Basic TAPI Implementation

This chapter outlines the TAPI 2.1 functions, events, and messages that the Cisco Unified TAPI Service Provider (TSP) supports. This chapter contains functions in the following sections:

- Overview, on page 1
- TAPI Line Functions, on page 1
- TAPI Line Messages, on page 57
- TAPI Line Device Structures, on page 74
- TAPI Phone Functions, on page 132
- TAPI Phone Messages, on page 148
- TAPI Phone Structures, on page 156
- Wave Functions, on page 163

Overview

TAPI comprises a set of classes that expose the functionality of the Cisco Unified Communications Solutions. TAPI enables developers to create customized IP telephony applications for Unified Communications Manager without specific knowledge of the communication protocols between the Unified Communications Manager and the service provider. For example, a developer could create a TAPI application that communicates with an external voice-messaging system.

TAPI Line Functions

The number of TAPI devices that are configured in the Unified Communications Manager determines the number of available lines. Cisco Media Driver is used to terminate a media stream in the first-party call control models.

Table 1: TAPI Line Functions Supported

TAPI line functions supported
lineAccept, on page 4
lineAddProvider, on page 4
lineAddToConference, on page 5

TAPI line functions supported
lineAnswer, on page 6
lineBlindTransfer, on page 6
lineCallbackFunc, on page 7
lineClose, on page 8
lineCompleteTransfer, on page 8
lineConfigProvider, on page 9
lineDeallocateCall, on page 10
lineDevSpecific, on page 10
lineDevSpecificFeature, on page 12
lineDial, on page 13
lineDrop, on page 14
lineForward, on page 15
lineGenerateDigits, on page 17
lineGenerateTone, on page 18
lineGetAddressCaps, on page 19
lineGetAddressID, on page 20
lineGetAddressStatus, on page 21
lineGetCallInfo, on page 21
lineGetCallStatus, on page 22
lineGetConfRelatedCalls, on page 22
lineGetDevCaps, on page 23
lineGetID, on page 24
lineGetLineDevStatus, on page 25
lineGetMessage, on page 25
lineGetNewCalls, on page 26
lineGetNumRings, on page 27
lineGetProviderList, on page 28
lineGetRequest, on page 29

TAPI line functions supported
lineGetStatusMessages, on page 30
lineGetTranslateCaps, on page 30
lineHandoff, on page 31
lineHold, on page 32
lineInitialize, on page 33
lineInitializeEx, on page 34
lineMakeCall, on page 35
lineMonitorDigits, on page 36
lineMonitorTones, on page 36
lineNegotiateAPIVersion, on page 37
lineNegotiateExtVersion, on page 38
lineOpen, on page 39
linePark, on page 40
linePrepareAddToConference, on page 41
lineRedirect, on page 43
lineRegisterRequestRecipient, on page 43
lineRemoveFromConference, on page 44
lineSetAppPriority, on page 46
lineSetCallPrivilege, on page 47
lineSetNumRings, on page 48
lineSetStatusMessages, on page 49
lineSetTollList, on page 50
lineSetupConference, on page 51
lineSetupTransfer, on page 52
lineShutdown, on page 52
lineTranslateAddress, on page 53
lineTranslateDialog, on page 54
lineUnhold, on page 56

TAPI line functions supported
lineUnpark, on page 56

lineAccept

The lineAccept function accepts the specified offered call.

Function Details

```
LONG lineAccept(HCALL hCall,
LPCSTR lpsUserUserInfo,
DWORD dwSize
);
```

Parameters

hCall

A handle to the call to be accepted. The application must be an owner of the call. Call state of hCall must be offering.

lpsUserUserInfo

A pointer to a string that contains user-user information to be sent to the remote party as part of the call accept. Leave this pointer NULL if you do not want to send user-user information. User-user information is sent only if supported by the underlying network. The protocol discriminator member for the user-user information, if required, should appear as the first byte of the buffer that is pointed to by lpsUserUserInfo and must be accounted for in dwSize.

Note The Cisco Unified TSP does not support user-user information.

dwSize

The size in bytes of the user-user information in lpsUserUserInfo. If lpsUserUserInfo is NULL, no user-user information gets sent to the calling party, and dwSize is ignored.

lineAddProvider

The lineAddProvider function installs a new telephony service provider into the telephony system.

Function Details

```
LONG WINAPI lineAddProvider(  LPCSTR lpszProviderFilename,
  HWND hwndOwner,
  LPDWORD lpdwPermanentProviderID
);
```

Parameters

lpszProviderFilename

A pointer to a null-terminated string that contains the path of the service provider to be added.

hwndOwner

A handle to a window in which dialog boxes that need to be displayed as part of the installation process (for example, by the service provider's TSPI_providerInstall function) would be attached. Can be NULL to indicate that any window created during the function should have no owner window.

lpdwPermanentProviderID

A pointer to a DWORD-sized memory location into which TAPI writes the permanent provider identifier of the newly installed service provider.

Return Values

Returns zero if request succeeds or a negative number if an error occurs. Possible return values are:

- LINEERR_INIFILECORRUPT
- LINEERR_NOMEM
- LINEERR_INVALPARAM
- LINEERR_NOMULTIPLEINSTANCE
- LINEERR_INVALPOINTER
- LINEERR_OPERATIONFAILED

lineAddToConference

This function takes the consult call that is specified by hConsultCall and adds it to the conference call that is specified by hConfCall.

Function Details

```
LONG lineAddToConference(  HCALL hConfCall,
  HCALL hConsultCall
);
```

Parameters

hConfCall

A pointer to the conference call handle. The state of the conference call must be OnHoldPendingConference or OnHold.

hConsultCall

A pointer to the consult call that will be added to the conference call. The application must be the owner of this call, and it cannot be a member of another conference call. The allowed states of the consult call comprise connected, onHold, proceeding, or ringback

lineAnswer

The lineAnswer function answers the specified offering call.

Note CallProcessing requires previous calls on the device to be in connected call state before answering further calls on the same device. If calls are answered without checking for the call state of previous calls on the same device, then Cisco Unified TSP might return a successful answer response but the call will not go to connected state and needs to be answered again.

Function Details

```
LONG lineAnswer(  HCALL hCall,
  LPCSTR lpsUserUserInfo,
  DWORD dwSize
);
```

Parameters

hCall

A handle to the call to be answered. The application must be an owner of this call. The call state of hCall must be offering or accepted.

lpsUserUserInfo

A pointer to a string that contains user-user information to be sent to the remote party at the time the call is answered. You can leave this pointer NULL if no user-user information will be sent.

User-user information only gets sent if supported by the underlying network. The protocol discriminator field for the user-user information, if required, should be the first byte of the buffer that is pointed to by lpsUserUserInfo and must be accounted for in dwSize.

Note The Cisco Unified TSP does not support user-user information.

dwSize

The size in bytes of the user-user information in lpsUserUserInfo. If lpsUserUserInfo is NULL, no user-user information gets sent to the calling party, and dwSize is ignored.

lineBlindTransfer

The lineBlindTransfer function performs a blind or single-step transfer of the specified call to the specified destination address.

Note The lineBlindTransfer function that is implemented until Cisco Unified TSP 3.3 does not comply with the TAPI specification. This function actually gets implemented as a consultation transfer and not a single-step transfer. From Cisco Unified TSP 4.0, the lineBlindTransfer complies with the TAPI specs wherein the transfer is a single-step transfer.

If the application tries to blind transfer a call to an address that requires a FAC, CMC, or both, then the lineBlindTransfer function will return an error. If a FAC is required, the TSP will return the error LINEERR_FACREQUIRED. If a CMC is required, the TSP will return the error LINEERR_CMCREQUIRED. If both a FAC and a CMC are required, the TSP will return the error LINEERR_FACANDCMCREQUIRED. An application that wants to blind transfer a call to an address that requires a FAC, CMC, or both, should use the lineDevSpecific -BlindTransferFACCMC function.

Function Details

```
LONG lineBlindTransfer(  HCALL hCall,
  LPCSTR lpszDestAddress,
  DWORD dwCountryCode
);
```

Parameters

hCall

A handle to the call to be transferred. The application must be an owner of this call. The call state of hCall must be connected.

lpszDestAddress

A pointer to a NULL-terminated string that identifies the location to which the call is to be transferred. The destination address uses the standard dial number format.

dwCountryCode

The country code of the destination. The implementation uses this parameter to select the call progress protocols for the destination address. If a value of 0 is specified, the defined default call-progress protocol is used.

lineCallbackFunc

The lineCallbackFunc function provides a placeholder for the application-supplied function name.

Function Details

```
VOID FAR PASCAL lineCallbackFunc(  DWORD hDevice,
  DWORD dwMsg,
  DWORD dwCallbackInstance,
  DWORD dwParam1,
  DWORD dwParam2,
  DWORD dwParam3
);
```

Parameters

hDevice

A handle to either a line device or a call that is associated with the callback. The context that dwMsg provides determines the nature of this handle (line handle or call handle). Applications must use the DWORD type for this parameter because using the HANDLE type may generate an error.

dwMsg

A line or call device message.

dwCallbackInstance

Callback instance data that is passed back to the application in the callback. TAPI does not interpret DWORD.

dwParam1

A parameter for the message.

dwParam2

A parameter for the message.

dwParam3

A parameter for the message.

Further Details

For information about parameter values that are passed to this function, see TAPI Line Functions, on page 1.

lineClose

The lineClose function closes the specified open line device.

Function Details

```
LONG lineClose(  HLINE hLine
);
```

Parameter

hLine

A handle to the open line device to be closed. After the line has been successfully closed, this handle no longer remains valid.

lineCompleteTransfer

The lineCompleteTransfer function completes the transfer of the specified call to the party that is connected in the consultation call.

Function Details

```
LONG lineCompleteTransfer(  HCALL hCall,
  HCALL hConsultCall,
  LPHCALL lphConfCall,
  DWORD dwTransferMode
);
```

Parameters

hCall

A handle to the call to be transferred. The application must be an owner of this call. The call state of hCall must be onHold, onHoldPendingTransfer.

hConsultCall

A handle to the call that represents a connection with the destination of the transfer. The application must be comprise an owner of this call. The call state of hConsultCall must be connected, ringback, busy, or proceeding.

lphConfCall

A pointer to a memory location where an hCall handle can be returned. If dwTransferMode is LINETRANSFERMODE_CONFERENCE, the newly created conference call is returned in lphConfCall and the application becomes the sole owner of the conference call. Otherwise, TAPI ignores this parameter.

dwTransferMode

Specifies how the initiated transfer request is to be resolved. This parameter uses the following LINETRANSFERMODE_constant:

- LINETRANSFERMODE_TRANSFER—Resolve the initiated transfer by transferring the initial call to the consultation call.
- LINETRANSFERMODE_CONFERENCE—The transfer gets resolved by establishing a three-way conference among the application, the party connected to the initial call, and the party connected to the consultation call. Selecting this option creates a conference call.

lineConfigProvider

The lineConfigProvider function causes a service provider to display its configuration dialog box. This basically provides a straight pass-through to TSPI_providerConfig.

Function Details

```
LONG WINAPI lineConfigProvider(  HWND hwndOwner,
  DWORD dwPermanentProviderID
);
```

Parameters

hwndOwner

A handle to a window to which the configuration dialog box (displayed by TSPI_providerConfig) is attached. This parameter can equal NULL to indicate that any window that is created during the function should have no owner window.

dwPermanentProviderID

The permanent provider identifier of the service provider to be configured.

Return Values

Returns zero if the request succeeds or a negative number if an error occurs. Possible return values follow:

- LINEERR_INIFILECORRUPT
- LINEERR_NOMEM
- LINEERR_INVALPARAM
- LINEERR_OPERATIONFAILED

lineDeallocateCall

The lineDeallocateCall function deallocates the specified call handle.

Function Details

```
LONG lineDeallocateCall(  HCALL hCall
);
```

Parameter

hCall

The call handle to be deallocated. An application with monitoring privileges for a call can always deallocate its handle for that call. An application with owner privilege for a call can deallocate its handle unless it is the sole owner of the call and the call is not in the idle state. The call handle is invalid after it is deallocated.

lineDevSpecific

The lineDevSpecific function enables service providers to provide access to features that other TAPI functions do not offer. The extensions are device-specific and the applications must be able to read the extensions to take advantage of these extensions.

When used with the Cisco Unified TSP, lineDevSpecific can be used to:

- Enable the message waiting lamp for a particular line.
- Handle the audio stream (instead of using the provided Cisco wave driver).
- Turn On or Off the reporting of media streaming messages for a particular line.
- Register a CTI port or route point for dynamic media termination.

- Set the IP address and the UDP port of a call at a CTI port or route point with dynamic media termination.
- Redirect a Call and Reset the OriginalCalledID of the call to the party that is the destination of the redirect.
- Redirect a call and set the OriginalCalledID of the call to any party.
- Join two or more calls into one conference call.
- Redirect a Call to a destination that requires a FAC, CMC, or both.
- Blind Transfer a Call to a destination that requires a FAC, CMC, or both.
- Open a CTI port in third party mode.
- Set the SRTP algorithm IDs that a CTI port supports.
- Acquire any CTI-controllable device in the Cisco Unified Communications Manager system, which needs to be opened in super provider mode.
- Deacquire any CTI-controllable device in the Cisco Unified Communications Manager system.
- Trigger the actual line open from the TSP side. This is used for the delayed open mechanism.
- Initiate TalkBack on the Intercom Whisper call of the Intercom line
- Query SpeedDial and Label setting of a Intercom line.
- Set SpeedDial and Label setting of a Intercom line.
- Start monitoring a call
- Start recording of a call
- Stop recording of a call
- Direct call with feature priority (see Secure Conference for more information.
- Transfer without media
- Direct Transfer
- Message Summary
- Register call pickup group for notification
- Unregister call pickup group for notification
- Call pickup request
- Start send media to BIB
- Stop send media to BIB
- Agent zip tone
- Enable feature
- Add remote destination
- Remove remote destination
- Update remote destination
- Hold enhancement

Note In Cisco Unified TSP Releases 4.0 and later, the TSP no longer supports the ability to perform a SwapHold/SetupTransfer on two calls on a line in the CONNECTED and the ONHOLD call states. Therefore, these calls can be transferred by using lineCompleteTransfer. Cisco Unified TSP Releases 4.0 and later enable to transfer these calls using the lineCompleteTransfer function without having to perform the SwapHold/SetupTransfer beforehand.

