PDQTapi Help Contents

Technical Support Distribution Copyright and Trademarks

Welcome and thank you for choosing PDQTapi, the Microsoft Windows telephony control from the Crescent Division of Progress Software. This topic provides access to setup information and introduces the important programming concepts and techniques that you must know to use the PDQTapi control in a Visual Basic 4.0 (VB4) application. It also provides access to reference information for the PDQTapi control. To learn more about PDQTapi, select one of the following topics:

- What is Telephony and TAPI?
- What is PDQTapi?
- Using PDQTapi
- PDQTapi Control Reference

The information in this help file is designed for VB4 programmers who have a basic understanding of telecommunications. For more information about TAPI concepts and design, see the *Microsoft Win32 Telephony (TAPI) Programmer's Reference*.

What is Telephony and TAPI?

Related Topics

Telephony is a technology that allows computers to access the communications features and services of a telephone network. For example, a modem, the telephone line, specialized communications software, etc., all of these technologies combine to allow a computer to access a telephone network. The *Microsoft Windows Telephony Application Programming Interface (TAPI)* is a library and supporting services that allows programmers to build 32-bit applications that use the communications features and services available on a telephone network. TAPI is currently part of the Microsoft Windows 95 operating system and will soon be part of the Microsoft Windows NT 4.0 (Cairo) operating system.

Basically, TAPI allows you to build applications that can place, receive, and manage calls from your computer. It supports both voice and data calls. Your application can place calls using a variety of hardware devices on telephone networks that offer complex services, like voice mail, e-mail, conferencing, and call waiting. All these services use different technologies to manage calls and transmit voice and data; however, TAPI hides these servicespecific details from your applications.

The TAPI architecture allows you to create 32-bit applications that can use any available service on the telephone network without including service-specific code in your application. The TAPI library, called TAPI32.DLL, provides a set of functions that allow you to place, receive, and manage calls that access services using generic representations of devices and locations.

- A *location* is a set of values that identifies and provides information about the calling application on a telephone network. The location includes a name, phone number, and other information about the caller. The topic Locations provides more information about TAPI locations.
- A *device* is any communications device that facilitates calls on a telephone network, such as a modem.

Each device has an associated TAPI service provider. A *service provider* describes a device, the service it provides, and it allows you to configure and interact with the device. The term *service* represents the functionality provided by a service provider and its associated device. The TAPI service support application, called TAPIEXE.EXE, maintains information about the available service providers defined on the current system and facilitates communication between the TAPI32.DLL and the service providers. The topic TAPI Service Providers and Devices more information about TAPI services, service providers, and devices.

For more information about TAPI, see the *Microsoft Win32 Telephony (TAPI) Programmer's Reference*.

What is Telephony and TAPI?
    TAPI Service Providers and Devices
    Locations
What is PDQTapi?
Using PDQTapi
PDQTapi Control Reference

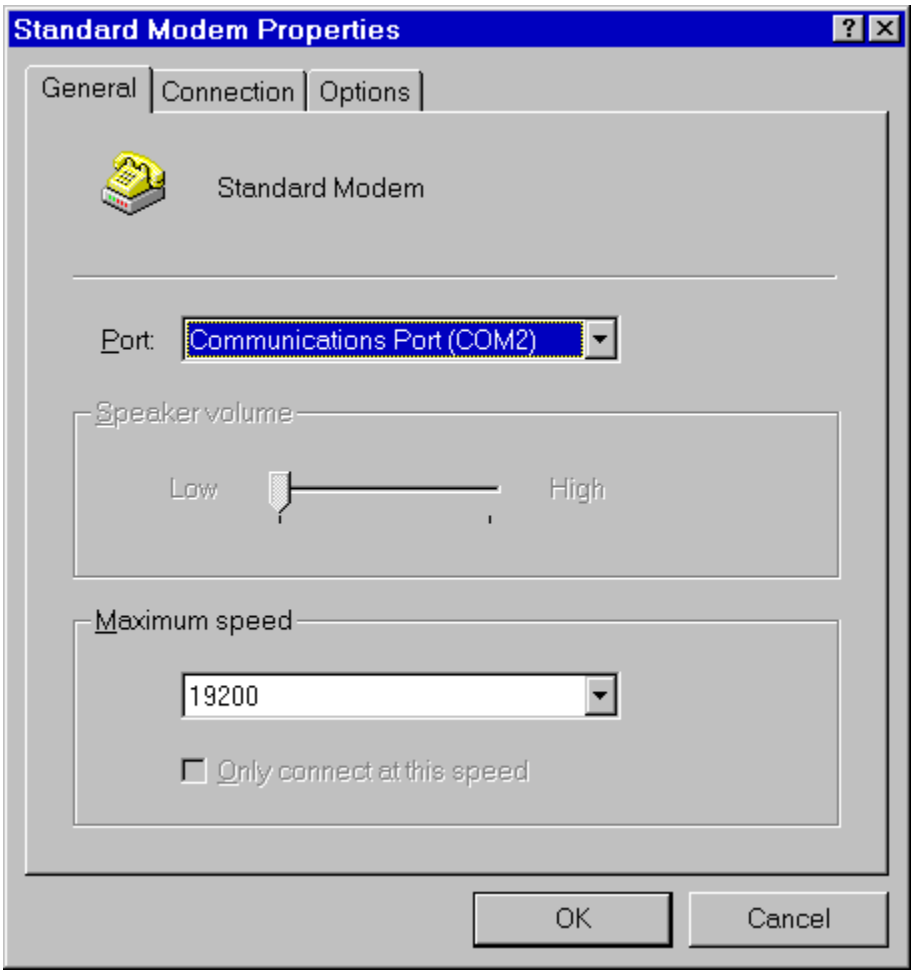
TAPI Service Providers and Devices

Related Topics

A TAPI service provider is actually a dynamic link library (DLL) that executes the device-specific functions required to complete a communications task using a hardware device, such as a telephone, a modem, a fax board, or an ISDN card. To be "device-independent", an application must access these functions through the TAPI32.DLL library and not through the service provider directly. To receive requests from the TAPI dynamic link library, the service provider must be implemented with the Telephony Service Provider Interface (TSPI).

A user can install and configure any number of service providers on a computer as long as the service providers do not access the same hardware device. Some service providers can access multiple devices. The user relates the device and the service provider during installation. The service provider can also provide common dialog boxes that allow you to define configuration information for the device(s) associated with the service provider.

For example, the following TAPI common dialog boxes allows you to define and configure a modem.

