

Users can send call reminders to each other, simplifying the management of calls which need returning. Call reminders are sent using [TSendCallReminder](#).

See also

[Events overview](#) | [TSendCallReminder](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.2.6.3 OnCallState

Signals the state of one's own calls.

```
Event OnCallState(  
    Long IState,  
    Long ICallID  
)
```

Parameters

IState

The call's [call state](#)

ICallID

Call ID of call whose state has changed.

Remarks

A call has changed state. You should keep a list of active calls in the program. Each time **OnCallState** is received, check if the call ID is in the list and add it to the list if necessary. If a state of [LINECALLSTATE_IDLE](#) is sent, remove the call from the list. The list will generally only contain one or two calls.

States of group members are sent on event [OnSendUserState](#).

Example

[Simply CTI client \(Delphi\)](#)

See also

[Events overview](#) | [OnSendUserState](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.2.6.4 OnChatMessage

The chat partner is sending a chat message.

```
Event OnChatMessage(  
    String bstrMessage  
)
```

Parameters

bstrMessage

Message, up to 255 characters

Remarks

A chat session is in progress with another user, and the user has sent us a message. Display the message in the output window for these messages. The whole text is sent, not just the last character.

See also

[Events overview](#) | [EChatMessage](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.2.6.5 OnChatStart

A chat session is starting.

```
Event OnChatStart(  
    Bool bStartOK,  
    String bstrUser  
)
```

Parameters

bStartOK

Shows whether the session was successfully opened

bstrUser

Extension of chat partner

Remarks

There are two reasons why this event might be received:

1. You have used [EChatStart](#) to request a chat session with another user. If relevant permissions are not set, the session may be refused and so *bStartOK* will be False. Otherwise, it will be True and you can display the relevant input and output windows for chatting.

2. Someone is trying to start a chat session with you. You can see from *bstrUser* who that is. If you would like to refuse to chat, call [EChatEnd](#). Otherwise, display the windows for input and output for chatting.

See also

[Events overview](#) | [EChatStart](#) | [EChatEnd](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.2.6.6 OnChatStop

The chat session should be terminated.

Event OnChatStop()**Parameters**

None

Remarks

The message is sent to both chat partners when one of them closes the chat session. The chat windows should then be closed.

See also

[Events overview](#) | [EChatEnd](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.2.6.7 OnClientClose

The client should be closed and prepared for restart.

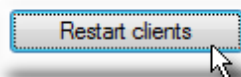
**Event OnClientClose(
Long IMinutes
)****Parameters**

IMinutes

Length of time in minutes until the client should be restarted – if the value is non-negative.

Remarks

The administrator sends this message when he wants to replace important files. This is useful when the clients were started from a shared network directory. To replace the files, all clients must be shut down. If you wish to support this feature, call [NPrepareRestart](#) upon receiving the event, passing *IMinutes* as a parameter. If *IMinutes* is negative, the client should not be restarted. Then close your application. **OnClientClose** is triggered from the server control panel (“User administration” page).

**See also**

[Events overview](#) | [NPrepareRestart](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.2.6.8 OnConfNotify

Conference member status has changed.

```
Event OnConfNotify(  
    Bool bConf  
)
```

Parameters

bConf

True, if the user has entered a conference. False, if the user leaves the conference.

Remarks

The message is sent upon entering and leaving a conference.

See also

[Events overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.2.6.9 OnConnectionState

The connection state has changed.

```
Event OnConnectionState(  
    Long IState  
)
```

Parameters

IState

State of the server connection

Remarks

The state of the server connection has changed.

Possible state values are:

0: Unknown

1: No network connection

2: Connection up, server inactive

3: Connection up, CTI server active

The connection state changes when the client connects to the server or when the server is shut down or when the server takes a break.

Example

[Simply CTI client \(Delphi\)](#)

[Integration in Excel and VB](#)

See also

[Events overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.2.6.10 OnDatabaseToUser

A user-defined message was sent from the server.

```
Event OnDatabaseToUser(  
    Long ICommand,  
    String bstrMessage  
)
```

Parameters

ICommand

Type of user-defined command

bstrMessage

Parameter sent

Remarks

TAPIMaster® has a second interface on the server side. Database queries are normally made via the ODBC server interface. Alternatively, the server interface can be linked to a database or other application. [EUserSendToDataBase](#) can then be used to exchange user-defined data.

See also

[Events overview](#) | [EUserSendToDataBase](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.2.6.11 OnDbBookRecord

Delivers a record from a telephone number search.

```
Event OnDbBookRecord(  
    String bstrName,  
    String bstrNumber,  
    String bstrExtra  
)
```

Parameters

bstrName

Name of party

bstrNumber

Telephone number of party

bstrExtra

Additional information

Remarks

The user has started a database query of type [SQLGetDbBook](#). The records are returned one by one. *bstrExtra* may be empty. Extent and format of the data arriving can be configured in the CTI server's database settings.

See also

[Events overview](#) | [SQLGetDbBook](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.2.6.12 OnForwardingState

Delivers one's own call forwarding settings.

```
Event OnForwardingState(  
    Bool bSet,  
    String bstrNumber  
)
```

Parameters

bSet

True if forwarding is active.

bstrNumber

Destination telephone number for forwarding.

Remarks

The forwarding settings are stored centrally on the CTI server. These messages are therefore sent after log-on. **OnForwardingState** is also called when forwarding settings are changed.

Example

[Integration in Excel and VB](#)

See also

[Events overview](#) | [TSetForwarding](#) | [TForwardingState](#) | [TForwardingDest](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.2.6.13 OnGroupMember

Informs clients on start-up of the members of their group and their respective states.

```
Event OnGroupMember(  
    String bstrNumber,  
    String bstrName,  
    Bool bOnline,  
    Bool bActiveCall,  
    Bool bOfferings,  
    Bool bRedirection  
)
```

Parameters

bstrNumber

Extension of group member, 2 – 4 characters in length.

bstrName

Name of the group member. This is the log-on name. Since this name is determined automatically, it is only available when that group member has already logged on at least once.

bOnline

True when the group member is currently logged on.

bActiveCall

True when the group member has an active call.

bOfferings

True when unanswered calls are waiting for the group member in the queue. Many switches allow these calls to be picked up.

bRedirection

True when call forwarding is active for the group member.

Remarks

The group member data are sent after each log-on, one message per member. An empty string for *bstrName* means that the transmission is complete. These events ensure that the program knows the call states of the group members after starting up. [OnSendUserState](#) keeps the program informed of subsequent changes.

See also

[Events overview](#) | [OnSendUserState](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.2.6.14 OnIdentity

The caller number identification arrives.

```
Event OnIdentity(  
    Long ICallID,  
    String bstrName  
)
```

Parameters*ICallID*

Anruf-ID of the call

bstrName

Caller name, up to 255 characters

Remarks

Caller number identification has been performed on the server. This can come from the ODBC interface or the server programming interface. In addition to the name, *bstrName* may contain other data. The message is also sent when caller identification via telephone book CDs is performed in the client.

Example

[Integration in Excel and VB](#)

See also

[Events overview](#) | [SendIdentity](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.2.6.15 OnInvalidStringFormat

The server and client platforms are incompatible.

Event OnInvalidStringFormat()**Parameters**

None

Remarks

TAPIMaster® exists in ANSI-standard and Unicode versions, the latter operating internally with 2-byte characters. Client and server must be the same version.

See also

[Events overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.2.6.16 OnLicenseInfo

Delivers extended licence information.

**Event OnLicenseInfo(
String bstrData
)****Parameters**

bstrData

Extended licence information. The data are enclosed in curly brackets and separated by semi-colons. Example format:

{License for: Anyfirm;Street: High street 1;City: 12345 Anytown;Phone: +49 (5555) 664488;Web: http://www.anyfirm.de}

Remarks

This message is sent after log-on.

See also

[Events overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.2.6.17 OnLocaleInfo

Delivers switch and country settings.

```
Event OnLocaleInfo(  
    Long lInternLength,  
    Long lCountry,  
    String bstrCity,  
    String bstrPBX  
)
```

Parameters

lInternLength

Maximum length of internal extensions – value must be in the range 2 - 4.

lCountry

Country code, e.g, 44 for England.

bstrCity

Prefix without the leading null

bstrPBX

Switch main number. Useful for switches connected to the public network. In the number +49 (69) 123456-789, 123456 is the main number.

Remarks

Upon log-on this function delivers the local parameters for use by the program.

See also

[Events overview](#)

[Send feedback to TAPI/Master@](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.2.6.18 OnLogin

Confirms log-on to CTI server.

```
Event OnLogin(  
    Bool bSuccess  
)
```

Parameters

bSuccess

Success or failure of log-on: True means the user logged on successfully.

Remarks

The user must be logged on before the majority of commands can be issued and events can be received.

See also

[Events overview](#) | [NLogin](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.2.6.19 OnMenuCommand

The menu of the connection icon was clicked.

```
Event OnMenuCommand(  
    UShort wCommand  
)
```

Parameters

wCommand
Command ID

Remarks

The user has clicked on the menu of the connection symbol. The command triggered is conveyed in *wCommand*.

See also

[Events overview](#) | [OConnectionIconOptions](#) | [OShowIcon](#) | [OShowMenu](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.2.6.20 OnObjectOfRedirect

Call forwarding has been activated to this user.

```
Event OnObjectOfRedirect(  
    Bool bSet,  
    String bstrNumber  
)
```

Parameters

bSet
True when forwarding has been activated to this user. False when the forwarding has been deactivated again.

bstrNumber
Extension of user who has activated forwarding to this device.

Remarks

This event is sent when another user activates or deactivates forwarding to this device. The event is also sent after log-on if any users have active forwarding to this device.

See also

[Events overview](#) | [TSetForwarding](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.2.6.21 OnProtocolData

Delivers call data records.

```
Event OnProtocolData(  
    String bstrOther,  
    String bstrOwn,  
    Long lStart,  
    Long lEnd,  
    Bool bSuccess,  
    Bool blnbound  
)
```

Parameters

bstrOther

Telephone number of the other party in the call

bstrOwn

Own extension

lStart

Call start time, format: time_t

lEnd

Call end time, format: time_t

bSuccess

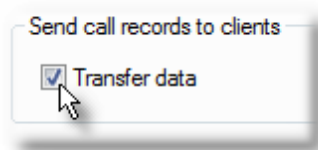
True when the call was established successfully and the parties were connected with each other

blnbound

True for incoming call, false for outgoing

Remarks

Such a call data record is delivered at the end of each call. If the user is not logged on, the data are buffered on the CTI server and sent after the next log-on. The data are only sent when the option is enabled on the server. Delivery can be activated on the "Database interface" page of the server control panel.

**Example**

[Integration in Excel and VB](#)

See also

[Events overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.2.6.22 OnRaiseClient

Tells the client it should raise itself into the foreground.

Event OnRaiseClient()**Parameters**

None

Remarks

Double-clicking on the network symbol triggers this event. The user would like to see the telephony application in the foreground.

See also

[Events overview](#) | [OShowIcon](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.2.6.23 OnSendAddress

Sends the address for which the call arrived.

```
Event OnSendAddress(  
    Bool blsAddress,  
    Long lCallID,  
    String bstrAddress  
)
```

Parameters

blsAddress

If True, *bstrAddress* contains the telephone address. If False, *bstrAddress* contains the telephone number.

lCallID

Call ID of the call

bstrAddress

Address or number called

Remarks

This message is sent when a call arrives. An ISDN telephone may answer to several MSN's.

Consider, for example, a home office work place. Here there will be one MSN for private calls and one for business calls. At the end of the month, we need to work out what proportion of the calls were for business. The calls can be assigned correctly using the called MSN in *bstrAddress*.

Another possibility exists in a switch context. A single Leitung may have several internal extensions. 210 may be for normal internal calls and 310 is a hotline connection. *bstrAddress* would contain either 210 or 310.

TAPIMaster® attempts to discover and signal one of these two values. However, most drivers do not support addresses. What can be signaled depends on what the switch, telephone and drivers support. It is also possible for this event not to be sent at all, or alternatively, to be sent several times during a call.

See also

[Events overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.2.6.24 OnSendUserState

The status of a group member has changed.

```
Event OnSendUserState(  
    String bstrNumber,  
    Bool bLogin,  
    Bool bActiveCall,  
    Bool bOfferings,  
    Bool bRedirection  
)
```

Parameters*bstrNumber*

Extension of group member

bLogin

True when the group member is logged on

bActiveCall

True when the group member is currently in a call

bOfferings

True when the group member has been called and he has not yet answered

bRedirection

True when forwarding is active for the group member

Remarks

The message is sent when the state of a group member changes. It allows the user to take action when a group member fails to answer a call or when he receives a waiting call whilst he is already on the phone. His availability can also always be seen.

Example[Integration in Excel and VB](#)**See also**[Events overview](#) | [OnGroupMember](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.2.6.25 OnServerMessage

The administrator sends a message to the user.

```
Event OnServerMessage(  

```

String bstrMessage

)

Parameters

bstrMessage
Message text

Remarks

The administrator sends a message to the user which is contained in *bstrMessage*. Up to 255 characters can be sent. The messages are sent from the server control panel "User administration" page.

**See also**

[Events overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.2.6.26 OnShow PhoneList

The user has double-clicked the yellow telephone symbol.

Event OnShowPhoneList()**Parameters**

None

Remarks

A telephone symbol may be displayed in the task bar when a call arrives. Double-clicking the icon leads to this event. The program should react by showing the caller list, if there is one.

See also

[Events overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.2.6.27 OnSQLData

Answers an SQL query.

```
Event OnSQLData(
    ULONG dwRequest,
    Long ICount,
    String bstrData
)
```

Parameters

dwRequest

Number of the request, which was initiated with [SQLBegin](#)

ICount

Number of data records returned. 0 means no records were found. -1 means the query failed.

bstrData

Text with the queried data.

Remarks

This event is always sent when data are queried using [SQLBegin](#) or [SQLNext](#). The data may arrive in XML format or as text. The query may fail when the query has already been terminated or when all the data have already been read.

Example

[Database query](#)

See also

[Events overview](#) | [SQLBegin](#) | [SQLNext](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.2.6.28 OnSQLEnd

An SQL query was ended.

```
Event OnSQLEnd(  
    ULONG dwRequest,  
    Bool bSuccess  
)
```

Parameters

dwRequest

Number of the database query, as assigned during [SQLBegin](#)

bSuccess

True when the query was successfully ended.

Remarks

The event is sent as a consequence of the command [SQLEnd](#). The event comes of its own accord if all the data have been read using [SQLBegin](#) or [SQLNext](#). If a query does not get terminated, the server terminates it itself after a number of minutes and sends this event.

See also

[Events overview](#) | [SQLEnd](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.2.6.29 OnSQLLogin

Confirms log-on to the database.

```
Event OnSQLLogin(  
    Bool bSuccess  
)
```

Parameters

bSuccess

True when log-on succeeded.

Remarks

Before a user can use the database, he must log-on with [SQLLogin](#). As a result of this, the server sends this event. The log-on is rejected if the wrong password is supplied or if the database connection is not active. The database connection can be established on the “Database interface” page of the server control panel.

**See also**

[Events overview](#) | [SQLLogin](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.2.6.30 OnUserToUser

Another user sends a message.

```
Event OnUserToUser(  
    String bstrOther,  
    String bstrMessage  
)
```

Parameters

bstrOther

Extension of the other user

bstrMessage

Text message the other user has sent, maximum 255 characters

Remarks

Users can send each other messages using [ESendUserToUser](#). Implementing this function may allow some workplaces to do without a mail client.

See also

[Events overview](#) | [ESendUserToUser](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.3 C++ CTI API

The interface for the C++ API can be found in the file FCTICLNT.DLL. You will also need files CTICLNT.H, CTITYPES.H and FCTICLNT.LIB from the CTI SDK. Copy these files to the relevant directories of your development environment. Perform a client setup on the target machine to install or register all necessary files.

[Network](#)

[Options](#)

[Telephony functions](#)

[SQL](#)

[Additional functions](#)

[Events](#)

[Structures](#)

[Constants](#)

See also

[Command reference - Client](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.3.1 Network

Use of network functions

The network functions allow you to control the connection to the CTI server.

Prefix

The network functions start with the prefix "CTI_N".

Server identification

CTI_NGetServerName	Query server name
CTI_NSetServerName	Change server name
CTI_NGetPortNumber	Query port number
CTI_NSetPortNumber	Change port number
CTI_NSetNetworkOptions	Change network connection options, no new connection
CTI_NNetworkConfigDialog	Show configuration dialog for network connections

Controlling the network connection

CTI_N_StartClientCALLBACK	Initialise the interface, output via callback function
CTI_NStartClientHWND	Initialise the interface, output via Windows messages
CTI_NLogin	Log on to CTI server with user name
CTI_NLogoff	Log off CTI server
CTI_NClientRestart	Reestablish network connection
CTI_NClientStop	Terminate network connection
CTI_NClientActive	Query whether network connection is active
CTI_NPrepareRestart	Prepare client for restart
CTI_NIsNetworkVersion	Is the DLL in use the network version?

See also

[C++ CTI API overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.3.1.1 CTI_NClientActive

Shows the status of the network connection.

```
BOOL WINAPI CTI_NClientActive();
```

Parameters

None

Return value

BOOL

TRUE or FALSE, depending on whether the connection is active or not.

Remarks

The network connection is initialised using [CTI_N_StartClientCALLBACK](#) or [CTI_NStartClientHWND](#). It can be interrupted using [CTI_NClientStop](#) and re-established using [CTI_NClientRestart](#).

See also

[Network functions overview](#) | [WM_CTL_NETWORKCLOSE](#) | [WM_CTL_NETWORKOPEN](#) | [CTI_SERVICEPAUSE](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.3.1.2 CTI_NClientRestart

Re-establishes network connection.

```
VOID WINAPI CTI_NClientRestart();
```

Parameters

None

Return value

None

Remarks

Re-establishes a network connection which was interrupted using [CTI_NClientStop](#).

See also

[Network functions overview](#) | [CTI_NClientStop](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.3.1.3 CTI_NClientStop

Interrupts the network connection.

```
VOID WINAPI CTI_NClientStop();
```

Parameters

None

Return value

None

Remarks

Using this function you can end the network connection to the CTI Server. The function is automatically called when the client interface gets unloaded.

See also

[Network functions overview](#) | [CTI_NClientRestart](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.3.1.4 CTI_NGetPortNumber

Returns the port number of the network connection.

LONG WINAPI CTI_NGetPortNumber();

Parameters

None

Return value

LONG

Port number of the network connection

Remarks

The port number can be changed using [CTI_NSetPortNumber](#).

See also

[Network functions overview](#) | [CTI_NNetworkConfigDialog](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.3.1.5 CTI_NGetServerName

Returns the server name of the network connection.

PTCHAR WINAPI CTI_NGetServerName();

Parameters

None

Return value

PTCHAR

Null-terminated string containing the server name

Remarks

The server name can be changed using **CTI_NGetServerName**. The IP address or “localhost” may also be returned in place of the server name.

See also

[Network functions overview](#) | [CTI_NNetworkConfigDialog](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.3.1.6 CTI_NIsNetworkVersion

Information about the type of FCTICLNT.DLL.

BOOL WINAPI CTI_NIsNetworkVersion();**Parameters**

None

Return value

BOOL

TRUE or FALSE, depending on whether the network version of FCTICLNT.DLL is in use or not.

Remarks

The set of commands available in the single-workplace version is restricted.

See also

[Network functions overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.3.1.7 CTI_NLogin

Logs client on to server.

VOID WINAPI CTI_NLogin();**Parameters**

None

Return value

None

Remarks

This function is performed automatically if [CTICLIENT_AUTOLOGIN](#) is active and the network connection is up. **CTI_NLogin** activates the functionality for the client in the CTI Server. The other members of the group then see the client as logged on. When you use the interface from a program which is active all day long, it may make sense to turn off the CTI functionality occasionally, such as during lunch. The other group members then see that the user is not available. [CTI_NLogoff](#) turns CTI off and **CTI_NLogin** turns it on again.

See also

[Network functions overview](#) | [CTICLIENT_AUTOLOGIN](#) | [CTI_NLogoff](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.3.1.8 CTI_NLogoff

Logs client off from server.

VOID WINAPI CTI_NLogoff();

Parameters

None

Return value

None

Remarks

Turns off the CTI function in the CTI server. The members of the group then see the client as logged off. Alternatively, the whole network connection be torn down using [CTI_NClientStop](#).

See also

[Network functions overview](#) | [CTI_NLogin](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.3.1.9 CTI_NNetworkConfigDialog

Shows the network dialog.

VOID WINAPI CTI_NNetworkConfigDialog();

Parameters

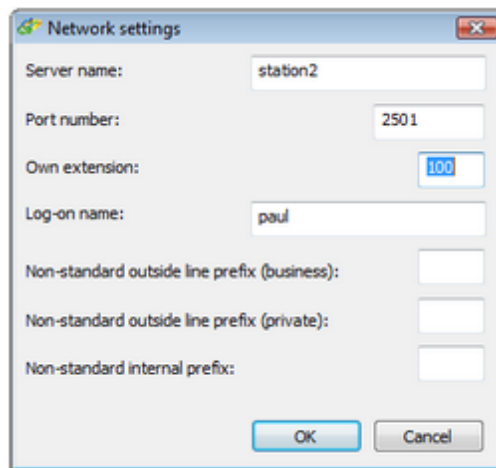
None

Return value

None

Remarks

A dialog for entering the connection parameters is opened. Alternatively you can use [CTI_NSetNetworkOptions](#) and [CTI_OSetOwnerNumber](#). . If the first free parameters are not supplied and the connection option [CTICLIENT_CHECKCONNECTION](#) is active, then the dialog appears automatically when the program is started. When the dialog is displayed, the CTI server connection is interrupted. Pressing OK causes a new connection to be established. Use this dialog if you need to change the connection parameters and do not want to have to program a means of entering the parameters yourself.



Example[SQL sample](#)[Simply CTI client \(C++\)](#)**See also**[Network functions overview](#) | [CTI_OSetOptions](#) | [CTI_NSetNetworkOptions](#),

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.3.1.10 CTI_NPrepareRestart

Prepare client for restart.

```
VOID WINAPI CTI_NPrepareRestart(  
    LONG IMinutes  
);
```

Parameters*IMinutes*

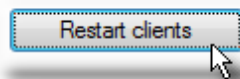
Time in minutes which should elapse before the restart. When *IMinutes* = 0, a restart occurs immediately. If the value is negative, the client is not restarted.

Return value

None

Remarks

The CTI server can shut down and restart the standard clients. This can be useful if the clients were started from the network and the files have to be changed. The restart can be triggered from the server control panel, "User administration" page.



You can also support these functions. When a [CTI_CLIENTCLOSE](#) event arrives, the IParam tells you the time until the client should restart. Call [CTI_NPrepareRestart](#) passing this time as a parameter, and end the application. The application will be restarted after the time has elapsed.

Example[Simply CTI client \(C++\)](#)**See also**[Network functions overview](#) | [CTI_CLIENTCLOSE](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.3.1.11 CTI_NSetNetworkOptions

Re-establishes the network connection using changed connection parameters.

```
VOID WINAPI CTI_NSetNetworkOptions(  
    PDWORD pdwPort,  
    PTCHAR pszServerName  
);
```

Parameters

pdwPort

Server port to which to establish the connection.

pszServerName

Name or IP address of CTI server.

Return value

None

Remarks

The function changes the connection parameters. The existing network connection to the CTI server is interrupted and re-established using the new connection parameters.

See also

[Network functions overview](#) | [CTI_NSetPortNumber](#) | [CTI_NSetServerName](#) | [CTI_NClientRestart](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.3.1.12 CTI_NSetPortNumber

Change the port number of the network connection.

```
VOID WINAPI CTI_NSetPortNumber(  
    LONG IPort  
);
```

Parameters

IPort

Port number for connection.

Return value

None

Remarks

The function changes the port number but does not re-establish the connection. In order to re-establish the connection with the changed parameters, use [CTI_NClientRestart](#).

See also

[Network functions overview](#) | [CTI_NSetNetworkOptions](#) | [CTI_NNetworkConfigDialog](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.3.1.13 CTI_NSetServerName

Changes the server name of the network connection.

```
VOID WINAPI CTI_NSetServerName(  
    PTCHAR pszServerName  
);
```

Parameters

pszServerName
Server name, IP address or "localhost".

Return value

None

Remarks

The function changes the server name but does not re-establish the connection. In order to re-establish the connection with the changed parameters, use [CTI_NClientRestart](#).

See also

[Network functions overview](#) | [CTI_NSetNetworkOptions](#) | [CTI_NNetworkConfigDialog](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.3.1.14 CTI_NStartClientCALLBACK

Initialises the interface.

```
VOID WINAPI CTI_NStartClientCALLBACK(  
    InfoParentProc pProc  
);
```

Parameters

pProc
Address of a callback function with the following format:

```
typedef VOID WINAPI CallbackFunc(DWORD, WPARAM, LPARAM);
```

If the callback is a class member function, it must be static.

Return value

None

Remarks

This function initialises the interface. All events from the interface are then delivered to the callback function. The function may not be used together with [CTI_NStartClientHWND](#).

Example

[Simply CTI client \(C++\)](#)

See also

[Network functions overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.3.1.15 CTI_NStartClientHWND

Initialises the interface.

```
VOID WINAPI CTI_NStartClientHWND(  
    HWND hWndParent  
);
```

Parameters

hWndParent

Handle of window which receives events.

Return value

None

Remarks

This function initialises the interface. All events are then delivered over the corresponding window function. The function may not be used together with [CTI_NStartClientCALLBACK](#).

Example

[SQL sample](#)

See also

[Network functions overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.3.2 Options

Use of options

Options determine the behaviour of the [CTI interface](#). Options are activated before initialising the interface with [CTI_NStartClientCALLBACK](#) or [CTI_NStartClientHWND](#).

Program options are stored internally by the interface, there is no need for you to save these values yourself. Upon [installation](#) the interface retrieves the initial values from the file FCTIINST.INI. If this file is not present, default settings are used. You do not normally require the options functions unless your program allows them to be changed. If several CTI programs use the interface on the same computer, each program should make its standard settings explicitly. Most options, with the exception of the internal extension, can also be set using the [Setup Configurator](#), which writes the FCTIINST.INI file.

Prefix

The options functions start with the prefix "CTI_O".

Setting the bit flags

CTI_OGetOptions	Returns the options as a bit flags.
CTI_OSetOptions	Sets the parameters as a bit flag.

Parameter options

CTI_OConnectionIconOptions	Changes the appearance of the connection icons.
CTI_OGetUserCallInfo	Queries the method of notification of incoming calls.
CTI_OSetUserCallInfo	Determines the method of notifying the user of incoming calls.
CTI_OGetDialKey	Queries the key-stroke for dialing highlighted text.
CTI_OSetDialKey	Sets the key-stroke for dialing highlighted text
CTI_OGetDropKey	Queries the key stroke for hanging up a call.
CTI_OSetDropKey	Sets the key-stroke for hanging up calls.
CTI_OGetIniFileName	Returns the name of the configuration file.
CTI_O_GetOwnerNumber	Queries ones own extension. Only available in the network version.
CTI_OSetOwnerNumber	Sets ones own extension. Only available in the network version.
CTI_OGetOwnerName	Queries one's own login name.
CTI_OSetOwnerName	Sets ones own login name.

See also

[C++ CTI API overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.3.2.1 CTI_OConnectionIconOptions

Changes the appearance of the connection icons.

```
VOID WINAPI CTI_OConnectionIconOptions(  
    HICON hIconOn,  
    HICON hIconOff,  
    HMENU hMenu,  
    PTCHAR pszText  
);
```

Parameters*hIconOn*

Handle of icon which is displayed when a connection exists to the CTI server. The value 0 can be specified if the standard icon should be used.

hIconOff

Handle of icon which is displayed when the connection to the CTI server is interrupted. The value 0 can be specified if the standard icon should be used.

hMenu

Handle of menu which is displayed over the connection icon if this is clicked with the right mouse button. The value 0 can be specified if no menu should be used.

pszText

Tool-tip which should be shown over the connection icon. Maximum length: 63 characters. The value 0 can be specified if the standard text should be displayed. Specify an empty string if no tip should be shown.