Function Details

```
LONG lineDevSpecific(  HLINE hLine,
  DWORD dwAddressID,
  HCALL hCall,
  LPVOID lpParams,
  DWORD dwSize
);
```

Parameters

hLine

A handle to a line device. This parameter is required.

dwAddressID

An address identifier on the given line device.

hCall

A handle to a call. Although this parameter is optional, if it is specified, the call that it represents must belong to the hLine line device. The call state of hCall is device specific.

lpParams

A pointer to a memory area that is used to hold a parameter block. The format of this parameter block specifies device specific, and TAPI passes its contents to or from the service provider.

dwSize

The size in bytes of the parameter block area.

lineDevSpecificFeature

The lineDevSpecificFeature function enables service providers to provide access to features that other TAPI functions do not offer. The extensions are device-specific and the applications must be able to read the extensions to take advantage of these extensions. When used with the Cisco TSP, lineDevSpecificFeature can be used to enable/disable Do-Not-Disturb feature on a device.

Function Details

```
LONG lineDevSpecificFeature(HLINE hLine,
DWORD dwFeature,
LPVOID lpParams,
DWORD dwSize
);
```

Parameters

hLine

A handle to a line device. This parameter is required.

dwFeature

Feature to invoke on the line device. This parameter uses the PHONEBUTTONFUNCTION_TAPI constants. When used with the Cisco TSP, the only value that is considered valid is PHONEBUTTONFUNCTION_DONOTDISTURB (0x0000001A).

lpParams

A pointer to a memory area used to hold a parameter block. The format of this parameter block is device-specific and TAPI passes its contents to or from the service provider.

dwSize

The size in bytes of the parameter block area.

Return Values

Returns a positive request identifier if the function is completed asynchronously or a negative number if an error occurs. The dwParam2 parameter of the corresponding LINE_REPLY message is zero if the function succeeds or it is a negative number if an error occurs.

Possible return values follow:

- LINEERR_INVALFEATURE
- LINEERR_OPERATIONUNAVAIL
- LINEERR_INVALLINEHANDLE
- LINEERR_OPERATIONFAILED
- LINEERR_INVALPOINTER
- LINEERR_RESOURCEUNAVAIL
- LINEERR_NOMEM
- LINEERR_UNINITIALIZED.

Error Codes

The following new error can be returned by Cisco TSP for Do-Not-Disturb feature:

LINERR_ALREADY_IN_REQUESTED_STATE 0xC0000009

lineDial

The lineDial function dials the specified number on the specified call.

The application can use this function to enter a FAC or CMC. The FAC or CMC can be entered one digit at a time or multiple digits at a time. The application may also enter both the FAC and CMC if required in one lineDial() request as long as the FAC and CMC are separated by a “#” character. If sending both a FAC and

CMC in one lineDial() request, Cisco recommends that you terminate the lpszDestAddress with a "#" character to avoid waiting for the T.302 interdigit time-out.

You cannot use this function to enter a dial string along with a FAC and/or a CMC. You must enter the FAC and/or CMC in a separate lineDial request.

Function Details

```
LONG lineDial(  HCALL hCall,
  LPCSTR lpszDestAddress,
  DWORD dwCountryCode
);
```

Parameters

hCall

A handle to the call on which a number is to be dialed. Ensure the application is an owner of the call. The call state of hCall can be any state except idle and disconnected.

lpszDestAddress

The destination to be dialed by using the standard dial number format.

dwCountryCode

The country code of the destination. The implementation uses this code to select the call progress protocols for the destination address. If a value of 0 is specified, the default call progress protocol is used.

lineDrop

The lineDrop function drops or disconnects the specified call. The application can specify user-user information to be transmitted as part of the call disconnect.

Function Details

```
LONG lineDrop(  HCALL hCall,
  LPCSTR lpsUserUserInfo,
  DWORD dwSize
);
```

Parameters

hCall

A handle to the call to be dropped. Ensure the application is an owner of the call. The call state of hCall can be any state except an Idle state.

lpsUserUserInfo

A pointer to a string that contains user-user information to be sent to the remote party as part of the call disconnect. You can leave this pointer NULL if no user-user information is to be sent. User-user information is sent only if it is supported by the underlying network. The protocol discriminator field for the user-user information, if required, should appear as the first byte of the buffer that is pointed to by lpsUserUserInfo and must be accounted for in dwSize.

Note The Cisco Unified TSP does not support user-user information.

dwSize

The size in bytes of the user-user information in lpsUserUserInfo. If lpsUserUserInfo is NULL, no user-user information gets sent to the calling party, and dwSize is ignored.

lineForward

The lineForward function forwards calls that are destined for the specified address on the specified line, according to the specified forwarding instructions. When an originating address (dwAddressID) is forwarded, the switch deflects the specified incoming calls for that address to the other number. This function provides a combination of forward all feature. This API allows calls to be forwarded unconditionally to a forwarded destination. This function can also cancel forwarding that is currently in effect.

To indicate that the forward is set/reset, upon completion of lineForward, TAPI fires LINEADDRESSSTATE events that indicate the change in the line forward status.

Change forward destination with a call to lineForward without canceling the current forwarding set on that line.

Note lineForward implementation of Cisco Unified TSP allows user to set up only one type for forward as dwForwardMode = UNCOND. The lpLineForwardList data structure accepts LINEFORWARD entry with dwForwardMode = UNCOND.

Function Details

```
LONG lineForward(  HLINE hLine,
  DWORD bAllAddresses,
  DWORD dwAddressID,
  LPLINEFORWARDLIST const lpForwardList,
  DWORD dwNumRingsNoAnswer,
  LPHCALL lphConsultCall,
  LPLINECALLPARAMS const lpCallParams
);
```

Parameters

hLine

A handle to the line device.

bAllAddresses

Specifies whether all originating addresses on the line or just the one that is specified gets forwarded. If TRUE, all addresses on the line get forwarded, and dwAddressID is ignored; if FALSE, only the address that is specified as dwAddressID gets forwarded.

dwAddressID

The address of the specified line whose incoming calls are to be forwarded. This parameter gets ignored if bAllAddresses is TRUE.

Note If bAllAddresses is FALSE, dwAddressID must equal 0.

lpForwardList

A pointer to a variably sized data structure that describes the specific forwarding instructions of type LINEFORWARDLIST.

Note To cancel the forwarding that currently is in effect, ensure lpForwardList Parameter is set to NULL.

dwNumRingsNoAnswer

The number of rings before a call is considered a no answer. If dwNumRingsNoAnswer is out of range, the actual value gets set to the nearest value in the allowable range.

Note This parameter is not used because this version of Cisco Unified TSP does not support call forward no answer.

lphConsultCall

A pointer to an HCALL location. In some telephony environments, this location is loaded with a handle to a consultation call that is used to consult the party to which the call is being forwarded, and the application becomes the initial sole owner of this call. This pointer must be valid even in environments where call forwarding does not require a consultation call. This handle is set to NULL if no consultation call is created.

Note This parameter is also ignored because a consult call is not created for setting up lineForward.

lpCallParams

A pointer to a structure of type LINECALLPARAMS. This pointer gets ignored unless lineForward requires the establishment of a call to the forwarding destination (and lphConsultCall is returned; in which case, lpCallParams is optional). If NULL, default call parameters get used. Otherwise, the specified call parameters get used for establishing hConsultCall.

Note This parameter must be NULL for this version of Cisco Unified TSP because we do not create a consult call.

Return Values

Returns zero if the request succeeds or a negative number if an error occurs. Possible return values follow:

- LINEERR_INVALLINEHANDLE
- LINEERR_NOMEM
- LINEERR_INVALADDRESSID
- LINEERR_OPERATIONUNAVAIL
- LINEERR_INVALADDRESS
- LINEERR_OPERATIONFAILED
- LINEERR_INVALCOUNTRYCODE
- LINEERR_RESOURCEUNAVAIL
- LINEERR_INVALPOINTER
- LINEERR_STRUCTURETOOSMALL
- LINEERR_INVALPARAM
- LINEERR_UNINITIALIZED

Note For lpForwardList[0].dwForwardMode other than UNCOND, lineForward returns LINEERR_OPERATIONUNAVAIL. For lpForwardList.dwNumEntries more than 1, lineForward returns LINEERR_INVALPARAM

lineGenerateDigits

The lineGenerateDigits function initiates the generation of the specified digits on the specified call as out-of-band tones by using the specified signaling mode.

Note The Cisco Unified TSP supports neither invoking this function with a NULL value for lpszDigits to abort a digit generation that is currently in progress nor invoking lineGenerateDigits while digit generation is in progress. Cisco Unified IP Phones pass DTMF digits out of band. This means that the tone is not injected into the audio stream (in-band) but is sent as a message in the control stream. The phone on the far end then injects the tone into the audio stream to present it to the user. CTI port devices do not inject DTMF tones. Also, be aware that some gateways will not inject DTMF tones into the audio stream on the way out of the LAN.

Function Details

```
LONG lineGenerateDigits(  HCALL hCall,
  DWORD dwDigitMode,
  LPCSTR lpszDigits,
  DWORD dwDuration
);
```

Parameters

hCall

A handle to the call. The application must be an owner of the call. Call state of hCall can be any state.

dwDigitMode

The format to be used for signaling these digits. The dwDigitMode can have only a single flag set. This parameter uses the following LINEDIGITMODE_constant:

- LINEDIGITMODE_DTMF -Uses DTMF tones for digit signaling. Valid digits for DTMF mode include ‘0’ -‘9’, ‘*’, ‘#’.

  lpszDigits

  Valid characters for DTMF mode in the Cisco Unified TSP include ‘0’ through ‘9’, ‘*’, and ‘#’.

  dwDuration

  Duration in milliseconds during which the tone should be sustained.

Note Cisco Unified TSP does not support dwDuration.

lineGenerateTone

The lineGenerateTone function generates the specified tone over the specified call.

Note The Cisco Unified TSP supports neither invoking this function with a 0 value for dwToneMode to abort a tone generation that is currently in progress nor invoking lineGenerateTone while tone generation is in progress. Cisco Unified IP Phones pass tones out of band. This means that the tone is not injected into the audio stream (in-band) but is sent as a message in the control stream. The phone on the far end then injects the tone into the audio stream to present it to the user. Also, be aware that some gateways will not inject tones into the audio stream on the way out of the LAN.

Function Details

```
LONG lineGenerateTone(  HCALL hCall,
  DWORD dwToneMode,
  DWORD dwDuration,
  DWORD dwNumTones,
  LPLINEGENERATETONE const lpTones
);
```

Parameters

hCall

A handle to the call on which a tone is to be generated. The application must be an owner of the call. The call state of hCall can be any state.

dwToneMode

Defines the tone to be generated. Tones can be either standard or custom tones. A custom tone comprises a set of arbitrary frequencies. A small number of standard tones are predefined. The duration of the tone gets specified with dwDuration for both standard and custom tones. The dwToneMode parameter can have only one bit set. If no bits are set (the value 0 is passed), tone generation gets canceled.

This parameter uses the following LINETONEMODE_constant:

- LINETONEMODE_BEEP -The tone is a beep, as used to announce the beginning of a recording. The service provider defines the exact beep tone.

dwDuration

Duration in milliseconds during which the tone should be sustained.

Note Cisco Unified TSP does not support dwDuration.

dwNumTones

The number of entries in the lpTones array. This parameter is ignored if dwToneMode ≠ CUSTOM.

lpTones

A pointer to a LINEGENERATETONE array that specifies the components of the tone. This parameter gets ignored for non-custom tones. If lpTones is a multifrequency tone, the various tones play simultaneously.

lineGetAddressCaps

The lineGetAddressCaps function queries the specified address on the specified line device to determine its telephony capabilities.

Function Details

```
LONG lineGetAddressCaps(  HLINEAPP hLineApp,
  DWORD dwDeviceID,
  DWORD dwAddressID,
  DWORD dwAPIVersion,
  DWORD dwExtVersion,
  LPLINEADDRESSCAPS lpAddressCaps
);
```

Parameters

hLineApp

The handle by which the application is registered with TAPI.

dwDeviceID

The line device that contains the address to be queried. Only one address gets supported per line, so dwAddressID must be zero.

dwAddressID

The address on the given line device whose capabilities are to be queried.

dwAPIVersion

The version number, obtained by lineNegotiateAPIVersion, of the API that is to be used. The high-order word contains the major version number; the low-order word contains the minor version number.

dwExtVersion

The version number of the extensions to be used. This number can be left zero if no device-specific extensions are to be used. Otherwise, the high-order word contains the major version number and the low-order word contains the minor version number.

lpAddressCaps

A pointer to a variably sized structure of type LINEADDRESSCAPS. Upon successful completion of the request, this structure gets filled with address capabilities information. Prior to calling lineGetAddressCaps, the application should set the dwTotalSize member of this structure to indicate the amount of memory that is available to TAPI for returning information.

lineGetAddressID

The lineGetAddressID function returns the address identifier that is associated with an address in a different format on the specified line.