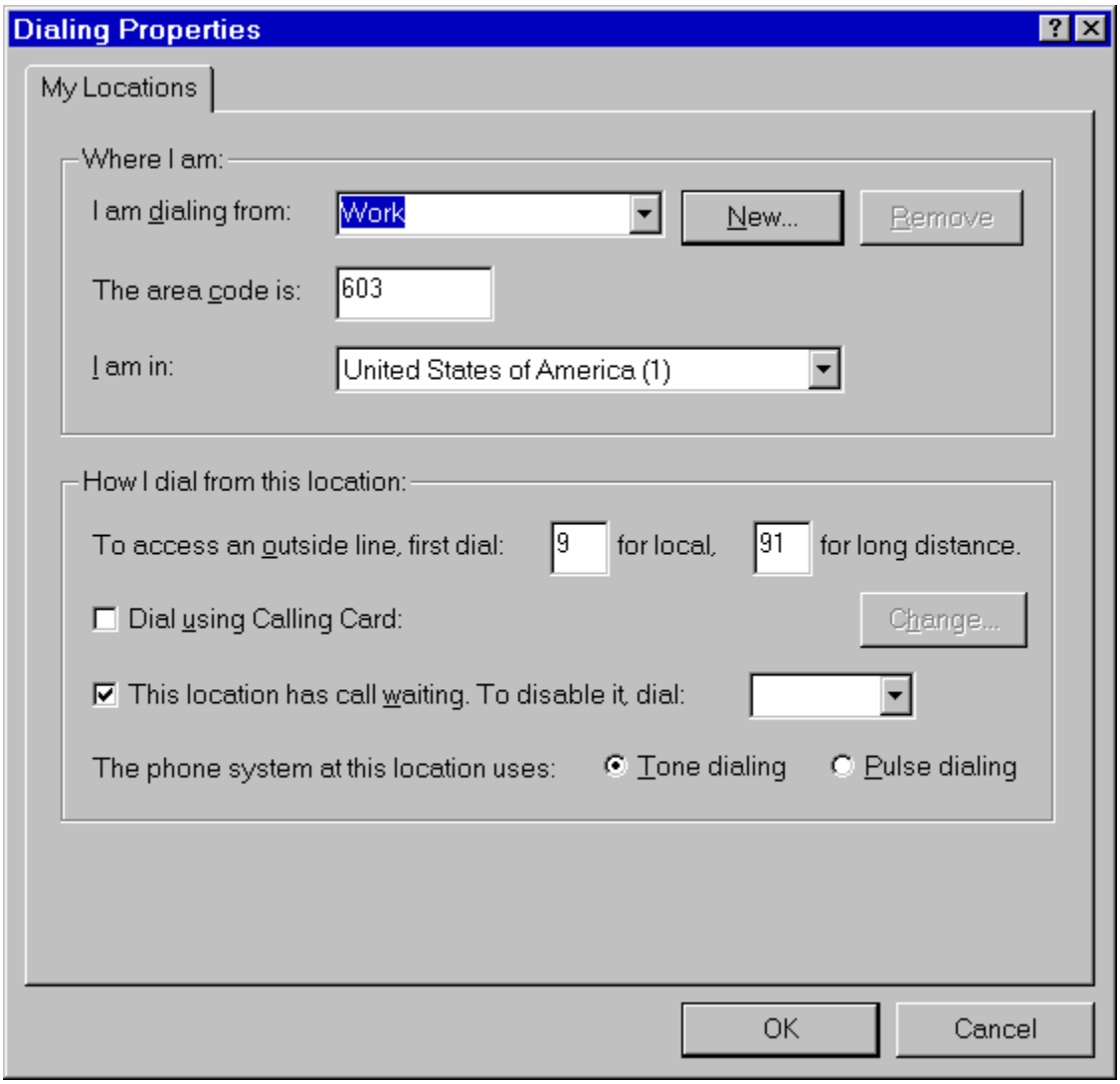


There can be only one active TAPI service provider on a computer at a time. The TAPI32.DLL provides functions that determine what service providers are available on a system and allow you to select one for use in an application. In this way, any number of applications can request services from the same service provider; the TAPI dynamic link library manages all access to the service provider.

Locations

Related Topics

A *location* is a set of values that identifies and provides information about the calling application on a telephone network. TAPI provides a common dialog boxes that allows you to define one or more locations.



Multiple location definitions provide the ability access the network through TAPI regardless of the location of your computer (i.e. at home, at work, etc.). There can be only one active TAPI location on a computer at a time and all TAPI applications on a machine share that location definition.

What is PDQTapi?

Related Topics

The Crescent PDQTapi Telephony control (PDQTapi) is a 32-bit, OLE-enabled control that provides a simple, easy-to-use interface to TAPI from a VB4 application. The properties, methods, and events of the PDQTapi control allow a VB4 application to do the following on a telephone network:

- Set and configure devices and locations
- Place outgoing calls
- Receive incoming data calls
- Monitor the status of calls and handle call errors
- Terminate calls

The control file for the PDQTapi control is called PDQTAPI.OCX.

Voice vs. Data Calls

The PDQTapi control provides the ability to place both data and voice calls. You can also receive data calls with this control, but you cannot receive voice calls.

NOTE Data calls with the PDQTapi control require the assistance of a PDQComm control. PDQComm is an OLE-enabled control that allows you to embed serial communications functionality in a VB4 application. The PDQComm control provides the ability to control the transmission line and creates and maintains buffers for sending and receiving data.

The topic Using PDQTapi and PDQComm for Data Calls provides information about placing, receiving, and managing data calls with the PDQTapi control. For information about the properties, methods, and events of the PDQTapi control, see the PDQTapi Control Reference. For more information about the PDQComm control, see the *PDQComm User's Guide*.

What is Telephony and TAPI?
What is PDQTapi?
    Hardware and Software Requirements
    Installation
    File Distribution
    PDQTapi Demonstrations
    Technical Support
Using PDQTapi
PDQTapi Control Reference

Hardware and Software Requirements

Related Topics

PDQTapi has the following hardware and software requirements:

- Any IBM-compatible machine with an 80386 processor or higher
- A 3 1/2 inch floppy drive
- 2 MB of free space on your hard disk
- A modem.
- Microsoft Windows 95

NOTE The Microsoft Windows TAPI library (TAPI32.DLL) must be installed and registered properly on your system before you can use the PDQTapi control. Currently, the Microsoft Windows TAPI library is available only for the Microsoft Windows 95 operating system.

- Microsoft Visual Basic 4.0 (32-bit) for Windows and the hardware and software that it requires to function
- The PDQCOM32.OCX control file for data calls.

Installation

Related Topics

The installation program installs the PDQTAPI.OCX control file into the \SYSTEM subdirectory of the Microsoft Windows 95 operating system. It also registers and licenses the control in the Windows 95 registry. All demonstration projects and the PDQTAPI.HLP and PDQTAPI.CNT help files for the PDQTapi reside in the \PDQTapi subdirectory of the product installation directory.

File Distribution

Related Topics

You may distribute any application you create using PDQTapi routines, as long as you distribute the application only as an executable. You may also distribute the PDQTAPI.OCX control file with your compiled application, but not in such a way that other people can reuse it to build other applications.

See the "End-user Product License Agreement" at the beginning of the *PDQTapi Users Guide* for more information about the license to use this product.

PDQTapi Demonstrations

Related Topics

The PDQTapi control has two sample projects that demonstrate the power and simplicity of the PDQTapi control.

Sample Project	Description
DEALER.VBP	This project allows you to browse a list of Crescent product distributors and select a distributor to dial for a voice call. It uses the PDQTapi control to place and manage the call. This sample also demonstrates several different techniques that allow users to define a device and location for a call.
TAPICOMM.VBP	This project allows you to specify a number, place a call, and perform a file transfer between a remote site and your local machine. This sample demonstrates the use of the PDQTapi control with the PDQComm control to place and manage a data call.