Return value

None

Remarks

It is possible to alter the appearance, menu and tool-tips of the connection icons. The [CTICLIENT_SHOWICON](#) option must be active to allow use of the connection icon. For menus, [CTICLIENT_SHOWMENU](#) is also necessary.

Example

[Simply CTI client \(C++\)](#)

See also

[Options overview](#) | [CTI_OSetOptions](#) | [CTICLIENT_SHOWICON](#) | [CTICLIENT_SHOWMENU](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.3.2.2 CTI_OGetDialKey

Queries the key-stroke for dialing highlighted text.

```
LONG WINAPI CTI_OGetDialKey();
```

Parameters

None

Return value

LONG

The less significant word holds the key for dialing highlighted text, the more significant word holds the additional special keys. The special keys can be:

0x00 = No other key

0x01 = Shift key

0x02 = Control key

0x03 = Shift and control keys

Remarks

The values for the hot-key correspond to the virtual key-code of the function key and should be in the range 0x70 (F1) to 0x7B (F12). The key-stroke for dialing highlighted text should not be the same as the key-stroke for hanging up.

See also

[Options overview](#) | [CTICLIENT_HOTKEY](#) | [CTI_OSetDialKey](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.3.2.3 CTI_OGetDropKey

Queries the key stroke for hanging up a call.

LONG WINAPI CTI_OGetDropKey();

Parameters

None

Return value

LONG

The less significant word holds the key for hanging up, the more significant word holds the additional special keys. The special keys can be:

0x00 = No other key

0x01 = Shift key

0x02 = Control key

0x03 = Shift and control keys

Remarks

The values for the drop-key correspond to the virtual key-code of the function key and should be in the range 0x70 (F1) to 0x7B (F12). The key for dialing should not be the same as the key for hanging up.

See also

[Options overview](#) | [CTICLIENT_HOTKEY](#) | [CTI_OSetDropKey](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.3.2.4 CTI_OGetIniFileName

Returns the name of the configuration file.

PTCHAR WINAPI CTI_OGetIniFileName();

Parameters

None

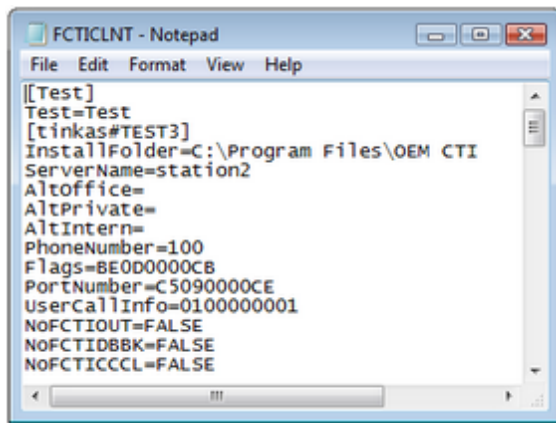
Return value

PTCHAR

Null-terminated string containing the name of the configuration file of the CTI interface.

Remarks

The interface stores its settings in a configuration file. You may use this file yourself if you do not wish to program the creation of a separate file for storing the client settings. The file is in Windows INI format. The section names have the form: user name + # + computer name, for example, [smith#wk12]. Take a look at one of the INI files provided in order to avoid using existing key names.

**See also**

[Options overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.3.2.5 CTI_OGetOptions

Returns the options as a bit flags.

```
DWORD WINAPI CTI_OGetOptions();
```

Parameters

None

Return value

DWORD

The value contains bit flags corresponding to the options.

Remarks

Returns the [connection options](#) for the interface.

See also

[Options overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.3.2.6 CTI_OGetOwnerName

Queries one's own login name.

```
PTCHAR WINAPI CTI_OGetOwnerName();
```

Parameters

None

Return value

PTCHAR

Null-terminated string containing one's own login name

Remarks

Returns one's own extension.

See also

[Options overview](#) | [CTI_NNetworkConfigDialog](#) | [CTI_OSetOwnerName](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.3.2.7 CTI_OGetOwnerNumber

Queries one's own extension.

```
PTCHAR WINAPI CTI_OGetOwnerNumber();
```

Parameters

None

Return value

PTCHAR

Null-terminated string containing one's own extension

Remarks

Returns one's own extension. The string may be 2 – 4 characters long (excluding the terminating null). If the string is empty, one's own extension has not yet been configured. Configuration can be performed using [CTI_OSetOwnerNumber](#).

See also

[Options overview](#) | [CTI_NNetworkConfigDialog](#) | [CTI_OSetOwnerNumber](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.3.2.8 CTI_OGetUserCallInfo

Queries the method of notification of incoming calls.

```
DWORD WINAPI CTI_OGetUserCallInfo();
```

Parameters

None

Return value

DWORD

Determines how the interface client should be informed of incoming calls. Possible values are:

- 0 No notification
- 1 The telephone client receives an event, upon which it should be raise itself to the foreground.
- 2 Customer-specific display

Remarks

The standard client uses the customer-specific display.

See also

[Options overview](#) | [CTI_OSetUserCallInfo](#)

[Send feedback to TAPIMaster@](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.3.2.9 CTI_OSetDialKey

Sets the key-stroke for dialing highlighted text.

```
VOID WINAPI CTI_OSetDialKey(  
    LONG IValue,  
    DWORD _dwSpecialKey = 2  
);
```

Parameters

IValue

Virtual key code of the function key. The value should be in the range 0x70 (F1) - 0x7B (F12).

dwSpecialKey

Additional key(s) to be pressed

0x00 = No additional key

0x01 = Shift key

0x02 = Control key

0x03 = Shift and control key

Return value

None

Remarks

If [CTICLIENT_HOTKEY](#) is set active, highlighted text can be dialed using a function key. The function key can be combined with a control or shift key. The key code should not have the same value as that used in [CTI_OSetDropKey](#). The default additional key is the control key.

See also

[Options overview](#) | [CTI_OSetOptions](#) | [CTI_OGetDialKey](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.3.2.10 CTI_OSetDropKey

Sets the key-stroke for hanging up calls.

```
VOID WINAPI CTI_OSetDropKey(  
    LONG IValue,  
    DWORD _dwSpecialKey = 2  
);
```

Parameters

IValue

Virtual key code of the function key. The value should be in the range 0x70 (F1) - 0x7B (F12).

dwSpecialKey

Additional key(s) to be pressed

0x00 = No additional key

0x01 = Shift key

0x02 = Control key

0x03 = Shift and control key

Return value

None

Remarks

If [CTICLIENT_HOTKEY](#) is active, a call can be dialed using a function key. The function key can be combined with a control and shift key. Do not choose a key-stroke already in use by [CTI_OSetDialKey](#).

See also

[Options overview](#) | [CTI_OSetOptions](#) | [CTI_OGetDropKey](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.3.2.11 CTI_OSetOptions

Sets the parameters as a bit flag.

```
VOID WINAPI CTI_OSetOptions(  
    DWORD dwAdd,  
    DWORD dwRemove  
);
```

Parameters

dwAdd

Value to add.

dwRemove

Flag to remove.

Return value

None

Remarks

Changes take effect immediately. The numerical values of the bit-flags are defined in the [connection options](#). The settings are stored internally by the interface. Within a single call, some options may be set on and others off.

Example

[SQL sample](#)

[Simply CTI client \(C++\)](#)

See also

[Options overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.3.2.12 CTI_OSetOwnerName

Sets ones own login name.

```
VOID WINAPI CTI_OSetOwnerName(  
    PTCHAR pszName  
);
```

Parameters

pszName

Own login name, maximum length: 31.

Return value

None

Remarks

Sets ones own login name. The value will be saved by the CTI interface. The interface will be reconnected.

See also

[Options overview](#) | [CTI_NNetworkConfigDialog](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.3.2.13 CTI_OSetOwnerNumber

Sets ones own extension.

```
VOID WINAPI CTI_OSetOwnerNumber(  
    PTCHAR pszNumber
```

```
);
```

Parameters

pszNumber

Own extension, maximum length: 4.

Return value

None

Remarks

Sets ones own extension. The value can be up to four characters long. Call [CTI_NClientRestart](#) to make the client log on to the CTI server using the new extension.

See also

[Options overview](#) | [CTI_NNetworkConfigDialog](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.3.2.14 CTI_OSetUserCallInfo

Determines the method of notifying the user of incoming calls.

```
VOID WINAPI CTI_OSetUserCallInfo(  
    DWORD dwUserCallInfo  
);
```

Parameters

dwUserCallInfo

Determines how the interface client should be informed of incoming calls.

Possible values are:

- 0 No notification
- 1 The telephone client receives an event, upon which it should be raise itself to the foreground.
- 2 Customer-specific display

Return value

None

Remarks

The standard client uses the customer-specific display.

See also

[Options overview](#) | [CTI_CALLNUMBER](#) | [CTI_OGetUserCallInfo](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.3.3 Telephony functions

Use of telephony functions

The TAPIMaster® interface encapsulates the TAPI interface, providing a substantial reduction in the number of interface commands. The functions listed below provide control of the telephone or similar functionality. Not all functions are possible with all switches – it depends on how much the driver supports. To allow differentiation of calls, each call has a unique ID (not related to the telephone number) in the TAPIMaster® interface. You should store these numbers internally as they are required by many of the telephony commands. The numbers are known as Call IDs.

Prefix

The telephony functions start with the prefix "CTI_T".

Single call functions

CTI_TMakeCall	Initiate a new call.
CTI_TCallAnswer	Answer a call.
CTI_TCallDrop	Hang up a call.

Multicall functions

CTI_TAddCallToConference	Add an existing call to a conference.
CTI_TAddNumberToConference	Call a new party and add the party to a conference.
CTI_TCallHold	Hold call.
CTI_TCallRedirect	Redirect call.
CTI_TCallToggle	Switch back and forth between active and held calls.
CTI_TCallTransfer	Connect two calls.
CTI_TCallUnHold	Retrieve a held call.
CTI_TPickup	Pick up a call from a third party.
CTI_TProxyMakeCall	Initiate call from another telephone and, when established, connect through to the target device.

Additional functions

CTI_TLineReset	Reset TAPI line.
CTI_TReDial	Start redial.
CTI_TRemovePhoneIcon	Remove call symbol.
CTI_TSendCallReminder	Send call reminder.
CTI_TGetForwardingState	Query call forwarding status.
CTI_TGetForwardingDest	Query call forwarding destination.
CTI_TSetForwarding	Activate or deactivate call forwarding.

See also[C++ CTI API overview](#)

[Send feedback to TAPIMaster®](#)[© 2020 Tino Kasubke. All rights reserved.](#)

1.2.1.3.3.1 CTI_TAddCallToConference

Add a call to a conference.

```
VOID WINAPI CTI_TAddCallToConference(  
    LONG ICallID = 0  
);
```

Parameters*ICallID*

Call ID of the call which should be added to the conference. If a value of 0 is supplied, the CTI function tries either to establish a conference automatically or to add the most likely participant.

Return value

None

Remarks

This function allows existing calls to be added to a conference. In most cases a held call is brought into conference with an active consultation call. Some switches do not support this function. In particular, many manufacturers do not support conferences with more than three participants.

See also[Telephony functions overview](#) | [CTI_TAddNumberToConference](#)

[Send feedback to TAPIMaster®](#)[© 2020 Tino Kasubke. All rights reserved.](#)

1.2.1.3.3.2 CTI_TAddNumberToConference

Initiates a call and adds it to the conference.

```
VOID WINAPI CTI_TAddNumberToConference(  
    PTCHAR pszNumber  
);
```

Parameters*pszNumber*

Telephone number of the party who should be called and added to the conference.

Return value

None

Remarks

The function calls *bstrNumber*. If this called party answers, the party is added to the conference / the active call is extended to a three-way conference. Some switches do not support this function. In particular, many manufacturers do not support conferences with more than three participants.

See also

[Telephony functions overview](#) | [CTI_TAddCallToConference](#)

[Send feedback to TAPIMaster@](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.3.3.3 CTI_TCallAnswer

Answers a call.

```
VOID WINAPI CTI_TCallAnswer(  
    LONG ICallID = 0  
);
```

Parameters

ICallID

Call ID of the call to answer. Supplying 0 attempts to answer the first call in [OFFERING](#) state in one's own queue.

Return value

None

Remarks

Answers an incoming call. The function works only if the telephone can switch to hands-free mode, a feature supported by most telephones.

Example

[Simply CTI client \(C++\)](#)

See also

[Telephony functions overview](#) | [CTI_CALLSTATE](#)

[Send feedback to TAPIMaster@](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.3.3.4 CTI_TCallDrop

Hang up a call.

```
VOID WINAPI CTI_TCallDrop(  
    LONG ICallID  
);
```

Parameters

ICallID

Anruf-ID of the call to be hung up. Supply 0 to hang up the first call on the Leitung..

Return value

None

Remarks

Hangs up a call. Almost all drivers support this function.

Example

[Simply CTI client \(C++\)](#)

See also

[Telephony functions overview](#) | [CTI_TMakeCall](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.3.3.5 CTI_TCallHold

Put a call on hold.

```
VOID WINAPI CTI_TCallHold(  
    LONG ICallID = 0  
);
```

Parameters

ICallID

Call ID of the call to put on hold. If you supply a value of 0, the most likely candidate call is held.

Return value

None

Remarks

Puts an active call on hold. The caller hears on-hold music. [CTI_TCallUnHold](#) retrieves the call. Some switches do not support this function.

See also

[Telephony functions overview](#) | [CTI_TCallToggle](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.3.3.6 CTI_TCallRedirect

Redirect a call.

```
VOID WINAPI CTI_TCallRedirect(  
    LONG ICallID = 0,  
    PTCHAR pszNumber  
);
```

Parameters

ICallID

Call ID of the call to redirect.

pszNumber

Destination telephone number to which the call should be redirected.

Return value

None

Remarks

The function redirects a call to a different telephone. It is not important whether the call is already connected or not. Good switches also allow redirection of call-waiting calls. Some switches do not support this function. A value of 0 for *ICallID* causes the CTI function to redirect the most likely call in OFFERING or ACCEPTED state. If no call is in such a state, the most likely connected call is redirected.

See also

[Telephony functions overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.3.3.7 CTI_TCallToggle

Switch back and forth between two calls.

```
VOID WINAPI CTI_TCallToggle(  
    LONG IOldCall = 0,  
    LONG INewCall = 0  
);
```

Parameters

IOldCall

Call ID of the call which is currently active and should be put on hold.

INewCall

Call ID of the call which is currently on hold and should be made active.

Return value

None

Remarks

The function alternates back and forth between two calls. One call gets connected and the other held. The function will also work if the parameters are supplied the wrong way around. If 0 is supplied for the parameters, the CTI function attempts automatically to find the two calls which should be switched. Some switches do not support this function.

See also

[Telephony functions overview](#) | [CTI_TCallHold](#) | [CTI_TCallUnHold](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.3.3.8 CTI_TCallTransfer

Connect two calls.

```
VOID WINAPI CTI_TCallTransfer(  
    LONG ICall1 = 0,  
    LONG ICall2 = 0  
);
```

Parameters*ICall1*

Call ID of the first call.

ICall2

Call ID of the second call.

Return value

None

Remarks

Connect two calls together. In most cases one call is active and the other on hold (hearing music). The parameter order is unimportant. Some switches do not support this function. If one or both of the parameters are 0, the CTI function connects the two most likely calls.

See also[Telephony functions overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.3.3.9 CTI_TCallUnHold

Retrieve held call.

```
VOID WINAPI CTI_TCallUnHold(  
    LONG ICallID = 0  
);
```

Parameters*ICallID*

Call ID of the call to retrieve. If a value of 0 is supplied, the last held call is retrieved.

Return value

None

Remarks

The function can retrieve a call held by [CTI_TCallHold](#) so that the call is active once more. Some switches do not support this function.

See also[Telephony functions overview](#) | [CTI_TCallToggle](#) | [CTI_TCallHold](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.3.3.10 CTI_TGetForwardingDest

Query the call forwarding destination.

```
PTCHAR WINAPI CTI_TGetForwardingDest();
```

Parameters

None

Return value

PTCHAR

String containing the call forwarding destination

Remarks

This function may return a number even when call forwarding is not active. Use [CTI_TGetForwardingState](#) to find out if call forwarding is active.

See also

[Telephony functions overview](#) | [CTI_TSetForwarding](#) | [CTI_TGetForwardingState](#) | [CTI_FORWARDINGSTATE](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.3.3.11 CTI_TGetForwardingState

Query call forwarding status.

```
BOOL WINAPI CTI_TGetForwardingState();
```

Parameters

None

Return value

BOOL

TRUE when forwarding is active.

Remarks

Shows whether call forwarding is active. Use [CTI_TGetForwardingDest](#) to query the call forwarding destination.

See also

[Telephony functions overview](#) | [CTI_TGetForwardingDest](#) | [CTI_TSetForwarding](#) | [CTI_FORWARDINGSTATE](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.3.3.12 CTI_TLineReset

Reset TAPI line.

```
VOID WINAPI CTI_TLineReset();
```

Parameters

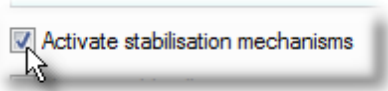
None

Return value

None

Remarks

Occasionally the TAPI driver may fail to respond. This function resets the line, in which usually corrects the problem. It is also possible to activate auto-correction in the CTI server, a mechanism which tries to resolve such problems automatically. This can be activated from the server control panel on the “TAPI special treatment” page.

**See also**

[Telephony functions overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.3.3.13 CTI_TMakeCall

Initiate a call.

```
VOID WINAPI CTI_TMakeCall(  
    PTCHAR pszNumber  
);
```

Parameters

pszNumber

Telephone number of the device to call.

Return value

None

Remarks

If an active call already exists, an attempt will be made to put this call on hold and to initiate a new call. You can also use this function for consultation calls.

Example

[Simply CTI client \(C++\)](#)

See also

[Telephony functions overview](#) | [CTI_TCallDrop](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.3.3.14 CTI_TPickup

Pick up an incoming call which has not yet been answered.

```
VOID WINAPI CTI_T_Pickup(  
    PTCHAR pszPickupNumber  
);
```

Parameters

pszPickupNumber

Extension of the device from which a call should be picked up.

Return value

None

Remarks

This function allows calls to be picked up from a different device. TAPIMaster® employs to some extent substitute functions with respect to the TAPI, thus allowing cross-switch pick-up. If this function should fail none the less, TAPIMaster® can issue direct hardware commands to pick the call up. On some switches it is possible to pick up call-waiting calls.

See also

[Telephony functions overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.3.3.15 CTL_TProxyMakeCall

Initiate a call from a proxy telephone. Useful for some types of cordless (DECT) telephone.

```
VOID WINAPI CTL_TProxyMakeCall(  
    PTCHAR pszNumber,  
    PTCHAR pszProxy,  
);
```

Parameters

pszNumber

Number of the device to call.

pszProxy

Extension of the mediating device. Must be an internal phone belonging to the switch.

Return value

None

Remarks

Say a user device cannot use TAPI to make a call. A proxy device calls the user. The user answers, whereupon the proxy device establishes a consultation call to the actual target device. The call is transferred to the user after dialing has started.

See also

[Telephony functions overview](#) | [CTL_TCallDrop](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.3.3.16 CTL_TReDial

Start redial.

```
VOID WINAPI CTI_TReDial();
```

Parameters

None

Return value

None

Remarks

Initiates redial to the number last called.

See also

[Telephony functions overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.3.3.17 CTI_TRemovePhoneIcon

Remove the telephone symbol.

```
VOID WINAPI CTI_TRemovePhoneIcon();
```

Parameters

None

Return value

None

Remarks

By setting the [CTICLIENT_PHONEICON](#) option, a small telephone is displayed when a call arrives. This function allows the symbol to be removed again when the call has been processed.

**See also**

[Telephony functions overview](#) | [CTI_OSetOptions](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.3.3.18 CTI_TSendCallReminder

Send a call reminder.

```
VOID WINAPI CTI_TSendCallReminder(  
    PTCHAR pszOther,  
    PTCHAR pszNumber,  
    PTCHAR pszInfo = NULL  
);
```

Parameters*pszOther*

Extension of the party to inform.

pszNumber

Number of caller whose call should be returned.

pszInfo

Short message (up to 255 characters) describing the caller's problem.

Return value

None

Remarks

A call arrives which turns out to be for a colleague who is currently not available. This function provides a quick and simple way to notify the colleague about the call. It leads to event [CTI_CALLREMINDER](#) in the client of the receiving colleague. If the colleague is not logged on, the event will be delivered when he does log on. This method assures less information loss and, because the data sent can be used in the colleague's client as a template for returning the call, the process is quicker than using e-mail.

See also[Telephony functions overview](#)[Send feedback to TAPIMaster®](#)[© 2020 Tino Kasubke. All rights reserved.](#)**1.2.1.3.3.19 CTI_TSetForwarding**

Activate or deactivate call forwarding.

```
VOID WINAPI CTI_TSetForwarding(  
    BOOL bSet,  
    PTCHAR pszNumber = _T("")  
);
```

Parameters*bSet*

Activate or deactivate forwarding.

pszNumber

Destination number to which to forward.

Return value

None

Remarks

Activates or deactivates call forwarding. In response, the user receives event [CTI_FORWARDINGSTATE](#). The function may fail if the number already has forwarding active, permission for forwarding has not been granted, or the switch's TAPI function does not support this feature. In the case that the switch does not support the feature, it is possible to control forwarding via telephone codes, which can be set in the server control panel page "TAPI special treatment".

See also

[Telephony functions overview](#) | [CTI_TGetForwardingState](#) | [CTI_TGetForwardingDest](#) | [CTI_FORWARDINGSTATE](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.3.4 SQL

Use of SQL functions

In telephony applications it is often necessary to access a database, in order, for example, to display further information beyond a telephone number: address, notes, etc. The CTI-server already uses database connections in order to identify caller numbers and save call data records. You may use a third database connection to perform your own queries. First configure the data source on the CTI-Server. This can be set on the "Database interface" page of the server control panel.



It is then recommended to test the commands [CTI_SQLLogin](#), [CTI_SQLBegin](#), [CTI_SQLNext](#) and [CTI_SQLEnd](#) using the [CTI-Browser](#).

Prefix

The SQL functions start with the prefix "SQL_".

Retrieve addresses from the master database

The database for caller number identification can also be used in reverse to get a number from a name.

CTI_SQLGetDbBook	Retrieves particular records or a complete address list from the master database.
----------------------------------	---

Queries with additional database connections

Use the following commands for your own database queries.

CTI_SQLLogin	Log client on to database.
CTI_SQLBegin	Start a database query
CTI_SQLNext	Continue a query
CTI_SQLEnd	End a query

Steps in an example session

1. Log on to database with [CTI_SQLLogin](#)("Password");
2. [SQL_LOGIN](#) returns TRUE in wParam: logon succeeded
3. Request 50 records using [CTI_SQLBegin](#)("SELECT * FROM Members",50); Function returns Query ID of 1.
4. Data records arrive in event [SQL_DATA](#)
5. A further 30 records are requested for query number 1: [CTI_SQLNext](#)(1,30);
6. Data records arrive in event [SQL_DATA](#)
7. End query number 1 using [CTI_SQLEnd](#)(1);
8. Event [SQL_END](#) confirms the end of the query.

See also

[C++ CTI API overview](#)

[Send feedback to TAPIMaster@](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.3.4.1 CTI_SQLBegin

Start an SQL query.

```
DWORD WINAPI CTI_SQLBegin(  
    PTCHAR pszCommand,  
    DWORD dwMaxCount  
);
```

Parameters

pszCommand
SQL command to send to the database

dwMaxCount
Maximum number of records to query

Return value

DWORD
Abfrage-ID

Remarks

The function starts a database query. Any SQL string can be passed in, so long as the administrator has not made any restrictions. Since a large number of records may be discovered, *dwMaxCount* should be used to limit the number. The query can be continued with [CTI_SQLNext](#) and ended with [CTI_SQLEnd](#). Queries started with **CTI_SQLBegin** are terminated in the server automatically after a few minutes. A Query ID is returned from each query and is required in subsequent calls to [CTI_SQLNext](#) and [CTI_SQLEnd](#). A return value of 0 means the query could not be initiated. This occurs when the user is not logged on to the database.

Example

[SQL sample](#)

See also

[SQL overview](#) | [SQL_BEGIN](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.3.4.2 CTI_SQLEnd

End an SQL query.

```
VOID WINAPI CTI_SQLEnd(  
    DWORD dwRequestID  
);
```

Parameters

dwRequestID
Query ID, as returned by [CTI_SQLBegin](#).

Return value

None

Remarks

End a query. This leads to event [SQL_END](#) being sent to the client. Queries which are no longer required should be terminated as quickly as possible because a statement handle is kept in the connection to the database for each active query. Queries which have not been terminated are terminated automatically after a few minutes.

Example

[SQL sample](#)

See also

[SQL overview](#) | [SQL_END](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.3.4.3 CTI_SQLGetDbBook

Search for telephone number records from name or part of name.

```
VOID WINAPI CTI_SQLGetDbBook(  
    PTCHAR pszText  
);
```

Parameters

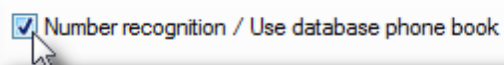
pszText
Search Term

Return value

None

Remarks

The function searches the calling number identification database for a number. A name or part of a name is supplied and records with names and telephone numbers are returned. Supplying "Wil", for example, would search for names like Williamson and Wilson. If an empty string is supplied, the database creates a complete telephone book and returns it to the client. The returned records lead to event of type [SQL_DBBOOK](#) in the client. The database interface of the CTI server must be configured before this function can be used. Caller number identification should be activated and configured on the "Database interface" page of the server control panel.

**See also**

[SQL overview](#) | [SQL_DBBOOK](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.3.4.4 CTI_SQLLogin

Log on to the database.

```
VOID WINAPI CTI_SQLLogin(  
    PTCHAR pszPassword  
);
```

Parameters

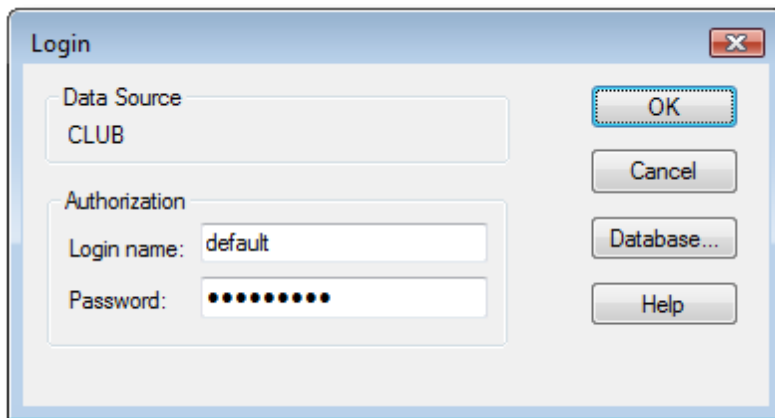
pszPassword
Database password

Return value

None

Remarks

The client logs on to the database with a password in order to use the SQL interface. The administrator sets the password when configuring the ODBC data source.



An empty string can be supplied in *pszPassword* if no password has been set. In response, the client receives an event of type [SQL_LOGIN](#).

Example

[SQL sample](#)

See also

[SQL overview](#) | [SQL_LOGIN](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.3.4.5 CTI_SQLNext

Continue SQL query.

```
BOOL WINAPI CTI_SQLNext(  
    DWORD dwRequestID,  
    DWORD dwMaxNextCount
```

```
);
```

Parameters

dwRequestID

Query ID

dwMaxNextCount

Maximum number of data records

Return value

BOOL

TRUE, if the query was successfully sent

Bemerkungen

Continue a query which was started using [CTI_SQLBegin](#). Supply the query ID returned from [CTI_SQLBegin](#) as the parameter. The maximum number of data records should be limited to the number which can be processed in a reasonable time frame. Each call to **CTI_SQLNext** extends the query lifetime. If the client is not logged on to the database, the function fails and returns FALSE.