Function Details

```
LONG lineGetAddressID(  HLINE hLine,
  LPDWORD lpdwAddressID,
  DWORD dwAddressMode,
  LPCSTR lpsAddress,
  DWORD dwSize
);
```

Parameters

hLine

A handle to the open line device.

lpdwAddressID

A pointer to a DWORD-sized memory location that returns the address identifier.

dwAddressMode

The address mode of the address that is contained in lpsAddress. The dwAddressMode parameter can have only a single flag set. This parameter uses the following LINEADDRESSMODE_constant:

- LINEADDRESSMODE_DIALABLEADDR -The address is specified by its dialable address. The lpsAddress parameter represents the dialable address or canonical address format.

lpsAddress

A pointer to a data structure that holds the address that is assigned to the specified line device. dwAddressMode determines the format of the address. Because the only valid value equals

LINEADDRESSMODE_DIALABLEADDR, lpsAddress uses the common dialable number format and is NULL-terminated.

dwSize

The size of the address that is contained in lpsAddress.

lineGetAddressStatus

The lineGetAddressStatus function allows an application to query the specified address for its current status.

Function Details

```
LONG lineGetAddressStatus(  HLINE hLine,
  DWORD dwAddressID,
  LPLINEADDRESSSTATUS lpAddressStatus
);
```

Parameters

hLine

A handle to the open line device.

dwAddressID

An address on the given open line device. This parameter specifies the address to be queried.

lpAddressStatus

A pointer to a variably sized data structure of type LINEADDRESSSTATUS. Prior to calling lineGetAddressStatus, the application should set the dwTotalSize member of this structure to indicate the amount of memory that is available to TAPI for returning information.

lineGetCallInfo

The lineGetCallInfo function enables an application to obtain fixed information about the specified call.

Function Details

```
LONG lineGetCallInfo(  HCALL hCall,
  LPLINECALLINFO lpCallInfo
);
```

Parameters

hCall

A handle to the call to be queried. The call state of hCall can be any state.

lpCallInfo

A pointer to a variably sized data structure of type LINECALLINFO. Upon successful completion of the request, call-related information fills this structure. Prior to calling lineGetCallInfo, the application

should set the dwTotalSize member of this structure to indicate the amount of memory that is available to TAPI for returning information.

lineGetCallStatus

The lineGetCallStatus function returns the current status of the specified call.

Function Details

```
LONG lineGetCallStatus(  HCALL hCall,
  LPLINECALLSTATUS lpCallStatus
);
```

Parameters

hCall

A handle to the call to be queried. The call state of hCall can be any state.

lpCallStatus

A pointer to a variably sized data structure of type LINECALLSTATUS. Upon successful completion of the request, call status information fills this structure. Prior to calling lineGetCallStatus, the application should set the dwTotalSize member of this structure to indicate the amount of memory available to TAPI for returning information.

lineGetConfRelatedCalls

The lineGetConfRelatedCalls function returns a list of call handles that are part of the same conference call as the specified call. The specified call represents either a conference call or a participant call in a conference call. New handles get generated for those calls for which the application does not already have handles, and the application receives monitor privilege to those calls.

Function Details

```
LONG WINAPI lineGetConfRelatedCalls(  HCALL hCall,
  LPLINECALLLIST lpCallList
);
```

Parameters

hCall

A handle to a call. This represents either a conference call or a participant call in a conference call. For a conference parent call, the call state of hCall can be any state. For a conference participant call, it must be in the conferenced state.

lpCallList

A pointer to a variably sized data structure of type LINECALLLIST. Upon successful completion of the request, call handles to all calls in the conference call return in this structure. The first call in the list represents the conference call, the other calls represent the participant calls. The application receives monitor privilege to those calls for which it does not already have handles; the privileges to calls in the

list for which the application already has handles remains unchanged. Prior to calling lineGetConfRelatedCalls, the application should set the dwTotalSize member of this structure to indicate the amount of memory that is available to TAPI for returning information.

Return Values

Returns zero if request succeeds or a negative number if an error occurs. Possible return values follow:

- LINEERR_INVALCALLHANDLE
- LINEERR_OPERATIONFAILED
- LINEERR_NOCONFERENCE
- LINEERR_RESOURCEUNAVAIL
- LINEERR_INVALPOINTER
- LINEERR_STRUCTURETOOSMALL
- LINEERR_NOMEM
- LINEERR_UNINITIALIZED

lineGetDevCaps

The lineGetDevCaps function queries a specified line device to determine its telephony capabilities. The returned information applies for all addresses on the line device.

Function Details

```
LONG lineGetDevCaps(  HLINEAPP hLineApp,
  DWORD dwDeviceID,
  DWORD dwAPIVersion,
  DWORD dwExtVersion,
  LPLINEDEVCAPS lpLineDevCaps
);
```

Parameters

hLineApp

The handle by which the application is registered with TAPI.

dwDeviceID

The line device to be queried.

dwAPIVersion

The version number, obtained by lineNegotiateAPIVersion, of the API to be used. The high-order word contains the major version number; the low-order word contains the minor version number.

dwExtVersion

The version number, obtained by lineNegotiateExtVersion, of the extensions to be used. It can be zero if no device-specific extensions are to be used. Otherwise, the high-order word contains the major version number; the low-order word contains the minor version number.

lpLineDevCaps

A pointer to a variably sized structure of type LINEDEVCAPS. Upon successful completion of the request, this structure gets filled with line device capabilities information. Prior to calling lineGetDevCaps, the application should set the dwTotalSize member of this structure to indicate the amount of memory that is available to TAPI for returning information.

lineGetID

The lineGetID function returns a device identifier for the specified device class that is associated with the selected line, address, or call.

Function Details

```
LONG lineGetID(  HLINE hLine,
  DWORD dwAddressID,
  HCALL hCall,
  DWORD dwSelect,
  LPVARSTRING lpDeviceID,
  LPCSTR lpszDeviceClass
);
```

Parameters

hLine

A handle to an open line device.

dwAddressID

An address on the given open line device.

hCall

A handle to a call.

dwSelect

Specifies whether the requested device identifier is associated with the line, address or a single call. The dwSelect parameter can only have a single flag set. This parameter uses the following LINECALLSELECT_constants:

- LINECALLSELECT_LINE Selects the specified line device. The hLine parameter must be a valid line handle; hCall and dwAddressID are ignored.
- LINECALLSELECT_ADDRESS Selects the specified address on the line. Both hLine and dwAddressID must be valid; hCall is ignored.
- LINECALLSELECT_CALL Selects the specified call. hCall must be valid; hLine and dwAddressID are both ignored.

lpDeviceID

A pointer to a memory location of type VARSTRING, where the device identifier is returned. Upon successful completion of the request, the device identifier fills this location. The format of the returned information depends on the method that the device class API uses for naming devices. Before calling

lineGetID, the application must set the dwTotalSize member of this structure to indicate the amount of memory that is available to TAPI for returning information.

lpszDeviceClass

A pointer to a NULL-terminated ASCII string that specifies the device class of the device whose identifier is requested. Device classes include wave/in, wave/out and tapi/line.

Valid device class strings are those that are used in the SYSTEM.INI section to identify device classes.

lineGetLineDevStatus

The lineGetLineDevStatus function enables an application to query the specified open line device for its current status.

Function Details

```
LONG lineGetLineDevStatus(  HLINE hLine,
  LPLINEDEVSTATUS lpLineDevStatus
);
```

Parameters

hLine

A handle to the open line device to be queried.

lpLineDevStatus

A pointer to a variably sized data structure of type LINEDEVSTATUS. Upon successful completion of the request, the device status of the line fills this structure. Prior to calling lineGetLineDevStatus, the application should set the dwTotalSize member of this structure to indicate the amount of memory that is available to TAPI for returning information.

lineGetMessage

The lineGetMessage function returns the next TAPI message that is queued for delivery to an application that is using the Event Handle notification mechanism (see lineInitializeEx, on page 34 for more information).

Function Details

```
LONG WINAPI lineGetMessage(  HLINEAPP hLineApp,
  LPLINEMESSAGE lpMessage,
  DWORD dwTimeout
);
```

Parameters

hLineApp

The handle that lineInitializeEx returns. Ensure that the application has set the LINEINITIALIZEEXOPTION_USEEVENT option in the dwOptions member of the LINEINITIALIZEEXPARAMS structure.

lpMessage

A pointer to a LINEMESSAGE structure. Upon successful return from this function, the structure contains the next message that had been queued for delivery to the application.

dwTimeout

The time-out interval, in milliseconds. The function returns if the interval elapses, even if no message can be returned. If dwTimeout is zero, the function checks for a queued message and returns immediately. If dwTimeout is INFINITE, the function time-out interval never elapses.

Return Values

Returns zero if the request succeeds or returns a negative number if an error occurs. Possible return values follow:

- LINEERR_INVALAPPHANDLE
- LINEERR_OPERATIONFAILED
- LINEERR_INVALPOINTER
- LINEERR_NOMEM

lineGetNewCalls

The lineGetNewCalls function returns call handles to calls on a specified line or address for which the application currently does not have handles. The application receives monitor privilege for these calls.

An application can use lineGetNewCalls to obtain handles to calls for which it currently has no handles. The application can select the calls for which handles are to be returned by basing this selection on scope (calls on a specified line, or calls on a specified address). For example, an application can request call handles to all calls on a given address for which it currently has no handle.

Function Details

```
LONG WINAPI lineGetNewCalls(  HLINE hLine,
  DWORD dwAddressID,
  DWORD dwSelect,
  LPLINECALLLIST lpCallList
);
```

Parameters

hLine

A handle to an open line device.

dwAddressID

An address on the given open line device. An address identifier permanently associates with an address; the identifier remains constant across operating system upgrades.

dwSelect

The selection of calls that are requested. This parameter uses one and only one of the LINECALLSELECT_Constants.

lpCallList

A pointer to a variably sized data structure of type LINECALLLIST. Upon successful completion of the request, call handles to all selected calls get returned in this structure. Prior to calling lineGetNewCalls, the application should set the dwTotalSize member of this structure to indicate the amount of memory that is available to TAPI for returning information.

Return Values

Returns zero if the request succeeds or a negative number if an error occurs. Possible return values follow:

- LINEERR_INVALADDRESSID
- LINEERR_OPERATIONFAILED
- LINEERR_INVALCALLSELECT
- LINEERR_RESOURCEUNAVAIL
- LINEERR_INVALLINEHANDLE
- LINEERR_STRUCTURETOOSMALL
- LINEERR_INVALPOINTER
- LINEERR_UNINITIALIZED
- LINEERR_NOMEM

lineGetNumRings

The lineGetNumRings function determines the number of rings that an incoming call on the given address should ring before the call is answered.

Function Details

```
LONG WINAPI lineGetNumRings(  HLINE hLine,
  DWORD dwAddressID,
  LPDWORD lpdwNumRings
);
```

Parameters

hLine

A handle to the open line device.

dwAddressID

An address on the line device. An address identifier permanently associates with an address; the identifier remains constant across operating system upgrades.

lpdwNumRings

The number of rings that is the minimum of all current lineSetNumRings requests.

Return Values

Returns zero if request succeeds or a negative number if an error occurs. Possible return values follow:

- LINEERR_INVALADDRESSID
- LINEERR_OPERATIONFAILED
- LINEERR_INVALLINEHANDLE
- LINEERR_RESOURCEUNAVAIL
- LINEERR_INVALPOINTER
- LINEERR_UNINITIALIZED
- LINEERR_NOMEM

lineGetProviderList

The lineGetProviderList function returns a list of service providers that are currently installed in the telephony system.

Function Details

```
LONG WINAPI lineGetProviderList(  DWORD dwAPIVersion,
  LPLINEPROVIDERLIST lpProviderList
);
```

Parameters

dwAPIVersion

The highest version of TAPI that the application supports (not necessarily the value that lineNegotiateAPIVersion negotiates on some particular line device).

lpProviderList

A pointer to a memory location where TAPI can return a LINEPROVIDERLIST structure. Prior to calling lineGetProviderList, the application should set the dwTotalSize member of this structure to indicate the amount of memory that is available to TAPI for returning information.

Return Values

Returns zero if request succeeds or a negative number if an error occurs. Possible return values follow:

- LINEERR_INCOMPATIBLEAPIVERSION
- LINEERR_NOMEM
- LINEERR_INIFILECORRUPT
- LINEERR_OPERATIONFAILED
- LINEERR_INVALPOINTER
- LINEERR_STRUCTURETOOSMALL

lineGetRequest

The lineGetRequest function retrieves the next by-proxy request for the specified request mode.

Function Details

```
LONG WINAPI lineGetRequest(  HLINEAPP hLineApp,
  DWORD dwRequestMode,
  LPVOID lpRequestBuffer
);
```

Parameters

hLineApp

The application's usage handle for the line portion of TAPI.

dwRequestMode

The type of request that is to be obtained. dwRequestMode can have only one bit set. This parameter uses one and only one of the LINEREQUESTMODE_Constants.

lpRequestBuffer

A pointer to a memory buffer where the parameters of the request are to be placed. The size of the buffer and the interpretation of the information that is placed in the buffer depends on the request mode. The application-allocated buffer provides sufficient size to hold the request. If dwRequestMode is LINEREQUESTMODE_MAKECALL, interpret the content of the request buffer by using the LINEREQMAKECALL structure. If dwRequestMode is LINEREQUESTMODE_MEDIACALL, interpret the content of the request buffer by using the LINEREQMEDIACALL structure.

Return Values

Returns zero if the request succeeds or a negative number if an error occurs. Possible return values follow:

- LINEERR_INVALAPPHANDLE
- LINEERR_NOTREGISTERED
- LINEERR_INVALPOINTER
- LINEERR_OPERATIONFAILED
- LINEERR_INVALREQUESTMODE
- LINEERR_RESOURCEUNAVAIL
- LINEERR_NOMEM
- LINEERR_UNINITIALIZED
- LINEERR_NOREQUEST

lineGetStatusMessages

The lineGetStatusMessages function enables an application to query the notification messages that the application receives for events related to status changes for the specified line or any of its addresses.