These sample projects reside in the \PDQTapi directory. Explore these projects to learn about the PDQTapi control and the common programming techniques used in a VB4 telephony application.

Technical Support

Related Topics

The Crescent technical support staff is ready to help you with problems that you encounter when installing or using PDQTapi. It does not matter what component of PDQTapi you are having a problem with; the Crescent technical support staff will do its best to help you succeed with PDQTapi.

If you need technical support, contact Crescent using any of the following methods:

By Telephone	Contact Crescents North American technical support staff at (**617) 280-3000** -- Monday through Friday from 9:00 a.m. to 5:00 p.m. EST.
By FAX	Contact Crescent by FAX at (**617) 280-4025**.
Via BBS	Contact Crescent through our 24-hour bulletin board service at (**617) 280-4221**.
Via CompuServe	Contact Crescent through CompuServe address **70662,2605** Crescent also maintains a section in the MS Windows Components A+ Forum on CompuServe. To reach the Crescent section, type the following at the CompuServe prompt: **GO CRESCENT**
By Electronic Mail	Contact Crescent using the Internet address **crescent-support@progress.com**
Via the WWW	View the Crescent Web page at **http://www.progress.com/crescent**
By Mail	Address your correspondence to: **Technical Support** **Crescent Division, Progress Software Corporation** **14 Oak Park** **Bedford, Massachusetts 01730**

Please have your product name, version number, serial number, and system configuration information available so that the Crescent technical support staff can process your support requests as efficiently as possible.

Using PDQTapi

Related Topics

This topics describes how to use the PDQTapi control to initiate calls and perform basic call-management tasks in a VB4 application. It provides information on the following subjects:

- Selecting Locations and Devices
- Placing, Receiving, and Terminating Calls
- Monitoring Call Status and Error Handling

For more information about the PDQTapi methods, properties, and events, see the PDQTapi Control Reference.

What is Telephony and TAPI?
What is PDQTapi?
Using PDQTapi
    Selecting Locations and Devices
    Placing, Receiving, and Terminating Calls
    Monitoring Call Status and Error Handling
PDQTapi Control Reference

Selecting Locations and Devices

Related Topics

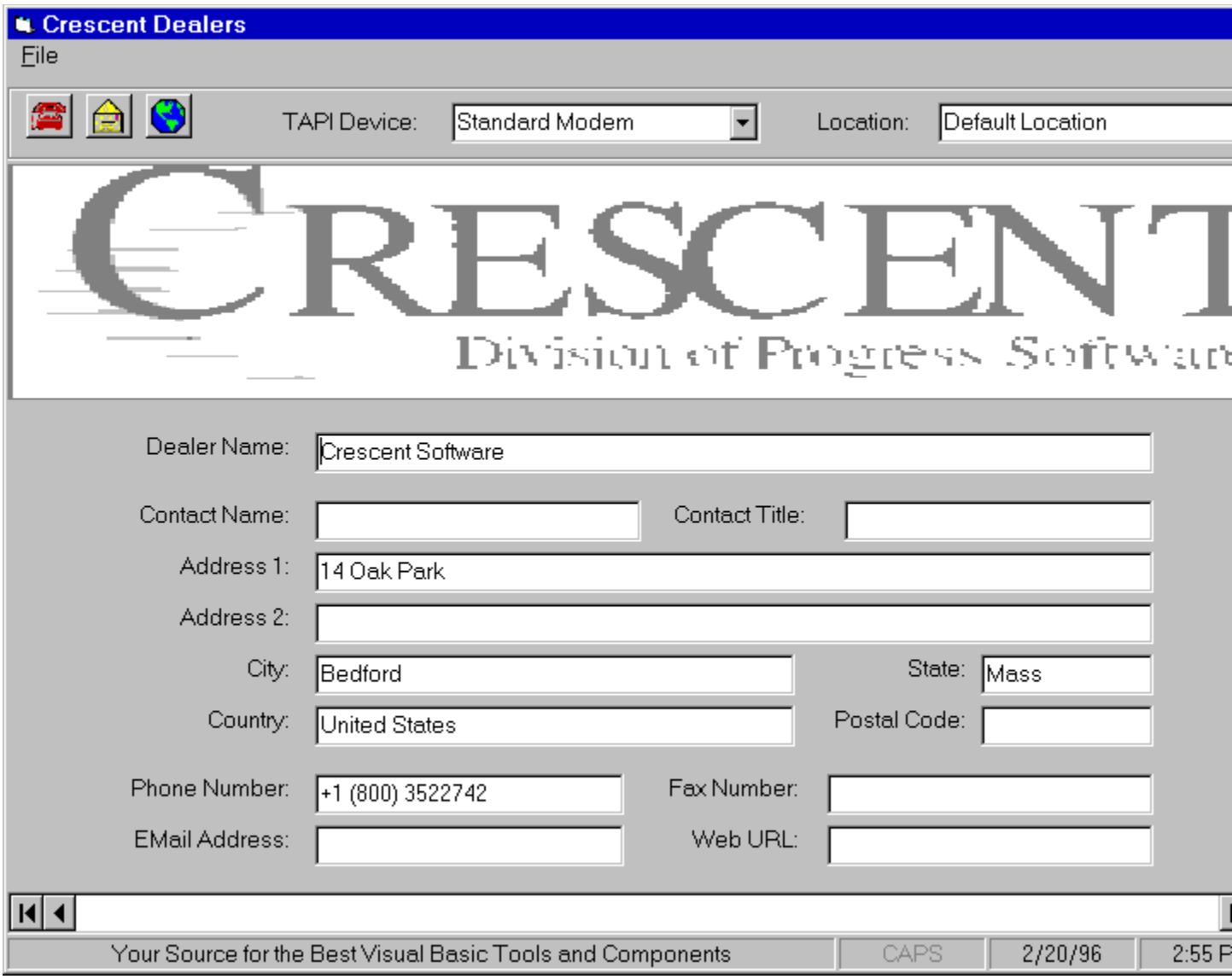
Before you can place a call using the PDQTapi control, there must be a current device and location defined for your system. For a basic description of a TAPI device or location, see the topic What is Telephony and TAPI?.

The PDQTapi has the following methods, properties, and events that allow you to set, read, and manage the device and location information for a call.

Method/Property/Event	Description
AreaCode property	Contains the area code of the currently selected location.
CountryCode property	Contains the country code of the currently selected location.
CurrentDevice property	Specifies the current TAPI device.
Devices property	Identifies the number of TAPI devices installed on the current system.
EnumTapiDevices method	Populates a list or combo box (or any control with an AddItem method) with the list of currently installed TAPI devices.
EnumTapiLocations method	Populates a list or combo box (or any control with an AddItem method) with the list of currently configured TAPI locations.
Location property	Identifies the currently active location.
LocationChange event	Occurs whenever the current location is changed by the user or another application.
SetCallSettings method	Displays the TAPI Location common dialog boxes to allow a user to change, modify, or add a new or current location.
SetModemSettings method	Displays the TAPI Device common dialog boxes to allow a user to change, modify, or add a new or current device.

NOTE There must be at least one modem and location already defined on your system before you can use these properties and methods to access location and device information for a call. To setup a device and location on your system, use the Modems utility located in the Control Panel application in Microsoft Windows 95.