Example

[SQL sample](#)

See also

[SQL overview](#) | [SQL DATA](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.3.5 Additional functions

Use of additional functions

The CTI interface also provides functions which, although largely unrelated to telephony, can be used to add useful program features. For example, chat and the sending of messages are supported.

Prefix

Die additional functions start with the prefix "CTI_E".

Sending messages

CTI_EChatStart	Start a chat session
CTI_EChatMessage	Send a chat message
CTI_EChatEnd	End an active chat session
CTI_ESendUserToUser	Send a short message
CTI_EUserSendToDatabase	Send user-defined messages

Other

CTI_EExpandInternNumber	Expand an extension
CTI_EGetDateTime	Converts a time_t var into a standard string.
CTI_EGetFormatDateTime	Converts a time_t var into an individual formatted string.
CTI_ESetAcidReady	Signal readiness to receive calls in an ACD group
CTI_ESetMainWindow	Pass the handle of the main window to the interface
CTI_EShowInfoDialog	Show interface version information
CTI_EConvertToCanonical	Convert a telephone number to canonical form

See also

[C++ CTI API overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.3.5.1 CTI_EChatEnd

End the active chat session.

```
VOID WINAPI CTI_EChatEnd();
```

Parameters

None

Return value

None

Remarks

End the active chat session. Both of the parties chatting may call this function. After the call is made, both parties receive an event of type [CTI_CHATSTOP](#). The application can then close or deactivate the chat window.

See also

[Additional functions overview](#) | [CTI_EChatStart](#) | [CTI_EChatMessage](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.3.5.2 CTI_EChatMessage

Send chat message.

```
VOID WINAPI CTI_EChatMessage(  
    PTCHAR pszMsg  
);
```

Parameters

pszMsg

Message text from chat box

Return value

None

Remarks

The whole text from the chat box should be sent. The function should be called as soon as something changes in the entry field. The function leads to event [CTI_CHATMESSAGE](#) at the other chatting party.

See also

[Additional functions overview](#) | [CTI_EChatStart](#) | [CTI_EChatEnd](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.3.5.3 CTI_EChatStart

Start a chat session.

```
VOID WINAPI CTI_EChatStart(  
    PTCHAR pszDest  
);
```

Parameters

pszDest

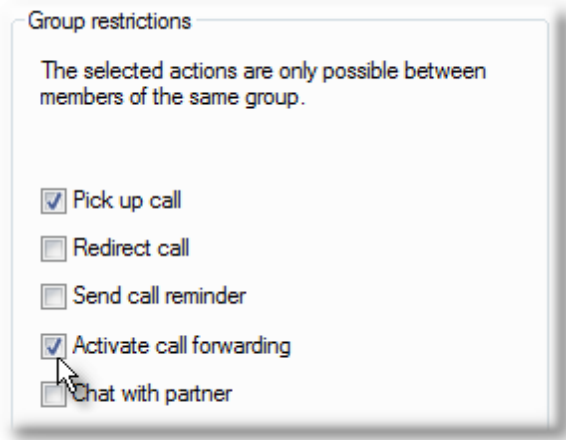
Extension person with whom to chat

Return value

None

Remarks

Start a chat session. The person with whom chatting is to start must be logged on and chat permission must be granted. The person must not be in a chat session. If the chat initiation was successful, both chat parties receive an event of type [CTI_CHATSTART](#). The input and output fields for the session may then be displayed. If a chat session cannot be raised, this may be due to chat being restricted to members of the same group. You can remove this restriction from the “Groups” page of the server control panel.



See also

[Additional functions overview](#) | [CTI_EChatMessage](#) | [CTI_EChatEnd](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.3.5.4 CTI_EConvertToCanonical

Convert telephone number to canonical form.

```
VOID WINAPI CTI_EConvertToCanonical(
    PTCHAR pszInput,
    PTCHAR pszOutput
)
```

Parameters

pszInput

Number to convert to canonical form

pszOutput

Converted number

Return value

None

Remarks

You can use this function to convert a dataset containing telephone numbers to canonical form. The output string should be at least 32 characters long.

See also

[Additional functions overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.3.5.5 CTI_EExpandInternNumber

Expand an extension.

```
PTCHAR WINAPI CTI_EExpandInternNumber(  
    PTCHAR pszNumber  
);
```

Parameters

pszNumber

Internal telephone number to be expanded

Return value

PTCHAR

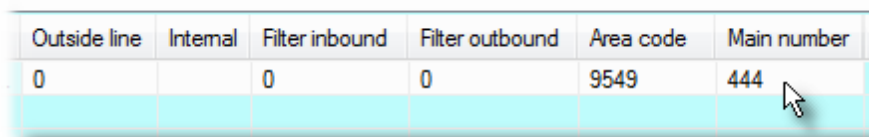
Null-terminate string containing the expanded number

Remarks

Extends a telephone number with the country code, prefix, and, if necessary, main number.

Example

Presuming the settings shown below, CTI_EExpandInternNumber("22") returns the value +49 (6033) 44422 in Germany.



Outside line	Internal	Filter inbound	Filter outbound	Area code	Main number
0		0	0	9549	444

See also

[Additional functions overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.3.5.6 CTI_EGetDateTime

Converts a time_t var into a standard string.

```
PTCHAR CTI_EGetDateTime(  
    __time32_t wTime,  
);
```

Parameters*wTime*time_t var, e. g. delivered in the event [CTI_PROTOCOLDATA](#).**Return value**

Zero terminated string in a standard format: 2007-01-12T00:53:32.875

Remarks

Use this function to get a sortable date format string. Any programs like Excel use this format.

See also[Additional functions overview](#)[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.3.5.7 CTI_EGetFormatDateTime

Converts a time_t var into an individual formatted string.

```
PTCHAR CTI_EGetFormatDateTime(
    __time32_t wTime,
    PTCHAR pszFormat
);
```

Parameter*wTime*time_t var, e. g. delivered in the event [CTI_PROTOCOLDATA](#).*pszFormat*

Date format. The format can be created with this format tokens.

Token	Description
%a	Abbreviated weekday name
%A	Full weekday name
%b	Abbreviated month name
%B	Full month name
%c	Date and time representation appropriate for locale
%d	Day of month as decimal number (01 – 31)
%H	Hour in 24-hour format (00 – 23)
%I	Hour in 12-hour format (01 – 12)
%j	Day of year as decimal number (001 – 366)
%m	Month as decimal number (01 – 12)
%M	Minute as decimal number (00 – 59)
%p	Current locale's A.M./P.M. indicator for 12-hour clock
%S	Second as decimal number (00 – 59)

%U	Week of year as decimal number, with Sunday as first day of week (00 – 53)
%w	Weekday as decimal number (0 – 6; Sunday is 0)
%W	Week of year as decimal number, with Monday as first day of week (00 – 53)
%x	Date representation for current locale
%X	Time representation for current locale
%y	Year without century, as decimal number (00 – 99)
%Y	Year with century, as decimal number
%z,%Z	Either the time-zone name or time zone abbreviation, depending on registry settings; no characters if time zone is unknown
%%	Percent sign

Return value

Zero terminated string in an individual format.

Remarks

The format tokens can be combined.

See also

[Additional functions overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.3.5.8 CTI_ESendUserToUser

Send a short message.

```
VOID WINAPI CTI_ESendUserToUser(  
    PTCHAR pszOtherUser,  
    PTCHAR pszMessage  
);
```

Parameters

pszOtherUser

Extension of the other party

pszMessage

Short message (max 220 characters) to send to other party

Return value

None

Remarks

Using this function, various users can send short messages to each other without having to use a mail client. [CTI_USERTOUSER](#) is triggered in the receiving client.

See also

[Additional functions overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.3.5.9 CTI_ESetAcidReady

The client of a service (ACD) group member informs the server he is ready to receive calls.

```
VOID WINAPI CTI_ESetAcidReady(  
    BOOL bReady,  
);
```

Parameters

bReady

TRUE if the client is available for the hotline.

Return value

None

Remarks

The client must be a member of the service group.

See also

[Additional functions overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.3.5.10 CTI_ESetMainWindow

Passes the handle of the main window to the interface.

```
VOID WINAPI CTI_ESetMainWindow(  
    HWND hWnd,  
);
```

Parameters

hWnd

Applications window handle

Return value

None

Remarks

Passes the handle of the application's main window to the interface. This is necessary so that the interface can display child dialogs in the correct position.

Example

[SQL sample](#)

[Simply CTI client \(C++\)](#)

See also

[Additional functions overview](#) | [CTI_EShowInfoDialog](#) | [CTI_NNetworkConfigDialog](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.3.5.11 CTI_EShowInfoDialog

Displays interface version information.

```
VOID WINAPI CTI_EShowInfoDialog();
```

Parameters

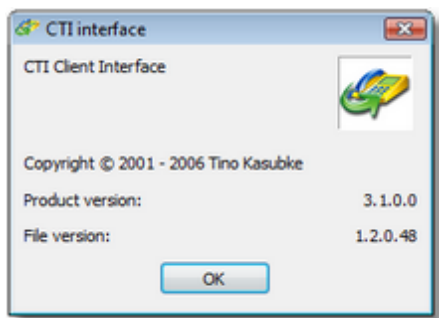
None

Return value

None

Remarks

Displays a small dialog with information about the interface.



See also

[Additional functions overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.3.5.12 CTI_EUserSendToDataBase

Send user-defined messages.

```
VOID WINAPI CTI_EUserSendToDataBase(  
    LONG ICommand,  
    PTCHAR pszMessage  
);
```

Parameters

ICommand

Command number

pszMessage

Text to send

Return value

None

Remarks

This function sends user-defined messages to the server interface, where they cause the [CTI_USERTODATABASE](#) event to be generated. *ICommand* is used to differentiate commands arriving there. The server interface can send an answer to the message, leading to the [CTI_DATABASETOUSER](#) event on the client side.

See also

[Additional functions overview](#) | [C++ CTI API \(Server\)](#) | [CTI_DATABASETOUSER](#) | [CTI_USERTODATABASE](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.3.6 Events

Use of events

The client interface events may arrive as Windows messages or via a callback function, depending on how the interface was initialised. The callback function is like the Windows function except that the window handle is missing:

```
typedef VOID WINAPI CallbackFunc(DWORD, WPARAM, LPARAM);
```

If the callback is a class member function, it must be static.

Types

The Windows messages listed below are among those which may be delivered in DWORD.

Type	Value	Description
WM_CTI_MENUCOMMAND	WM_USER + 31	Message from the context menu of the connection symbol
WM_CTI_SHOWPHONELIST	WM_USER + 32	Message from call symbol
WM_CTI_NETWORKCLOSE	WM_USER + 23	Network connection interrupted
WM_CTI_NETWORKOPEN	WM_USER + 22	Network connection active
WM_CTI_RAISECLIENT	WM_USER + 33	Client should be raised to the foreground
WM_CTI_NETWORK_DISPATCH	WM_USER + 21	CTI message reception

See also

[C++ CTI API overview](#) | [CTI NStartClientCALLBACK](#) | [CTI NStartClientHWND](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.3.6.1 WM_CTI_MENUCOMMAND

Receive menu messages.

Message WM_CTI_MENUCOMMAND (WM_USER + 31)

WPARAM

Less significant word contains the command ID.

LPARAM

Not used.

Remarks

This message is sent when the user clicks the menu of the connection icon

Example

[Simply CTI client \(C++\)](#)

See also

[Events overview](#) | [CTIClient_SHOWICON](#) | [CTIClient_SHOWMENU](#) | [CTI_OConnectionIconOptions](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.3.6.2 WM_CTI_SHOWPHONELIST

The caller list should be displayed.

Message WM_CTI_SHOWPHONELIST (WM_USER + 32)**WPARAM**

Not used

LPARAM

Not used

Remarks

This message is triggered by a double-click on the telephone symbol in the task bar. This symbol is displayed when a call arrives if [CTICLIENT_PHONEICON](#) is active. The program should react by showing the caller list, if there is one.

**Example**

[Simply CTI client \(C++\)](#)

See also

[Events overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.3.6.3 WM_CTI_NETWORKCLOSE

Informs the client that the network connection has been interrupted.

Message WM_CTI_NETWORKCLOSE (WM_USER + 23)**WPARAM**

Not used

LPARAM

Not used

Remarks

The connection was interrupted. If [CTIClient_AUTOCONNECT](#) is active, the interface attempts to reconnect at regular intervals.

Example

[SQL sample](#)

[Simply CTI client \(C++\)](#)

See also

[Events overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.3.6.4 WM_CTI_NETWORKOPEN

Informs the client that the network connection is active again.

Message WM_CTI_NETWORKOPEN (WM_USER + 22)**WPARAM**

Not used

LPARAM

Not used

Remarks

The network connection is active. [CTI NLogin](#) can now be called, unless the [CTICLIENT_AUTOLOGIN](#) option is active, in which case this is not necessary.

Example

[SQL sample](#)

[Simply CTI client \(C++\)](#)

See also

[Events overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.3.6.5 WM_CTI_RAISECLIENT

Tells the client it should raise itself into the foreground.

Message WM_CTI_RAISECLIENT (WM_USER + 33)**WPARAM**

Not used

LPARAM

Not used

Remarks

The command is usually sent following a double-click on the network symbol. The application should then be raised to the foreground.

See also

[Events overview](#) | [CTICLIENT_SHOWICON](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.3.6.6 WM_CTI_NETWORK_DISPATCH

CTI message sent.

Message WM_CTI_NETWORK_DISPATCH (WM_USER + 21)
WPARAM

Contains the size of the structure sent.

LPARAM

Pointer to a [CTI_COMMAND_MSG](#) or [CTI_PHONE_DATA](#) structure.

Remarks

Almost all CTI events arrive in these structures. The interpretation depends on the m_dwCommand parameter, which is located at the same position in both structures. LPARAM should first be cast to LP_CTI_COMMAND_MSG in order to interpret m_dwCommand. The possible values of m_dwCommand are listed in the table below:

Parameter	Value	Description	Interpretation of LPARAM
CTI_LOGIN	0x00011001	Successful log-on	LP_CTI_COMMAND_MSG
CTI_CALLNUMBER	0x00012001	Telephone number sent	LP_CTI_COMMAND_MSG
CTI_FORWARDINGSTATE	0x00012004	Notification of call forwarding	LP_CTI_COMMAND_MSG
CTI_OBJECTOFREDIRECT	0x00012006	Notification of call forwarding destination	LP_CTI_COMMAND_MSG
CTI_PROTOCOLDATA	0x0001200B	Call data records sent	LP_CTI_PHONE_DATA
CTI_CALLREMINDER	0x0001200C	Call reminder sent	LP_CTI_PHONE_DATA
CTI_IDENTITY	0x0001200D	Telephone number identification	LP_CTI_COMMAND_MSG
CTI_LICENSEINFO	0x0001300E	Licence information	LP_CTI_COMMAND_MSG
CTI_SENDUSERSTATE	0x00012011	Status of group member	LP_CTI_COMMAND_MSG
CTI_CONFNOTIFY	0x00012014	Joining conference	LP_CTI_COMMAND_MSG
CTI_SENDADDRESS	0x00012016	MSN or address of call	LP_CTI_COMMAND_MSG
CTI_SERVICEPAUSE	0x00013001	CTI server is taking a break	LP_CTI_COMMAND_MSG
CTI_CLIENTCLOSE	0x00013002	Client stop and restart	LP_CTI_COMMAND_MSG
CTI_SERVERMESSAGE	0x00013003	Message from the administrator	LP_CTI_COMMAND_MSG

CTI_INVALIDSTRINGFORMAT	0x00013004	Character widths differ	LP_CTI_COMMAND_MSG
CTI_USERTOUSER	0x00013005	Messages between users	LP_CTI_COMMAND_MSG
CTI_DATABASETOUSER	0x00013007	Messages from server interface	LP_CTI_COMMAND_MSG
CTI_GROUPMEMBER	0x00013008	Group list	LP_CTI_COMMAND_MSG
CTI_CHATSTART	0x0001300A	Chat session start	LP_CTI_COMMAND_MSG
CTI_CHATSTOP	0x0001300B	Chat session end	LP_CTI_COMMAND_MSG
CTI_CHATMESSAGE	0x0001300C	Chat message sent	LP_CTI_COMMAND_MSG
CTI_LOCALEINFO	0x0001300D	Switch parameter	LP_CTI_COMMAND_MSG
CTI_CALLSTATE	0x00030002	Status of own calls	LP_CTI_COMMAND_MSG
SQL_LOGIN	0x00040001	SQL log-on success	LP_CTI_COMMAND_MSG
SQL_DATA	0x00040002	Data records	LP_CTI_COMMAND_MSG
SQL_END	0x00040004	Query terminated	LP_CTI_COMMAND_MSG
SQL_DBBOOK	0x00040006	Telephone numbers from database	LP_CTI_COMMAND_MSG

Example[SQL sample](#)[Simply CTI client \(C++\)](#)**See also**[Events overview](#)[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

Signals telephone number.

Sub-message CTI_CALLNUMBER (0x00012001)**Message type**

CTI_COMMAND_MSG

Used values*m_wParam*

If the least significant word is TRUE, it is a primary call. If it is FALSE, the number is for a waiting call. The most significant word signifies the call direction: TRUE denotes an incoming call.

m_lParam

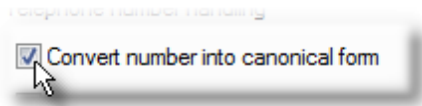
Call ID.

m_szMessage

Telephone number of other party.

Remarks

The message is only sent when a telephone number has been signalled. This should always be the case for an outgoing call. This message may be the first information received about a call. That is, it may not have been preceded by a [CTI_CALLSTATE](#) message. The telephone number can change throughout the course of the call, for example, when someone has transferred the call to the user. The server can be configured to send the number in canonical format ("Switch settings" page).



Example

[Simply CTI client \(C++\)](#)

See also

[WM_CTINetworkDispatch overview](#)

[Send feedback to TAPIMaster@](#)

© 2020 Tino Kasubke. All rights reserved.

A call reminder has been sent to the user.

Sub-message CTI_CALLREMINDER (0x0001200C)

Message type

CTI_PHONE_DATA

Used values

m_unDest

Extension of user who sent the call reminder.

m_szNumber

Number of caller whose call is to be returned.

m_szMessage

Additional message about the problem. The sender might describe in this field why *m_szNumber* needs calling back.

Remarks

Users can send call reminders to each other, simplifying the management of calls which need returning. Call reminders are sent using [CTI_TSendCallReminder](#).

See also

[WM_CTINetworkDispatch overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

Signals the state of one's own calls.

Sub-message CTI_CALLSTATE (0x00030002)**Message type**

CTI_COMMAND_MSG

Used values

m_wParam

The call's [call state](#)

m_lParam

Call ID of the

Remarks

A call has changed state. You should keep a list of active calls in the program. Each time **CTI_CALLSTATE** is received, check if a call with that ID is in the list and add it to the list if necessary. If a state of [LINECALLSTATE_IDLE](#) is sent, remove the call from the list. The list will generally only contain one or two calls. States of group members are sent on event [CTI_SENDUSERSTATE](#).

Example

[Simply CTI client \(C++\)](#)

See also

[WM_CTI_NETWORK_DISPATCH overview](#) | [Gesprächszustände](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

The chat session should be terminated.

Sub-message CTI_CHATSTOP (0x0001300B)**Message type**

CTI_COMMAND_MSG

Used values

None

Remarks

The message is sent to both chat partners when one of them closes the chat session. The chat windows should then be closed.

See also

[WM_CTL_NETWORK_DISPATCH overview](#) | [CTI_EChatEnd](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

The chat partner is sending a chat message.

Sub-message CTI_CHATMESSAGE (0x0001300C)

Message type

CTI_COMMAND_MSG

Used values

m_szMessage

Message, up to 255 characters

Remarks

A chat session is in progress with another user, and the user has sent us a message. Display the message in the output window for these messages. The whole text is sent, not just the last character.

See also

[WM_CTL_NETWORK_DISPATCH overview](#) | [CTI_EChatMessage](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

Start a chat session with another user.

Sub-message CTI_CHATSTART (0x0001300A)

Message type

CTI_COMMAND_MSG

Used values

m_wParam

TRUE if chat was successfully started.

m_unNumber

Extension of chat partner

Remarks

There are two reasons why this event might be received:

1. You have used [CTI_EChatStart](#) to request a chat session with another user. If relevant permissions are not set or the partner is already chatting, the session may be refused and so *m_wParam* will be FALSE. Otherwise, it will be TRUE and you can display the relevant input and output windows for chatting.
2. Someone is trying to start a chat session with you. You can see from *m_unNumber* who that is. If you would like to refuse to chat, call [CTI_EChatEnd](#). Otherwise, display the windows for input and output for chatting.

See also

[WM_CTI_NETWORK_DISPATCH overview](#) | [CTI_EChatStart](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

The client should be closed and prepared for restart.

Sub-message CTI_CLIENTCLOSE (0x00013002)**Message type**

CTI_COMMAND_MSG

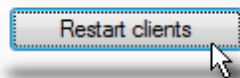
Used values

m_IParam

Length time in minutes until the client should be restarted – if the value is non-negative.

Remarks

The administrator sends this message when he wants to replace important files. This is useful when the clients were started from a shared network directory. To replace the files, all clients must be shut down. If you wish to support this feature, call [CTI_NPrepareRestart](#) upon receiving the event, passing *m_IParam* as a parameter. If *m_IParam* is negative, the client should not be restarted. Then close your application. **CTI_CLIENTCLOSE** is triggered from the server control panel (“Users” page).

**Example**

[Simply CTI client \(C++\)](#)

See also

[WM_CTI_NETWORK_DISPATCH overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

Conference member status has changed.

Sub-message CTI_CONFNOTIFY (0x00012014)**Message type**

CTI_COMMAND_MSG

Used values

m_wParam

TRUE, the user has entered a conference. FALSE, if the user leaves the conference.

Remarks

The message is sent upon entering and leaving a conference.

See also

[WM_CTL_NETWORK_DISPATCH overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

A user-defined message was sent from the server.

Sub-message CTI_DATABASETouser (0x00013007)**Message type**

CTI_COMMAND_MSG

Used values

m_lParam

Type of user-defined command

m_szMessage

Parameter sent

Remarks

TAPIMaster® has a second interface on the server side. Database queries are normally made via the ODBC server interface. Alternatively, the server interface can be linked to a database or other application. [CTI_EUserSendToDataBase](#) can then be used to exchange user-defined data.

See also

[WM_CTL_NETWORK_DISPATCH overview](#) | [CTI_EUserSendToDataBase](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

Delivers one's own call forwarding settings.

Sub-message CTI_FORWARDINGSTATE (0x00012004)**Message type**

CTI_COMMAND_MSG

Used values

m_wParam

TRUE if forwarding is active.

m_szMessage

Destination telephone number for forwarding.

Remarks

The forwarding settings are stored centrally on the CTI server. These messages are therefore sent after log-on. **CTI_FORWARDINGSTATE** is also called when forwarding settings are changed.

See also

[WM_CTL_NETWORK_DISPATCH overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

Informs clients on start-up of the members of their group and their respective states.

Sub-message CTI_GROUPMEMBER (0x00013008)

Message type

CTI_COMMAND_MSG

Used values

m_wParam

Bit field. Meaning of flags:

0x00000001 Group member is logged on.

0x00000002 There are active calls.

0x00000004 There are incoming calls which have not been answered.

m_lParam

If bit 0x01000000 is set, call forwarding is active for the group member.

m_unNumber

Extension of group member

m_szMessage

Name of the group member. This is the log-on name. Since this name is determined automatically, it is only available when that group member has already logged on at least once.

Remarks

The group member data are sent after each log-on, one message per member. An empty string for *m_unNumber* means that the transmission is complete. These events ensure that the program knows the call states of the group members after starting up [CTI_SENDUSERSTATE](#) keeps the program informed of subsequent changes.

See also

[WM_CTI_NETWORK_DISPATCH overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

The caller number identification arrives.

Sub-message CTI_IDENTITY (0x0001200D)

Message type

CTI_COMMAND_MSG

Used values

m_lParam

Call ID of the call

m_szMessage

Caller name, up to 255 characters

Remarks

Caller number identification has been performed on the server. This can come from the ODBC interface or the server programming interface. In addition to the name, *m_szMessage* may contain other data. The message is also sent when caller identification via telephone book CDs is performed in the client.

See also

[WM_CTL_NETWORK_DISPATCH overview](#) | [CTI_OSetOptions](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

The server and client platforms are incompatible.

Sub-message CTI_INVALIDSTRINGFORMAT (0x00013004)**Message type**

CTI_COMMAND_MSG

Used values

None

Remarks

TAPIMaster® exists in ANSI-standard and Unicode versions, the latter operating internally with 2-byte characters. Client and server must be the same version.

See also

[WM_CTL_NETWORK_DISPATCH overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

Delivers extended licence information.

Sub-message CTI_LICENSEINFO (0x0001300E)**Message type**

CTI_COMMAND_MSG

Used values

m_szMessage

Extended licence information. The data are enclosed in curly brackets and separated by semi-colons. Example format:

{License for: Anyfirm;Street: High street 1;City: 12345 Anytown;Phone: +49 (5555) 664488;Web: <http://www.anyfirm.de>}

Remarks

This message is sent after log-on.

See also

[WM_CTL_NETWORK_DISPATCH overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

Delivers switch and country settings.

Sub-message CTI_LOCALEINFO (0x0001300D)**Message type**

CTI_COMMAND_MSG

Used values

m_wParam

The less significant word contains the length of internal extensions, the more significant word contains the country code.

m_lParam

Switch main number. Used for switches connected to the public network. In the number +49 (69) 123456-789, 123456 is the main number.

m_szMessage

Prefix, without the leading null

Remarks

Upon log-on, this function delivers the local parameters for use by the program.

See also

[WM_CTI_NETWORK_DISPATCH overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

Confirms log-on to CTI server.

Sub-message CTI_LOGIN (0x00011001)**Message type**

CTI_COMMAND_MSG

Used values

m_wParam

Success failure of log-on: TRUE means the user logged on successfully.

m_lParam

TRUE for the licensed version

Remarks

The user must be logged on before the majority of commands can be issued and events can be received.

See also

[WM_CTI_NETWORK_DISPATCH overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

Call forwarding has been activated to this user.

Sub-message CTI_OBJECTOFREDIRECT (0x00012006)

Message type

CTI_COMMAND_MSG

Used values

m_wParam

TRUE when forwarding has been activated to this user. FALSE when the forwarding has been deactivated again.

m_szNumber

Extension of user who has activated forwarding to this user.

Remarks

This event is sent when another user activates or deactivates forwarding to this device. The event is also sent after log-on if any users have active forwarding to this device.

See also

[WM_CTI_NETWORK_DISPATCH overview](#) | [CTI_TSetForwarding](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

Delivers call data records.

Sub-message CTI_PROTOCOLDATA (0x0001200B)

Message type

CTI_PHONE_DATA

Used values

m_dwStart

Call start time, format: time_t

m_dwEnd

Call end time, format: time_t

m_dwFlags

Bit flags for success and direction of call:

0x00000002 The call was successful

0x00000004 The call was incoming

m_unActor

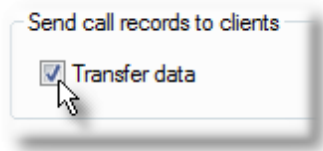
Own extension

m_szNumber

Telephone number of the other party in the call

Remarks

Such a call data record is delivered at the end of each call. If the user is not logged on, the data are buffered on the CTI server and sent after the next log-on. The data are only sent when the option is enabled on the server. Delivery can be activated on the "Database interface" page of the server control panel.



See also

[WM_CTI_NETWORK_DISPATCH overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

Sends the address for which the call arrived.

Sub-message CTI_SENDADDRESS (0x00012016)

Message type

CTI_COMMAND_MSG

Used values

m_wParam

If TRUE, *m_szMessage* contains the telephone address. If FALSE, *m_szMessage* contains the telephone.

m_lParam

Call ID of the call

m_szMessage

Address or number called

Remarks

This message is sent when a call arrives. An ISDN telephone may answer to several MSN's.