Function Details

```
LONG WINAPI lineGetStatusMessages(  HLINE hLine,
  LPDWORD lpdwLineStates,
  LPDWORD lpdwAddressStates
);
```

Parameters

hLine

Handle to the line device.

lpdwLineStates

A bit array that identifies the line device status changes for which a message is to be sent to the application. If a flag is TRUE, that message is enabled; if FALSE, it is disabled. This parameter uses one or more LINEDEVSTATE_Constants.

lpdwAddressStates

A bit array that identifies for which address status changes a message is to be sent to the application. If a flag is TRUE, that message is enabled; if FALSE, disabled. This parameter uses one or more LINEADDRESSSTATE_Constants.

Return Values

Returns zero if the request succeeds or a negative number if an error occurs. Possible return values follow:

- LINEERR_INVALLINEHANDLE
- LINEERR_OPERATIONFAILED
- LINEERR_INVALPOINTER
- LINEERR_RESOURCEUNAVAIL
- LINEERR_NOMEM
- LINEERR_UNINITIALIZED

lineGetTranslateCaps

The lineGetTranslateCaps function returns address translation capabilities.

Function Details

```
LONG WINAPI lineGetTranslateCaps(  HLINEAPP hLineApp,
  DWORD dwAPIVersion,
```

```
  LPLINETRANSLATECAPS lpTranslateCaps
);
```

Parameters

hLineApp

The application handle that lineInitializeEx returns. If an application has not yet called the lineInitializeEx function, it can set the hLineApp parameter to NULL.

dwAPIVersion

The highest version of TAPI that the application supports (not necessarily the value that lineNegotiateAPIVersion negotiates on some particular line device).

lpTranslateCaps

A pointer to a location to which a LINETRANSLATECAPS structure is loaded. Prior to calling lineGetTranslateCaps, the application should set the dwTotalSize member of this structure to indicate the amount of memory that is available to TAPI for returning information.

Return Values

Returns zero if the request succeeds or a negative number if an error occurs. Possible return values follow:

- LINEERR_INCOMPATIBLEAPIVERSION
- LINEERR_NOMEM
- LINEERR_INIFILECORRUPT
- LINEERR_OPERATIONFAILED
- LINEERR_INVALAPPHANDLE
- LINEERR_RESOURCEUNAVAIL
- LINEERR_INVALPOINTER
- LINEERR_STRUCTURETOOSMALL
- LINEERR_NODRIVER.

lineHandoff

The lineHandoff function gives ownership of the specified call to another application. Specify the application either directly by its file name or indirectly as the highest priority application that handles calls of the specified media mode.

Function Details

```
LONG WINAPI lineHandoff(  HCALL hCall,
  LPCSTR lpszFileName,
  DWORD dwMediaMode
);
```

Parameters

hCall

A handle to the call to be handed off. The application must be an owner of the call. The call state of hCall can be any state.

lpszFileName

A pointer to a null-terminated string. If this pointer parameter is non-NULL, it contains the file name of the application that is the target of the handoff. If NULL, the handoff target represents the highest priority application that has opened the line for owner privilege for the specified media mode. A valid file name does not include the path of the file.

dwMediaMode

The media mode that is used to identify the target for the indirect handoff. The dwMediaMode parameter indirectly identifies the target application that is to receive ownership of the call. This parameter gets ignored if lpszFileName is not NULL. This parameter uses one and only one of the LINEMEDIAMODE_Constants.

Return Values

Returns zero if the request succeeds or a negative number if an error occurs. Possible return values follow:

- LINEERR_INVALCALLHANDLE
- LINEERR_OPERATIONFAILED
- LINEERR_INVALMEDIAMODE
- LINEERR_TARGETNOTFOUND
- LINEERR_INVALPOINTER
- LINEERR_TARGETSELF
- LINEERR_NOMEM
- LINEERR_UNINITIALIZED
- LINEERR_NOTOWNER

lineHold

The lineHold function places the specified call on hold.

Function Details

```
LONG lineHold(  HCALL hCall
);
```

Parameter

hCall

A handle to the call that is to be placed on hold. Ensure that the application is an owner of the call and the call state of hCall is connected.

lineInitialize

Although the lineInitialize function is obsolete, tapi.dll and tapi32.dll continue to export it for backward compatibility with applications that are using API versions 1.3 and 1.4.

Function Details

```
LONG WINAPI lineInitialize(  LPHLINEAPP lphLineApp,
  HINSTANCE hInstance,
  LINECALLBACK lpfnCallback,
  LPCSTR lpszAppName,
  LPDWORD lpdwNumDevs
);
```

Parameters

lphLineApp

A pointer to a location that is filled with the application's usage handle for TAPI.

hInstance

The instance handle of the client application or DLL.

lpfnCallback

The address of a callback function that is invoked to determine status and events on the line device, addresses, or calls. For more information, see lineCallbackFunc.

lpszAppName

A pointer to a null-terminated text string that contains only displayable characters. If this parameter is not NULL, it contains an application-supplied name for the application. The LINECALLINFO structure provides this name to indicate, in a user-friendly way, which application originated, originally accepted, or answered the call. This information can prove useful for call logging purposes. If lpszAppName is NULL, the application's file name gets used instead.

lpdwNumDevs

A pointer to a DWORD-sized location. Upon successful completion of this request, this location gets filled with the number of line devices that is available to the application.

Return Values

Returns zero if the request succeeds or a negative number if an error occurs. Possible return values follow:

- LINEERR_INVALAPPNAME
- LINEERR_OPERATIONFAILED
- LINEERR_INIFILECORRUPT
- LINEERR_RESOURCEUNAVAIL
- LINEERR_INVALPOINTER

- LINEERR_REINIT
- LINEERR_NODRIVER
- LINEERR_NODEVICE
- LINEERR_NOMEM
- LINEERR_NOMULTIPLEINSTANCE.

lineInitializeEx

The lineInitializeEx function initializes the use of TAPI by the application for the subsequent use of the line abstraction. It registers the specified notification mechanism of the application and returns the number of line devices that are available. A line device represents any device that provides an implementation for the line-prefixed functions in the telephony API.

Function Details

```
LONG lineInitializeEx(  LPHLINEAPP lphLineApp,
  HINSTANCE hInstance,
  LINECALLBACK lpfnCallback,
  LPCSTR lpszFriendlyAppName,
  LPDWORD lpdwNumDevs,
  LPDWORD lpdwAPIVersion,
  LPLINEINITIALIZEEXPARAMS lpLineInitializeExParams
);
```

Parameters

lphLineApp

A pointer to a location that is filled with the TAPI usage handle for the application.

hInstance

The instance handle of the client application or DLL. The application or DLL can pass NULL for this parameter, in which case TAPI uses the module handle of the root executable of the process (for purposes of identifying call handoff targets and media mode priorities).

lpfnCallback

The address of a callback function that is invoked to determine status and events on the line device, addresses, or calls, when the application is using the “hidden window” method of event notification. This parameter gets ignored and should be set to NULL when the application chooses to use the “event handle” or “completion port” event notification mechanisms.

lpszFriendlyAppName

A pointer to a NULL-terminated ASCII string that contains only standard ASCII characters. If this parameter is not NULL, it contains an application-supplied name for the application. The LINECALLINFO structure provides this name to indicate, in a user-friendly way, which application originated, originally accepted, or answered the call. This information can prove useful for call-logging purposes. If lpszFriendlyAppName is NULL, the module filename of the application gets used instead (as returned by the Windows API GetModuleFileName).

lpdwNumDevs

A pointer to a DWORD-sized location. Upon successful completion of this request, this location gets filled with the number of line devices that are available to the application.

lpdwAPIVersion

A pointer to a DWORD-sized location. The application must initialize this DWORD, before calling this function, to the highest API version that it is designed to support (for example, the same value that it would pass into dwAPIHighVersion parameter of lineNegotiateAPIVersion). Make sure that artificially high values are not used; ensure that the value is set to 0x00020000. TAPI translates any newer messages or structures into values or formats that the application supports. Upon successful completion of this request, this location is filled with the highest API version that TAPI supports, which allows the application to adapt to being installed on a system with an older TAPI version.

lpLineInitializeExParams

A pointer to a structure of type LINEINITIALIZEEXPARAMS that contains additional parameters that are used to establish the association between the application and TAPI (specifically, the selected event notification mechanism of the application and associated parameters).

lineMakeCall

The lineMakeCall function places a call on the specified line to the specified destination address. Optionally, you can specify call parameters if anything but default call setup parameters are requested.

Function Details

```
LONG lineMakeCall(HLINE hLine,
LPHCALL lphCall,
LPCSTR lpszDestAddress,
DWORD dwCountryCode,
LPLINECALLPARAMS const lpCallParams
);
typedef struct LineParams {
DWORD FeaturePriority;
```

Parameters

hLine

A handle to the open line device on which a call is to be originated.

lphCall

A pointer to an HCALL handle. The handle is only valid after the application receives LINE_REPLY message that indicates that the lineMakeCall function successfully completed. Use this handle to identify the call when you invoke other telephony operations on the call. The application initially acts as the sole owner of this call. This handle registers as void if the reply message returns an error (synchronously or asynchronously).

lpszDestAddress

A pointer to the destination address. This parameter follows the standard dialable number format. This pointer can be NULL for non-dialed addresses or when all dialing is performed by using lineDial. In the latter case, lineMakeCall allocates an available call appearance that would typically remain in the dial tone state until dialing begins.

dwCountryCode

The country code of the called party. If a value of 0 is specified, the implementation uses a default.

lpCallParams

The dwNoAnswerTimeout attribute of the lpCallParams field is checked and if specified as non-zero, automatically disconnects a call if not answered after the specified time. For more information, see LINECALLPARAMS.

lineMonitorDigits

The lineMonitorDigits function enables and disables the unbuffered detection of digits that are received on the call. Each time that a digit of the specified digit mode is detected, a message gets sent to the application to indicate which digit has been detected.

Function Details

```
LONG lineMonitorDigits(  HCALL hCall,
  DWORD dwDigitModes
);
```

Parameters

hCall

A handle to the call on which digits are to be detected. The call state of hCall can be any state except idle or disconnected.

dwDigitModes

The digit mode or modes that are to be monitored. If dwDigitModes is zero, the system cancels digit monitoring. This parameter which can have multiple flags set, uses the following LINEDIGITMODE_constant:

LINEDIGITMODE_DTMF -Detect digits as DTMF tones. Valid digits for DTMF include ‘0’ through ‘9’, ‘*’, and ‘#’.

lineMonitorTones

The lineMonitorTones function enables and disables the detection of inband tones on the call. Each time that a specified tone is detected, a message gets sent to the application.

Function Details

```
LONG lineMonitorTones(  HCALL hCall,
  LPLINEMONITORTONE const lpToneList,
  DWORD dwNumEntries
);
```

Parameters

hCall

A handle to the call on which tones are to be detected. The call state of hCall can be any state except idle.

lpToneList

A list of tones to be monitored, of type LINEMONITORTONE. Each tone in this list has an application-defined tag field that is used to identify individual tones in the list to report a tone detection. Calling this operation with either NULL for lpToneList or with another tone list cancels or changes tone monitoring in progress.

dwNumEntries

The number of entries in lpToneList. This parameter gets ignored if lpToneList is NULL.

lineNegotiateAPIVersion

The lineNegotiateAPIVersion function allows an application to negotiate an API version to use. The Cisco Unified TSP supports TAPI 2.0 and 2.1.

Function Details

```
LONG lineNegotiateAPIVersion(  HLINEAPP hLineApp,
  DWORD dwDeviceID,
  DWORD dwAPILowVersion,
  DWORD dwAPIHighVersion,
  LPDWORD lpdwAPIVersion,
  LPLINEEXTENSIONID lpExtensionID
);
```

Parameters

hLineApp

The handle by which the application is registered with TAPI.

dwDeviceID

The line device to be queried.

dwAPILowVersion

The least recent API version with which the application is compliant. The high-order word specifies the major version number; the low-order word specifies the minor version number.

dwAPIHighVersion

The most recent API version with which the application is compliant. The high-order word specifies the major version number; the low-order word specifies the minor version number.

lpdwAPIVersion

A pointer to a DWORD-sized location that contains the API version number that was negotiated. If negotiation succeeds, this number falls in the range between dwAPILowVersion and dwAPIHighVersion.

lpExtensionID

A pointer to a structure of type LINEEXTENSIONID. If the service provider for the specified dwDeviceID supports provider-specific extensions, upon a successful negotiation, this structure gets filled with the extension identifier of these extensions. This structure contains all zeros if the line provides no extensions. An application can ignore the returned parameter if it does not use extensions.

The Cisco Unified TSP extensionID specifies 0x8EBD6A50, 0x138011d2, 0x905B0060, 0xB03DD275.

lineNegotiateExtVersion

The lineNegotiateExtVersion function allows an application to negotiate an extension version to use with the specified line device. Do not call this operation if the application does not support extensions.

Function Details

```
LONG lineNegotiateExtVersion(  HLINEAPP hLineApp,
  DWORD dwDeviceID,
  DWORD dwAPIVersion,
  DWORD dwExtLowVersion,
  DWORD dwExtHighVersion,
  LPDWORD lpdwExtVersion
);
```

Parameters

hLineApp

The handle by which the application is registered with TAPI.

dwDeviceID

The line device to be queried.

dwAPIVersion

The API version number that was negotiated for the specified line device by using lineNegotiateAPIVersion.

dwExtLowVersion

The least recent extension version of the extension identifier that lineNegotiateAPIVersion returns and with which the application is compliant. The high-order word specifies the major version number; the low-order word specifies the minor version number.

dwExtHighVersion

The most recent extension version of the extension identifier that lineNegotiateAPIVersion returns and with which the application is compliant. The high-order word specifies the major version number; the low-order word specifies the minor version number.

lpdwExtVersion

A pointer to a DWORD-sized location that contains the extension version number that was negotiated. If negotiation succeeds, this number falls between dwExtLowVersion and dwExtHighVersion.

lineOpen

The lineOpen function opens the line device that its device identifier specifies and returns a line handle for the corresponding opened line device. Subsequent operations on the line device use this line handle.