The DEALER.VBP sample project demonstrates the use of most of these PDQTapi methods, properties, and events. The following picture shows the DEALER.FRM module of DEALER.VBP.


Crescent Dealers
File
TAPI Device:
Standard Modem
Location:
Default Location
CRESCENT
Division of Progress Software
Dealer Name:
Crescent Software
Contact Name:
Contact Title:
Address 1:
14 Oak Park
Address 2:
City:
Bedford
State:
Mass
Country:
United States
Postal Code:
Phone Number:
+1 (800) 3522742
Fax Number:
EMail Address:
Web URL:
Your Source for the Best Visual Basic Tools and Components
CAPS
2/20/96
2:55 P

What is Telephony and TAPI?
What is PDQTapi?
Using PDQTapi
    Selecting Locations and Devices
        Displaying and Setting Locations and Devices for Calls
        Using the TAPI Common Dialogs to Select Locations and Devices
        Understanding the LocationChange Event
    Placing, Receiving, and Terminating Calls
    Monitoring Call Status and Error Handling
PDQTapi Control Reference

▪ Displaying and Setting Locations and Devices for Calls

Related Topics

The TAPI Device (cmbTapi(0)) and Location (cmbTapi(1)) combo boxes of the DEALER.VBP project demonstrate the use of the EnumTapiDevices and EnumTapiLocations methods. The Form_Load event procedure of the DEALER.FRM module demonstrates how to use these methods to populate the combo boxes with the devices and locations currently defined on the system.

DEALER.FRM:

```
Private Sub Form_Load()
   dtaEnQuiry1.DatabaseName = App.Path & "\Dealers.mdb"
   PDQTapi1.CurrentDevice = 0
   PDQTapi1.EnumTapiDevices cmbTapi(0)
   cmbTapi(0).ListIndex = PDQTapi1.CurrentDevice
   cmbTapi(1).ListIndex = PDQTapi1.EnumTapiLocations(cmbTapi(1))
End Sub
```

This event procedure also sets the CurrentDevice property and then uses this property to set the ListIndex property of the cmbTapi(0) combo box.

When a user selects a new device or location from the TAPI Device (cmbTapi(0)) or Location (cmbTapi(1)) combo box, the Click event procedure for the cmbTapi control array sets the Location and CurrentDevice properties for the PDQTapi1 control.

DEALER.FRM:

```
Private Sub cmbTapi_Click(Index As Integer)
   If Index Then
      PDQTapi1.Location = cmbTapi(1).Text
   Else
      PDQTapi1.CurrentDevice = cmbTapi(0).ListIndex
   End If
End Sub
```

▪ Using the TAPI Common Dialogs to Select Locations and Devices

Related Topics

The DEALER.VBP project also allows a user to select and define TAPI devices and locations using the TAPI common dialog boxes. The Device Settings and Location Settings commands on the File menu of the DEALER.VBP project use the SetCallSettings and SetModemSettings methods to display TAPI common dialog boxes that allow users to select or define new TAPI locations or devices. The Click event procedure for the m_File menu of the DEALER.FRM module demonstrates how to call the SetCallSettings and SetModemSettings methods.

DEALER.FRM:

```
Private Sub m_File_Click(Index As Integer)

Select Case Index
   Case 0
      Dim strPhoneNumber As String
      strPhoneNumber = txtEnQuiry10.Text
      DialStatus.Dial txtEnQuiry1.Text
      gFunction% = TAPI_PLACECALL
      PDQTapi1.PlaceCall "", strPhoneNumber
   Case 1
      MsgBox "EMail support is currently not implemented", 48,_
         "Crescent Dealer Network"
   Case 2
      MsgBox "Web support is currently not implemented", 48,_
         "Crescent Dealer Network"
   Case 4
      PDQTapi1.SetModemSettings
   Case 5
      Dim rtnPNum As String
      rtnPNUM = PDQTapi1.SetCallSettings ""
   Case 7
      Unload Me
      End
End Select

End Sub
```

▪ Understanding the LocationChange Event

Related Topics

It is important to note that the current location setting is shared by all other TAPI applications on the current system. When another application changes the location setting, your application might need to know about the change. The LocationChange event of the PDQTapi control occurs whenever another application changes the current location setting. For example, the LocationChange event procedure of the PDQTapi1 control located in the DEALER.FRM module of the DEALER.VBP project resets the contents and selection of the Locations (cmbTapi(1)) combo box whenever the event occurs.

DEALER.FRM:

```
Private Sub PDQTapi1_LocationChange()
   cmbTapi(1).Clear
   cmbTapi(1).ListIndex = PDQTapi1.EnumTapiLocations(cmbTapi(1))
End Sub
```

For more information about the PDQTapi methods, properties, and events, see the PDQTapi Control Reference.

▪ Placing, Receiving, and Terminating Calls

Related Topics

The PDQTapi control has three methods that perform the most basic TAPI call management functions.

Methods	Descriptions
PlaceCall	Directs TAPI to place a call using the current location and device, and a specified number.
WaitForCall	Directs TAPI to open a line and monitor it for incoming calls.
HangUp	Disconnects an active call.

To place a call with the PDQTapi control, there must be an active TAPI device and location defined on the current system. You can place both voice and data calls using the PDQTapi control. If you are placing a data call, you must specify a PDQComm control as a parameter to PlaceCall method.

To receive a call with the PDQTapi control, there must be an active device defined on the current system. The PDQTapi control can receive incoming data calls, but not voice calls. The WaitForCall method toggles a "call-monitoring" mode for the active TAPI device on the current system. To activate call-monitoring, execute the WaitForCall method with a PDQComm control as a parameter to the method. To deactivate call-monitoring, execute the WaitForCall method with a null string in place of a PDQComm control parameter.

While the HangUp method is simple to understand, the topic Monitoring Call Status and Error Handling provides information about when to hang-up a TAPI call. For more information about handling data calls with PDQTapi, see the topic Using PDQTapi and PDQComm for Data Calls.

What is Telephony and TAPI?
What is PDQTapi?
Using PDQTapi
    Selecting Locations and Devices
    Placing, Receiving, and Terminating Calls
        Understanding Canonical Addresses for Calls
        Using PDQTapi and PDQComm for Data Calls
    Monitoring Call Status and Error Handling
PDQTapi Control Reference

Understanding Canonical Addresses for Calls

Related Topics

The *address* of a call is the phone number used to place a call. The *canonical address format* is universal storage format for phone numbers that accomodates both domestic and international calls regardless of the location of the caller. A canonical address is an ASCII string with the following structure:

```
+CountryCode [(AreaCode)] SubscriberNumber
```

The plus sign (+) is a required indicator that the phone number is in canonical address format. The following table describe the parameters.

Parameter	Description
CountryCode	A string containing one or more digit characters (0-9). This string represents the country in which the address is located.
AreaCode	An optional string containing one or more digit characters (0-9). This string identifies a region in the country in which the address is located. If you specify an area code, you must enclose it in parentheses.
SubscriberNumber	A string containing one or more digit characters (0-9). It can also contain dahses (-), spaces, and periods (.). This string identifies the number of a phone or device on the telephone network.