Consider, for example, a home office work place. Here there will be one MSN for private calls and one for business calls. At the end of the month, we need to work out what proportion of the calls were for business. The calls can be assigned correctly using the called MSN in *m_szMessage*.

Another possibility exists in a switch context. A single line may have several internal extensions. 210 may be for normal internal calls and 310 is a hotline connection. *m_szMessage* would contain either 210 or 310.

TAPIMaster® attempts to discover and signal one of these two values. However, most drivers do not support addresses. What can be signalled depends on what the switch, telephone and drivers support. It is also possible for this event not to be sent at all, or alternatively, to be sent several times during a call.

See also

[WM_CTL_NETWORK_DISPATCH overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

The status of a group member has changed.

Sub-message CTI_SENDUSERSTATE (0x00012011)**Message type**

CTI_COMMAND_MSG

Used values

m_wParam

Bit field. Meaning of flags:

0x00000001 Group member is logged on.

0x00000002 There are active calls.

0x00000004 There are incoming calls which have not been answered.

m_lParam

If bit 0x01000000 is set, call forwarding is active for the group member.

m_unNumber

Extension of group member

Remarks

The message is sent when the state of a group member changes. It allows the user to take action when a group member fails to answer a call or when he receives a waiting call whilst he is already on the phone. His availability can also always be seen.

See also

[WM_CTL_NETWORK_DISPATCH overview](#), [CTI_GROUPMEMBER](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

The administrator sends a message to the user.

Sub-message CTI_SERVERMESSAGE (0x00013003)**Message type**

CTI_COMMAND_MSG

Used values

m_szMessage

Message, of length up to 255 characters.

Remarks

The administrator sends a message to the user which is contained in *m_szMessage*. The messages are sent from the server control panel "User" page.

**See also**

[WM_CTI_NETWORK_DISPATCH overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

The CTI server has been stopped.

Sub-message CTI_SERVICEPAUSE (0x00013001)**Message type**

CTI_COMMAND_MSG

Used values

m_wParam

TRUE if the server is running, FALSE if it was stopped.

Remarks

The CTI server may sometimes be stopped for maintenance work. There are certain utilities in the program which you should then deactivate, such as a dial button. When the CTI-Server is running again, these utilities may be activated again.

See also

[WM_CTI_NETWORK_DISPATCH overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

Another user sends a message.

Sub-message CTI_USERTOUSER (0x00013005)**Message type**

CTI_COMMAND_MSG

Used values

m_unNumber

Extension of the other user

m_szMessage

Text message the other user has sent, maximum 255 characters

Remarks

Users can send each other messages using [CTI_ESendUserToUser](#). Implementing this function may allow some workplaces to do without a mail client.

See also

[WM_CTL_NETWORK_DISPATCH overview](#) | [CTI_ESendUserToUser](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

Answers an SQL query.

Sub-message SQL_BEGIN (0x00040002)**Message type**

CTI_COMMAND_MSG

Used values

m_wParam

Number of the request, which was initiated with [CTI_SQLBegin](#)

m_lParam

Number data records returned. 0 means no records were found. -1 means the query failed.

m_szMessage

Text with the queried data.

Remarks

This event is always sent when data are queried using [CTI_SQLBegin](#) or [CTI_SQLNext](#). The data may arrive in XML format or as text. The query may fail when the query has already been terminated or when all the data have already been read.

Example

[SQL sample](#)

[Database queries](#)

See also

[WM_CTL_NETWORK_DISPATCH overview](#) | [CTI_SQLBegin](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

Delivers records from a telephone number search.

Sub-message SQL_DBBOOK (0x00040006)**Message type**

CTI_COMMAND_MSG

Used values

m_wParam

Number of records sent

m_szMessage

Text containing the records

Remarks

The has started a database query of type [CTI_SQLGetDbBook](#). The records are enclosed in curly brackets. Name, number and additional information are separated by semi-colons. The additional information is not always present. Extent and format of the data arriving can be configured in the CTI server's database settings.

See also

[WM_CTI_NETWORK_DISPATCH overview](#) | [CTI_SQLGetDbBook](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

An SQL query was ended.

Sub-message SQL_END (0x00040004)**Message type**

CTI_COMMAND_MSG

Used values

m_wParam

Number of the database query, as assigned during [CTI_SQLBegin](#).

m_lParam

TRUE when the query was successfully ended.

Remarks

The event is sent as a consequence of the command [CTI_SQLEnd](#). The event comes of its own accord if all the data have been read using [CTI_SQLBegin](#) or [CTI_SQLNext](#). If a query does not get terminated, the server terminates it itself after a number of minutes and sends this event.

Example

[SQL sample](#)

See also

[WM_CTI_NETWORK_DISPATCH overview](#) | [CTI_SQLEnd](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

Confirms log-on to the database.

Sub-message SQL_LOGIN (0x00040001)**Message type**

CTI_COMMAND_MSG

Used values

m_wParam

TRUE when log-on succeeded.

Remarks

Before a user can use the database, he must log on with [CTI SQLLogin](#). As a result of this, the server sends this event. The log-on is rejected if the wrong password is supplied or if the database connection is not active. The database connection can be established on the “Database interface” page of the server control panel.

**Example**

[SQL sample](#)

See also

[WM CTI NETWORK DISPATCH overview](#) | [CTI SQLLogin](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.3.7 Structures

The [callback functions](#) send pointers to structures in their return values. These structures contain data. Database queries are further structured according to XML.

[CTI_COMMAND_MSG](#)

[CTI_PHONE_DATA](#)

[Database queries](#)

See also

[C++ CTI API overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.3.7.1 CTI_COMMAND_MSG

Definition

```
#define CTI_NUMBER_SIZE      8
#define CTI_MESSAGE_SIZE    256

typedef struct cti_command_tag_msg
{
    DWORD m_dwSize;
    DWORD m_dwCommand;
    WPARAM m_wParam;
    LPARAM m_lParam;
    union tag_number
    {
        CTI_NUMBER wNumber;
        TCHAR szNumber[CTI_NUMBER_SIZE];
    }m_unNumber;
    TCHAR m_szMessage[CTI_MESSAGE_SIZE];
}CTI_COMMAND_MSG,FAR* LP_CTI_COMMAND_MSG;
```

Remarks

These structures are used to send most data over the network. m_dwSize contains the structure size, comprising usually the message length plus the length of the data preceding it. m_dwCommand contains the [command parameter](#). remaining values are command-dependent. SQL messages, such as [SQL_DATA](#), are particularly likely to contain text longer than 256 characters.

See also

[Structures overview](#) | [WM_CTI_NETWORK_DISPATCH](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.3.7.2 CTI_PHONE_DATA

Definition

```
#define CTI_NUMBER_SIZE      8
#define PHONE_NUMBER_SIZE   32
#define CTI_MESSAGE_SIZE    256

typedef struct cti_phone_data_tag
{
    DWORD m_dwSize;
    DWORD m_dwCommand;
    time_t m_dwStart;
    time_t m_dwEnd;
    DWORD m_dwFlags;
    union tag_Actor
    {
        CTI_NUMBER wNumber;
        TCHAR szNumber[CTI_NUMBER_SIZE];
    }m_unActor;
    TCHAR m_szNumber[PHONE_NUMBER_SIZE];
    union tag_Dest
    {
        CTI_NUMBER wNumber;
        TCHAR szNumber[CTI_NUMBER_SIZE];
    }m_unDest;
    TCHAR m_szMessage[CTI_MESSAGE_SIZE];
}CTI_PHONE_DATA, FAR* LP_CTI_PHONE_DATA;
```

Remarks

This structure is used to convey [call data records](#) and [call reminders](#).

See also

[Structures overview](#) | [WM_CTI_NETWORK_DISPATCH](#)

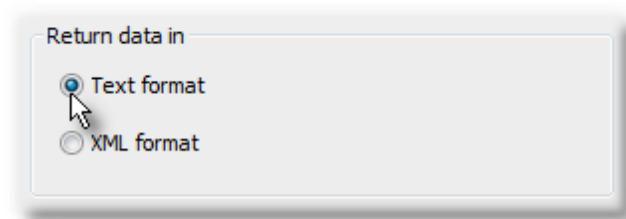
[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.3.7.3 Database queries

Configuration of the return type

Data from TAPIMaster®'s database interface can either be formatted as text or as XML. The format can be set on the "Database interface" page of the server control panel.



XML-Format

Here is an example of a query sent to the example database contained in the "Database" folder of the programmer's package. The query is:

```
SQLBegin(„SELECT * FROM Members“,2)
```

The answer is:

```
<?xml version="1.0" standalone="yes"?>
<DATAPACKET Version="2.0">
  <METADATA>
    <FIELDS>
      <FIELD tagname="MemberNumber" fieldtype="i4" />
      <FIELD tagname="GivenName" fieldtype="string" WIDTH="40" />
      <FIELD tagname="Surname" fieldtype="string" WIDTH="32" />
      <FIELD tagname="DayOfBirth" fieldtype="datetime" WIDTH="19" />
      <FIELD tagname="Tax" fieldtype="numeric" WIDTH="19" />
      <FIELD tagname="VoteRights" fieldtype="i1" WIDTH="3" />
      <FIELD tagname="DrivingLicense" fieldtype="boolean" />
      <FIELD tagname="ThrowWidth" fieldtype="r8" WIDTH="53" />
    </FIELDS>
  </METADATA>
  <ROWDATA>
    <ROW>
      <MemberNumber>1</MemberNumber>
      <GivenName>Walter</GivenName>
      <Surname>Miller</Surname>
      <DayOfBirth>1962-11-14 00:00:00</DayOfBirth>
      <Tax>15.5000</Tax>
      <VoteRights>20</VoteRights>
      <DrivingLicense>1</DrivingLicense>
      <ThrowWidth>12.5</ThrowWidth>
    </ROW>
    <ROW>
      <MemberNumber>2</MemberNumber>
      <GivenName>Paul</GivenName>
      <Surname>Smith</Surname>
      <DayOfBirth>1959-12-26 00:00:00</DayOfBirth>
      <Tax>12.0000</Tax>
      <VoteRights>10</VoteRights>
      <DrivingLicense>1</DrivingLicense>
      <ThrowWidth>11.99</ThrowWidth>
    </ROW>
  </ROWDATA>
</DATAPACKET>
```

Text format

The data presented above take the following form in plain text:

```
{HEADER ColCount=8}
{COLUMNS (Type=4 Text=MemberNumber);(Type=12 Text=GivenName);(Type=12 Text=Surname);
(Type=93 Text=DayOfBirth);(Type=2 Text=Tax);(Type=-6 Text=VoteRights);(Type=-7
Text=DrivingLicense);(Type=8 Text=ThrowWidth);}
{(Value0=1);(Value1=Walter);(Value2=Miller);(Value3=1962-11-14 00:00:00);
(Value4=15.5000);(Value5=20);(Value6=1);(Value7=12.5);}
```

```
{ (Value0=2); (Value1=Klaus); (Value2=Smith); (Value3=1959-12-26 00:00:00);  
(Value4=12.0000); (Value5=10); (Value6=1); (Value7=11.99); }
```

SQL data types

The table lists the types supported by the TAPIMaster® interface:

Typ	Wert
Char	1
Numeric	2
Decimal	3
i4	4
i2	5
Float	6
r4	7
r8	8
Date	9
Time	10
Timestamp	11
String	12
Datetime	93
string	-1
i8	-5
i1	-6
Boolean	-7

See also

[Structures overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.3.8 Constants

Use of constants

The TAPIMaster® API makes use of a number of constants. Some of these, such as the [call states](#), come from the TAPI interface. A number of flags are also used.

[Call states](#)

[Connection options](#)

See also

[C++ CTI API overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.3.8.1 Call states

A call always has a definite state, which depends on whether the call is being established, is active, or has just ended. The call ends with state IDLE, at which point it you may remove it from your internal call management structures. You can also find the call states in file TAPI.H in your development environment. There the states have the prefix LINECALLSTATE_.

Identifier	Value (hexadecimal)	Description
IDLE	0x0001	Switch ready
OFFERING	0x0002	Incoming call
ACCEPTED	0x0004	User / device is handling the call
DIALTONE	0x0008	Receiver off hook
DIALING	0x0010	Caller is dialing
RINGBACK	0x0020	Ringling at other end
BUSY	0x0040	Telephone busy
SPECIALINFO	0x0080	Manufacturer-specific information
CONNECTED	0x0100	Parties connected
PROCEEDING	0x0200	Number complete, call is being established
ONHOLD	0x0400	Call held
CONFERENCED	0x0800	Call in conference
ONHOLDPENDCONF	0x1000	Conference is initiated
ONHOLDPENDTRANSFER	0x2000	Call is waiting for transfer
DISCONNECTED	0x4000	Call terminated
UNKNOWN	0x8000	Status unknown

See also

[Constants overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.1.3.8.2 Connection options

Use of the connection options

These options are set before connecting using the function [CTI_OSetOptions](#). These constants are defined in file CTYPES.H.

Constant	Value (hexadecimal)	Description
CTCLIENT_AUTOCONNE CT	0x00000002	Keep connection up automatically
CTCLIENT_SHOWICON	0x00000004	Display connection icon
CTCLIENT_CHECKCONN ECTION	0x00000008	Check connection settings when starting
CTCLIENT_AUTOLOGIN	0x00000010	Automatic log-on
CTCLIENT_SHOWMENU	0x00000020	Tray icon has a menu
CTCLIENT_HOTKEY	0x00000040	Text-dial using keyboard short-cut
CTCLIENT_PHONEICON	0x00000080	Call symbol
CTCLIENT_ASSISTEDTE LEPHONY	0x00000100	Client supports Assisted Telephony
CTCLIENT_AUTOANSWE R	0x00000200	Answer incoming calls automatically

See also

[Constants overview](#) | [CTI_OSetOptions](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

Activate support of Assisted Telephony.

Flag CTCLIENT_ASSISTEDTELEPHONY (0x00000100)**Remarks**

Accepts Assisted Telephony commands. When a third-party program calls the Windows TAPI function `tapiRequestMakeCall`, the interface starts calling.

Default value

Enabled

See also

[Connection options overview](#) | [CTI_OSetOptions](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

Calls answered automatically.

Flag CTICLIENT_AUTOANSWER (0x00000200)**Remarks**

calls are answered automatically. This option is of use in call centers. The telephone or headset must be capable of switching to hands-free mode and support must be present in the TAPI driver.

Default value

Disabled

See also

[Connection options overview](#) | [CTI_OSetOptions](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

The network connection is maintained automatically.

Flag CTICLIENT_AUTOCONNECT (0x00000002)**Remarks**

At program start-up, the CTI interface automatically establishes the connection to the CTI server. If the connection is interrupted, the client tries automatically to re-establish it.

Default value

Enabled

See also

[Connection options overview](#) | [CTI_OSetOptions](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

Automatically log on to the CTI server.

Flag CTICLIENT_AUTOLOGIN (0x00000010)**Remarks**

After the network connection has been established, the interface logs on to the server automatically. There is no need to call [CTI_NLogin](#) separately.

Default value

Enabled

See also

[Connection options overview](#) | [CTI_OSetOptions](#) | [CTI_OSetOwnerNumber](#) | [CTI_NLogin](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

Connection parameters are checked at start-up.

Flag CTICLIENT_CHECKCONNECTION (0x00000008)**Remarks**

At start-up, the interface checks the connection parameters – server name, port number and internal extension number – for completeness. If a parameter is missing, the interface raises the [logon dialog](#) to the foreground.

Default value

Enabled

See also

[Connection options overview](#) | [CTI_OSetOptions](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

Activate text-dial via key-stroke.

Flag CTICLIENT_HOTKEY (0x00000040)**Remarks**

Determines whether highlighted text can be dialed by key-stroke. The key will be in the range F1 to F12. Additionally, the shift and/or control key can be pressed. If the text contains a dialable number, the number will be called. The text-dial and additional key are set using [CTI_OSetDialKey](#).

Default value

Disabled

See also

[Connection options overview](#) | [CTI_OSetOptions](#) | [CTICLIENT_QUICKMOUSEDIAL](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

A telephone symbol is displayed when incoming calls arrive.

Flag CTICLIENT_PHONEICON (0x00000080)**Remarks**

A small telephone symbol can be displayed in the task bar when a call arrives. This function parallels the display of a letter by MS Outlook. The telephone number, if present, is shown in the symbol's tool-tip. The icon has a short menu, which allows the call to be returned. [CTI_TRemovePhoneIcon](#) removes the icon. The [WM_CTIP_SHOWPHONELIST](#) event is sent when the icon is double-clicked and you should then display the caller list, if present.

**Default value**

Enabled

See also

[Connection options overview](#) | [CTI_OSetOptions](#)

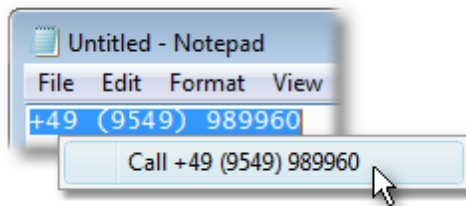
[Send feedback to TAPIMaster@](#)

© 2020 Tino Kasubke. All rights reserved.

Activates text-dial via context menu.

Flag CTICLIENT_QUICKMOUSEDIAL (0x00000400)**Remarks**

Using this option, a context menu can be used to dial highlighted text. Such a context menu is displayed if the text contains a dialable number:

**Default value**

Disabled

See also

[Connection options overview](#) | [CTI_OSetOptions](#) | [CTICLIENT_HOTKEY](#)

[Send feedback to TAPIMaster@](#)

© 2020 Tino Kasubke. All rights reserved.

Show a connection symbol.

Flag CTICLIENT_SHOWICON (0x00000004)**Remarks**

A connection icon is shown in the task bar reflecting the state of the network connection. The icon can also have a user-defined menu and tool-tip. You can also supply your own icons for active and interrupted connections instead of using the standard icons.

**Default value**

Enabled

See also

[Connection options overview](#) | [CTI_OSetOptions](#) | [CTI_OConnectionIconOptions](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

The connection icon has a menu.

Flag CTIClient_SHOWMENU (0x00000020)

Remarks

A menu can be assigned to the connection item in the task bar. The menu appears when the user right-clicks the icon with the mouse. [CTI_OConnectionIconOptions](#) is used to specify the menu handle. Menu events are delivered by OnMenuCommand. [CTIClient_SHOWICON](#) must be active.

Default value

Disabled

See also

[Connection options overview](#) | [CTI_OSetOptions](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.2 Sample programs

This section contains example programs for C++, Delphi and VB/Excel. The use of the C++ and ActiveX interface is illustrated. You should install the CTI client before working with the examples.

[Integration in VB](#)

[Simply CTI client \(C++\)](#)

[Simply CTI client \(Delphi\)](#)

[SQL sample](#)

See also

[Client interface](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

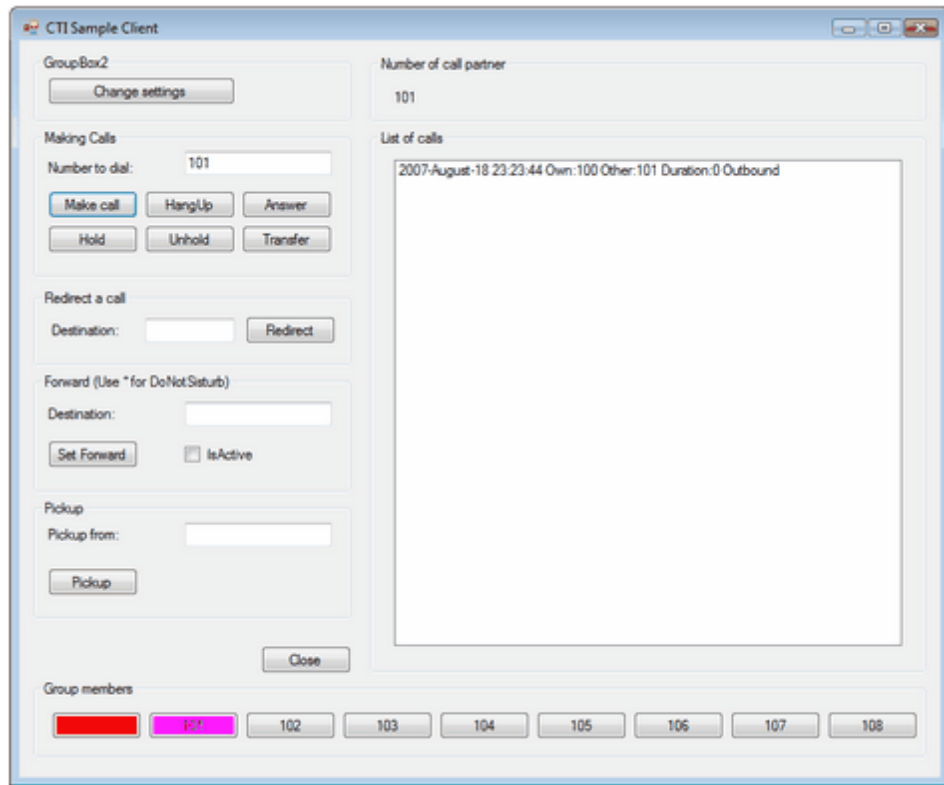
1.2.2.1 Integration in VB

Required files

Several of the [files](#) from the client installation are required. FCTICLNT.DLL must be registered.

Example

This VB client is available.



See also

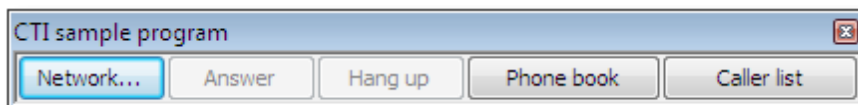
[Sample programs](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.2.2 Simply CTI client C++

This simple client is written in C++ (MFC, Visual C++ 7.0) and contains the major telephony functions.



There is a small telephone book. New names can be added using the insert key can and removed using the delete key.

Name	Number
Paula	089-65465456
Walter	069564445646

Incoming calls are recorded in a list. The delete key removes them. The telephone book is used here for caller identification. Double-clicking on a list entry initiates a return call to the caller.

Last caller: 102		
Time	Number	Caller
05.10.2006 21:32:43	102	
05.10.2006 21:32:38	102	
05.10.2006 21:32:29	102	

See also

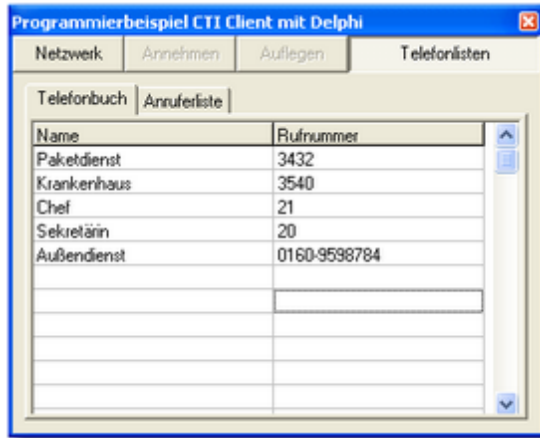
[Sample programs](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.2.2.3 Simply CTI client Delphi

This simple client is written in Delphi 6 and contains the major telephony functions. It uses the ActiveX interface.



See also

[Sample programs](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

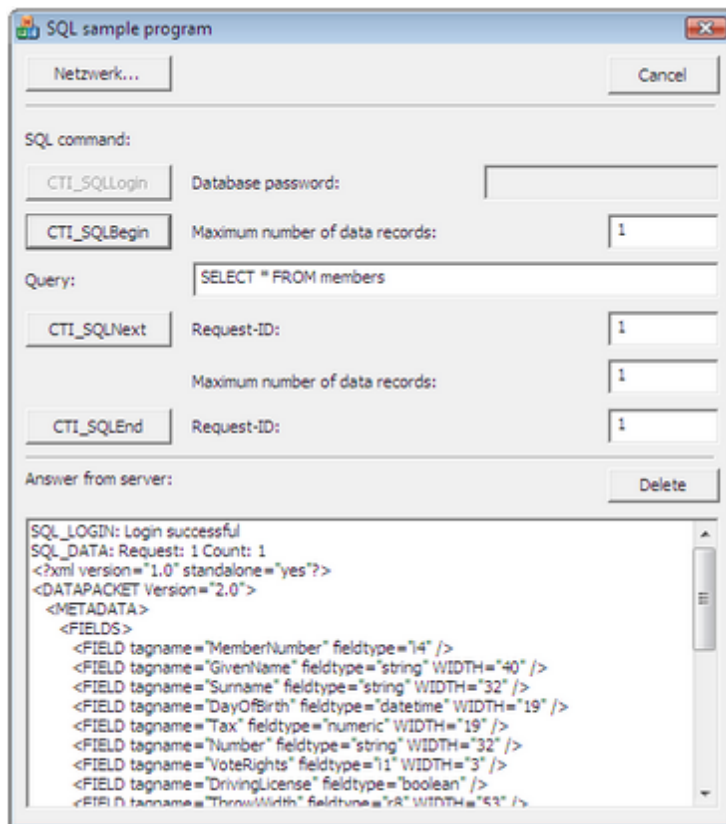
1.2.2.4 SQL sample

The SQL Demo is a C++ program based on MFC, Visual C++ 7.0. It is located in folder CPlusPlus\SQLDemo of the CTI SDK. The example allows you to try out some SQL commands. The example just illustrates database access and does not contain any telephony functionality. A database must be connected via ODBC from the server control panel. The client should be installed and operational on the machine.



Please familiarise yourself first with the [SQL interface's commands](#) and events. Then start the program. Enter the database password and press SQL_Login. If the password was correct or not necessary, "Login successful" will appear in the output field underneath.

The remaining commands are now available for use. The query has already been set up for "SELECT * FROM members" and 1 data record. It is intended for use with database "CLUB.MDB" in the "database" directory of the developer kit. You will need to adjust the query to work with other databases. When the command has been sent, you will get a request ID back, which is used to sequentially number the queries. You may query more records using this ID by pressing SQL-Next. SQL_End ends the query before all records have been read. After a few minutes, unused queries are terminated automatically.



See also

[Sample programs](#) | [SQL](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.3 Server interface

The server interface can be integrated into a database such as Access, into an Excel table, or into an application with the function of a database. The application could also be a service ultimately responsible for database access. Usually, however, you will want to connect to a database using the ODBC interface of the CTI server.

As on the client side, here there are also two interfaces in one DLL. This DLL, however, contains far fewer functions.

[Command reference](#)

[Sample programs](#)

See also

[Contents](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.3.1 Command reference

The command reference is split into two sections: one for ActiveX and the other for the C++ API. The C++ functions start with the prefix CTI_. Otherwise, functions and events are largely identical for the two interfaces.

[.NET](#)
[ActiveX](#)
[C++ CTI API](#)

See also

[Server interface](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.3.1.1 .NET

The server interface is composed of these four classes. The class is mandatory Telephony needed, because here takes place the initialization of the interface.

ClientCom	The class is used to send messages from server to Client.und vice versa..
CtiConvert	The class converts numbers and date formats.
PBXInfo	The class provides information to the telephone system and the registered clients.
Telephony	The class contains all functions and events which have to do directly with the calls.

See also

[Command reference - Server](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.3.1.1.1 ClientCom

Class ClientCom

The class is used to send messages from server to Client.und vice versa.

Functions

OnUserToDatabase	User-defined data are received for further processing.
SendAdminMessage	Sending text messages from server to the client .
SendIdentity	Send a caller identification message back.
SendUserData	Send user-defined data.

See also

[.NET overview](#)

[Send feedback to TAPIMaster®](#)

<%FULLCOPYRIGHT%

1.3.1.1.1.1 OnUserToDatabase

User-defined data are received for further processing.

```
Event OnUserToDataBase  
(  
    object sender,  
    uint e.dwCommand,  
    String e.strUser,  
    String e.strMessage  
)
```

Parameters

sender

object [ClientCom](#)

e.dwCommand

Command number

e.strUser

Extension of the user who sent the command.

e.strMessage

Command parameter

Remarks

A client has sent a user-defined command to the server interface. This can be achieved using [UserSendToDataBase](#) in the client. The server handles the event and can respond to the client using [SendUserData](#). The parameters *dwCommand* and *strUser* are required for the answer to the client.