Function Details

```
LONG lineOpen(  HLINEAPP hLineApp,
  DWORD dwDeviceID,
  LPHLINE lphLine,
  DWORD dwAPIVersion,
  DWORD dwExtVersion,
  DWORD dwCallbackInstance,
  DWORD dwPrivileges,
  DWORD dwMediaModes,
  LPLINECALLPARAMS const lpCallParams
);
```

Parameters

hLineApp

The handle by which the application is registered with TAPI.

dwDeviceID

Identifies the line device to be opened. It can either be a valid device identifier or the value LINEMAPPER

Note The Cisco Unified TSP does not support LINEMAPPER at this time.

lphLine

A pointer to an HLINE handle that is then loaded with the handle that represents the opened line device. Use this handle to identify the device when you are invoking other functions on the open line device.

dwAPIVersion

The API version number under which the application and Telephony API operate. Obtain this number with lineNegotiateAPIVersion.

dwExtVersion

The extension version number under which the application and the service provider operate. This number remains zero if the application does not use any extensions. Obtain this number with lineNegotiateExtVersion.

dwCallbackInstance

User-instance data that is passed back to the application with each message that is associated with this line or with addresses or calls on this line. The Telephony API does not interpret this parameter.

dwPrivileges

The privilege that the application wants for the calls for which it is notified. This parameter can be a combination of the LINECALLPRIVILEGE_ constants. For applications that are using TAPI version 2.0 or later, values for this parameter can also be combined with the LINEOPENOPTION_constants:

- LINECALLPRIVILEGE_NONE -The application can make only outgoing calls.
- LINECALLPRIVILEGE_MONITOR -The application can monitor only incoming and outgoing calls.
- LINECALLPRIVILEGE_OWNER -The application can own only incoming calls of the types that are specified in dwMediaModes.
- LINECALLPRIVILEGE_MONITOR + LINECALLPRIVILEGE_OWNER -The application can own only incoming calls of the types that are specified in dwMediaModes, but if the application does not represent an owner of a call, it acts as a monitor.
- Other flag combinations return the LINEERR_INVALPRIVSELECT error.

dwMediaModes

The media mode or modes of interest to the application. Use this parameter to register the application as a potential target for incoming call and call handoff for the specified media mode. This parameter proves meaningful only if the bit LINECALLPRIVILEGE_OWNER in dwPrivileges is set (and ignored if it is not).

This parameter uses the following LINEMEDIAMODE_constant:

- LINEMEDIAMODE_INTERACTIVEVOICE -The application can handle calls of the interactive voice media type; that is, it manages voice calls with the user on this end of the call. Use this parameter for third-party call control of physical phones and CTI port and CTI route point devices that other applications opened.
- LINEMEDIAMODE_AUTOMATEDVOICE -Voice energy exists on the call. An automated application locally handles the voice. This represents first-party call control and is used with CTI port and CTI route point devices.

lpCallParams

The dwNoAnswerTimeout attribute of the lpCallParams field is checked and if it is non-zero, automatically disconnects a call if it is not answered after the specified time.

linePark

The linePark function parks the specified call according to the specified park mode.

Function Details

```
LONG WINAPI linePark(HCALL hCall,
DWORD dwParkMode,
LPCSTR lpszDirAddress,
LPVARSTRING lpNonDirAddress
);
```

Parameters

hCall

Handle to the call to be parked. The application must act as an owner of the call. The call state of hcall must be connected.

dwParkMode

Park mode with which the call is parked. This parameter can have only a single flag set and uses one of the LINEPARKMODE_Constants.

Note Ensure that LINEPARKMODE_Constants is set to LINEPARKMODE_NONDIRECTED.

lpszDirAddress

Pointer to a null-terminated string that indicates the address where the call is to be parked when directed park is used. The address specifies in dialable number format. This parameter gets ignored for nondirected park.

Note This parameter gets ignored.

lpNonDirAddress

Pointer to a structure of type VARSTRING. For nondirected park, the address where the call is parked gets returned in this structure. This parameter gets ignored for directed park. Within the VARSTRING structure, ensure that dwStringFormat is set to STRINGFORMAT_ASCII (an ASCII string buffer that contains a null-terminated string), and the terminating NULL must be accounted for in the dwStringSize. Before calling linePark, the application must set the dwTotalSize member of this structure to indicate the amount of memory that is available to TAPI for returning information.

linePrepareAddToConference

The linePrepareAddToConference function prepares an existing conference call for the addition of another party.

If LINEERR_INVALLINESTATE is returned, that means that the line is currently not in a state in which this operation can be performed. The dwLineFeatures member includes a list of currently valid operations (of the type LINEFEATURE) in the LINEDEVSTATUS structure. (Calling lineGetLineDevStatus updates the information in LINEDEVSTATUS.)

Obtain a conference call handle with lineSetupConference or with lineCompleteTransfer that is resolved as a three-way conference call. The linePrepareAddToConference function typically places the existing conference call in the onHoldPendingConference state and creates a consultation call that can be added later to the existing conference call with lineAddToConference.

You can cancel the consultation call by using lineDrop. You may also be able to swap an application between the consultation call and the held conference call with lineSwapHold.

Function Details

```
LONG WINAPI linePrepareAddToConference(  HCALL hConfCall,
  LPHCALL lphConsultCall,
  LPLINECALLPARAMS const lpCallParams
);
```

Parameters

hConfCall

A handle to a conference call. The application must act as an owner of this call. Ensure that the call state of hConfCall is connected.

lphConsultCall

A pointer to an HCALL handle. This location then gets loaded with a handle that identifies the consultation call to be added. Initially, the application serves as the sole owner of this call.

lpCallParams

A pointer to call parameters that gets used when the consultation call is established. You can set this parameter to NULL if no special call setup parameters are desired.

Return Values

Returns a positive request identifier if the function completes asynchronously, or a negative number if an error occurs. The dwParam2 parameter of the corresponding LINE_REPLY message specifies zero if the function succeeds, or it is a negative number if an error occurs.

Possible return values follow:

- LINEERR_BEARERMODEUNAVAIL
- LINEERR_INVALPOINTER
- LINEERR_CALLUNAVAIL
- LINEERR_INVALRATE
- LINEERR_CONFERENCEFULL
- LINEERR_NOMEM
- LINEERR_INUSE
- LINEERR_NOTOWNER
- LINEERR_INVALADDRESSMODE
- LINEERR_OPERATIONUNAVAIL
- LINEERR_INVALBEARERMODE
- LINEERR_OPERATIONFAILED
- LINEERR_INVALCALLPARAMS
- LINEERR_RATEUNAVAIL
- LINEERR_INVALCALLSTATE
- LINEERR_RESOURCEUNAVAIL
- LINEERR_INVALCONFCALLHANDLE
- LINEERR_STRUCTURETOOSMALL
- LINEERR_INVALLINESTATE

- LINEERR_USERUSERINFOTOOBIG
- LINEERR_INVALMEDIAMODE
- LINEERR_UNINITIALIZED

lineRedirect

The lineRedirect function redirects the specified offered or accepted call to the specified destination address.

Note If the application tries to redirect a call to an address that requires a FAC, CMC, or both, the lineRedirect function returns an error. If a FAC is required, the TSP returns the message LINEERR_FACREQUIRED. If a CMC is required, the TSP returns the message LINEERR_CMCREQUIRED. If both a FAC and a CMC are required, the TSP returns the message LINEERR_FACANDCMCREQUIRED. An application that wants to redirect a call to an address that requires a FAC, CMC, or both, should use the lineDevSpecific RedirectFACCMC function.

Function Details

```
LONG lineRedirect(  HCALL hCall,
  LPCSTR lpszDestAddress,
  DWORD dwCountryCode
);
```

Parameters

hCall

A handle to the call to be redirected. The application must act as an owner of the call. The call state of hCall must be offering, accepted, or connected.

Note The Cisco Unified TSP supports redirecting of calls in the connected call state.

lpszDestAddress

A pointer to the destination address. This follows the standard dialable number format.

dwCountryCode

The country code of the party to which the call is redirected. If a value of 0 is specified, the implementation uses a default.

lineRegisterRequestRecipient

The lineRegisterRequestRecipient function registers the invoking application as a recipient of requests for the specified request mode.

Function Details

```
LONG WINAPI lineRegisterRequestRecipient(  HLINEAPP hLineApp,
  DWORD dwRegistrationInstance,
  DWORD dwRequestMode,
  DWORD bEnable
);
```

Parameters

hLineApp

The application's usage handle for the line portion of TAPI.

dwRegistrationInstance

An application-specific DWORD that is passed back as a parameter of the LINE_REQUEST message. This message notifies the application that a request is pending. This parameter gets ignored if bEnable is set to zero. TAPI examines this parameter only for registration, not for deregistration. The dwRegistrationInstance value that is used while deregistering need not match the dwRegistrationInstance that is used while registering for a request mode.

dwRequestMode

The type or types of request for which the application registers. This parameter uses one or more LINEREQUESTMODE_Constants.

bEnable

If TRUE, the application registers the specified request modes; if FALSE, the application deregisters for the specified request modes.

Return Values

Returns zero if the request succeeds or a negative number if an error occurs. Possible return values follow:

- LINEERR_INVALAPPHANDLE
- LINEERR_OPERATIONFAILED
- LINEERR_INVALREQUESTMODE
- LINEERR_RESOURCEUNAVAIL
- LINEERR_NOMEM
- LINEERR_UNINITIALIZED

lineRemoveFromConference

The lineRemoveFromConference function removes a specified call from the conference call to which it currently belongs. The remaining calls in the conference call are unaffected.

Function Details

```
LONG WINAPI lineRemoveFromConference(  HCALL hCall
);
```

Parameters

hCall

Handle to the call that is to be removed from the conference. The application must be an owner of this call. The call state of hCall must be conference.

Return Values

Returns a positive request identifier if the function is completed asynchronously, or a negative number if an error occurs. The dwParam2 parameter of the corresponding LINE_REPLY message is zero if the function succeeds or it is a negative number if an error occurs. The following table shows the return values for this function:

Value	Description
LINEERR_INVALCALLHANDLE	The handle to the call that is to be removed is invalid.
LINEERR_OPERATIONUNAVAIL	The operation is unavailable.
LINEERR_INVALCALLSTATE	The call state is something other than conferenced.
LINEERR_OPERATIONFAILED	The operation failed.
LINEERR_NOMEM	Not enough memory.
LINEERR_RESOURCEUNAVAIL	The resources are unavailable.
LINEERR_NOTOWNER	The application is not the owner of this call.
LINEERR_UNINITIALIZED	A parameter is uninitialized.

lineRemoveProvider

The lineRemoveProvider function removes an existing telephony service provider from the system.

Function Details

```
LONG WINAPI lineRemoveProvider(  DWORD dwPermanentProviderID,
  HWND hwndOwner
);
```

Parameters

dwPermanentProviderID

The permanent provider identifier of the service provider that is to be removed.

hwndOwner

A handle to a window to which any dialog boxes that need to be displayed as part of the removal process (for example, a confirmation dialog box by the service provider's TSPI_providerRemove function) would be attached. The parameter can be a NULL value to indicate that any window that is created during the function should have no owner window.

Return Values

Returns zero if the request succeeds or a negative number if an error occurs. Possible return values follow:

- LINEERR_INIFILECORRUPT
- LINEERR_NOMEM
- LINEERR_INVALPARAM
- LINEERR_OPERATIONFAILED

lineSetAppPriority

The lineSetAppPriority function allows an application to set its priority in the handoff priority list for a particular media type or Assisted Telephony request mode or to remove itself from the priority list.

Function Details

```
LONG WINAPI lineSetAppPriority(  LPCSTR lpszAppFilename,
  DWORD dwMediaMode,
  LPLINEEXTENSIONID lpExtensionID,
  DWORD dwRequestMode,
  LPCSTR lpszExtensionName,
  DWORD dwPriority
);
```

Parameters

lpszAppFilename

A pointer to a string that contains the application executable module filename (without directory information). In TAPI version 2.0 or later, the parameter can specify a filename in either long or 8.3 filename format.

dwMediaMode

The media type for which the priority of the application is to be set. The value can be one LINEMEDIAMODE_Constant; only a single bit may be on. Use the value zero to set the application priority for Assisted Telephony requests.

lpExtensionID

A pointer to a structure of type LINEEXTENSIONID. This parameter gets ignored.

dwRequestMode

If the dwMediaMode parameter is zero, this parameter specifies the Assisted Telephony request mode for which priority is to be set. It must be either LINEREQUESTMODE_MAKECALL or LINEREQUESTMODE_MEDIACALL. This parameter gets ignored if dwMediaMode is nonzero.

lpszExtensionName

This parameter gets ignored.

dwPriority

The new priority for the application. If the value 0 is passed, the application gets removed from the priority list for the specified media or request mode (if it was already not present, no error gets generated). If the value 1 is passed, the application gets inserted as the highest priority application for the media or request mode (and removed from a lower-priority position, if it was already in the list). Any other value generates an error.