For example, the following is the canonical address of the Crescent bulletin board:

```
+1 (617) 280-4221
```

TAPI operations that involve a phone number require the number in canonical address format. The PDQTapi method TranslateNumber allows you to generate the canonical address format for a specified telephone number. The PlaceCall and SetCallSettings methods can also take phone numbers that are not in canonical format and then translate and use them.

Using PDQTapi and PDQComm for Data Calls

Related Topics

To place or receive a data call using the PDQTapi control, you must use a PDQComm control. It is important to understand the roles of the PDQTapi and PDQComm controls in this tandem relationship. Use the PDQTapi control to manage the call and use the PDQComm control to manage the line configuration and data transmission.

When you execute a PlaceCall or WaitForCall method with a specified PDQComm control, PDQTapi sets the properties of the PDQComm control to the appropriate settings to handle the call. You then use the properties, methods, and events of the PDQComm control to configure the communications line, initiate data transfers, and configure and manage data buffers. Use the PDQTapi HangUp method to terminate a call and all PDQComm processing associated with the call.

NOTE When using the PDQTapi and PDQComm controls together, do not open the communications port with the PDQComm control.

The TAPICOMM.VBP sample project demonstrates the use of the PDQTapi and PDQComm controls to manage a data call. See the topic Placing, Receiving, and Terminating Calls for more information about PDQTapi and call management. For more information about the PDQComm control and data transmission, see the *PDQComm User's Guide*.

Monitoring Call Status and Error Handling

Related Topics

The PDQTapi has the following events that allow you to monitor calls and handle errors that occur during call processing.

Event	Description
OnTapi	Occurs when an operation initiated by a PlaceCall and HangUp method completes. See the topic Using the OnTapi Event for more information.
CallState	Occurs when the state changes during the progress of a call. See the topic Using the CallStateEvent for more information.

The topic Understanding OnTapi and CallState Event Relationships provides information about how to use these two event together.

What is Telephony and TAPI?
What is PDQTapi?
Using PDQTapi
    Selecting Locations and Devices
    Placing, Receiving, and Terminating Calls
    Monitoring Call Status and Error Handling
        Using the OnTapi Event
        Using the CallState Event
        Understanding OnTapi and CallState Event Relationships
PDQTapi Control Reference

▪ Using the OnTapi Event

Related Topics

The OnTapi event monitors the success of the asynchronous methods of the PDQTapi control: HangUp and PlaceCall. *Asynchronous methods* start an action and immediately returns control to your program before the completion of the action. For example, when you execute the PlaceCall method to place a call, control returns to your application while TAPI attempts to allocate handles for the line and call. When TAPI successfully allocates the handles in response to the PlaceCall method, an OnTapi event occurs.

The OnTapi event returns two parameters: a request ID and a status value.

The request ID identifies the PlaceCall or HangUp method that caused the event to occur. Both the PlaceCall and HangUp methods return a request ID that you can use to direct processing in an OnTapi event procedure.

The status parameter of the OnTapi event returns a zero (0) to indicate that the request succeeded or a non-zero value to indicate the request failed. A PlaceCall method is successful when TAPI allocates valid line and call handles for the request. A HangUp method is successful when the TAPI deallocates line and call handles for the request. The CallHandle and LineHandle properties contain the handles for the call associated with a PDQTapi control.

The OnTapi event procedure of the PDQTapi1 control located in the DEALER.FRM module of the DEALER.VBP demonstration project, shows how to use this event to track the status of asynchronous method calls.

DEALER.FRM:

```
Private Sub PDQTapi1_OnTapi(ByVal lRequestID As Long, ByVal lStatus As
Long)
.
.
.
If lRequestID = PLACECALL_ID And lStatus <> 0 Then
    DialStatus.SetStatus "Call could not be completed"
    DialStatus!cmdLog.Enabled = True
End If

End Sub
```

A PDQTapi control has an "active" call when the CallHandle and LineHandle properties contain non-zero values, or valid handles. The successful execution of a PlaceCall method yields an active call. The WaitForCall method allocates a line handle when upon receiving an incoming data call and then allocates a call handle.

Using the CallState Event

Related Topics

Once you have an active call, use the CallState event to monitor state changes for the call. The CallStateConstants object of the PDQTapi control establishes constants for different call states. The following table describes the different call states and associated constants:

Constant	Value	Description
PDQ_CALLSTATE_ACCEPTED	4	The call has been accepted by a remote application.
PDQ_CALLSTATE_BUSY	64	The call is receiving a busy tone and that the call cannot be completed.
PDQ_CALLSTATE_CONFERENCED	2048	The call is part of a multi-party conference call.
PDQ_CALLSTATE_CONNECTED	256	The call has established a connection to a remote party. Information can now flow over the call between the calling and remote parties.
PDQ_CALLSTATE_DIALING	16	A phone number is being sent to the switch.
PDQ_CALLSTATE_DIALTONE	8	The call is receiving a dial tone from the switch. This means that the switch is ready to receive a phone number for dialing.
PDQ_CALLSTATE_DISCONNECTED	16384	The remote location has disconnected the call.
PDQ_CALLSTATE_IDLE	1	No call exists.
PDQ_CALLSTATE_OFFERING	2	The call is being offered to the station, signaling the arrival of a new call.
PDQ_CALLSTATE_ONHOLD	1024	The call is on hold by the switch.
PDQ_CALLSTATE_ONHOLDPENDCONF	4096	The call is currently on hold while it is being added to a multi-party conference.
PDQ_CALLSTATE_ONHOLDPENDTRANSFER	8192	The call is currently on hold while it is being transferred.
PDQ_CALLSTATE_PROCEEDING	512	The dialing process has succeeded and the call is now proceeding through the switch or telephone network.
PDQ_CALLSTATE_RINGBACK	32	The remote location has been alerted to the call.
PDQ_CALLSTATE_SPECIALINFO	128	The network has sent special information, usually in response to a failure to connect.
PDQ_CALLSTATE_UNKNOWN	32768	The state of the call is unknown.

The CallState event procedure of the PDQTapi1 control located in the DEALER.FRM module of the

DEALER.VBP demonstration project, shows how to use this event to track the status of a call. Notice the use of the call state constants in the Case statement to direct processing.