See also

[ClientCom overview](#)

[Send feedback to TAPIMaster@](#)
[<%FULLCOPYRIGHT%](#)

1.3.1.1.1.2 SendAdminMessage

Sending text messages from server to the client .

```
void SendAdminMessage  
(  
    String strUser,  
    String strMessage  
)
```

Parameters

strUser

Extension of the user

strText

Text to send.

Return value

None

Remarks

This messages will be showed at the client side as [OnServerMessage](#) events.

See also

[ClientCom overview](#)

[Send feedback to TAPIMaster®](#)

<%FULLCOPYRIGHT%

1.3.1.1.1.3 SendIdentity

Send a caller identification message back.

```
void SendIdentity  
(  
    uint dwCallID,  
    String strUser,  
    String strText  
)
```

Parameters

dwCallID

Call ID of call

strUser

Extension of user

strText

Text containing the data about the caller.

Return value

None

Remarks

The server interface has received event [OnIdentity](#), whereupon information about the calling number was determined. The call ID and extension of the user, which were received on **OnGetIdentity**, are also sent back. *strText* should be formatted suitably for display by the client in question.

See also

[ClientCom overview](#)

[Send feedback to TAPIMaster@](#)
<%FULLCOPYRIGHT%

1.3.1.1.1.4 SendUserData

Send user-defined data.

```
void SendUserData  
(  
    uint dwCommand,  
    String strUser,  
    String strText  
)
```

Parameters

dwCommand

Command number

strUser

Extension of the user who sent the command.

strText

Command parameter

Return value

None

Remarks

TAPIMaster® allows user-defined data to be sent from client to server and back. In this way you may transmit additional data between client and server without having to open an additional connection.

The client sends commands to the server using [UserSendToDataBase](#). Event [OnUserToDataBase](#) is triggered upon their arrival at the server.

See also

[ClientCom overview](#)

[Send feedback to TAPIMaster®](#)

<%FULLCOPYRIGHT%

1.3.1.1.2 CtiConvert

Class CtiConvert

The class converts numbers and date formats.

Functions

ConvertToCanonical	Convert telephone number to canonical form.
GetDateTime	Converts a time_t var into a standard string.
GetFormatDateTime	Converts a time_t var into an individual formatted string.

See also

[.NET overview](#)

[Send feedback to TAPIMaster®](#)

<%FULLCOPYRIGHT%

1.3.1.1.2.1 ConvertToCanonical

Convert telephone number to canonical form.

```
String ConvertToCanonical  
(  
    String strInput  
)
```

Parameters

strInput

Number to convert to canonical form.

Return value

Converted Number

Remarks

You can use this function to convert a dataset containing telephone numbers to canonical form. The output string should be at least 32 characters long.

See also

[CtiConvert overview](#)

[Send feedback to TAPIMaster®](#)

<%FULLCOPYRIGHT%

```
Function ConvertToCanonical(  
    String bstrInput,  
    StringPtr pbstrOutput  
)
```

Parameters*bstrInput*

Number to convert to canonical form

pbstrOutput

Converted Number

Remarks

You can use this function to convert a dataset containing telephone numbers to canonical form. The output string should be at least 32 characters long.

1.3.1.1.2.2 GetDateTime

Converts a time_t var into a standard string.

```
String GetDateTime  
{  
    uint dwTime  
}
```

Parameters*lTime*time_t var, e. g. delivered in the event [OnProtocolData](#).**Return value**

Return value in a standard format: 2007-01-12T00:53:32.875

Remarks

Use this function to get a sortable date format string. Any programs like Excel use this format.

See also[CtiConvert overview](#)

[Send feedback to TAPIMaster®](#)
<%FULLCOPYRIGHT%

1.3.1.1.2.3 GetFormatDateTime

Converts a `time_t` var into an individual formatted string.

```
String GetFormatDateTime
(
    uint dwTime,
    String strFormat
)
```

Parameters

dwTime

`time_t` var, e. g. delivered in the event [OnProtocolData](#).

strFormat

String with date format. This can be formed of the following format characters.

Token	Description
%a	Abbreviated weekday name
%A	Full weekday name
%b	Abbreviated month name
%B	Full month name
%c	Date and time representation appropriate for locale
%d	Day of month as decimal number (01 – 31)
%H	Hour in 24-hour format (00 – 23)
%I	Hour in 12-hour format (01 – 12)
%j	Day of year as decimal number (001 – 366)
%m	Month as decimal number (01 – 12)
%M	Minute as decimal number (00 – 59)
%p	Current locale's A.M./P.M. indicator for 12-hour clock
%S	Second as decimal number (00 – 59)
%U	Week of year as decimal number, with Sunday as first day of week (00 – 53)
%w	Weekday as decimal number (0 – 6; Sunday is 0)
%W	Week of year as decimal number, with Monday as first day of week (00 – 53)
%x	Date representation for current locale
%X	Time representation for current locale
%y	Year without century, as decimal number (00 – 99)
%Y	Year with century, as decimal number
%z,%Z	Either the time-zone name or time zone abbreviation, depending on registry settings; no characters if time zone is unknown
%%	Percent sign

Return value

Null-terminated string in a specific format

Remarks

The format characters can be combined.

See also

[CtiConvert overview](#)

[Send feedback to TAPIMaster®](#)

<%FULLCOPYRIGHT%

1.3.1.1.3 PBXInfo

Class PBXInfo

The class provides information to the telephone system and the registered clients.

Functions

GetExternPrefix	Prefix for external calls.
GetInboundFilter	Digits filtered for incoming calls.
GetInternPrefix	Prefix for internal calls.
GetNameFromExtension	User name of an extension.
GetOutboundFilter	Digits filtered for outgoing calls.
LicenseCount	Returns the number of licences.
OnLoginData	A user has logged in or logged out.
TMServiceStatus	State of TAPIMaster service.

See also

[.NET overview](#)

[Send feedback to TAPIMaster@](#)

[<%FULLCOPYRIGHT%](#)

1.3.1.1.3.1 GetExternPrefix

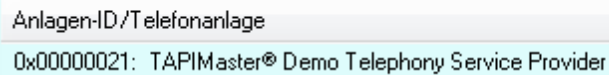
Prefix for external calls.

```
String GetExternPrefix  
(  
    String strPBX  
)
```

Parameters

strPBX

Name of the PBX system. You can see it in the CTI server control panel.



Anlagen-ID/Telefonanlage
0x00000021: TAPIMaster® Demo Telephony Service Provider

Return value

This digits will be pre-dialed.

Remarks

If no prefix used, the return value will be empty.

See also

[PBXInfo overview](#)

[Send feedback to TAPIMaster®](#)

<%FULLCOPYRIGHT%

1.3.1.1.3.2 GetInboundFilter

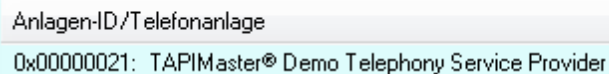
Digits filtered for incoming calls.

```
String GetInboundFilter  
(  
    String strPBX  
)
```

Parameters

strPBX

Name of the PBX system. You can see it in the CTI server control panel.



Anlagen-ID/Telefonanlage
0x00000021: TAPIMaster® Demo Telephony Service Provider

Return value

filtered digits.

Remarks

If no filter used, the return value will be empty.

See also

[PBXInfo overview](#)

[Send feedback to TAPIMaster®](#)

<%FULLCOPYRIGHT%

1.3.1.1.3.3 GetInternPrefix

Prefix for internal calls.

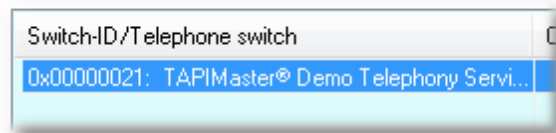
String GetInternPrefix

```
(  
    String strPBX  
)
```

Parameters

strPBX

Name of the PBX system. You can see it in the CTI server control panel.

**Return value**

This digits will be pre-dialed.

Remarks

If no prefix used, the return value will be empty.

See also

[PBXInfo overview](#)

[Send feedback to TAPIMaster®](#)

<%FULLCOPYRIGHT%

1.3.1.1.3.4 GetNameFromExtension

User name of an extension.

```
String GetNameFromExtension  
(  
    String strExtension  
)
```

Parameters

strExtension
Extension of the user.

Return value

User name.

Remarks

The user must have at that extension connected at least once.

See also

[PBXInfo overview](#)

[Send feedback to TAPIMaster®](#)
<%FULLCOPYRIGHT%

1.3.1.1.3.5 GetOutboundFilter

Digits filtered for outgoing calls.

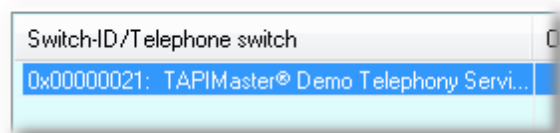
String GetOutboundFilter

```
(  
    String strPBX  
)
```

Parameters

strPBX

Name of the PBX system. You can see it in the CTI server control panel.

**Return value**

filtered digits.

Remarks

If no filter used, the return value will be empty.

See also

[PBXInfo overview](#)

[Send feedback to TAPIMaster®](#)

<%FULLCOPYRIGHT%

1.3.1.1.3.6 LicenseCount

Returns the number of licences.

Property LicenseCount**Type**

int, read only

Remarks

The number 0 indicates an unlicensed version or that a licence with limited lifetime has expired.

See also

[PBXInfo overview](#)

[Send feedback to TAPIMaster®](#)

<%FULLCOPYRIGHT%

1.3.1.1.3.7 OnLoginData

A user has logged in or logged out.

```
Event OnLoginData
(  
    object sender,  
    String e.strDevice  
    String e.strName,  
    bool e.bLogin  
)
```

Parameters

sender

object [PBXInfo](#)

e.strDevice

Extension of the user who has logged in or logged out.

e.strName

Login name, if known.

e.bLogin

1 User has logged in.

0 User has logged out.

Remarks

Will be sent the client has logged on.

See also

[PBXInfo overview](#)

[Send feedback to TAPIMaster®](#)

<%FULLCOPYRIGHT%

1.3.1.1.3.8 TMServiceStatus

State of TAPIMaster service.

Property TMServiceStatus
Type

uint, read only. Definition:

- 0 Unknown
- 1 SERVICE_STOPPED
- 2 SERVICE_START_PENDING
- 3 SERVICE_STOP_PENDING
- 4 SERVICE_RUNNING
- 5 SERVICE_CONTINUE_PENDING
- 6 SERVICE_PAUSE_PENDING
- 7 SERVICE_PAUSED

Remarks

The returned constants are similar to those in winsvc.h.

See also

[PBXInfo overview](#)

[Send feedback to TAPIMaster®](#)

<%FULLCOPYRIGHT%

1.3.1.1.4 Telephony

Class Telephony

The class contains all functions and events which have to do directly with the calls.

Functions

Alternate	Switch back and forth between two calls.
Answer	Answers a call.
Conference	Connect two calls to a telephone conference.
Forward	Enables call forwarding.
GetPBXMonitoredLines	Lists all monitored lines.
Hangup	Hang up a call.
Init	Initialise the server interface.
MakeCall	Initiates a call from the server.
OnCallNumber	Signals a telephone number.
OnCallState	Signals the state of one's own calls.
OnForwardingState	Signals the state of forwarding.

OnIdentity	Request for calling number identification
OnNewCall	Preliminary informations for a call come.
OnProtocolData	A call data record should be stored.
Pickup	Picks up a call from an other phone device.
Redirect	Redirect a call on to another telephone.
SendAllUserCallStates	Delivers the states of all recent calls.
SendLastCallRecords	Returns the data of the last calls.
SendPBXLoginLines	Returns the registered users.
SetPhoneDisplay	Displays text in the phone display.
Transfer	Connect two calls.

See also

[.NET overview](#)

[Send feedback to TAPIMaster®](#)

<%FULLCOPYRIGHT%

1.3.1.1.4.1 Alternate

Switch back and forth between two calls.

```
void Alternate  
(  
    String strDevice  
)
```

Parameters

strDevice

Extension of the user who shall change between the calls.

Return value

None

Remarks

A licence must be present for the user.

See also

[Telephony overview](#)

[Send feedback to TAPIMaster®](#)

<%FULLCOPYRIGHT%

1.3.1.1.4.2 Answer

Answers a call.

```
void Answer  
(  
    String strDevice  
)
```

Parameters

strDevice

Extension of user who is answer.

Return value

None

Remarks

A licence must be present for the user.

See also

[Telephony overview](#)

[Send feedback to TAPIMaster®](#)

<%FULLCOPYRIGHT%

1.3.1.1.4.3 Conference

Connect two calls to a telephone conference.

```
void Conference  
(  
    String strDevice  
)
```

Parameters

strDevice

Extension of the user who starts the conference.

Return value

None

Remarks

A licence must be present for the user.

See also

[Telephony overview](#)

[Send feedback to TAPIMaster®](#)

<%FULLCOPYRIGHT%

1.3.1.1.4.4 Forward

Enables call forwarding.

```
void Forward  
(  
    String strDevice,  
    String strTraget  
)
```

Parameters

strDevice

Extension of the user.

strTarget

Target of the call forwarding.

Return value

None

Remarks

A licence must be present for the user.

See also

[Telephony overview](#)

[Send feedback to TAPIMaster®](#)

<%FULLCOPYRIGHT%

1.3.1.1.4.5 GetPBXMonitoredLines

Lists all monitored lines.

```
String GetPBXMonitoredLines  
(  
    int nMaxCount  
)
```

Parameters

nMaxCount
Maximale Anzahl der Leitungen.

Return value

List of extension numbers, separated by commas.

Remarks

These are all extensions that are supported by the license.

See also

[Telephony overview](#)

[Send feedback to TAPIMaster®](#)
<%FULLCOPYRIGHT%

1.3.1.1.4.6 Hangup

Hang up a call.

```
void Hangup  
(  
    String strDevice  
)
```

Parameters

strDevice

Extension of user who is drop.

Return value

None

Remarks

A licence must be present for the user.

See also

[Telephony overview](#)

[Send feedback to TAPIMaster®](#)

<%FULLCOPYRIGHT%

1.3.1.1.4.7 Init

Initialise the server interface.

```
void Init()
```

Parameters

None

Return value

None

Remarks

This function initialises the .NET interface in FCTIDATA.DLL and should be called before the interface is used.

See also

[Telephony overview](#)

[Send feedback to TAPIMaster®](#)

<%FULLCOPYRIGHT%

1.3.1.1.4.8 MakeCall

Initiates a call from the server.

```
void MakeCall  
(  
    String strDevice,  
    String strTarget  
)
```

Parameters

strDevice

Extension of user who is calling.

strTarget

Number to call

Return value

None

Remarks

A licence must be present for the user.

See also

[Telephony overview](#)

[Send feedback to TAPIMaster®](#)

<%FULLCOPYRIGHT%

1.3.1.1.4.9 OnCallNumber

Signals a telephone number.

```
Event OnCallNumber
(  
    object sender,  
    String e.strDevice  
    uint e.dwCallID  
    String e.strNumber  
)
```

Parameters

sender

object [Telephony](#)

e.strDevice

Extension of user to whom the number – the number of the party he is in a call with – is being sent.

e.dwCallID

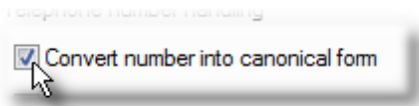
Call ID

e.strNumber

Telephone number conveyed, up to 31 characters.

Remarks

Signals a telephone number, if one has been received. No number is signalled for calls from the analogue network or for anonymous calls. This event is also triggered for outgoing calls. The server can be configured to send the number in canonical format (“Switch settings” page).



A new call is normally signalled via event [OnCallState](#) with a status of [LINECALLSTATE_OFFERING](#) (incoming) or [LINECALLSTATE_DIALTONE](#) (outgoing). **OnCallNumber** can, however, also be sent before all other events. The telephone number can change throughout the course of the call, for example, when the call is transferred to the user.

See also

[Telephony overview](#)

[Send feedback to TAPIMaster®](#)

<%FULLCOPYRIGHT%

1.3.1.1.4.10 OnCallState

Signals the state of one's own calls.

```
Event OnCallState  
(  
    object sender,  
    String e.strDevice,  
    uint e.dwState,  
    uint e.dwCallID  
)
```

Parameters

sender

object [Telephony](#)

e.strDevice

Extension of user whose call states are signalled.

e.dwState

[Call state](#) which the call is in.

e.dwCallID

Call ID of call whose state has changed.

Remarks

The state of a call has changed.

See also

[Telephony overview](#)

[Send feedback to TAPIMaster®](#)

<%FULLCOPYRIGHT%

1.3.1.1.4.11 OnForwardingState

The forwarding state has changed.

Event OnForwardingState

```
(  
    object sender,  
    String e.strDevice,  
    uint e.bSet,  
    uint e.strTarget  
)
```

Parameters

sender

object [Telephony](#)

e.strDevice

Extension of user whose forwarding state is being sent.

e.bSet

true for active forwarding, false if not.

e.strTarget

Target of the forwarding

Remarks

The forwarding state has changed.

See also

[Telephony overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.3.1.1.4.12 OnIdentity

Request for calling number identification

```
Event OnIdentity
(  
    object sender,  
    uint e.dwCallID,  
    String e.strDevice,  
    String e.strNumber  
)
```

Parameters

sender

object [Telephony](#)

e.dwCallID

Call ID

e.strDevice

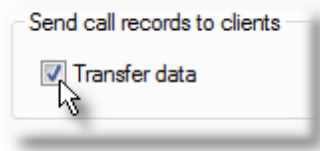
Extension which was called or which is calling.

e.strNumber

Telephone number of other party in call.

Remarks

This event is triggered when a call occurs and a telephone number is recognised. This only occurs if the signalling of telephone numbers to the clients is active. This can be activated on the “Database interface” page of the server control panel.



Upon reception of this event, determine the information for the telephone number in the server. The parameters *ICallID* and *strUser* are required for the answer. [SendIdentity](#) is used to return the data to the client.

See also

[Telephony overview](#)

[Send feedback to TAPIMaster®](#)

<%FULLCOPYRIGHT%

1.3.1.1.4.13 OnNew Call

Preliminary informations for a call come.

Event OnNewCall

```
(  
    object sender,  
    String e.strDevice,  
    uint e.dwCallID,  
    boole. blnbound,  
    uint e.dwStart,  
    String e.strNumber  
)
```

Parameters

sender

object [Telephony](#)

e.strDevice

Extension which was called or which is calling

e.dwCallID

Call ID of call whose state has changed.

e.bInbound

Is true for incoming calls.

e.dwStart

Call start time, as time_t

e.strNumber

Phone number of the partner

Remarks

One provides first of all information with important data of the call.

See also

[Telephony overview](#)

[Send feedback to TAPIMaster®](#)

<%FULLCOPYRIGHT%

1.3.1.1.4.14 OnProtocolData

A call data record should be stored.

```
Event OnProtocolData  
(  
    object sender,  
    uint e.dwStart,  
    uint e.dwEnd,  
    String e.strDevice,  
    String e.strNumber,  
    bool e.bSuccess,  
    bool e.bInbound,  
    String e.strAddress  
)
```

Parameters

sender

object [Telephony](#)

e.dwStart

Call start time, as time_t

e.dwEnd

Call end time, as time_t

e.strDevice

User extension

e.strNumber

Telephone number of the other party in the call

e.bSuccess

True if the call was successful.

e.bInbound

True if the call was incoming.

e.strAddress

Address called. This is relevant for incoming calls for a device which responds to several internal extensions (addresses). The telephone can be configured so as to be reachable from outside by different numbers, reflecting different projects or profit centers. Using the address, it is possible to determine the project for which the telephone service was rendered.

Remarks

This event is generated when a call with a telephone number is recognized. The server interface can then save the data.

See also

[Telephony overview](#)

[Send feedback to TAPIMaster®](#)
<%FULLCOPYRIGHT%

1.3.1.1.4.15 Pickup

Picks up a call from an other phone device.

```
void Pickup  
(  
    String strDevice,  
    String strRinging  
)
```

Parameters

strDevice

Extension of the user who picks a call.

strRinging

Extension of the device from which the a call should be picked up.

Return value

None

Remarks

A licence must be present for the user.

See also

[Telephony overview](#)

[Send feedback to TAPIMaster®](#)
<%FULLCOPYRIGHT%

1.3.1.1.4.16 Redirect

Redirect a call on to another telephone.

```
void Redirect  
(  
    String strDevice,  
    String strTarget  
)
```

Parameters

strDevice

Extension of the user who passes a call on.

strTarget

Destination telephone number to which the call should be redirected.

Return value

None

Remarks

The user has to be a license.

See also

[Telephony overview](#)

[Send feedback to TAPIMaster®](#)

<%FULLCOPYRIGHT%

1.3.1.1.4.17 SendAllUserCallStates

Delivers the states of all recent calls.

```
void SendAllUserCallStates()
```

Parameters

None

Return value

None

Remarks

After calling this function, [OnCallState](#) Events will be generated.

See also

[Telephony overview](#)

[Send feedback to TAPIMaster®](#)

<%FULLCOPYRIGHT%

1.3.1.1.4.18 SendLastCallRecords

Returns the data of the last calls.

```
void SendLastCallRecords(  
    String strDevice,  
    int nMaxCount,  
    bool bDelete  
)
```

Parameters

strDevice

Extension of the user.

nMaxCount

Maximum number of data records.

bDelete

Tells if the call buffer should be cleared

Return value

None

Remarks

The data will be sent on events of type [OnProtocolData](#).

See also

[Telephony overview](#)

[Send feedback to TAPIMaster®](#)

<%FULLCOPYRIGHT%

1.3.1.1.4.19 SendPBXLoginLines

Returns the registered users.

```
void SendPBXLoginLines()
```

Parameters

None

Return value

None

Remarks

After calling this function, [OnLoginData](#) Events will be generated.

See also

[Telephony overview](#)

[Send feedback to TAPIMaster®](#)

<%FULLCOPYRIGHT%

1.3.1.1.4.20 SetPhoneDisplay

Displays text in the phone display.

```
void SetPhoneDisplay  
(  
    String strDevice,  
    String strText  
)
```

Parameters

strDevice

Extension where the text should be displayed.

strText

Displayed text.

Return value

None

Remarks

A licence must be present for the user. Only a few PBX systems support this feature.

See also

[Telephony overview](#)

[Send feedback to TAPIMaster@](#)
<%FULLCOPYRIGHT%

1.3.1.1.4.21 Transfer

Connect two calls.

```
void Transfer  
(  
    String strDevice  
)
```

Parameters

strDevice

Extension of the user who shall connect two calls.

Return value

None

Remarks

A licence must be present for the user.

See also

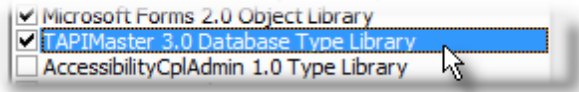
[Telephony overview](#)

[Send feedback to TAPIMaster®](#)

<%FULLCOPYRIGHT%

1.3.1.2 ActiveX

The ActiveX interface can be found in file FCTIDATA.DLL. The DLL must be registered. Perform a server set-up on the target computer so that all necessary files are present and registered there.



You can see how to use the server interface in the example for [Integration in Excel and VB](#).

[Functions](#)

[Events](#)

See also

[Command reference - Server](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.3.1.2.1 Functions

Initialisation

InitCOM	Initialise the server interface.
-------------------------	----------------------------------

Data exchange

SendIdentity	Send caller number identification back
SendUserData	Send user-defined data

Telephony functions

Alternate	Switch back and forth between two calls.
Answer	Answers a call.
Conference	Connect two calls to a telephone conference.
Drop	Hang up a call.
MakeCall	Initiate call
Pickup	Picks up a call from an other phone device.
Redirect	Redirect a call on to another telephone.
Transfer	Connect two calls.

Additional functions

ConvertToCanonical	Convert a number into canonical form
GetAllUserCallStates	Starts information events for all active calls.
GetDateTime	Converts a time_t var into a standard string.
GetExtension	Internal function
GetExternPrefix	Prefix for external calls.
GetFilterCalibrationState	Internal function
GetFormatDateTime	Converts a time_t var into an individual formatted string.
GetInboundFilter	Digits filtered for incoming calls.
GetInternPrefix	Prefix for internal calls.
GetLastCallRecords	Sends a list of the last calls as an event.
GetOutboundFilter	Digits filtered for outgoing calls.
GetPBXFiltersAndPrefixes	Returns filter and primaries.
GetPBXMonitoredLines	Lists all monitored lines.
LicenseCount	Returns the number of CTI licences
ResetCalibration	Internal function
SetCalibrationParams	Internal function

SetPhoneDisplay	Internal function
TMServiceStatus	State of TAPIMaster service.
UserCount	Internal function
UserNameFromNumber	Returns the last user logged back for an extension.

Not listed functions are obsolete, or only for internal use.

See also

[ActiveX overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.3.1.2.1.1 Alternate

Switch back and forth between two calls.

```
Function Alternate(  
    String bstrUser  
)
```

Parameters

bstrUser

Extension of the user who shall change between the calls.

Remarks

A licence must be present for the user.

Example

[ServerInterface](#)

See also

[Server functions overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.3.1.2.1.2 Answer

Answers a call.

```
Function Answer(  
    String bstrUser  
)
```

Parameters

bstrUser

Extension of user who is answer.

Remarks

A licence must be present for the user.

Example[ServerInterface](#)**See also**[Server functions overview](#)

[Send feedback to TAPIMaster®](#)[© 2020 Tino Kasubke. All rights reserved.](#)

1.3.1.2.1.3 Conference

Connect two calls to a telephone conference.

```
Function Conference(  
    String bstrUser  
)
```

Parameters*bstrUser*

Extension of the user who starts the conference.

Remarks

A licence must be present for the user.

Example[ServerInterface](#)**See also**[Server functions overview](#)

[Send feedback to TAPIMaster®](#)[© 2020 Tino Kasubke. All rights reserved.](#)

1.3.1.2.1.4 ConvertToCanonical

Convert telephone number to canonical form.

```
Function ConvertToCanonical(  
    String bstrInput,  
    StringPtr pbstrOutput  
)
```

Parameters*bstrInput*

Number to convert to canonical form

pbstrOutput

Converted Number

Remarks

You can use this function to convert a dataset containing telephone numbers to canonical form. The output string should be at least 32 characters long.

Example

[ServerInterface](#)

See also

[Server functions overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.3.1.2.1.5 Drop

Hang up a call.

```
Function Drop(  
    String bstrUser  
)
```

Parameters

bstrUser

Extension of user who is drop.

Remarks

A licence must be present for the user.

Example

[ServerInterface](#)

See also

[Server functions overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.3.1.2.1.6 GetAllUserCallStates

Starts information events for all active calls.

```
Function GetAllUserCallStates()
```

Parameters

none

Example

Starts an [OnCallState](#) and [OnNewCall](#) event.

See also

[Server functions overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.3.1.2.1.7 GetDateTime

Converts a time_t var into a standard string.

```
Function GetDateTime(  
    Long ITime,  
    StringPtr pbstrTimeString,  
)
```

Parameters

ITime

time_t var, e. g. delivered in the event [OnProtocolData](#).

pbstrTimeString

Return value in a standard format: 2007-01-12T00:53:32.875

Remarks

Use this function to get a sortable date format string. Any programs like Excel use this format.

See also

[Server functions overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.3.1.2.1.8 GetExtension

Internal function.

[Send feedback to TAPIMaster®](#)

<%FULLCOPYRIGHT%

1.3.1.2.1.9 GetExternPrefix

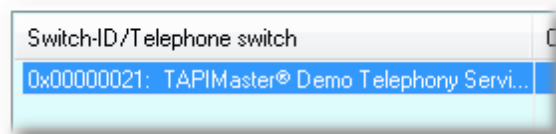
Prefix for external calls.

```
Function GetExternPrefix(  
    String bstrPBX,  
    StringPtr pbstrPrefix,  
)
```

Parameters

bstrPBX

Name of the PBX system. You can see it in the CTI server control panel.



pbstrPrefix

This digits will be pre-dialed.