Return Values

Returns zero if the request succeeds or a negative number if an error occurs. Possible return values follow:

- LINEERR_INIFILECORRUPT
- LINEERR_INVALREQUESTMODE
- LINEERR_INVALAPPNAME
- LINEERR_NOMEM
- LINEERR_INVALMEDIAMODE
- LINEERR_OPERATIONFAILED
- LINEERR_INVALPARAM
- LINEERR_RESOURCEUNAVAIL
- LINEERR_INVALPOINTER

lineSetCallPrivilege

The lineSetCallPrivilege function sets the application privilege to the specified privilege.

Function Details

```
LONG WINAPI lineSetCallPrivilege(  HCALL hCall,
  DWORD dwCallPrivilege
);
```

Parameters

hCall

A handle to the call whose privilege is to be set. The call state of hCall can be any state.

dwCallPrivilege

The privilege that the application can have for the specified call. This parameter uses one and only one LINECALLPRIVILEGE_Constant.

Return Values

Returns zero if the request succeeds or a negative number if an error occurs. Possible return values follow:

- LINEERR_INVALCALLHANDLE
- LINEERR_OPERATIONFAILED
- LINEERR_INVALCALLSTATE
- LINEERR_RESOURCEUNAVAIL
- LINEERR_INVALCALLPRIVILEGE
- LINEERR_UNINITIALIZED
- LINEERR_NOMEM

lineSetNumRings

The lineSetNumRings function sets the number of rings that must occur before an incoming call is answered. Use this function to implement a toll saver-style function. It allows multiple, independent applications to each register the number of rings. The function lineGetNumRings returns the minimum number of rings that are requested. The application that answers incoming calls can use it to determine the number of rings that it should wait before answering the call.

Function Details

```
LONG WINAPI lineSetNumRings(  HLINE hLine,
  DWORD dwAddressID,
  DWORD dwNumRings
);
```

Parameters

hLine

A handle to the open line device.

dwAddressID

An address on the line device. An address identifier permanently associates with an address; the identifier remains constant across operating system upgrades.

dwNumRings

The number of rings before a call should be answered to honor the toll saver requests from all applications.

Return Values

Returns zero if the request succeeds or a negative number if an error occurs. Possible return values follow:

- LINEERR_INVALLINEHANDLE
- LINEERR_OPERATIONFAILED
- LINEERR_INVALADDRESSID

- LINEERR_RESOURCEUNAVAIL
- LINEERR_NOMEM
- LINEERR_UNINITIALIZED

lineSetStatusMessages

The lineSetStatusMessages function enables an application to specify the notification messages to receive for events that are related to status changes for the specified line or any of its addresses.

Function Details

```
LONG lineSetStatusMessages(  HLINE hLine,
  DWORD dwLineStates,
  DWORD dwAddressStates
);
```

Parameters

hLine

A handle to the line device.

dwLineStates

A bit array that identifies for which line-device status changes a message is to be sent to the application. This parameter uses the following LINEDEVSTATE_constants:

- LINEDEVSTATE_OTHER -Device-status items other than the following ones changed. The application should check the current device status to determine which items changed.
- LINEDEVSTATE_RINGING -The switch tells the line to alert the user. Service providers notify applications on each ring cycle by sending LINE_LINEDEVSTATE messages that contain this constant. For example, in the United States, service providers send a message with this constant every 6 seconds.
- LINEDEVSTATE_NUMCALLS -The number of calls on the line device changed.
- LINEDEVSTATE_REINIT -Items changed in the configuration of line devices. To become aware of these changes (as with the appearance of new line devices) the application should reinitialize its use of TAPI. New lineInitialize, lineInitializeEx, and lineOpen requests get denied until applications have shut down their usage of TAPI. The hDevice parameter of the LINE_LINEDEVSTATE message remains NULL for this state change as it applies to any lines in the system. Because of the critical nature of LINEDEVSTATE_REINIT, such messages cannot be masked, so the setting of this bit is ignored, and the messages always get delivered to the application.
- LINEDEVSTATE_REMOVED -Indicates that the service provider is removing the device from the system (most likely through user action, through a control panel or similar utility). Normally, a LINE_CLOSE message on the device immediately follows LINE_LINEDEVSTATE message with this value. Subsequent attempts to access the device prior to TAPI being reinitialized result in LINEERR_NODEVICE being returned to the application. If a service provider sends a LINE_LINEDEVSTATE message that contains this value to TAPI, TAPI passes it along to applications that have negotiated TAPI version 1.4 or later; applications that negotiate a previous TAPI version do not receive any notification.

dwAddressStates

A bit array that identifies for which address status changes a message is to be sent to the application. This parameter uses the following LINEADDRESSSTATE_constant:

- LINEADDRESSSTATE_NUMCALLS -The number of calls on the address changed. This change results from events such as a new incoming call, an outgoing call on the address, or a call changing its hold status.

lineSetTollList

The lineSetTollList function manipulates the toll list.

Function Details

```
LONG WINAPI lineSetTollList(  HLINEAPP hLineApp,
  DWORD dwDeviceID,
  LPCSTR lpszAddressIn,
  DWORD dwTollListOption
);
```

Parameters

hLineApp

The application handle that lineInitializeEx returns. If an application has not yet called the lineInitializeEx function, it can set the hLineApp parameter to NULL.

dwDeviceID

The device identifier for the line device upon which the call is intended to be dialed, so variations in dialing procedures on different lines can be applied to the translation process.

lpszAddressIn

A pointer to a null-terminated string that contains the address from which the prefix information is to be extracted for processing. Ensure that this parameter is not NULL, and also ensure that it is in the canonical address format.

dwTollListOption

The toll list operation to be performed. This parameter uses one and only one of the LINETOLLLISTOPTION_Constants.

Return Values

Returns zero if the request succeeds or a negative number if an error occurs. Possible return values follow:

- LINEERR_BADDEVICEID
- LINEERR_NODRIVER
- LINEERR_INVALAPPHANDLE
- LINEERR_NOMEM
- LINEERR_INVALADDRESS

- LINEERR_OPERATIONFAILED
- LINEERR_INVALPARAM
- LINEERR_RESOURCEUNAVAIL
- LINEERR_INIFILECORRUPT
- LINEERR_UNINITIALIZED
- LINEERR_INVALLOCATION

lineSetupConference

The lineSetupConference function initiates a conference for an existing two-party call that the hCall parameter specifies. A conference call and consult call are established, and the handles return to the application. Use the consult call to dial the third party and the conference call replaces the initial two-party call. The application can also specify the destination address of the consult call that will allow the PBX to dial the call for the application.

Function Details

```
LONG lineSetupConference (HCALL hCall,
HLINE hLine,
LPHCALL lphConfCall,
LPHCALL lphConsultCall,
DWORD dwNumParties,
LPLINECALLPARAMS const lpCallParams
);
```

Parameters

hCall

The handle of the existing two-party call. Ensure that the application is the owner of the call.

hLine

The line on which the initial two-party call was made. This parameter is not used because hCall must be set.

lphConfCall

A pointer to the conference call handle. The service provider allocates this call and returns the handle to the application.

lphConsultCall

A pointer to the consult call. If the application does not specify the destination address in the call parameters, it should use this call handle to dial the consult call. If the destination address is specified, the consult call will be made using this handle.

dwNumParties

The number of parties in the conference call. Currently the Cisco Unified TAPI Service Provider supports a three-party conference call.

lpCallParams

The call parameters that are used to set up the consult call. The application can specify the destination address if it wants the consult call to be dialed for it in the conference setup.

lineSetupTransfer

The lineSetupTransfer function initiates a transfer of the call that the hCall parameter specifies. It establishes a consultation call, lphConsultCall, on which the party can be dialed that can become the destination of the transfer. The application acquires owner privilege to the lphConsultCall parameter.

Function Details

```
LONG lineSetupTransfer(  HCALL hCall,
  LPHCALL lphConsultCall,
  LPLINECALLPARAMS const lpCallParams
);
```

Parameters

hCall

The handle of the call to be transferred. Ensure that the application is an owner of the call and ensure that the call state of hCall is connected.

lphConsultCall

A pointer to an hCall handle. This location is then loaded with a handle that identifies the temporary consultation call. When setting up a call for transfer, a consultation call automatically gets allocated that enables lineDial to dial the address that is associated with the new transfer destination of the call. The originating party can carry on a conversation over this consultation call prior to completing the transfer. The call state of hConsultCall does not apply.

This transfer procedure may not be valid for some line devices. The application may need to ignore the new consultation call and remove the hold on an existing held call (using lineUnhold) to identify the destination of the transfer. On switches that support cross-address call transfer, the consultation call can exist on a different address than the call that is to be transferred. It may also be necessary to set up the consultation call as an entirely new call, by lineMakeCall, to the destination of the transfer. The address capabilities of the call specifies which forms of transfer are available.

lpCallParams

The dwNoAnswerTimeout attribute of the lpCallParams field is checked and, if is non-zero, used to automatically disconnect a call if it is not answered after the specified time.

lineShutdown

The lineShutdown function shuts down the usage of the line abstraction of the API.

Function Details

```
LONG lineShutdown(  HLINEAPP hLineApp
);
```

Parameters

hLineApp

The usage handle of the application for the line API.

lineTranslateAddress

The lineTranslateAddress function translates the specified address into another format.

Function Details

```
LONG WINAPI lineTranslateAddress(  HLINEAPP hLineApp,
  DWORD dwDeviceID,
  DWORD dwAPIVersion,
  LPCSTR lpszAddressIn,
  DWORD dwCard,
  DWORD dwTranslateOptions,
  LPLINETRANSLATEOUTPUT lpTranslateOutput
);
```

Parameters

hLineApp

The application handle that lineInitializeEx returns. If a TAPI 2.0 application has not yet called the lineInitializeEx function, it can set the hLineApp parameter to NULL. TAPI 1.4 applications must still call lineInitialize first.

dwDeviceID

The device identifier for the line device upon which the call is intended to be dialed, so variations in dialing procedures on different lines can be applied to the translation process.

dwAPIVersion

Indicates the highest version of TAPI that the application supports (not necessarily the value that is negotiated by lineNegotiateAPIVersion on some particular line device).

lpszAddressIn

Pointer to a null-terminated string that contains the address from which the information is to be extracted for translation. This parameter must either use the canonical address format or an arbitrary string of dialable digits (non-canonical). This parameter must not be NULL. If the AddressIn contains a subaddress or name field, or additional addresses separated from the first address by CR and LF characters, only the first address gets translated.

dwCard

The credit card to be used for dialing. This parameter proves valid only if the CARDOVERRIDE bit is set in dwTranslateOptions. This parameter specifies the permanent identifier of a Card entry in the [Cards] section in the registry (as obtained from lineTranslateCaps) that should be used instead of the PreferredCardID that is specified in the definition of the CurrentLocation. It does not cause the PreferredCardID parameter of the current Location entry in the registry to be modified; the override applies only to the current translation operation. This parameter gets ignored if the CARDOVERRIDE bit is not set in dwTranslateOptions.

dwTranslateOptions

The associated operations to be performed prior to the translation of the address into a dialable string. This parameter uses one of the LINETRANSLATEOPTION_Constants.

Note If you have set the LINETRANSLATEOPTION_CANCELCALLWAITING bit, also set the LINECALLPARAMFLAGS_SECURE bit in the dwCallParamFlags member of the LINECALLPARAMS structure (passed in to lineMakeCall through the lpCallParams parameter). This action prevents the line device from using dialable digits to suppress call interrupts.

lpTranslateOutput

A pointer to an application-allocated memory area to contain the output of the translation operation, of type LINETRANSLATEOUTPUT. Prior to calling lineTranslateAddress, the application should set the dwTotalSize member of this structure to indicate the amount of memory that is available to TAPI for returning information.

Return Values

Returns zero if the request succeeds or a negative number if an error occurs. Possible return values follow:

- LINEERR_BADDEVICEID
- LINEERR_INVALPOINTER
- LINEERR_INCOMPATIBLEAPIVERSION
- LINEERR_NODRIVER
- LINEERR_INIFILECORRUPT
- LINEERR_NOMEM
- LINEERR_INVALADDRESS
- LINEERR_OPERATIONFAILED
- LINEERR_INVALAPPHANDLE
- LINEERR_RESOURCEUNAVAIL
- LINEERR_INVALCARD
- LINEERR_STRUCTURETOOSMALL
- LINEERR_INVALPARAM

lineTranslateDialog

The lineTranslateDialog function displays an application-modal dialog box that allows the user to change the current location of a phone number that is about to be dialed, adjust location and calling card parameters, and see the effect.

Function Details

```
LONG WINAPI lineTranslateDialog(  HLINEAPP hLineApp,
  DWORD dwDeviceID,
  DWORD dwAPIVersion,
  HWND hwndOwner,
  LPCSTR lpszAddressIn
);
```

Parameters

hLineApp

The application handle that lineInitializeEx returns. If an application has not yet called the lineInitializeEx function, it can set the hLineApp parameter to NULL.

dwDeviceID

The device identifier for the line device upon which the call is intended to be dialed, so variations in dialing procedures on different lines can be applied to the translation process.

dwAPIVersion

Indicates the highest version of TAPI that the application supports (not necessarily the value that lineNegotiateAPIVersion negotiates on the line device that dwDeviceID indicates).

hwndOwner

A handle to a window to which the dialog box is to be attached. Can be a NULL value to indicate that any window that is created during the function should have no owner window.

lpszAddressIn

A pointer to a null-terminated string that contains a phone number that is used, in the lower portion of the dialog box, to show the effect of the user's changes on the location parameters. Ensure that the number is in canonical format; if noncanonical, the phone number portion of the dialog box does not display. You can leave this pointer NULL, in which case the phone number portion of the dialog box does not display. If the lpszAddressIn parameter contains a subaddress or name field, or additional addresses separated from the first address by CR and LF characters, only the first address gets used in the dialog box.