DEALER.FRM:

```
Private Sub PDQTapi1_CallState(ByVal lCallState As Long, ByVal lExtraInfo
As Long)
.
.
If gFunction% = TAPI_PLACECALL Then
   Select Case lCallState
      Case PDQ_CALLSTATE_IDLE         'PlaceCall done
         gFunction% = 0
         Debug.Print "PDQ_CALLSTATE_IDLE"
      Case PDQ_CALLSTATE_DIALTONE
         DialStatus.SetStatus "Dialtone available ..."
         Debug.Print "PDQ_CALLSTATE_DIALTONE"
      Case PDQ_CALLSTATE_DIALING
         DialStatus.SetStatus "Dialing ..."
         Debug.Print "PDQ_CALLSTATE_DIALING"
      Case PDQ_CALLSTATE_BUSY
         DialStatus.SetStatus "Busy"
         Debug.Print "PDQ_CALLSTATE_BUSY"
      Case PDQ_CALLSTATE_CONNECTED
         DialStatus.SetStatus "Connected"
         DialStatus.Connected True
         Debug.Print "PDQ_CALLSTATE_CONNECTED"
      Case PDQ_CALLSTATE_DISCONNECTED
         DialStatus.SetStatus "Call complete"
         DialStatus.Connected False
         Debug.Print "PDQ_CALLSTATE_DISCONNECTED"
      Case Else
         Debug.Print "PlaceCall "; lCallState; lExtraInfo
      End Select
ElseIf gFunction% = TAPI_HANGUP Then
   If lCallState = PDQ_CALLSTATE_IDLE Then
      gFunction% = 0
      Debug.Print "PDQ_CALLSTATE_IDLE"
      DialStatus.SetStatus "Call complete"
      DialStatus.Connected False
   End If
End If

End Sub
```

Understanding OnTapi and CallState Event Relationships

Related Topics

It is important to note that the PlaceCall and HangUp methods cause both OnTapi and CallState events to occur. An OnTapi event associated with a PlaceCall method occurs before all related CallState events for that action. The CallState events associated with the PlaceCall method vary depending upon the success of the call and the telephone network configuration.

An OnTapi event associated with a HangUp method occurs after all related CallState events occur for that action. A successful HangUp method causes a single CallState event to occur for the PDQ_CALLSTATE_IDLE state.

It is also important to note that a PDQ_CALLSTATE_DISCONNECTED call state event is not the same as issuing a HangUp method. Unlike the HangUp method, the PDQ_CALLSTATE_DISCONNECTED call state event does not occur an OnTapi event or deallocate the active line and call handles. For more information about TAPI call states, see the *Microsoft Win32 Telephony (TAPI) Programmer's Reference*.

What is Telephony and TAPI?
What is PDQTapi?
Using PDQTapi

PDQTapi Control

Related Topics

The Crescent PDQTapi Telephony control allows a Visual Basic application to easily place, receive and manage calls, and configure devices on a telephone network using the Microsoft Windows Telephony Application Programming Interface (TAPI).

NOTE The PDQTapi control is a 32-bit control. The Microsoft Windows TAPI library (TAPI32.DLL) must be installed and registered properly on your system before you can use the PDQTapi control. Currently, the Microsoft Windows TAPI library is available only for the Microsoft Windows 95 operating system.

File Name

PDQTAPI.OCX

Control Name

PDQTapi

Comments

To place a call with the PDQTapi control, do the following:

1. Define a location. A *location* is a set of values that identifies the calling application on the telephone network. Use the SetCallSettings method to display a standard TAPI dialog box that allows a user to access existing location definitions or define a new one. The Location property allows you to set the location for a call programmatically.

2. Define a device. A *device* is the communications device that facilitates the call on the telephone network, such as a modem. Use the SetModemSettings method to display a standard TAPI dialog box that allows a user to access existing device definitions or define a new one. The CurrentDevice property allows you to set the device for a call programmatically.

3. Place the call using the PlaceCall method.

Use the HangUp method to terminate the call.

Error handling for the PDQTapi control involves monitoring OnTapi and CallState events. The PlaceCall and HangUp methods of the PDQTapi control are the only asynchronous methods for the control. Use the OnTapi event to monitor the success or failure of these commands. Once the PDQTapi control establishes a call, use the CallState event to monitor activity on the line. The CallState event occurs when the status of a call changes and it has a parameter that returns the status of the call. You can use predefined constants to test the status value and execute code in response to the call status change.

The PDQTapi control only places and manages calls. If you intend to manage data transmissions during a call, you must use the PDQComm control with the PDQTapi control. The PDQComm control creates and manages buffers for data transfers. It also allows you to configure the line for data transmission. Both the PlaceCall and WaitForCall methods of the PDQTapi control allow you to designate a PDQComm control to handle data transmission for a call.

Properties

About	Devices	Object
AreaCode	Index	Parent

CallHandle
CountryCode
CurrentDevice
DeviceName
LineHandle
Location
Name
Tag
TapiName
TapiVersion

Methods

EnumTapiDevices
EnumTapiLocations
HangUp
PlaceCall
SetCallSettings
SetDeviceByName
SetModemSettings
TranslateNumber
WaitForCall

Events

CallState
LocationChange
OnTapi

Constants

The CallState event returns a status parameter that is an enumerated integer value representing the state of the call. This value is from a discrete list of integer values. To make your code more readable, the PDQTapi control supports the use of constants for reading the status parameter of the CallState event. For more information about these constants, see the PDQTapi Control Constants topic.

AreaCode Property

The AreaCode property contains the area code of the currently selected location.

Applies To

PDQTapi

Syntax

AreaCode$ = **[***form***.]***PDQTapi1*.AreaCode

Data Type

String

Usage

Read only at runtime.

Comments

The PDQTapi control uses this value when you submit a telephone number without country and area codes.

See Also

CountryCode

CallHandle Property

The CallHandle property contains the handle of the currently allocated call.

Applies To

PDQTapi

Syntax

hCall& = **[***form***.]***PDQTapi1*.CallHandle

Data Type

Long

Usage

Read only at runtime.

Comments

This property contains a zero (0) value when there is no active call and a non-zero value when there is an active call.

See Also

LineHandle

CallState Event

The CallState event occurs when the state changes during the progress of a call.

Applies To

PDQTapi

Syntax

```
Sub PDQTapi1_CallState(State&, Extra&)
```

Parameters

State& - The state of the current call.

Extra& - Aditional information if applicable.

Comments

The values for *State&* are defined in the CallStateConstants object shown in the section PDQTapi Control Constants.

Example

The following example from the DEALER.VBP sample project demonstrates the use of constants and a Case statement in a CallState event procedure to test the state of a call.

```
Sub PDQTapi1_CallState(lCallState, lExtraInfo)
Debug.Print "CallState fired: CallHandle "; Hex$(PDQTapi1.CallHandle); "
CallState "; lCallState; " Call Detail "; lExtraInfo

If gFunction% = TAPI_PLACECALL Then
    Select Case lCallState
        Case PDQ_CALLSTATE_IDLE         'PlaceCall done
            gFunction% = 0
            Debug.Print "PDQ_CALLSTATE_IDLE"
        Case PDQ_CALLSTATE_DIALTONE
            DialStatus.SetStatus "Dialtone available ..."
            Debug.Print "PDQ_CALLSTATE_DIALTONE"
        Case PDQ_CALLSTATE_DIALING
            DialStatus.SetStatus "Dialing ..."
            Debug.Print "PDQ_CALLSTATE_DIALING"
        Case PDQ_CALLSTATE_BUSY
            DialStatus.SetStatus "Busy"
            Debug.Print "PDQ_CALLSTATE_BUSY"
        Case PDQ_CALLSTATE_CONNECTED
            DialStatus.SetStatus "Connected"
            DialStatus.Connected True
            Debug.Print "PDQ_CALLSTATE_CONNECTED"
            gFunction% = 0
        Case PDQ_CALLSTATE_DISCONNECTED
            DialStatus.SetStatus "Call complete"
            DialStatus.Connected False
            'gRequest& = PDQTapi1.HangUp
            'gFunction% = TAPI_HANGUP
```

```
            Debug.Print "PDQ_CALLSTATE_DISCONNECTED"
            gFunction% = 0
        Case Else
            Debug.Print "PlaceCall "; lCallState; lExtraInfo
    End Select
ElseIf gFunction% = TAPI_HANGUP Then
    If lCallState = PDQ_CALLSTATE_IDLE Then
        gFunction% = 0
        Debug.Print "PDQ_CALLSTATE_IDLE"
        DialStatus.SetStatus "Call complete"
        DialStatus.Connected False
    End If
End If
End Sub
```

See Also

WaitForCall

CountryCode Property

The CountryCode property contains the country code of the currently selected location.