Remarks

If no prefix used, the pbstrPrefix return value will be empty.

See also

[Server functions overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.3.1.2.1.10 GetFilterCalibrationState

Internal function.

[Send feedback to TAPIMaster®](#)

<%FULLCOPYRIGHT%

1.3.1.2.1.11 GetFormatDateTime

Converts a time_t var into an individual formatted string.

```
Function GetFormatDateTime(
    Long lTime,
    String bstrFormat,
    StringPtr pbstrTimeString,
)
```

Parameters

lTime

time_t var, e. g. delivered in the event [OnProtocolData](#).

bstrFormat

String with date format. This can be formed of the following format characters.

Token	Description
%a	Abbreviated weekday name
%A	Full weekday name
%b	Abbreviated month name
%B	Full month name
%c	Date and time representation appropriate for locale
%d	Day of month as decimal number (01 – 31)
%H	Hour in 24-hour format (00 – 23)
%I	Hour in 12-hour format (01 – 12)
%j	Day of year as decimal number (001 – 366)
%m	Month as decimal number (01 – 12)
%M	Minute as decimal number (00 – 59)
%p	Current locale's A.M./P.M. indicator for 12-hour clock

%S	Second as decimal number (00 – 59)
%U	Week of year as decimal number, with Sunday as first day of week (00 – 53)
%w	Weekday as decimal number (0 – 6; Sunday is 0)
%W	Week of year as decimal number, with Monday as first day of week (00 – 53)
%x	Date representation for current locale
%X	Time representation for current locale
%y	Year without century, as decimal number (00 – 99)
%Y	Year with century, as decimal number
%z,%Z	Either the time-zone name or time zone abbreviation, depending on registry settings; no characters if time zone is unknown
%%	Percent sign

pbstrTimeString

Null-terminated string in a specific format.

Remarks

The format characters can be combined.

See also

[Server functions overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.3.1.2.1.12 GetInboundFilter

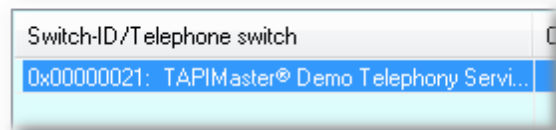
Digits filtered for incoming calls.

```
Function GetInboundFilter(
    String bstrPBX,
    StringPtr pbstrFilter,
)
```

Parameters

bstrPBX

Name of the PBX system. You can see it in the CTI server control panel.



pbstrFilter

filtered digits.

Remarks

If no filter used, the pbstrFilter return value will be empty.

See also

[Server functions overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.3.1.2.1.13 GetInternPrefix

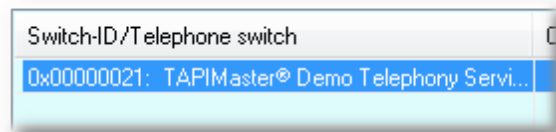
Prefix for internal calls.

```
Function GetInternPrefix(  
    String bstrPBX,  
    StringPtr pbstrPrefix,  
)
```

Parameters

bstrPBX

Name of the PBX system. You can see it in the CTI server control panel.



pbstrPrefix

This digits will be pre-dialed.

Remarks

If no prefix used, the pbstrPrefix return value will be empty.

See also

[Server functions overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.3.1.2.1.14 GetLastCallRecords

Sends a list of the last calls as an event.

```
Function GetLastCallRecords(  
    String bstrUSER,  
    Long IMaxCount,  
    Bool bDelete  
)
```

Parameters

bstrUser

Extension of the user.

IMaxCount

Maximum number of sendet call records.

bDelete

If TRUE the program will delete the buffer data.

Remarks

The data will be sent by an [OnProtocolData](#) event.

See also

[Server functions overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.3.1.2.1.15 GetOutboundFilter

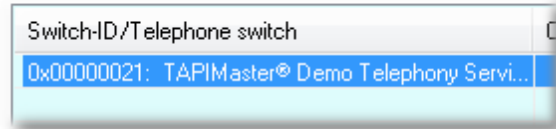
Digits filtered for outgoing calls.

```
Function GetOutboundFilter(
    String bstrPBX,
    StringPtr pbstrFilter,
)
```

Parameters

bstrPBX

Name of the PBX system. You can see it in the CTI server control panel.



pbstrFilter

filtered digits.

Remarks

If no filter used, the pbstrFilter return value will be empty.

See also

[Server functions overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.3.1.2.1.16 GetPBXFiltersAndPrefixes

Returns filter and primaries.

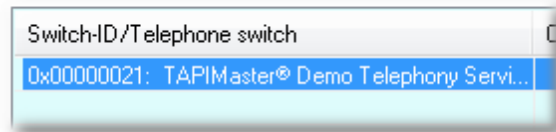
```
Function GetPBXFiltersAndPrefixes(
    String bstrPBX,
    StringPtr pbstrExtern,
    StringPtr pbstrIntern,
    StringPtr pbstrInbound,
```


StringPtr pbstrOutbound

)

Parameters*bstrPBX*

Name of the PBX system. You can see it in the CTI server control panel.

*pbstrExtern*

External code.

pbstrInternr

Internal code.

pbstrInbound

The filtered numbers for incoming calls.

pbstrOutbound

The filtered numbers for outgoing calls.

Remarks

The results can be completely or partially empty.

See also

[Server functions overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.3.1.2.1.17 GetPBXMonitoredLines

Lists all monitored lines.

**Function GetPBXMonitoredLines(
StringPtr pbstrLines**

)

Parameters*pbstrLines*

List of extension numbers, separated by commas.

Remarks

These are all extensions that are supported by the license.

See also

[Server functions overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.3.1.2.1.18 InitCOM

Initialise the server interface.

Function InitCOM()

Parameters

None

Remarks

This function initialises the COM interface in FCTIDATA.DLL and should be called before the interface is used.

Example

[ServerInterface](#)

See also

[Server functions overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.3.1.2.1.19 LicenseCount

Returns the number of licences.

Property LicenseCount

Type

Long, read only

Remarks

The number 0 indicates an unlicensed version or that a licence with limited lifetime has expired.

See also

[Server functions overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.3.1.2.1.20 MakeCall

Initiates a call from the server.

```
Function MakeCall(  
    String bstrUser,  
    String bstrNumber  
)
```

Parameters

bstrUser

Extension of user who is calling

bstrNumber

Number to call

Remarks

A licence must be present for the user.

Example

[ServerInterface](#)

See also

[Server functions overview](#)

[Send feedback to TAPIMaster@](#)

© 2020 Tino Kasubke. All rights reserved.

1.3.1.2.1.21 Pickup

Picks up a call from an other phone device.

```
Function Pickup(  
    String bstrUser,  
    String bstrNumber  
)
```

Parameters

bstrUser

Extension of the user who picks a call.

bstrNumber

Extension of the device from which the a call should be picked up.

Remarks

A licence must be present for the user.

Example

[ServerInterface](#)

See also

[Server functions overview](#)

[Send feedback to TAPIMaster@](#)

© 2020 Tino Kasubke. All rights reserved.

1.3.1.2.1.22 Redirect

Redirect a call on to another telephone.

```
Function Redirect(  
    String bstrUser,  
    String bstrNumber
```

```
)
```

Parameters*bstrUser*

Extension of the user who passes a call on.

bstrNumber

Destination telephone number to which the call should be redirected.

Remarks

The user has to be a license.

Example

[ServerInterface](#)

See also

[Server functions overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.3.1.2.1.23 ResetCalibration

Internal function.

[Send feedback to TAPIMaster®](#)

<%FULLCOPYRIGHT%

1.3.1.2.1.24 SendIdentity

Send a caller identification message back.

```
Function SendIdentity(  
    Long ICallID,  
    String bstrUser,  
    String bstrText  
)
```

Parameters*ICallID*

Call ID of call

bstrUser

Extension of user

bstrText

Text containing the data about the caller

Remarks

The server interface has received event [OnGetIdentity](#), whereupon information about the calling number was determined. The call ID and extension of the user, which were received on [OnGetIdentity](#), are also sent back. *bstrText* should be formatted suitably for display by the client in question.

Example[ServerInterface](#)**See also**[Server functions overview](#) | [OnGetIdentity](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.3.1.2.1.25 SendUserData

Send user-defined data.

```
Function SendUserData(  
    Long ICommand,  
    String bstrUser,  
    String bstrText  
)
```

Parameters*ICommand*

Command number

bstrUser

Extension of the user who sent the command.

bstrText

Command parameter

Remarks

TAPIMaster® allows user-defined data to be sent from client to server and back. In this way you may transmit additional data between client and server without having to open an additional connection.

The client sends commands to the server using [EUserSendToDataBase](#). Event [OnUserToDataBase](#) is triggered upon their arrival at the server.

Example[ServerInterface](#)**See also**[Server functions overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.3.1.2.1.26 SetCalibrationParams

Internal function.

[Send feedback to TAPIMaster®](#)

<%FULLCOPYRIGHT%

1.3.1.2.1.27 SetPhoneDisplay

Internal function.

[Send feedback to TAPIMaster®](#)
<%FULLCOPYRIGHT%

1.3.1.2.1.28 TMServiceStatus

State of TAPIMaster service.

Property TMServiceStatus**Type**

Long, Read only. Definition:

- 0 Unknown
- 1 SERVICE_STOPPED
- 2 SERVICE_START_PENDING
- 3 SERVICE_STOP_PENDING
- 4 SERVICE_RUNNING
- 5 SERVICE_CONTINUE_PENDING
- 6 SERVICE_PAUSE_PENDING
- 7 SERVICE_PAUSED

Remarks

The returned constants are similar to those in winsvc.h.

See also

[Server functions overview](#)

[Send feedback to TAPIMaster®](#)
© 2020 Tino Kasubke. All rights reserved.

1.3.1.2.1.29 Transfer

Connect two calls.

**Function Transfer(
 String bstrUser
)**

Parameters

bstrUser

Extension of the user who shall connect two calls.

Remarks

A licence must be present for the user.

Example

[ServerInterface](#)

See also

[Server functions overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.3.1.2.1.30 UserCount

Internal function.

[Send feedback to TAPIMaster®](#)
<%FULLCOPYRIGHT%

1.3.1.2.1.31 UserNameFromNumber

Returns the last user logged back for an extension.

```
Function UserNameFromNumber(
    String bstrNumber,
    StringPtr pbstrUser,
)
```

Parameters

bstrNumber
Extension to which the user name is searched.

pbstrUser
Username found.

Remarks

The result can be empty if no user has logged on.

See also

[Server functions overview](#)

[Send feedback to TAPIMaster®](#)
© 2020 Tino Kasubke. All rights reserved.

1.3.1.2.2 Events

Data exchange

OnCallNumber	Signals a telephone number
OnCallState	Signals call status
OnForwardingState	Signals the state of forwarding.
OnGetIdentity	Request for calling number identification
OnNewCall	Preliminary informations for a call come.
OnProtocolData	Signals call data for storage
OnUserToDataBase	Signals user-defined data

See also

[ActiveX overview](#)

[Send feedback to TAPIMaster®](#)
© 2020 Tino Kasubke. All rights reserved.

1.3.1.2.2.1 OnCallNumber

Signals a telephone number.

```
Event OnCallNumber(  
    String bstrUser  
    Long ICallID  
    String bstrNumber  
)
```

Parameters

bstrUser

Extension of user to whom the number – the number of the party he is in a call with – is being sent.

ICallID

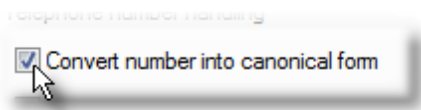
Call ID

bstrNumber

Telephone Telephone number conveyed, up to 31 characters.

Remarks

Signals a telephone number, if one has been received. No number is signalled for calls from the analogue network or for anonymous calls. This event is also triggered for outgoing calls. The server can be configured to send the number in canonical format (“Switch settings” page).



A new call is normally signalled via event [OnCallState](#) with a status of [LINECALLSTATE_OFFERING](#) (incoming) or [LINECALLSTATE_DIALTONE](#) (outgoing). **OnCallNumber** can, however, also be sent before all other events. The telephone number can change throughout the course of the call, for example, when the call is transferred to the user.

See also

[Server events overview](#) | [OnCallState](#) | [Call states](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.3.1.2.2.2 OnCallState

Signals the state of one's own calls.

```
Event OnCallState(  
    String bstrUser  
    Long IState,  
    Long ICallID
```

```
)
```

Parameters*bstrUser*

Extension of user whose call states are signalled.

IState

[Call state](#) which the call is in

ICallID

Call ID of call whose state has changed

Remarks

The state of a call has changed.

See also

[Server events overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.3.1.2.2.3 OnForwardingState

Signals the state of forwarding.

```
Event OnForwardingState(  
    String bstrUser  
    Bool bSet,  
    String bstrTarget  
)
```

Parameters*bstrUser*

Extension of user whose forwarding state is being sent.

bSet

true for active forwarding, false if not.

bstrTarget

Target of the forwarding

Remarks

The forwarding state has changed.

See also

[Server events overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.3.1.2.2.4 OnGetIdentity

Request for calling number identification

```
Event OnGetIdentity(  
    Long ICallID,  
    String bstrUser,  
    String bstrNumber  
)
```

Parameters

ICallID

Call ID of call

bstrUser

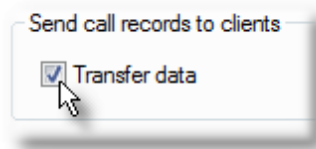
Extension which was called or which is calling

bstrNumber

Telephone number of other party in call

Remarks

This event is triggered when a call occurs and a telephone number is recognised. This only occurs if the signalling of telephone numbers to the clients is active. This can be activated on the “Database interface” page of the server control panel.



Upon reception of this event, determine the information for the telephone number in the server. The parameters *ICallID* and *bstrUser* are required for the answer. [SendIdentity](#) is used to return the data to the client.

Example

[Integration in Excel and VB](#)

See also

[Server events overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.3.1.2.2.5 OnLoginData

A user has logged in or logged out.

```
Event OnLoginData(  
    String bstrUser  
    String bstrName,  
    Bool bLogin
```

)

Parameters*bstrUser*

Extension of the user who has logged in or logged out.

bstrName

LOgin name if known.

bLogin

1 User has logged in.

0 User has logged out.

Remarks**See also**[Server events overview](#)[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.3.1.2.2.6 OnNew Call

Preliminary informations for a call come.

```

Event OnNewCall(
    String bstrUser,
    Long ICallID,
    Bool blnbound,
    Long IStart,
    String bstrNumber
)

```

Parameters*bstrUser*

Extension which was called or which is calling

ICallID

Call ID of call whose state has changed.

blnbound

Is true for incoming calls.

IStart

Call start time, as time_t

bstrNumber

Phone number of the partner

Remarks

One provides first of all information with important data of the call.

See also

[Server events overview](#) | [GetDateTime](#) | [GetFormatDateTime](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.3.1.2.2.7 OnProtocolData

A call data record should be stored.

```
Event OnProtocolData(  
    Long IStart,  
    Long IEnd,  
    String bstrUser,  
    String bstrNumber,  
    Bool bSuccess,  
    Bool blnbound,  
    String bstrAddress  
)
```

Parameters

IStart

Call start time, as time_t

IEnd

Call end time, as time_t

bstrUser

User extension

bstrNumber

Telephone number of the other party in the call

bSuccess

True if the call was successful

blnbound

True if the call was incoming

bstrAddress

Address called. This is relevant for incoming calls for a device which responds to several internal extensions (addresses). The telephone can be configured so as to be reachable from outside by different numbers, reflecting different projects or profit centers. Using the address, it is possible to determine the project for which the telephone service was rendered.

Remarks

This event is generated when a call with a telephone number is recognized. The server interface can then save the data.

Example

[Integration in Excel and VB](#)

See also

[Server events overview](#) | [GetDateTime](#) | [GetFormatDateTime](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.3.1.2.2.8 OnUserToDataBase

User-defined data are received for further processing.

```
Event OnUserToDataBase(  
    Long ICommand,  
    String bstrUser,  
    String bstrMessage  
)
```

Parameters

ICommand

Command number

bstrUser

Extension of the user who sent the command.

bstrMessage

Command parameter

Remarks

A client has sent a user-defined command to the server interface. This can be achieved using [EUserSendToDataBase](#) in the client. The server handles the event and can respond to the client using [SendUserData](#). The parameters *ICommand* and *bstrUser* are required for the answer to the client.

Example

[Integration in Excel and VB](#)

See also

[Server events overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.3.1.3 C++ CTI API

The interface for the C++ API is in file FCTIDATA.DLL. You will also need files CTIDATA.H, CTITYPES.H and FCTIDATA.LIB from the CTI SDK. Copy these files to the relevant directories in your development environment. Perform a server set-up on the target computer so that all necessary files are present and registered on the machine.

[Functions](#)

[Events](#)

See also

[Command reference - Server](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.3.1.3.1 Functions

Prefix

The server interface functions start with the prefix "CTI_T".

Initialisation

CTI_Init	Initialise the server interface.
--------------------------	----------------------------------

Data exchange

CTI_SendIdentity	Send caller number identification back.
CTI_SendUserData	Send user-defined data.

Additional functions

CTI_ConvertToCanonical	Convert a number into canonical form.
CTI_GetLicenseCount	Return the number of CTI licences.
CTI_GetDateTime	Converts a time_t var into a standard string.
CTI_GetFormatDateTime	Converts a time_t var into an individual formatted string.
CTI_MakeCall	Initiate call.
CTI_Answer	Answers a call.
CTI_Drop	Hang up a call.
CTI_Conference	Connect two calls to a telephone conference.
CTI_Pickup	Picks up a call from an other phone device.
CTI_Redirect	Redirect a call on to another telephone.
CTI_Toggle	Switch back and forth between two calls.
CTI_Transfer	Connect two calls.

See also[C++ CTI API overview](#)

[Send feedback to TAPIMaster®](#)[© 2020 Tino Kasubke. All rights reserved.](#)

1.3.1.3.1.1 CTI_Answer

Answers a call.

```
VOID WINAPI CTI_Answer(  
    PTCHAR pszUser  
);
```

Parameters*pszUser*

Extension of user who is answer.

Return value

None

Remarks

A licence must be present for the user.

See also[Server functions overview](#)

[Send feedback to TAPIMaster®](#)[© 2020 Tino Kasubke. All rights reserved.](#)

1.3.1.3.1.2 CTI_Conference

Connect two calls to a telephone conference.

```
VOID WINAPI CTI_Conference(  
    PTCHAR pszUser  
)
```

Parameters*pszUser*

Extension of the user who starts the conference.

Remarks

A licence must be present for the user.

Example[ServerInterface](#)**See also**[Server functions overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.3.1.3.1.3 CTI_ConvertToCanonical

Convert telephone number to canonical form.

```
VOID CTI_ConvertToCanonical(  
    PTCHAR pszInput,  
    PTCHAR pszOutput  
)
```

Parameters

pszInput

Number to convert to canonical form

pszOutput

Converted number

Return value

None

Remarks

You can use this function to convert a dataset containing telephone numbers to canonical form. The output string should be at least 32 characters long.

See also

[Server functions overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.3.1.3.1.4 CTI_Drop

Hang up a call.

```
VOID WINAPI CTI_Drop(  
    PTCHAR pszUser  
);
```

Parameters

pszUser

Extension of user who is drop.

Return value

None

Remarks

A licence must be present for the user.

See also

[Server functions overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.3.1.3.1.5 CTI_GetDateTime

Converts a time_t var into a standard string.

```
PTCHAR CTI_GetDateTime(
    __time32_t wTime,
);
```

Parameters

wTime

time_t var, e. g. delivered in the event [CTI_PROTOCOLDATA](#).

Return value

Zero terminated string in a standard format: 2007-01-12T00:53:32.875

Remarks

Use this function to get a sortable date format string. Any programs like Excel use this format.

See also

[Server functions overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.3.1.3.1.6 CTI_GetFormatDateTime

Converts a time_t var into an individual formatted string.

```
PTCHAR CTI_GetFormatDateTime(
    __time32_t wTime,
    PTCHAR pszFormat
);
```

Parameter

wTime

time_t var, e. g. delivered in the event [CTI_PROTOCOLDATA](#).

pszFormat

Date format. The format can be created with this format tokens.

Token	Description
%a	Abbreviated weekday name
%A	Full weekday name
%b	Abbreviated month name
%B	Full month name
%c	Date and time representation appropriate for locale

%d	Day of month as decimal number (01 – 31)
%H	Hour in 24-hour format (00 – 23)
%I	Hour in 12-hour format (01 – 12)
%j	Day of year as decimal number (001 – 366)
%m	Month as decimal number (01 – 12)
%M	Minute as decimal number (00 – 59)
%p	Current locale's A.M./P.M. indicator for 12-hour clock
%S	Second as decimal number (00 – 59)
%U	Week of year as decimal number, with Sunday as first day of week (00 – 53)
%w	Weekday as decimal number (0 – 6; Sunday is 0)
%W	Week of year as decimal number, with Monday as first day of week (00 – 53)
%x	Date representation for current locale
%X	Time representation for current locale
%y	Year without century, as decimal number (00 – 99)
%Y	Year with century, as decimal number
%z,%Z	Either the time-zone name or time zone abbreviation, depending on registry settings; no characters if time zone is unknown
%%	Percent sign

Return value

Zero terminated string in an individual format.

Remarks

The format tokens can be combined.

See also

[Server functions overview](#)

[Send feedback to TAPIMaster@](#)

[© 2020 Tino Kasubke. All rights reserved.](#)

1.3.1.3.1.7 CTI_GetLicenseCount

Returns the number of licences.

```
LONG WINAPI CTI_GetLicenseCount();
```

Parameters

None

Return value

LONG

Number of licences

Remarks

The 0 indicates an unlicensed version or that a licence with limited lifetime has expired.

See also

[Server functions overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.3.1.3.1.8 CTI_Init

Initialise the server interface.

```
VOID WINAPI CTI_Init(  
    DataCallbackProc pCallbackProc  
);
```

Parameters

pCallbackProc
Address of a [callback](#) function

Return value

None

Remarks

This function initialises the server interface in FCTIDATA.DLL and should be called before the interface is used.

See also

[Server functions overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.3.1.3.1.9 CTI_MakeCall

Initiates a call from the server.

```
VOID WINAPI CTI_MakeCall(  
    PTCHAR pszUser,  
    PTCHAR pszNumber  
);
```

Parameters

pszUser
Extension of user who is calling

pszNumber
Number to call

Return value

None

Remarks

A licence must be present for the user.

See also

[Server functions overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.3.1.3.1.10 CTL_Pickup

Picks up a call from an other phone device.

```
VOID WINAPI CTL_Pickup(  
    PTCHAR pszUser,  
    PTCHAR pszNumber  
)
```

Parameters

pszUser

Extension of the user who picks a call.

pszNumber

Extension of the device from which the a call should be picked up.

Remarks

A licence must be present for the user.

Example

[ServerInterface](#)

See also

[Server functions overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.3.1.3.1.11 CTL_Redirect

Redirect a call on to another telephone.

```
VOID WINAPI Redirect(  
    PTCHAR pszUser,  
    PTCHAR pszNumber  
)
```

Parameters

pszUser

Extension of the user who passes a call on.

pszNumber

Destination telephone number to which the call should be redirected.

Remarks

A licence must be present for the user.

Example

[ServerInterface](#)

See also

[Server functions overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.3.1.3.1.12 CTI_SendIdentity

Send a caller identification message back.

```
VOID WINAPI CTI_SendIdentity(  
    LONG ICallID,  
    PTCHAR pszUser,  
    PTCHAR pszText  
);
```

Parameters

ICallID

Call ID of call

pszUser

Extension of user

pszText

Text containing the data about the caller

Return value

None

Remarks

The server interface has received event [CTI_GETIDENTITY](#), whereupon information about the calling number was determined. The call ID and extension of the user, which were received on [CTI_GETIDENTITY](#), are also sent back. *pszText* should be formatted suitably for display by the client in question.

See also

[Server functions overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.3.1.3.1.13 CTI_SendUserData

Send user-defined data.

```
VOID WINAPI CTI_SendUserData(  
    LONG ICommand,  
    PTCHAR pszUser,  
    PTCHAR pszText  
);
```

Parameters

ICommand

Command number

pszUser

Extension of user who sent the command

pszText

Command parameter

Return value

None

Remarks

TAPIMaster® allows user-defined data to be sent from client to server and back. In this way you may transmit additional data between client and server without having to open an additional connection.

The client sends commands to the server using [CTI_EUserSendToDataBase](#). Event [CTI_DATABASETOUSER](#) is triggered upon their arrival at the server.

See also

[Server functions overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.3.1.3.1.14 CTI_Toggle

Switch back and forth between two calls.

```
VOID WINAPI CTI_Toggle(  
    PTCHAR pszUser  
)
```

Parameters

pszUser

Extension of the user who shall change between the calls.

Remarks

A licence must be present for the user.

Example

[ServerInterface](#)

See also

[Server functions overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.3.1.3.1.15 CTI_Transfer

Connect two calls.

```
VOID WINAPI CTI_Transfer(  
    PTCHAR pszUser  
)
```

Parameters

pszUser

Extension of the user who shall connect two calls.

Remarks

A licence must be present for the user.

Example

[ServerInterface](#)

See also

[Server functions overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.3.1.3.2 Events

Delivery

All events arrive via a callback function. The function is defined as follows:

```
typedef VOID (__stdcall* DataCallbackProc)(DWORD,LPVOID);
```

DWORD contains the type listed below, LPVOID points to a structure which, according to the function, is either [CTI_COMMAND_MSG](#) or [CTI_PHONE_DATA](#).

Types

Parameter	Value	Description
CTI_CALLNUMBER	0x00012001	Signals a number.
CTI_CALLSTATE	0x00030002	Signals the state of calls.
CTI_FORWARDINGS TATE	0x00012004	Signals the state of forwarding.
CTI_IDENTITY	0x0001200D	Request for calling number identification.
CTI_NEWCALL	0x00012018	Preliminary informations for a call come.
CTI_PROTOCOLDATA	0x0001200B	A call data record should be stored.
CTI_USERTODATAB ASE	0x00013007	User-defined data are received for further processing.

See also

[C++ CTI API overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.3.1.3.2.1 CTI_CALLNUMBER

Signals a number.

Sub-message CTI_CALLNUMBER (0x00012001)**Message type**

CTI_COMMAND_MSG

Used values

m_IParam

Call ID

m_unNumber

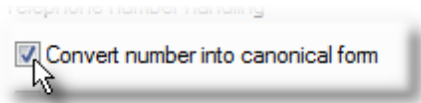
Extension user to whom the number – the number of the party he is in a call with – is being sent.

m_szMessage

Telephone number of other party in call.

Remarks

Signals a telephone number, if one has been received. This should always be the case for outgoing calls. This message may be the first message received for a particular call: in particular, it may come before any [CTI_CALLSTATE](#) message for the call. The telephone number may change during the call if, for example, someone forwarded the call to the user. The server can be configured to send the number in canonical format ("Switch settings" page).

**See also**

[Server events overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.3.1.3.2.2 CTI_CALLSTATE

Signals the state of calls.

Sub-message CTI_CALLSTATE (0x00030002)**Message type**

CTI_COMMAND_MSG

Used values

m_wParam

The call's [call state](#)

m_lParam

Call ID of call whose state has changed

m_unNumber

Extension of user whose call is having its state signalled.

Remarks

The state of a call has changed.

See also

[Server events overview](#) | [Call states](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.3.1.3.2.3 CTI_FORWARDINGSTATE

Signals the state of forwarding.

Sub-message CTI_FORWARDINGSTATE (0x00010004)**Message type**

CTI_COMMAND_MSG

Used values

m_unNumber

Extension of user whose forwarding state is being sent.

m_wParam

1 for active forwarding, 0 if not.

m_szMessage

Target of the forwarding.