Return Values

Returns zero if request succeeds or a negative number if an error occurs. Possible return values follow:

- LINEERR_BADDEVICEID
- LINEERR_INVALPARAM
- LINEERR_INCOMPATIBLEAPIVERSION
- LINEERR_INVALPOINTER
- LINEERR_INIFILECORRUPT
- LINEERR_NODRIVER
- LINEERR_INUSE
- LINEERR_NOMEM

- LINEERR_INVALADDRESS
- LINEERR_INVALAPPHANDLE
- LINEERR_OPERATIONFAILED

lineUnhold

The lineUnhold function retrieves the specified held call.

Function Details

```
LONG lineUnhold(  HCALL hCall
);
```

Parameters

hCall

The handle to the call to be retrieved. The application must be an owner of this call. The call state of hCall must be onHold, onHoldPendingTransfer, or onHoldPendingConference.

lineUnpark

The lineUnpark function retrieves the call that is parked at the specified address and returns a call handle for it.

Function Details

```
LONG WINAPI lineUnpark(HLINE hLine,
DWORD dwAddressID,
LPHCALL lphCall,
LPCSTR lpszDestAddress
);
```

Parameters

hLine

Handle to the open line device on which a call is to be unparked.

dwAddressID

Address on hLine at which the unpark is to be originated. An address identifier permanently associates with an address; the identifier remains constant across operating system upgrades.

lphCall

Pointer to the location of type HCALL where the handle to the unparked call is returned. This handle is unrelated to any other handle that previously may have been associated with the retrieved call, such as the handle that might have been associated with the call when it was originally parked. The application acts as the initial sole owner of this call.

lpszDestAddress

Pointer to a null-terminated character buffer that contains the address where the call is parked. The address displays in standard dialable address format.

TAPI Line Messages

This section describes the line messages that the Cisco Unified TSP supports. These messages notify the application of asynchronous events such as a new call arriving in the Cisco Unified Communications Manager. The messages get sent to the application by the method that the application specifies in lineInitializeEx

.

Table 2: TAPI Line Messages

TAPI Line Messages
LINE_ADDRESSSTATE, on page 58
LINE_APPNEWCALL, on page 59
LINE_CALLDEVSPECIFIC, on page 60
LINE_CALLINFO, on page 60
LINE_CALLSTATE, on page 61
LINE_CLOSE, on page 65
LINE_CREATE, on page 65
LINE_DEVSPECIFIC, on page 66
LINE_DEVSPECIFICFEATURE, on page 67
LINE_GATHERDIGITS, on page 68
LINE_GENERATE, on page 69
LINE_LINEDEVSTATE, on page 70
LINE_MONITORDIGITS, on page 71
LINE_MONITORTONE, on page 71
LINE_REMOVE, on page 72
LINE_REPLY, on page 73
LINE_REQUEST, on page 74

LINE_ADDRESSSTATE

The LINE_ADDRESSSTATE message gets sent when the status of an address changes on a line that is currently open by the application. The application can invoke lineGetAddressStatus to determine the current status of the address.

Function Details

```
LINE_ADDRESSSTATE
dwDevice = (DWORD) hLine;
dwCallbackInstance = (DWORD) hCallback;
dwParam1 = (DWORD) idAddress;
dwParam2 = (DWORD) AddressState;
dwParam3 = (DWORD) 0;
```

Parameters

dwDevice

A handle to the line device.

dwCallbackInstance

The callback instance supplied when the line is opened.

dwParam1

The address identifier of the address that changed status.

dwParam2

The address state that changed. Can be a combination of these values:

LINEADDRESSSTATE_OTHER

Address-status items other than those that are in the following list changed. The application should check the current address status to determine which items changed.

LINEADDRESSSTATE_DEVSPECIFIC

The device-specific item of the address status changed.

LINEADDRESSSTATE_INUSEZERO

The address changed to idle (it is now in use by zero stations).

LINEADDRESSSTATE_INUSEONE

The address changed from idle or from being used by many bridged stations to being used by just one station.

LINEADDRESSSTATE_INUSEMANY

The monitored or bridged address changed from being used by one station to being used by more than one station.

LINEADDRESSSTATE_NUMCALLS

The number of calls on the address has changed. This change results from events such as a new inbound call, an outbound call on the address, or a call changing its hold status.

LINEADDRESSSTATE_FORWARD

The forwarding status of the address changed, including the number of rings for determining a no-answer condition. The application should check the address status to determine details about the current forwarding status of the address.

LINEADDRESSSTATE_TERMINALS

The terminal settings for the address changed.

LINEADDRESSSTATE_CAPSCHANGE

Indicates that due to configuration changes that the user made, or other circumstances, one or more of the members in the LINEADDRESSCAPS structure for the address changed. The application should use lineGetAddressCaps to read the updated structure. Applications that support API versions earlier than 1.4 receive a LINEDEVSTATE_REINIT message that requires them to shut down and reinitialize their connection to TAPI to obtain the updated information.

dwParam3

This parameter is not used.

LINE_APPNEWCALL

The LINE_APPNEWCALL message informs an application when a new call handle is spontaneously created on its behalf (other than through an API call from the application, in which case the handle would have been returned through a pointer parameter that passed into the function).

Function Details

```
LINE_APPNEWCALL
dwDevice = (DWORD) hLine;
dwCallbackInstance = (DWORD) dwInstanceData;
dwParam1 = (DWORD) dwAddressID;
dwParam2 = (DWORD) hCall;
dwParam3 = (DWORD) dwPrivilege;
```

Parameters

dwDevice

The handle of the application to the line device on which the call was created.

dwCallbackInstance

The callback instance that is supplied when the line belonging to the call is opened.

dwParam1

Identifier of the address on the line on which the call appears.

dwParam2

The handle of the application to the new call.

dwParam3

The privilege of the application to the new call (LINECALLPRIVILEGE_OWNER or LINECALLPRIVILEGE_MONITOR).

LINE_CALLDEVSPECIFIC

The TSPI LINE_CALLDEVSPECIFIC message is sent to notify TAPI about device-specific events that occur on a call. The meaning of the message and the interpretation of the dwParam1 through dwParam3 parameters are device specific.

Function Details

```
LINE_CALLDEVSPECIFIC
htLine = (HTAPILINE) hLineDevice;
htCall = (HTAPICALL) hCallDevice;
dwMsg = (DWORD) LINE_CALLDEVSPECIFIC;
dwParam1 = (DWORD) DeviceData1;
dwParam2 = (DWORD) DeviceData2;
dwParam3 = (DWORD) DeviceData3;
```

Parameters

htLine

The TAPI opaque object handle to the line device.

htCall

The TAPI opaque object handle to the call device.

dwMsg

The value LINE_CALLDEVSPECIFIC.

dwParam1

Device specific

dwParam2

Device specific

dwParam3

Device specific

LINE_CALLINFO

The TAPI LINE_CALLINFO message gets sent when the call information about the specified call has changed. The application can invoke lineGetCallInfo to determine the current call information.

Function Details

```
LINE_CALLINFO
hDevice = (DWORD) hCall;
dwCallbackInstance = (DWORD) hCallback;
dwParam1 = (DWORD) CallInfoState;
dwParam2 = (DWORD) 0;
dwParam3 = (DWORD) 0;
```

Parameters

hDevice

A handle to the call.

dwCallbackInstance

The callback instance that is supplied when the call's line is opened.

dwParam1

The call information item that changed. Can be one or more of the

LINECALLINFOSTATE_constants.

dwParam2

This parameter is not used.

dwParam3

This parameter is not used.

LINE_CALLSTATE

The LINE_CALLSTATE message gets sent when the status of the specified call changes. Typically, several such messages occur during the lifetime of a call. Applications get notified of new incoming calls with this message; the new call exists in the offering state. The application can use the lineGetCallStatus function to retrieve more detailed information about the current status of the call.

Function Details

```
LINE_CALLSTATE
dwDevice = (DWORD) hCall;
dwCallbackInstance = (DWORD) hCallback;
dwParam1 = (DWORD) CallState;
dwParam2 = (DWORD) CallStateDetail;
dwParam3 = (DWORD) CallPrivilege;
```

Parameters

dwDevice

A handle to the call.

dwCallbackInstance

The callback instance that is supplied when the line that belongs to this call is opened.

dwParam1

The new call state. Cisco Unified TSP supports only the following LINECALLSTATE_values:

LINECALLSTATE_IDLE

The call remains idle; no call actually exists.

LINECALLSTATE_OFFERING

The call gets offered to the station, which signals the arrival of a new call. In some environments, a call in the offering state does not automatically alert the user. The switch that instructs the line to ring does alerts; it does not affect any call states.

LINECALLSTATE_ACCEPTED

The system offered the call and it has been accepted. This indicates to other (monitoring) applications that the current owner application claimed responsibility for answering the call. In ISDN, this also indicates that alerting to both parties started.

LINECALLSTATE_CONFERENCED

The call is a member of a conference call and is logically in the connected state.

LINECALLSTATE_DIALTONE

The call receives a dial tone from the switch, which means that the switch is ready to receive a dialed number.

LINECALLSTATE_DIALING

Destination address information (a phone number) is sent to the switch over the call. The lineGenerateDigits does not place the line into the dialing state.

LINECALLSTATE_RINGBACK

The call receives ringback from the called address. Ringback indicates that the call has reached the other station and is being alerted.

LINECALLSTATE_ONHOLDPENDCONF

The call currently remains on hold while it gets added to a conference.

LINECALLSTATE_CONNECTED

The call is established and the connection is made. Information can flow over the call between the originating address and the destination address.

LINECALLSTATE_PROCEEDING

Dialing completes and the call proceeds through the switch or telephone network.

LINECALLSTATE_ONHOLD

The switch keeps the call on hold.

LINECALLSTATE_ONHOLDPENDTRANSFER

The call that is currently on hold awaits transfer to another number.

LINECALLSTATE_DISCONNECTED

The remote party disconnected from the call.

LINECALLSTATE_UNKNOWN

The state of the call is not known. This state may occur due to limitations of the call-progress detection implementation.

Cisco Unified TSP supports two new call states that indicate more information about the call state within the Cisco Unified Communications Manager setup. The standard TAPI call state is set to

LINECALLSTATE_UNKNOWN and the following call states will be ORed with the unknown call state.

#define CLDSMT_CALL_PROGRESSING_STATE 0x0100000

The Progressing state indicates that the call is in progress over the network. The application must negotiate extension version 0x00050001 to receive this call state.

#define CLDSMT_CALL_WAITING_STATE 0x02000000

The waiting state indicates that the REFER request is in progress on Referrer's line and the application should not request any other function on this call. All the requests will result in LINEERR_INVALCALLSTATE. Application has to negotiate extension version 0x00070000 to receive this call state.

#define CLDSMT_CALL_WHISPER_STATE 0x03000000

The whisper state indicates that the Intercom call is connected in one-way audio mode. The Intercom originator cannot issue other function other that to drop the Intercom call. While at destination side, the system allows only Talkback and dropping call. All other requests result in LINEERR_OPERATIONUNAVAIL.

dwParam2

Call-state-dependent information.

- If dwParam1 is LINECALLSTATE_CONNECTED, dwParam2 contains details about the connected mode. This parameter uses the following LINECONNECTEDMODE_constants:

  LINECONNECTEDMODE_ACTIVE

  Call connects at the current station (the current station acts as a participant in the call).

  LINECONNECTEDMODE_INACTIVE

  Call stays active at one or more other stations, but the current station does not participate in the call.

  When a call is disconnected with cause code = DISCONNECTMODE_TEMPFAILURE and the lineState = LINEDEVSTATE_INSERVICE, applications must take care of dropping the call. If the application terminates media for a device, then it is also takes the responsibility to stop the RTP streams for the same call. Cisco Unified TSP will not provide Stop Transmission/Reception events to applications in this scenario. The behavior is exactly the same with IP phones. The user must hang up the disconnected -temp fail call on IP phone to stop the media. The application is also responsible for stopping the RTP streams in case the line goes out of service (LINEDEVSTATE_OUTOFSERVICE) and the call on a line is reported as IDLE.

Note If an application with negotiated extension version 0x00050001 or greater receives device-specific CLDSMT_CALL_PROGRESSING_STATE = 0x01000000 with LINECALLSTATE_UNKNOWN, the cause code is reported as the standard Q931 cause codes in dwParam2.

- If dwParam1 specifies LINECALLSTATE_DIALTONE, dwParam2 contains the details about the dial tone mode. This parameter uses the following LINEDIALTONEMODE_constant:

  LINEDIALTONEMODE_UNAVAIL

  The dial tone mode is unavailable and cannot become known.

- If dwParam1 specifies LINECALLSTATE_OFFERING, dwParam2 contains details about the connected mode. This parameter uses the following LINEOFFERINGMODE_constants:

  LINEOFFERINGMODE_ACTIVE

  The call alerts at the current station (accompanied by LINEDEVSTATE_RINGING messages) and, if an application is set up to automatically answer, it answers. For TAPI versions 1.4 and later, if the call state mode is ZERO, the application assumes that the value is active (which represents the situation on a non-bridged address).

Note The Cisco Unified TSP does not send LINEDEVSTATE_RINGING messages until the call is accepted and moves to the LINECALLSTATE_ACCEPTED state. IP_phones auto-accept calls. CTI ports and CTI route points do not auto-accept calls. Call the lineAccept() function to accept the call at these types of devices.

- If dwParam1 specifies LINECALLSTATE_DISCONNECTED, dwParam2 contains details about the disconnect mode. This parameter uses the following LINEDISCONNECTMODE_constants:

  LINEDISCONNECTMODE_NORMAL

  This specifies a normal disconnect request by the remote party; call terminated normally.

  LINEDISCONNECTMODE_UNKNOWN

  The reason for the disconnect request remains unknown.

  LINEDISCONNECTMODE_REJECT

  The remote user rejected the call.