Applies To

PDQTapi

Syntax

CountryCode$ = **[***form***.]***PDQTapi1*.CountryCode

Data Type

String

Usage

Read only at runtime.

Comments

The PDQTapi control uses this value when you submit a telephone number without country and area codes.

See Also

AreaCode

CurrentDevice Property

The CurrentDevice property contains a value identifying the currently selected TAPI device.

Applies To

PDQTapi

Syntax

[*form***.]***PDQTapi1*.CurrentDevice = *lDevice&*

Data Type

Long

Usage

Read/Write only at runtime.

Comments

This property allows you to select the TAPI device you want to use for a call. The value can be from 0 to one leas than the total number of TAPI devices (*PDQTapi1*.Devices - 1).

See Also

Devices, DeviceName, EnumTapiDevices, SetDeviceByName, SetModemSettings

DeviceName Property

The DeviceName property returns the string name of the current TAPI device.

Applies To

PDQTapi

Syntax

DeviceName$ = **[***form***.]***PDQTapi1*.DeviceName

Data Type

String

Usage

Read only at runtime.

Comment

While the integer ID associated with a particular TAPI device varies depending upon the installation and deinstallation of other TAPI devices, the name string associated with a TAPI device remains constant.

See Also

CurrentDevice, Devices, EnumTapiDevices, SetDeviceByName, SetModemSettings

Devices Property

The Devices property identifies the number of TAPI devices installed on the current system.

Applies To

PDQTapi

Syntax

lDevices& = **[***form***.]***PDQTapi1*.Devices

Data Type

Long

Usage

Read only at runtime.

See Also

CurrentDevice, DeviceName, EnumTapiDevices, SetDeviceByName, SetModemSettings

EnumTapiDevices Method

The EnumTapiDevices method populates a list or combo box (or any control with an AddItem method) with the list of currently installed TAPI devices.

Applies To

PDQTapi

Syntax

PDQTapi1.EnumTapiDevices *varControl*

Parameters

varControl - Any OLE control with an AddItem method.

Returns

Nothing.

Comments

If you pass EnumTapiDevices a control that does not have an AddItem method, or any other data type, an error occurs. You can use On Error to process the error.

See Also

CurrentDevice, Devices, DeviceName, SetDeviceByName, SetModemSettings

EnumTapiLocations Method

The EnumTapiLocations method populates a list or combo box (or any control with an AddItem method) with the list of currently configured TAPI locations.

Applies To

PDQTapi

Syntax

```
CurrentLocal% = PDQTapi1.EnumTapiDevices(varControl)
```

Parameters

varControl - Any OLE control with an AddItem method.

Returns

The ListIndex of the currently selected location.

Comments

If you pass EnumTapiLocations a control that does not have an AddItem method, or any other data type, an error occurs. You can use On Error to process the error.

Use the return value to set the ListIndex property of the control.

See Also

Locations, SetCallSettings

HangUp Method

The HangUp method disconnects an active call.

Applies To

PDQTapi

Syntax

```
RequestID% = PDQTapi1.HangUp
```

Parameters

None.

Returns

The request ID.

Comments

The HangUp method is an asynchronous method and it returns control to your application before the action is complete. If the return value is a positive number, the OnTapi event occurs with this ID to indicate success or failure.

See Also

PlaceCall, OnTapi, WaitForCall

LineHandle Property

The LineHandle property contains the handle of the currently allocated line.

Applies To

PDQTapi

Syntax

hLine& = **[***form***.]***PDQTapi1*.LineHandle

Data Type

Long

Usage

Read only at runtime.

Comments

This property contains a zero (0) value when there is no active line and a non-zero value when there is an active line. If you call the WaitForCall method to wait for incoming calls, the PDQTapi control allocates a line.

See Also

CallHandle

Location Property

The Location property identifies the currently active location.

Applies To

PDQTapi

Syntax

MyLocation$ = **[***form***.]***PDQTapi1*.Location

Data Type

String

Usage

Read/Write at runtime.

Comments

This property contains the text of the current location. The location is global to all TAPI applications, so if another application changes the location, the PDQTapi notifies you of the location change with a LocationChange event. You can also use this property to set the current location. It requires you to set the property to the exact location string you want.

See Also

EnumTapiLocations, LocationChange, SetCallSettings

LocationChange Event

The LocationChange event occurs whenever the current location is changed by the user or another application.

Applies To

PDQTapi

Syntax

```
Sub PDQTapi1_LocationChange()
```

Comments

This event occurs only when another TAPI control or TAPI-enabled application changes the current location. It allows you to update your user interface or do any other processing necessary when the location changes.

Example

The following example clears a combo box and reinitializes it when a location change occurs.

```
Sub PDQTapi1_LocationChange
   cmbTapiLocation.Clear
   cmbTapiLocation.ListIndex = _
      PDQTapi1.EnumTAPILocations(cmbTapiLocation)
End Sub
```

See Also

EnumTapiLocations, Location, SetCallSettings

OnTapi Event

The OnTapi event occurs when an operation initiated by a PlaceCall or HangUp method completes.

Applies To

PDQTapi

Syntax

```
Sub PDQTapi1_OnTapi(RequestID&, Status&)
```

Parameters

RequestID& - The request ID returned from the asynchronous method call.

Status& - The success or failure of the method call. Zero (0) indicates success, while a non-zero value indicates an error.

Comments

Both the PlaceCall and HangUp methods return a request ID value. The OnTapi event returns a parameter that you can use to map the event to a specific request. It also returns a status parameter that reports the success or failure of the request. A PlaceCall method is successful when TAPI allocates valid line and call handles for the request. A HangUp method is successful when TAPI deallocates line and call handles for the request. The CallHandle and LineHandle properties contain the handles for the call associated with a PDQTapi control.

Once a PlaceCall method succeeds, use the CallState event to monitor the state of the call on the telephone network.

See Also

CallHandle, CallState, HangUp, LineHandle, PlaceCall

PlaceCall Method

The PlaceCall method directs TAPI to place a call.

Applies To

PDQTapi

Syntax

```
RequestID% = PDQTapi1.PlaceCall(CommControl, strNumber$)
```

Parameters

CommControl - If this call is for data transmission, specify the PDQComm control that takes over processing if the call is successful. If this is a voice call, specify a null string.

strNumber$ - The number you are dialing. Ideally, this number will be in canonical (i.e. +1 (617) 280-4221) or dialable format (i.e. 16172804221). If it is not, the method will attempt to format it.

Returns

The request ID.

Comments

To place a call with the PDQTapi control, there must be an active TAPI device and location defined on the current system. You can place both voice and data calls using the PDQTapi control.