Remarks

The forwarding state has changed.

See also

[Server events overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.3.1.3.2.4 CTI_IDENTITY

Request for calling number identification.

Sub-message CTI_IDENTITY (0x0001200D)**Message type**

CTI_COMMAND_MSG

Used values

m_lParam

Call ID of call

m_unNumber

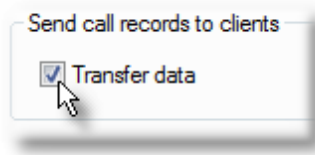
Extension which was called or which is calling

m_szMessage

Telephone number of other party in call

Remarks

This event is triggered when a call occurs and a telephone number is recognised. This only occurs if the signalling of telephone numbers to the clients is active. This can be activated on the “Database interface” page of the server control panel.



Upon reception of this event, determine the information for the telephone number in the server. The parameters *lCallID* and *bstrUser* are required for the answer. [CTI_SendIdentity](#) is used to return the data to the client.

See also

[Server events overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.3.1.3.2.5 CTI_NEWCALL

Preliminary informations for a call come.

Sub-message CTI_NEWCALL (0x0001200B)**Message type**

CTI_COMMAND_MSG

Used values

m_wParam

Call start time, as time_t

m_lParam

Call ID of call

m_unNumber

Extension which was called or which is calling.

m_unOther

wNumber is TRUE for incoming calls.

m_szMessage

Phone number of the partner.

Remarks

One provides first of all information with important data of the call.

See also

[Server events overview](#) | [CTI_GetDateTime](#) | [CTI_GetFormatDateTime](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.3.1.3.2.6 CTI_PROTOCOLDATA

A call data record should be stored.

Sub-message CTI_PROTOCOLDATA (0x0001200B)**Message type**

CTI_PHONE_DATA

Used values

m_dwStart

Call start time, as time_t

m_dwEnd

Call end time, as time_t

m_unActor

User extension

m_szNumber

Telephone number of the other party in the call

m_dwFlags

Bit flag. The bits are:

0x00000002 Call was successful

0x00000004 It was an incoming call

m_szMessage

Address called. This is relevant for incoming calls for a device which responds to several internal extensions (addresses). The telephone can be configured so as to be reachable from outside by different numbers, reflecting different projects or profit centers. Using the address, it is possible to determine the project for which the telephone service was rendered.

Remarks

This event is generated when a call with a telephone number is recognised. The server interface can then save the data.

See also

[Server events overview](#) | [CTI_GetDateTime](#) | [CTI_GetFormatDateTime](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.3.1.3.2.7 CTI_USERTODATABASE

User-defined data are received for further processing.

Sub-message CTI_USERTODATABASE (0x00013007)**Message type**

CTI_PHONE_DATA

Used values

m_IParam

Command number

m_unNumber

Extension of user who sent the command

m_szMessage

Command parameter

Remarks

A has sent a user-defined command to the server interface. This can be achieved using [CTI_EUserSendToDataBase](#) in the client. The server handles the event and can respond to the client using [CTI_SendUserData](#).

See also

[Server events overview](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.3.2 Sample programs

Here you can find an example using C++ and various code sequences.

[Dialing via the IP interface](#)

[Server interface](#)

See also

[Server interface](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

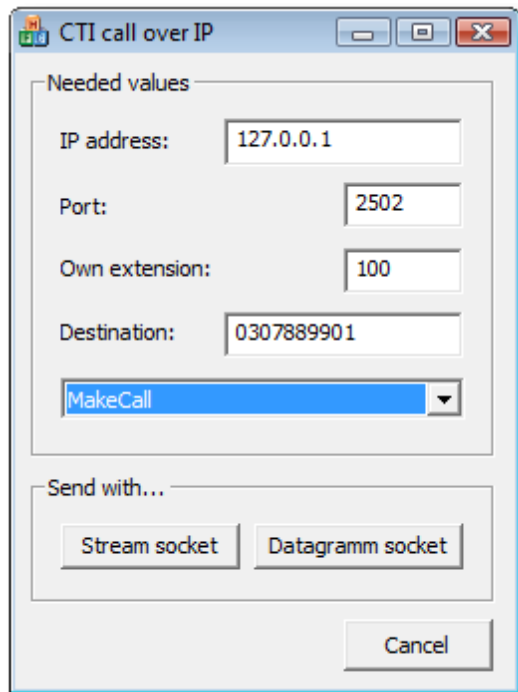
1.3.2.1 Dialing via the IP interface

An IP interface is provided by the licensed version. You can send dial commands to the server, allowing you to dial from other operating systems. The dial command has the format:

MakeCall <Caller> <Number>

e.g., MakeCall 301 01191

(Device 301 should phone 01191.)



The IPSend example contains the source text for dialing on Windows.

See also

[Sample programs](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.3.2.2 Server interface

Necessary files

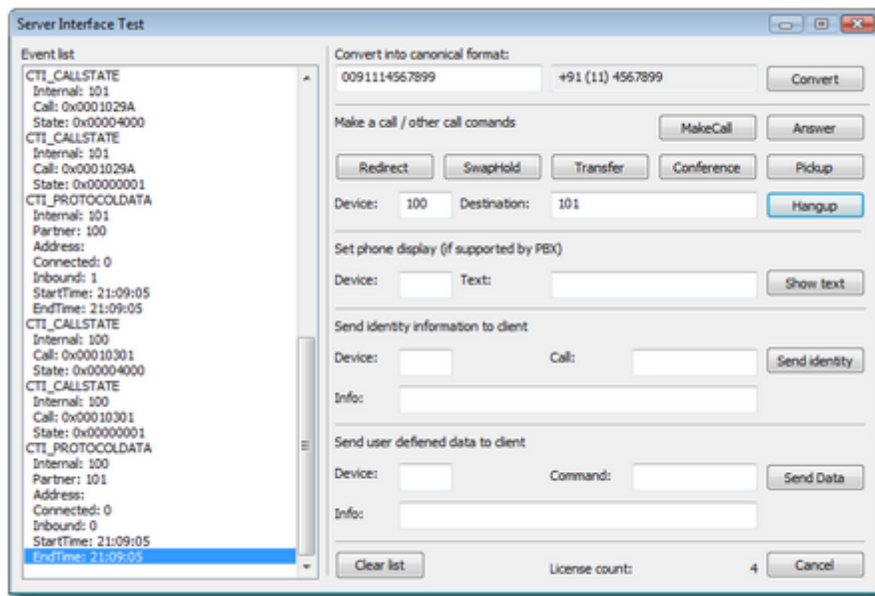
Please install the Server.

Source code

The code is included in the SDK.

Example

A short example of integrating the server interface can be found in the CPlusPlus folder in the SDK. The sample shows the translation into canonical format. Simple telephony functions like MakeCall, Answer and Drop can be processed. There also a VB an a C# version available in the SDK.



See also

[Sample programs](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.4 Install images

General

It would be unreasonable to expect an administrator to install the CTI interface on hundreds of clients. So TAPIMaster® provides several ways to simplify the procedure. A client installation is not necessary in all cases. Both client and server install images can be included in other install images. You can use the [Setup Configurator](#) to preconfigure an installation.

[Using clients without installing](#)
[Embedded client install image](#)
[Embedded server install image](#)
[Customised server install image](#)
[Installation configuration](#)
[Deinstallation](#)

See also

[Contents](#)

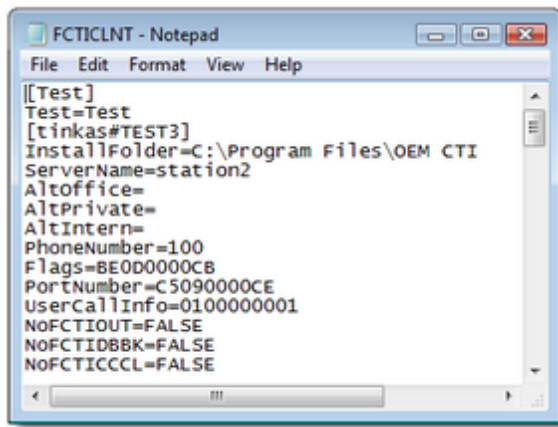
[Send feedback to TAPIMaster®](#)
© 2020 Tino Kasubke. All rights reserved.

1.4.1 Using clients without installing

The simplest kind of set-up is the one that can be avoided altogether. To do this proceed as follows:

- Run the CTI server set-up program.
- The files required for set-up can be found in folder ..\Program files\TAPIMaster®\Setup. Share this directory to the network or copy the files contained there to a shared directory.
- If you have written your own client, copy it into the shared directory too. This method does, however, prevent the client from being able to use any [ActiveX interface](#).
- Create a link to the standard client or to your client. The link should be well-defined in the network.
- E-mail the link to all users.

The clients can then be started without being installed. When first started, they will ask for the [connection parameters](#). Clients generally store their settings locally. In this type of set-up, however, it makes sense for all clients to use a common configuration file. To this end, install a client locally and then copy its FCTICLNT.INI file from the Windows directory into the shared network directory. The clients will then use this file after start-up. Each client uses its own section of the file and different users on the same machine also use different sections. So the different users will not overwrite each other's configuration. Here is an example FCTICLNT.INI file:

**See also**

[Install images](#) | [CTICLIENT_CHECKCONNECTION](#) | [CTI_OGetIniFileName](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.4.2 Embedded client install image

The client is usually installed using a dialog with several pages. This is, however, impractical when the client installation is part of a larger installation. It is therefore possible to embed the client install image into your own set-up method. This is done as follows.

- Copy the files from directory ..\Program files\TAPIMaster®\Setup into a sub-directory within your install image. Remove unnecessary [files](#). Add your own client as necessary.
- The directory contains a file FCTIINST.INI, which contains the settings for the set-up. Edit this file using the [Setup Configurator](#) so dass der Client mit den gewünschten Einstellungen gestartet wird. so that the client is started with the desired settings. Activate “automatic set-up”.
- Call the SETUP.EXE file during the installation so that it can install the client.

See also

[Install images](#) | [Setup Configurator](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.4.3 Embedded server install image

The server set-up can be embedded in a larger install image without the TAPIMaster® dialog being displayed. This is done as follows:

- Copy directory ..\Program files\TAPIMaster®\Setup into your installation. Remove unnecessary [files](#).
- At the start of your set-up, start the file LW.EXE with the parameter -I. The Line Watcher will not be displayed but will create an [installation configuration](#) for the machine.
- Display the server settings during your set-up. It may help you to compare also the [server setup](#) page of the [Setup Configurator](#).
- Save any values the user has changed back to the [installation configuration](#).
- Start the SETUP.EXE file with the parameter -S. The server will be installed and configured.

See also

[Install images](#) | [Setup Configurator](#) | [installation configuration](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.4.4 Customised server install image

Imagine a customer is having problems performing the server installation and you want to help him. You can do this as follows.

- The customer starts the [Line Watcher](#) and presses the “Save data” switch on the “TAPI line capabilities” page. An [installation configuration](#) will be generated. The customer e-mails this file (FCTIINST.INI) to you and tells you the internal extensions he would like to use.
- Open the [installation configuration](#) using the [Setup Configurator](#). You can now see the customer’s TAPI configuration.
- Make sure there is a switch driver installed on the customer’s machine. Configure the desired switch and activate the extensions. This is a good opportunity to issue the licences as well.
- Edit the “Basics” page of the [Setup Configurator](#). “Automated set-up” should be active.
- Create a self-unpacking file from the directory which then calls SETUP.EXE when unpacked.
- The customer starts the file and the server will be configured automatically.

See also

[Install images](#) | [Setup Configurator](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.4.5 Installation configuration

The file FCTIINST.INI contains the installation settings. The file is evaluated during program installation and the preset components are installed. The file can be created by the [Line Watcher](#) and by the [Setup Configurator](#). You used the [Setup Configurator](#) to make these settings while setting up the client. For the [embedded server install image](#) you should make the settings shown in green editable during set-up.

Settings applicable for client and server set-up.

[Default]

```
PortNumber=2501
AutoSetup=0
AutoSetupType=0
MaxIntern=2
CityCode=69
```

Settings for the client set-up.

[Client]

```
ServerName=PDC
Flags=414
InstallStandardClient=0
InstallOutlook=0
InstallDatabase=0
InstallUserDefined=0
InstallCOM=0
GenerateShortcuts=0
```

For these settings you may be able to use the preset defaults shown here.

[Server]

```
CanonicalFormat=1
SendCallData=1
SaveTextFile=1
SelfRepair=0
Pickup=1
Redirect=1
Remember=1
Forward=1
Chat=1
AddToGroup=1
License=
ComputerCode=494-113-823-124
```

If NumProvider is Null, no switch driver is installed and the server installation should be aborted.

[Provider]

```
Provider00=tipTel 195 Service Provider
Provider01=AGFEO TK-ServiceProvider3
NumProvider=2
SelectedProvider=0
```

The configuration for individual switches is shown below. Save the text for the desired switch again, retaining the format {line name;extension;activation flag}, since the lines will otherwise not be recognised during installation. Do not change the line names! For some switches the extension may be omitted.

[tipTel 195 Service Provider]

```
ProviderID=79
MaxNumActiveCalls=1
MaxNumConference=0
TransferModes=0
AddressFeatures=2
CallFeatures=2097348
```

```
CallFeatures2=0
NumLines=1
Line0000={tiptel phone1;;1}

[AGFEO TK-ServiceProvider3]
ProviderID=81
MaxNumActiveCalls=5
MaxNumConference=3
TransferModes=3
AddressFeatures=142
CallFeatures=235047149
CallFeatures2=0
NumLines=12
Line0000={Telefon 101;101;1}
Line0001={Telefon 102;102;1}
Line0002={Telefon 103;103;1}
Line0003={Telefon 104;104;0}
Line0004={Telefon 105;105;0}
Line0005={Telefon 106;106;0}
Line0006={Telefon 107;107;0}
Line0007={Telefon 20;20;0}
Line0008={Telefon 21;21;0}
Line0009={Telefon 31;31;1}
Line0010={Telefon 32;32;1}
Line0011={Telefon 33;33;1}
```

See also

[Install images](#) | [Setup Configurator](#)

[Send feedback to TAPIMaster@](#)

© 2020 Tino Kasubke. All rights reserved.

1.4.6 Deinstallation

You can also use the [Setup Configurator](#) to produce programs which remove the client and server from a machine. Removal then occurs without query. Activate “Automated set-up” on the “General” page of the Setup Configurator and activate “Deinstallation”. The same [files](#) are required as for server installation. Generate a self-unpacking file from the directory which calls SETUP.EXE when unpacked. The files will be removed.

See also

[Install images](#) | [Setup Configurator](#)

[Send feedback to TAPIMaster@](#)

© 2020 Tino Kasubke. All rights reserved.

1.5 Tools

Various development aids are available. You can use the CTI Browser to try command out without having to write any code. You can use the [Setup Configurator](#) to integrate the CTI set-up into your own program set-up.

[CTI browser](#)

[Line Watcher](#)

[Setup Configurator](#)

See also

[Contents](#)

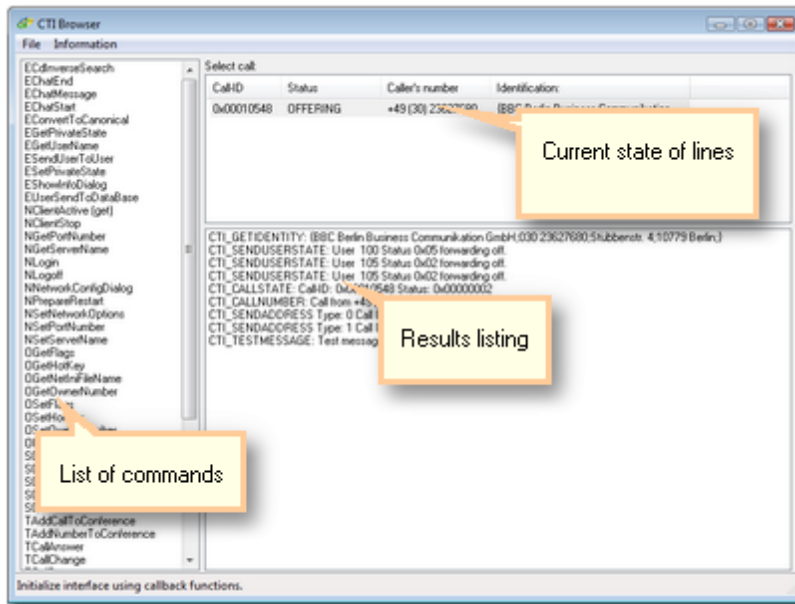
[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.5.1 CTI Browser

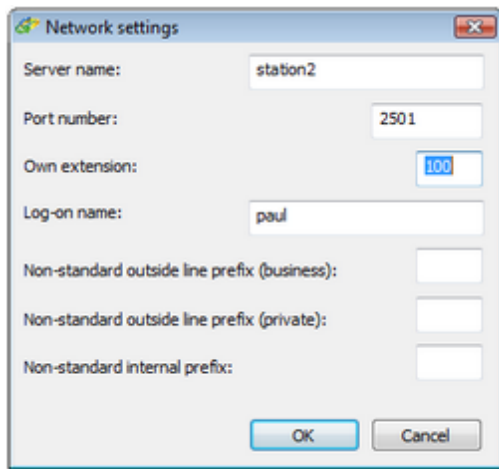
Purpose

The CTI Browser is a test client and can be used as a development tool to call individual methods of the client interface. The resulting data flow from the server is displayed. It is thus possible to test code sequences without writing any code. Before using the CTI Browser, you should take a little time to become familiar with the [Schnittstelle](#).

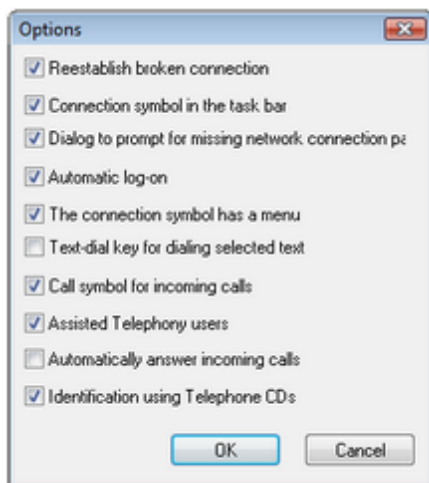


USE

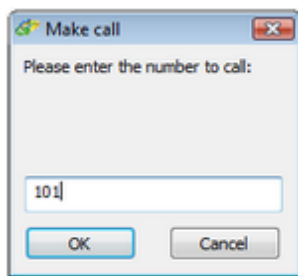
- First check that the CTI server is installed and running.
- The client should also be installed on the test machine. Stop the client if it is running.
- Then start the CTI Browser.
- On the left hand side you can see the commands from the [ActiveX reference](#). The corresponding C++ functions are named similarly but with a prefix of "CTI_". In the status line you can see a short description of the selected command. The commands can be launched by double-clicking. Some commands are only shown when the connection to the server is active.
- Start with [NNetworkConfigDialog](#). The following dialog should contain correct entries for server name, port number and your own extension.



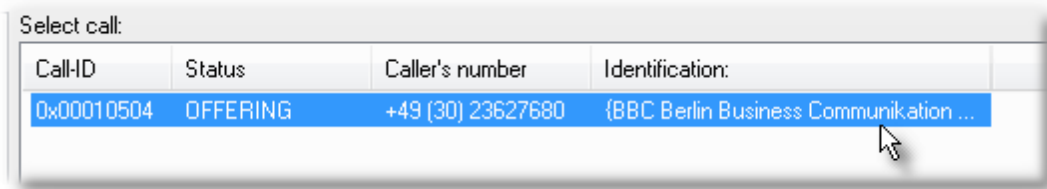
- Next launch command [NStartClient](#). This causes a dialog with the connection options to be displayed. Select the options as shown below. Press OK to confirm.



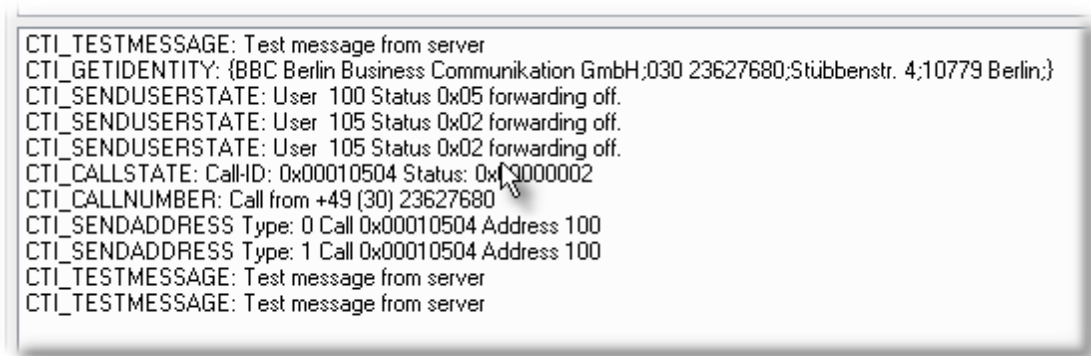
- The connection to the CTI server will now be established. The left hand list will now contain significantly more commands than before. Now launch TMakeCall. You can enter a number in the following dialog and press OK to dial it.



- The call state and telephone number are displayed in the top right of the Browser if they were able to be determined.



- the bottom right you can see all events which were sent to the client.



- You can end the call using [TCallDrop](#).
- Try the other commands out for yourself. Some switches do not support some functions. You can gain a rough overview of the switch's capabilities using the [Line Watcher](#). This program is contained in the Setup. You may also download the file [here](#).

See also

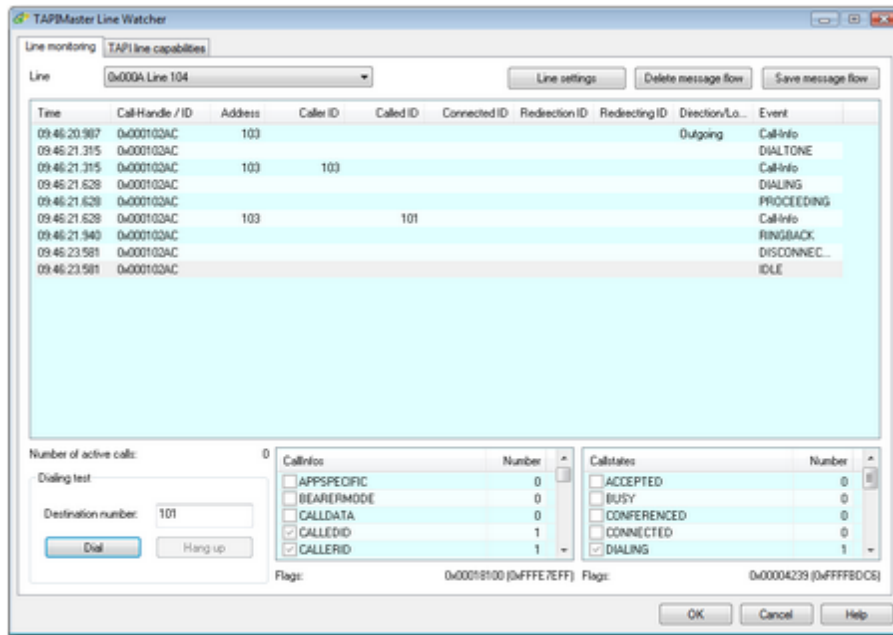
[Tools](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.5.2 Line Watcher

A slimmed-down version of the Line Watcher is provided in the installation package. The program allows you to see what is happening on a TAPI line. If a client has stopped responding to the switch, it is worth checking things out with the Line Watcher. This will show you, for example, whether telephone numbers are being signalled and whether TAPI is still active. On the second page of the program you can see the features which the manufacturer claims are supported. You can download the latest version of the program [here](#).



See also

[Tools](#)

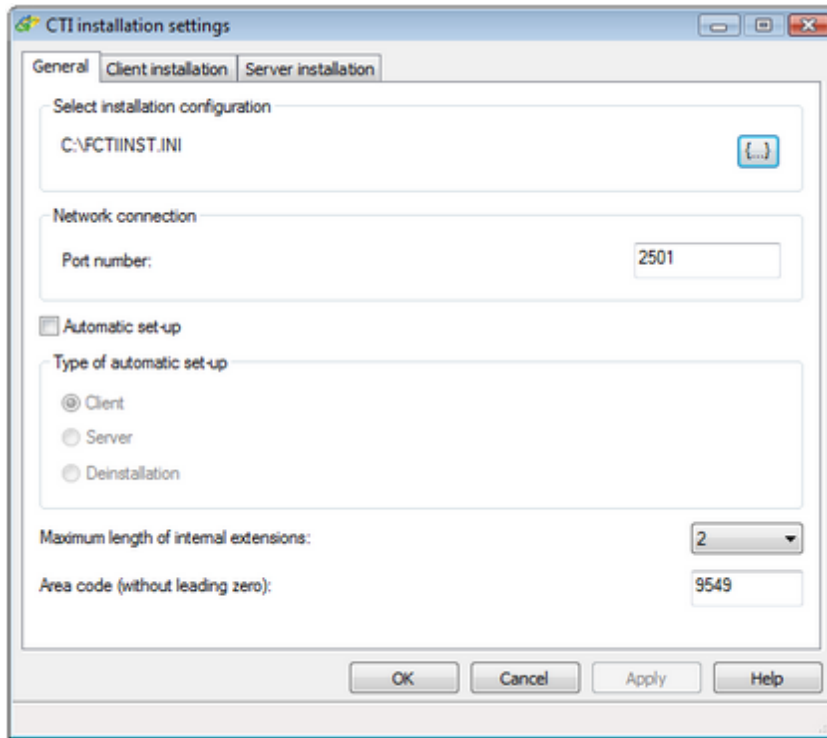
[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.5.3 Setup Configurator

You will generally need to install the CTI software before installing your own software for a customer. A manual installation methodology acceptable for a server may not be acceptable for hundreds of clients. The Setup Configurator provides a way to produce automated install images for [server](#) and [client](#), which can then be integrated into the main install image.

- You can find the settings for client and server on the "Basics" page. First choose the configuration file from the setup directory.
- The same port number can be used for client and server.
- Upon starting set-up, a wizard usually appears to guide you through the installation. This is not helpful for an integrated client set-up. Select "Automated set-up" to perform installation in one go. You may then generate an install image for client or server or generate a deinstallation. Alternatively, you may start SETUP.EXE with the parameter -D (deinstallation), -S (server) or -C (client) to force an automated installation. For the client, an extension can be supplied too: SETUP.EXE -C 251 installs a client with extension 251.
- The maximum length of internal extensions can be 2, 3 or 4.
- The local area code is signalled without the leading zero.



[Used files](#)

[Client install image](#)

[Server install image](#)

See also

[Tools](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.5.3.1 Used files

The files required depend on the type of install image. Copy these files into a directory and add them to your install image. The FCTIINST.INI file should then be prepared using the [Setup Configurator](#). Compress the directory with WinZip and arrange for SETUP.EXE to be called after unpacking.

File name	Server	Standard Client with Outlook and database telephone book	Standard client without additional telephone books	Third party application with COM and restart capability	Third party application without COM and restart capability
FCTIACD.DLL	X				
FCTIACVW.EXE	X				
FCTICAN.DLL	X	X	X	X	X
FCTICAN.TLB	X			X	
FCTICLNT.DLL		X	X	X	X
FCTICLNT.EXE		X	X		
FCTICLNT.TLB				X	
FCTICLNT.TSP		X			
FCTIDATA.DLL	X				
FCTIDATA.TLB	X				
FCTIDBBK.DLL		X			
FCTIHELP.CHM	X	X	X		X
FCTIHOOK.DLL		X	X		
FCTIINI.DLL	X	X	X	X	X
FCTIINST.INI	X	X	X	X	X
FCTILANG.DLL	X	X	X	X	X
FCTIOUT.DLL		X			

FCTIPHHK.DLL		X	X	X	X
FCTIREST.EXE		X	X	X	
FCTIRMVE.EXE	X	X	X	X	X
FCTISCUT.EXE			X		
FCTISRV.CPL	X				
FCTISRV.EXE	X				
LW.EXE	X				
SETUP.EXE	X	X	X	X	X
STOPTAPI.EXE	X				

See also

[Setup Configurator](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.5.3.2 Client install image**Connection options**

Here the options described in [connection options](#) can be activated individually. There is no longer any need to call [CTI_OSetOptions](#) or [OOptions](#). The client installation can be fully automated. The first time the program is started the user will have to supply his own extension, unless that is, the installation was initiated using `SETUP.EXE -C <extension>`, e.g., `SETUP.EXE -C 255`.

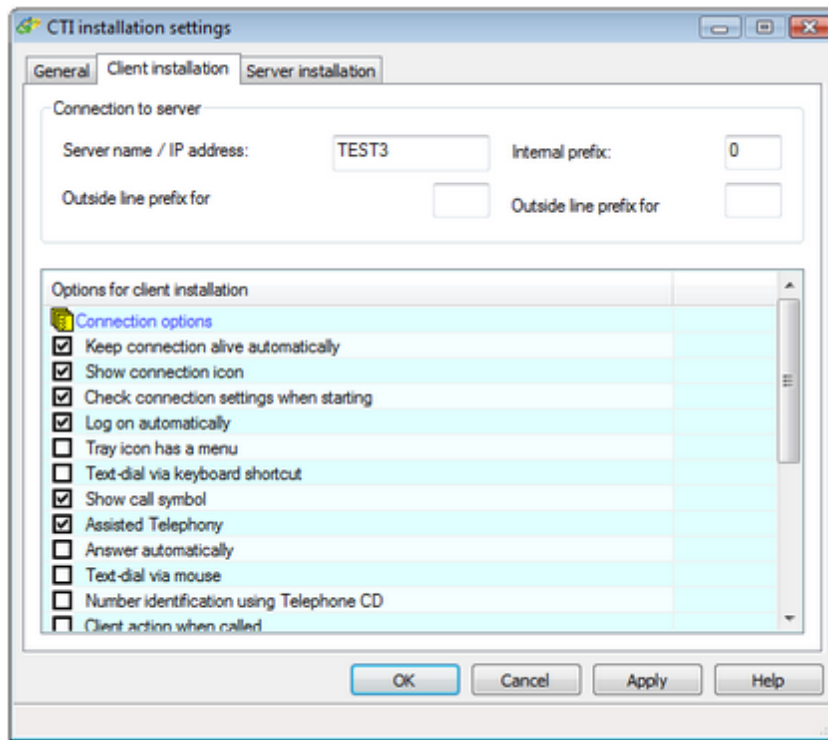
Components installed

The standard client is normally installed together with its telephone books. This is of course unnecessary if you are integrating the interface into your own products. Outlook may not be installed in the company where the client is used. You can deactivate what you do not need.

Other settings

If the [ActiveX](#) interface is not used, you do not need to register the interface. The desktop short-cuts can be suppressed.

The setup program does not overwrite any existing extensions. It can therefore also be used for performing software updates.



See also

[Setup Configurator](#)

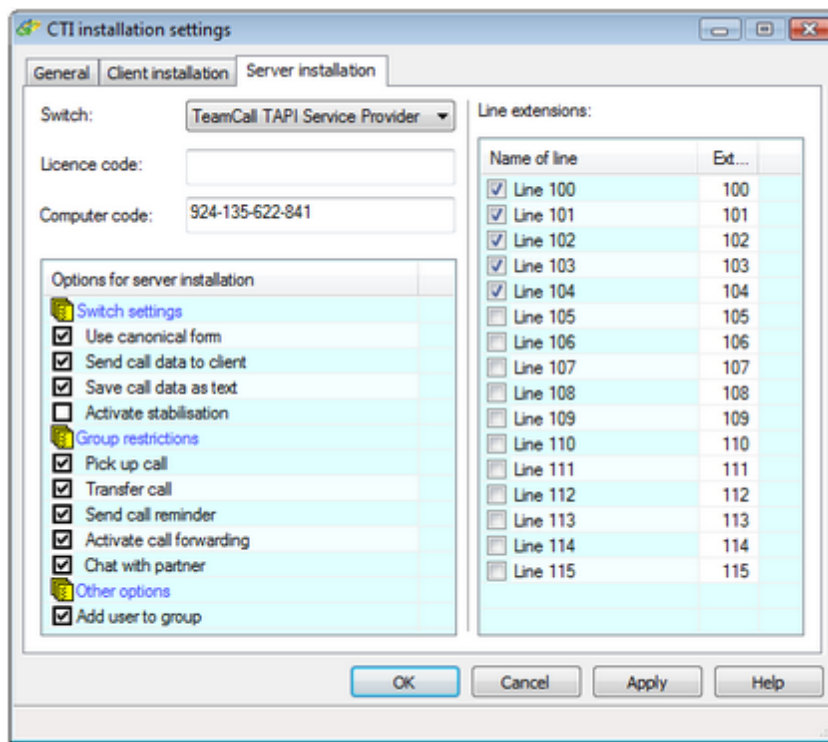
[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.5.3.3 Server install image

General

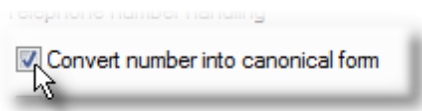
When you open an [installation configuration](#) which the customer created using the [Line Watcher](#) the fields "Switch", "Computer code" and "Line extensions" will already be filled in. First select the correct switch. Activate the extensions which are to be used. The licence code can be entered if it is known.



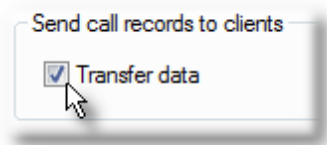
Switch settings

These options are overwritten during each installation.

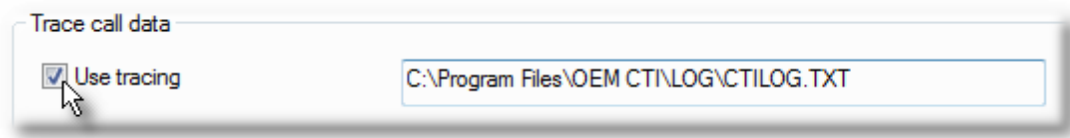
It makes sense for telephone numbers to be canonically formatted unless the client formats them in a special way. This option can easily be changed later on the "Switch settings" page of the server control panel.



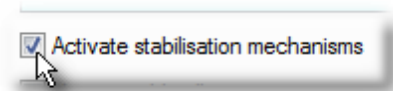
If the client keeps a list of calls, like the standard client does, call data should be sent to the client. The option can be found on the "Database interface" page of the server control panel.



If the database interface is not used, calls are not normally logged. You may, however, achieve minimal logging by having the data saved to a text file. The file can then be read by Excel for further processing. The option can be found on the "Database interface" page of the server control panel.



If a client working with a large switch stops responding – for example, calls can no longer be hung up – try activating stabilization. Calls may then, however, sometimes get hung up when TAPI delivers the wrong state for them. The option can be found on the “TAPI special treatment” page of the server control panel.

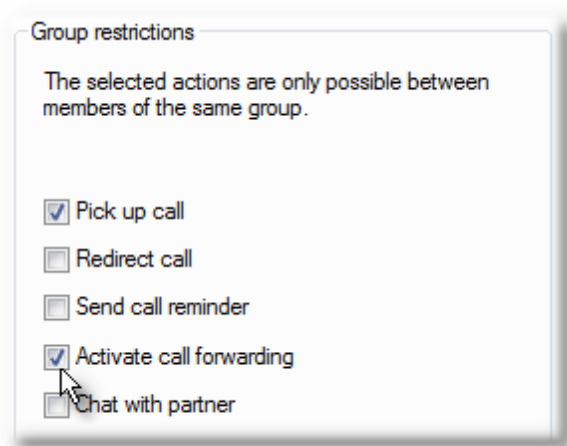


Group limitations

These options are also always overwritten. They restrict the following functions.

[CTI_TPickup / TPickup](#)
[CTI_TCallRedirect / TCallRedirect](#)
[CTI_TSendCallReminder / TSendCallReminder](#)
[CTI_TSetForwarding / TSetForwarding](#)
[CTI_EChatStart / EChatStart](#)

You can find these settings on the “Groups” page of the server control panel.



Other options

All the extensions present may be put in the first group. The display of group members and their status will then work correctly straight away. Do not activate this option if you have several hundred clients as it will cause too much network traffic.

See also

[Setup Configurator](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.6 Customization

As a software manufacturer or retailer you can customize the appearance of TAPIMaster® to your personal needs. The following adjustments can currently be made to names, text, colour and logo.

[Installation](#)

[Client interface](#)

[CTI server](#)

[Server control panel](#)

[Line Watcher](#)

[Languages](#)

[Required resources](#)

See also

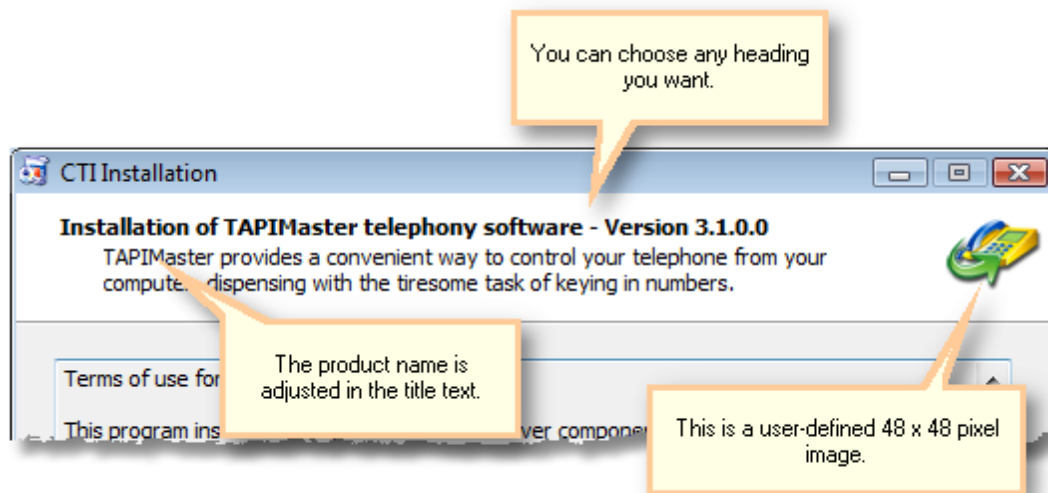
[Contents](#)

[Send feedback to TAPIMaster®](#)

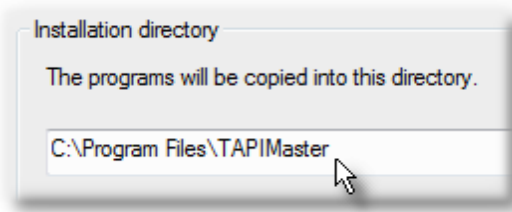
© 2020 Tino Kasubke. All rights reserved.

1.6.1 Installation

Title and product name and logo can be altered in the installation dialog.



The installation directory corresponds to the product name.

**See also**

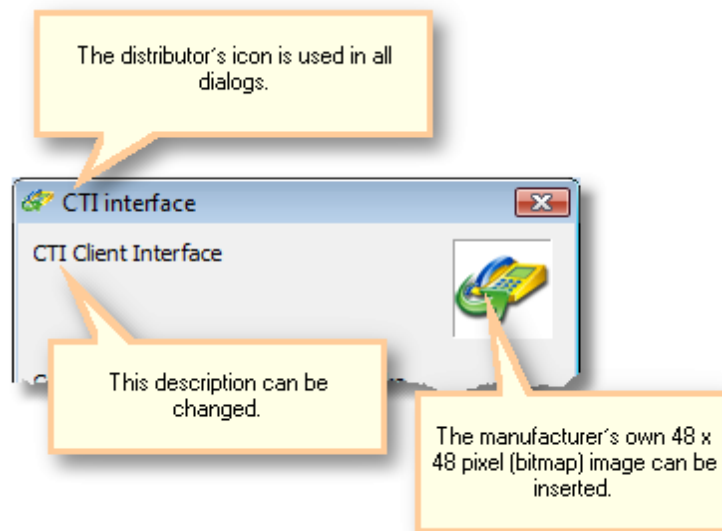
[Customization](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.6.2 Client interface

The client interface has an information dialog, which can be displayed using [EShowInfoDialog](#) or [CTI_EShowInfoDialog](#).



The connection symbol in the task bar can be changed at run-time with function [OConnectionIconOptions](#) / [CTI_OConnectionIconOptions](#).

See also

[Customization](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

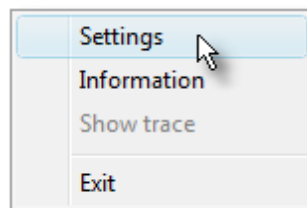
1.6.3 CTI Server

The server places an icon in the task bar when running. There is an icon to show the server is running and another for the case that the server could not start or has been stopped.

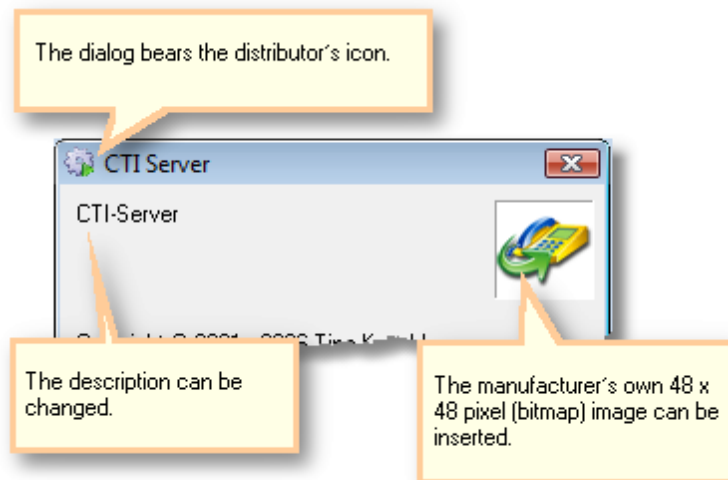


A user-defined tool-tip is displayed over the icon.

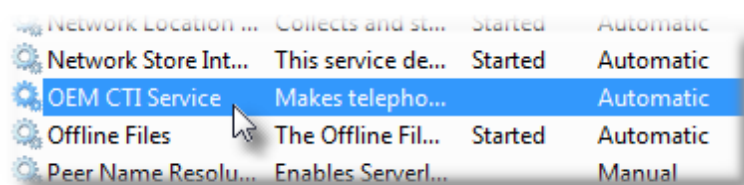
An information dialog can be called up from the server menu.



Some elements of this dialog can also be customized.



The CTI server usually runs as a Windows service. The name of this service can be changed.



See also

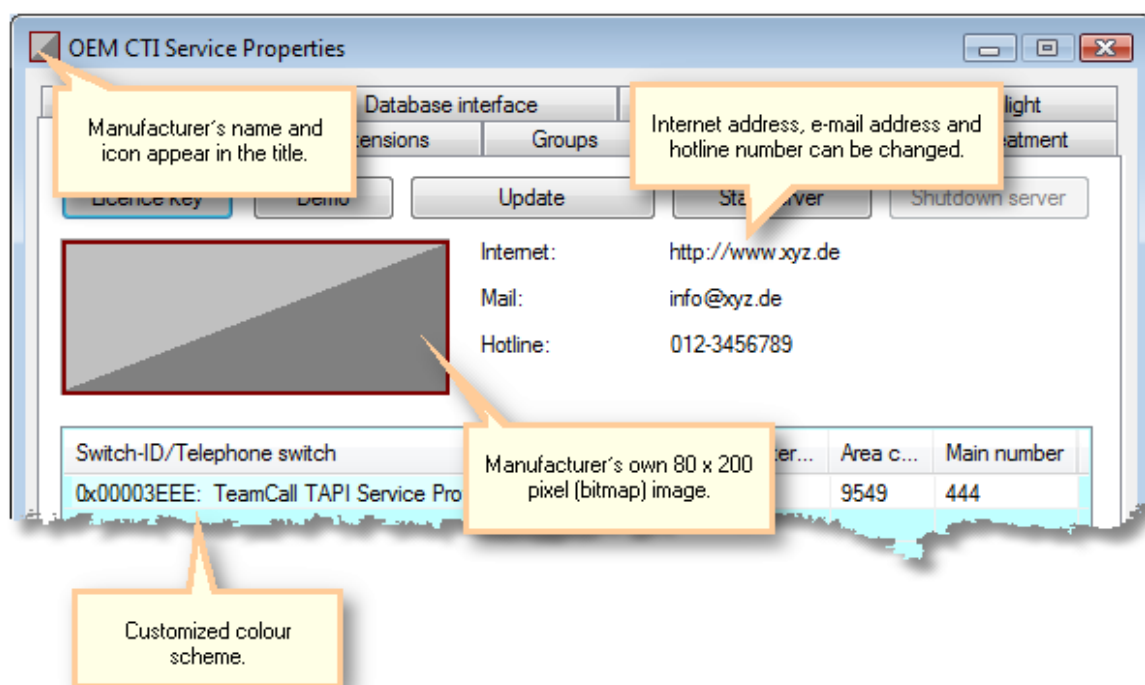
[Customization](#)

[Send feedback to TAPIMaster®](#)

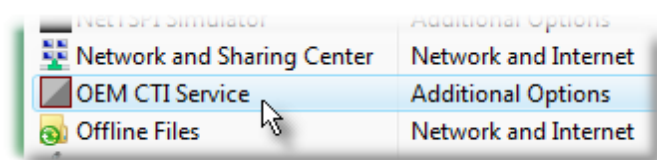
© 2020 Tino Kasubke. All rights reserved.

1.6.4 Server control panel

Text, icons, images and colours can be customized.



The server control panel can be reached under the Control Panel. The name shown there is can be changed.



See also

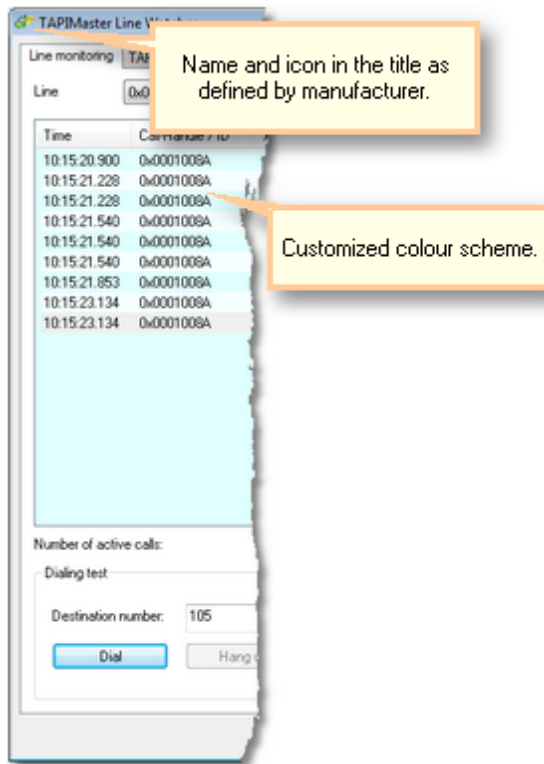
[Customization](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.6.5 Line Watcher

The Line Watcher can also be adapted to fit the manufacturers software.



See also

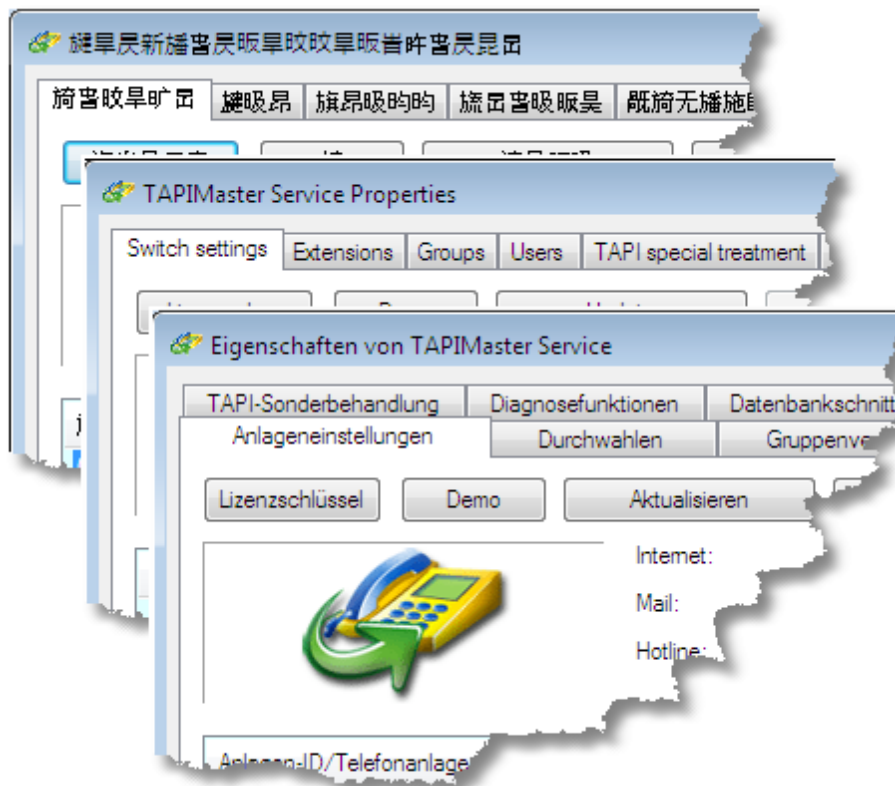
[Customization](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.6.6 Languages

TAPIMaster® gets its natural language resources from FCTILANG.DLL. By translating this file in a resource editor you can use TAPIMaster® in other languages. For Chinese, Japanese and certain other languages, please request the Unicode version of TAPIMaster®.



There are several guidelines to follow when translating the natural language resources. Do not remove any formatting symbols, e.g., %d, %s. Nor should you change the order of these directives. Do not delete or change any formatting instructions, e.g., \n, \r, \t. Breaking these guidelines can lead to program crashes and we can no longer guarantee that the program will function correctly.

190	Maximum length of internal
191	Country code: %d
192	Prefix: %s
193	Main number: %d
194	Main number: %d

See also

[Customization](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

1.6.7 Required resources

Summary of images, text strings and constants required

The following resources are required in order to customize the appearance of TAPIMaster® to fit the needs of a software manufacturer or distributor:

Bitmaps

Square bitmap 48 x 48 Pixel.

Example



Rectangular bitmap 200 x 80 Pixel

Example



Icons

One icon for the application, one for the active server and one for the inactive server.

Texte

- Product name
- Company name
- Service name, as it is to appear in the Control Panel and Services window.
- Internet address
- E-mail address
- A short description of the server [client interface](#) and [server interface](#)
- Title for the [installation dialog](#)

Colour constants

The Colour constants are specified as RGB-values, e.g. RGB(192,192,192) for pale grey.

- Colour of text in tables, normal
- Colour of text in tables, sub-headings.
- Colour of text in tables, deactivated text
- Colour of text background in tables, changeable data
- Colour of text background in tables, area outside the entries
- Colour of text background in tables, even lines
- Colour of text background in tables, odd lines

See also

[Customization](#)

[Send feedback to TAPIMaster®](#)

© 2020 Tino Kasubke. All rights reserved.

Index

- A -

Additional functions
 (ActiveX) 106
 (C++ CTI API) 170
 Assisted Telephony 79, 208
 Automated Setup files 315

- C -

Call 123, 192
 Address signalling 128, 196
 Answering 92, 155
 Call forwarding 93, 156
 Call retry 98, 161
 Connecting two calls 94, 157
 Dialing 96, 160
 Dialing by proxy 97, 161
 Hanging up 92, 155
 Holding a call 93, 156
 Identification by the server 274, 281, 293, 298
 Picking up a call 97, 160
 Retrieving a held call 95, 158
 Sending a call reminder 99, 117, 162, 187
 Telephone book CD 123, 192
 Telephone number identification 123, 192
 Telephone number signalling 116, 186
 Toggling between calls 94, 157
 Call data records 127, 195, 283, 300
 Call forwarding 95
 Activating forwarding 95, 99, 163
 Being the object of forwarding 126, 195
 Querying 95, 158, 159
 Signalling upon log-on 122, 191
 Call states
 List of states 117, 207
 Signalling 188
 Call symbol
 Display for incoming calls 86, 210
 Removing 98, 162
 Chat
 Ending a session 107, 119, 170, 188
 Sending text 107, 118, 171, 189

Starting a session 107, 118, 171, 189
 Check connection parameters 80, 210
 Client - restart 74, 119, 140, 190
 Command reference
 Client 4
 Server 220
 Conference
 Add call 91, 154
 Call and add new member 91, 154
 Show status 120, 190
 Configuration file
 Client 84, 146
 Setup 315
 Connection 72
 Automatic maintenance 79, 209
 Initialising 75, 142, 143
 Interrupting 72, 136
 Properties 84, 147, 150
 Querying status 72, 135
 Re-establishing 72, 136
 Status 120, 182, 183, 198
 Connection dialog 73, 139
 Connection icon
 In task bar 81, 86, 211
 Menu 87, 126, 180, 212
 Connection options 208
 Constants 206
 Contents 1
 Country settings 125, 194
 CTI Browser 311
 Customized server install image 307

- D -

Data structures 202
 Database queries 203
 Deinstallation 309
 Development aids 310
 Development tools
 CTI Browser 311
 Setup Configurator 315
 Dialing marked text 83, 210, 211
 Dialing prefix 125, 194

- E -

Embedded client install image 306

Embedded server install image 306

Embedding in install images 305

Events

(ActiveX) 114

(C++ CTI API) 179

- F -

Files for installation 316

Foreground 128, 184

Functions

Client 4

Server 220

- G -

Group members

Listing 122, 192

Status 129, 197

- I -

Incoming calls

Answering 92, 155

Automatic answer 79, 209

Notifying user 87, 149, 152

Information dialog 112, 177

Installation configuration 307

Interfaces

Client 3

Server 219

Testing commands 311

- K -

Keyboard shortcuts

Dialing 82, 83, 145, 149, 210

Hanging up a call 82, 83, 146, 150

- L -

Licence information 124, 193

Licences 2

Line Watcher 314

Logging off the CTI server 73, 138

Logging on to the CTI server 72, 80, 125, 138, 194, 209

- M -

Main telephone number of switch 125, 194

Menu 87, 126, 180, 212

Messages from the administrator 129, 197

- N -

Network connection 74, 75

Querying parameters 74, 75, 137, 148

Setting parameters 73, 74, 75, 85, 139, 140, 141, 142, 151

Network dialog 73, 139

Network functions

(ActiveX) 71

(C++ CTI API) 135

Network messages 185

- O -

Options

(ActiveX) 77

(C++ CTI API) 144

- P -

Prepare for restart 74, 119, 140, 190

Program examples

Client 213

CTI client with C++ 215

CTI client with Delphi 216

Dialing via IP interface 303

Excel as CTI client 214

Server 302

Server interface in Excel / VB 304

SQL handling 217

- R -

Redial 98, 161

References / links

Client 69

Server 261

- S -

Sending a call reminder 99, 117, 162, 187
Sending messages 111, 132, 175, 198
Server interface
 Initialising 272, 290
Setup Configurator 315
Setup program / install image 305
Setup programs / install images
 Automated Setup 315
 Client 317
 Files 316
 Installation configuration 307
 Server 318
Short messages 111, 132, 175, 198
Showing the list of calls 130, 181
SQL
 Continuing a query 104, 168
 Data types 203
 Ending a query 102, 131, 166, 200
 Example program 217
 Function overview (ActiveX) 101
 Function overview (C++ API) 165
 Logging on 103, 132, 168, 200
 Querying telephone book records 103, 121, 167, 199
 Receiving data 130, 199
 Starting a query 102, 166
Structures 202

- T -

Table of contents 1
Telephony functions
 (ActiveX) 89
 (C++ CTI API) 153
Test client 311
Tools 310

- U -

Unicode 124, 193
User-defined messages 113, 121, 177, 191, 284, 300
Using clients without installing 305

- V -

Version information 112, 177

Endnotes 2... (after index)

Back Cover