Use the TranslateNumber method to translate phone numbers into canonical address format. For more information about canonical address format, see the section "Understanding Canonical Addresses for Calls" in Chapter 1.

If you are placing a data call, you must specify a PDQComm control as a parameter to PlaceCall method.

The PlaceCall method is an asynchronous method and it returns control to your application before the action is complete. If the return value is a positive number, the OnTapi event occurs with this ID to indicate success or failure.

See Also

HangUp, OnTapi, TranslateNumber, WaitForCall

SetCallSettings Method

The SetCallSettings method displays the TAPI Location common dialog box to allow a user to change, modify, or add a new or current location.

Applies To

PDQTapi

Syntax

```
strDisplay$ = PDQTapi1.SetCallSettings(strNumberIn$)
```

Parameters

strNumberIn$ - An optional telephone number that will be displayed in the dialog box.

Returns

The specified phone number in a displayable format.

Comments

You must have a device and location defined on your current system to use the SetCallSettings method to display the TAPI Location dialog box.

See Also

EnumTapiLocations, Locations, SetModemSettings, TranslateNumber

SetDeviceByName Method

The SetDeviceByName method allows you to set the current TAPI device using the name string of the device.

Applies To

PDQTapi

Syntax

```
success = PDQTapi1.SetDeviceByName(DeviceName$)
```

Parameters

DeviceName$ - A string identifier for a registered TAPI device on the current system.

Returns

success - A boolean value representing the success (TRUE) or failure (FALSE) of the operation..

See Also

CurrentDevice, Devices, DeviceName, EnumTapiDevices, SetModemSettings

SetModemSettings Method

The SetModemSettings method displays the TAPI Device common dialog box to allow a user to change, modify, or add a new or current device.

Applies To

PDQTapi

Syntax

PDQTapi1.SetModemSettings

Parameters

None.

Returns

Nothing.

Comments

This method globally configures the device, meaning all changes made here affect other TAPI applications. You must have a device defined on your current system to use the SetModemSettings method to display the TAPI Device dialog box.

See Also

CurrentDevice, Devices, EnumTapiDevices, SetCallSettings

TapiName Property

The TapiName property contains a user friendly name that identifies your application to the TAPI service provider.

Applies To

PDQTapi

Syntax

TapiName$ = **[***form***.]***PDQTapi1*.TapiName

Data Type

String

Usage

Read/Write at design time, read only at runtime.

Comments

This property is optional. It tells the TAPI service provider who you are.This property is useful for debugging service providers.

TapiVersion Property

The TapiVersion property identifies the TAPI API level supported by the currently selected device.

Applies To

PDQTapi

Syntax

lVersion& = **[***form***.]***PDQTapi1*.TapiVersion

Data Type

Long

Usage

Read only at runtime.

Comments

This property has the major version of the TAPI API level stored in the high word and the minor version of the TAPI API level stored in the low word. To access them, do the following:

```
MajorVersion% = CINT(PDQTapi1.TapiVersion \ &H10000)

MinorVersion% = CINT(PDQTapi1.TapiVersion AND &HFFFF&)
```

TranslateNumber Method

The TranslateNumber method creates a canonical, a dialable and a displayable version of a telephone number based on the currently selected TAPI device and location.

Applies To

PDQTapi

Syntax

```
PDQTapi1.TranslateNumber strNumIn$, strCanonical$, strDial$, strDisplay$
```

Parameters

strNumIn$ - The phone number to translate

strCanonical - The canonical version of the number (modified) (i.e. +1 (617) 280-4221)

strDial$ - The dialable version based on location and device (modified) (i.e. 16172804221)

strDisplay$ - The displayable version based on location and device (modified)
(i.e. 1 (617) 280-4221)

Returns

The noted modified parameters.

Comments

If you pass in a number without country or area codes, TranslateNumber assumes it is the currently set location. TranslateNumber attempts to create a canonical address, but may not succeed depending on the format of the number passed.

See Also

PlaceCall

WaitForCall Method

The WaitForCall method directs TAPI to open a line and monitor it for incoming calls.

Applies To

PDQTapi

Syntax

```
Success% = PDQTapi1.WaitForCall(CommControl)
```

Parameters

CommControl - A PDQComm control or a null string. You call this method with a null string to disable call-monitoring.

Returns

An integer value indicating the success or failure of the method. A zero (0) value indicates success. An non-zero value indicates failure.

Comments

To receive a call with the PDQTapi control, there must be an active device defined on the current system. The PDQTapi control can receive incoming data calls, but not voice calls.

The WaitForCall method toggles a "call-monitoring" mode for the active TAPI device on the current system. To activate call-monitoring, execute the WaitForCall method with a PDQComm control as a parameter to the method. When you activate call-monitoring with the WaitForCall method, TAPI assigns a handle for the line and the incoming call. To deactive call-monitoring, execute the WaitForCall method with a null string in place of a PDQComm control parameter.

The CallState event occurs when the current TAPI device receives an incoming call. Use the PDQ_CALLSTATE_CONNECTED constant in a CallState event procedure to test for an incoming call and perform the desired processing.

See Also

CallHandle, HangUp, LineHandle, OnTapi, PlaceCall

PDQTapi Control Constants

The CallState event returns a status parameter that is an enumerated integer value representing the state of the call. This value is from a discrete list of integer values. To make your code more readable, the PDQTapi control supports the use of constants for reading the status parameter of the CallState event. These constants are defined as part of the CallStateConstants object.

CallStateConstants

Constant	Value	Description
PDQ_CALLSTATE_ACCEPTED	4	The call has been accepted by a remote application.
PDQ_CALLSTATE_BUSY	64	The call is receiving a busy tone and the call cannot be completed.
PDQ_CALLSTATE_CONFERENCED	2048	The call is part of a multi-party conference call.
PDQ_CALLSTATE_CONNECTED	256	The call has established a connection to a remote party. Information can now flow over the call between the calling and remote parties.
PDQ_CALLSTATE_DIALING	16	A phone number is being sent to the switch.
PDQ_CALLSTATE_DIALTONE	8	The call is receiving a dial tone from the switch. This means that the switch is ready to receive a phone number for dialing.
PDQ_CALLSTATE_DISCONNECTED	16384	The remote location has disconnected the call.
PDQ_CALLSTATE_IDLE	1	No call exists.
PDQ_CALLSTATE_OFFERING	2	The call is being offered to the station, signaling the arrival of a new call.
PDQ_CALLSTATE_ONHOLD	1024	The call is on hold by the switch.
PDQ_CALLSTATE_ONHOLDPENCONF	4096	The call is currently on hold while it is being added to a multi-party conference.
PDQ_CALLSTATE_ONHOLDPENDTRANSFER	8192	The call is currently on hold while it is being transferred.

PDQ_CALLSTATE_PROCEEDING	512	The dialing process has succeeded and the call is now proceeding through the switch or telephone network.
PDQ_CALLSTATE_RINGBACK	32	The remote location has been alerted to the call.
PDQ_CALLSTATE_SPECIALINFO	128	The network has sent special information, usually in response to a failure to connect.
PDQ_CALLSTATE_UNKNOWN	32768	The state of the call is unknown.