  LINEDISCONNECTMODE_BUSY

  The station that belongs to the remote user is busy.

  LINEDISCONNECTMODE_NOANSWER

  The station that belongs to the remote user does not answer.

  LINEDISCONNECTMODE_CONGESTION

  This message indicates that the network is congested.

  LINEDISCONNECTMODE_UNAVAIL

  The reason for the disconnect remains unavailable and cannot become known later.

  LINEDISCONNECTMODE_FACCMC

  Indicates that the FAC/CMC feature disconnected the call.

Note LINEDISCONNECTMODE_FACCMC is returned only if the extension version that is negotiated on the line is 0x00050000 (6.0(1)) or higher. If the negotiated extension version is not at least 0x00050000, TSP sets the disconnect mode to LINEDISCONNECTMODE_UNAVAIL.

dwParam3

If zero, this parameter indicates that no change in the privilege occurred for the call to this application.

If nonzero, this parameter specifies the privilege for the application to the call. This occurs in the following situations: (1) The first time that the application receives a handle to this call; (2) When the application is the target of a call hand-off (even if the application already was an owner of the call). This parameter uses the following LINECALLPRIVILEGE_ constants:

LINECALLPRIVILEGE_MONITOR

The application has monitor privilege.

LINECALLPRIVILEGE_OWNER

The application has owner privilege.

LINE_CLOSE

The LINE_CLOSE message gets sent when the specified line device has been forcibly closed. The line device handle or any call handles for calls on the line no longer remains valid after this message is sent.

Function Details

```
LINE_CLOSE
dwDevice = (DWORD) hLine;
dwCallbackInstance = (DWORD) hCallback;
dwParam1 = (DWORD) 0;
dwParam2 = (DWORD) 0;
dwParam3 = (DWORD) 0;
```

Parameters

dwDevice

A handle to the line device that was closed. This handle is no longer valid

dwCallbackInstance

The callback instance that is supplied when the line that belongs to this call is opened.

dwParam1

This parameter is not used.

dwParam2

This parameter is not used.

dwParam3

This parameter is not used.

LINE_CREATE

The LINE_CREATE message informs the application of the creation of a new line device.

Note CTI Manager cluster support, extension mobility, change notification, and user addition to the directory can generate LINE_CREATE events.

Function Details

```
LINE_CREATE
dwDevice = (DWORD) 0;
dwCallbackInstance = (DWORD) 0;
dwParam1 = (DWORD) idDevice;
dwParam2 = (DWORD) 0;
dwParam3 = (DWORD) 0;
```

Parameters

dwDevice

This parameter is not used.

dwCallbackInstance

This parameter is not used.

dwParam1

The dwDeviceID of the newly created device.

dwParam2

This parameter is not used.

dwParam3

This parameter is not used.

LINE_DEVSPECIFIC

The LINE_DEVSPECIFIC message notifies the application about device-specific events that occur on a line, address, or call. The meaning of the message and interpretation of the parameters are device specific.

Function Details

```
LINE_DEVSPECIFIC
dwDevice = (DWORD) hLineOrCall;
dwCallbackInstance = (DWORD) hCallback;
dwParam1 = (DWORD) DeviceSpecific1;
dwParam2 = (DWORD) DeviceSpecific2;
dwParam3 = (DWORD) DeviceSpecific3;
```

Parameters

dwDevice

This device-specific parameter specifies a handle to either a line device or call.

dwCallbackInstance

The callback instance that is supplied when the line opens.

dwParam1

is device specific

dwParam2

is device specific

dwParam3

is device specific

LINE_DEVSPECIFICFEATURE

This line message, added in Cisco Unified Communications Manager Release 6.0, enables a Do Not Disturb (DND) change notification event. Cisco TSP notifies applications by using the LINE_DEVSPECIFICFEATURE message about changes in the DND configuration or status. In order to receive change notifications an application needs to enable DEVSPECIFIC_DONOTDISTURB_CHANGED message flag by using lineDevSpecific SLDST_SET_STATUS_MESSAGES request.

LINE_DEVSPECIFICFEATURE message is sent to notify the application about device-specific events occurring on a line device. In case of a DND change notification, the message includes information about the type of change that occurred on a device and resulted feature status or configured option.

Function Details

```
dwDevice = (DWORD) hLine;
dwCallbackInstance = (DWORD) hCallback;
dwParam1 = (DWORD) PHONEBUTTONFUNCTION_DONOTDISTURB;
dwParam2 = (DWORD) typeOfChange;
dwParam3 = (DWORD) currentValue;

enum CiscoDoNotDisturbOption {
    DoNotDisturbOption_NONE      = 0,
    DoNotDisturbOption_RINGEROFF = 1,
    DoNotDisturbOption_REJECT    = 2
};

enum CiscoDoNotDisturbStatus {
    DoNotDisturbStatus_UNKNOWN  = 0,
    DoNotDisturbStatus_ENABLED  = 1,
    DoNotDisturbStatus_DISABLED = 2
};

enum CiscoDoNotDisturbNotification {
    DoNotDisturb_STATUS_CHANGED  = 1,
    DoNotDisturb_OPTION_CHANGED  = 2
};
```

Parameters

dwDevice

A handle to a line device.

dwCallbackInstance

The callback instance supplied when opening the line.

dwParam1

Always equal to PHONEBUTTONFUNCTION_DONOTDISTURB for the Do-Not-Disturb change notification

dwParam2

Indicates the type of change and can have one of the following enum values:

```
enum CiscoDoNotDisturbNotification {
    DoNotDisturb_STATUS_CHANGED  = 1,
    DoNotDisturb_OPTION_CHANGED  = 2
};
```

dwParam3

If the dwParm2 indicates status change (is equal to DoNotDisturb_STATUS_CHANGED) this parameter can have one of the following enum values:

```
enum CiscoDoNotDisturbStatus {
    DoNotDisturbStatus_UNKNOWN  = 0,
    DoNotDisturbStatus_ENABLED  = 1,
    DoNotDisturbStatus_DISABLED = 2
};
```

If the dwParm2 indicates option change (is equal to DoNotDisturb_OPTION_CHANGED) this parameter can have one of the following enum values:

```
enum CiscoDoNotDisturbOption {
    DoNotDisturbOption_NONE      = 0,
    DoNotDisturbOption_RINGEROFF = 1,
    DoNotDisturbOption_REJECT    = 2
};
```

LINE_GATHERDIGITS

The TAPI LINE_GATHERDIGITS message is sent when the current buffered digit-gathering request is terminated or canceled. You can examine the digit buffer after the application receives this message.

Function Details

```
LINE_GATHERDIGITS
hDevice = (DWORD) hCall;
dwCallbackInstance = (DWORD) hCallback;
dwParam1 = (DWORD) GatherTermination;
dwParam2 = (DWORD) 0;
dwParam3 = (DWORD) 0;
```

Parameters

hDevice

A handle to the call.

dwCallbackInstance

The callback instance that is supplied when the line opens.

dwParam1

The reason why digit gathering terminated. This parameter must be one and only one of the LINEGATHERTERM_constants.

dwParam2

Unused.

dwParam3

The tick count (number of milliseconds since Windows started) at which the digit gathering completes. For TAPI versions earlier than 2.0, this parameter is not used.

LINE_GENERATE

The TAPI LINE_GENERATE message notifies the application that the current digit or tone generation terminated. Only one such generation request can be in progress an a given call at any time. This message also gets sent when digit or tone generation is canceled.

Function Details

```
LINE_GENERATE
hDevice = (DWORD) hCall;
dwCallbackInstance = (DWORD) hCallback;
dwParam1 = (DWORD) GenerateTermination;
dwParam2 = (DWORD) 0;
dwParam3 = (DWORD) 0;
```

Parameters

hDevice

A handle to the call.

dwCallbackInstance

The callback instance that is supplied when the line opens.

dwParam1

The reason that digit or tone generation terminates. This parameter must be the only one of the LINEGENERATETERM_constants.

dwParam2

This parameter is not used.

dwParam3

The tick count (number of milliseconds since Windows started) at which the digit or tone generation completes. For API versions earlier than 2.0, this parameter is not used.

LINE_LINEDEVSTATE

The TAPI LINE_LINEDEVSTATE message gets sent when the state of a line device changes. The application can invoke lineGetLineDevStatus to determine the new status of the line.

Function Details

```
LINE_LINEDEVSTATE
hDevice = (DWORD) hLine;
dwCallbackInstance = (DWORD) hCallback;
dwParam1 = (DWORD) DeviceState;
dwParam2 = (DWORD) DeviceStateDetail1;
dwParam3 = (DWORD) DeviceStateDetail2;
```

Parameters

hDevice

A handle to the line device. This parameter is NULL when dwParam1 is LINEDEVSTATE_REINIT.

dwCallbackInstance

The callback instance that is supplied when the line is opened. If the dwParam1 parameter is LINEDEVSTATE_REINIT, the dwCallbackInstance parameter is not valid and is set to zero.

dwParam1

The line device status item that changed. The parameter can be one or more of the LINEDEVSTATE_constants.

LINEDEVSTATE_OUTOFSERVICE

Indicates the line device unregisters as it enters Energywise DeepSleep/PowersavePlus mode

dwParam2

The interpretation of this parameter depends on the value of dwParam1. If dwParam1 is LINEDEVSTATE_RINGING, dwParam2 contains the ring mode with which the switch instructs the line to ring. Valid ring modes include numbers in the range one to dwNumRingModes, where dwNumRingModes specifies a line device capability.

If dwParam1 is LINEDEVSTATE_REINIT, and TAPI issued the message as a result of translation of a new API message into a REINIT message, dwParam2 contains the dwMsg parameter of the original message (for example, LINE_CREATE or LINE_LINEDEVSTATE). If dwParam2 is zero, this indicates that the REINIT message represents a real REINIT message that requires the application to call lineShutdown at its earliest convenience.

If dw Param1 is LINEDEVSTATE_OUTOFSERVICE, dwParam2 contains the reason EnergyWisePowerSavePlus when the line device unregisters as it enters EnergywiseDeepSleep.

dwParam3

The interpretation of this parameter depends on the value of dwParam1. If dwParam1 is LINEDEVSTATE_RINGING, dwParam3 contains the ring count for this ring event. The ring count starts at zero.

If dwParam1 is LINEDEVSTATE_REINIT, and TAPI issued the message as a result of translation of a new API message into a REINIT message, dwParam3 contains the dwParam1 parameter of the original message (for example, LINEDEVSTATE_TRANSLATECHANGE or some other

LINEDEVSTATE_value, if dwParam2 is LINE_LINEDEVSTATE, or the new device identifier, if dwParam2 is LINE_CREATE).

LINE_MONITORDIGITS

The LINE_MONITORDIGITS message gets sent when a digit is detected. The lineMonitorDigits function controls the sending of this message.

Function Details

```
LINE_MONITORDIGITS
dwDevice = (DWORD) hCall;
dwCallbackInstance = (DWORD) hCallback;
dwParam1 = (DWORD) Digit;
dwParam2 = (DWORD) DigitMode;
dwParam3 = (DWORD) 0;
```

Parameters

dwDevice

A handle to the call.

dwCallbackInstance

The callback instance that is supplied when the line for this call is opened.

dwParam1

The low-order byte contains the last digit that is received in ASCII.

dwParam2

The digit mode that was detected. This parameter must be one and only one of the following LINEDIGITMODE_constant:

LINEDIGITMODE_DTMF

Detect digits as DTMF tones. Valid digits for DTMF include ‘0’ through ‘9’, ‘*’, and ‘#’.

dwParam3

The “tick count” (number of milliseconds after Windows started) at which the specified digit was detected. For API versions earlier than 2.0, this parameter is not used.

LINE_MONITORTONE

The LINE_MONITORTONE message gets sent when a tone is detected. The lineMonitorTones function controls the sending of this message.

Note Cisco Unified TSP supports only silent detection through LINE_MONITORTONE.

Function Details

```
LINE_MONITORTONE
dwDevice = (DWORD) hCall;
dwCallbackInstance = (DWORD) hCallback;
dwParam1 = (DWORD) dwAppSpecific;
dwParam2 = (DWORD) 0;
dwParam3 = (DWORD) tick count;
```

Parameters

dwDevice

A handle to the call.

dwCallbackInstance

The callback instance supplied when the line opens for this call.

dwParam1

The application-specific dwAppSpecific member of the LINE_MONITORTONE structure for the tone that was detected.

dwParam2

This parameter is not used.

dwParam3

The "tick count" (number of milliseconds after Windows started) at which the specified digit was detected.

LINE_REMOVE

The LINE_REMOVE message informs an application of the removal (deletion from the system) of a line device. Generally, this parameter is not used for temporary removals, such as extraction of PCMCIA devices, but only for permanent removals in which, the service provider would no longer report the device, if TAPI were reinitialized.

Note CTI Manager cluster support, extension mobility, change notification, and user deletion from the directory can generate LINE_REMOVE events.

Function Details

```
LINE_REMOVE
dwDevice = (DWORD) 0;
dwCallbackInstance = (DWORD) 0;
dwParam1 = (DWORD) dwDeviceID;
dwParam2 = (DWORD) 0;
dwParam3 = (DWORD) 0;
```

Parameters

dwDevice

Reserved. Set to zero.

dwCallbackInstance

Reserved. Set to zero.

dwParam1

Identifier of the line device that was removed.

dwParam2

Reserved. Set to zero.

dwParam3

Reserved. Set to zero.

LINE_REPLY

The LINE_REPLY message reports the results of function calls that completed asynchronously.

Function Details

```
LINE_REPLY
dwDevice = (DWORD) 0;
dwCallbackInstance = (DWORD) hCallback;
dwParam1 = (DWORD) idRequest;
dwParam2 = (DWORD) Status;
dwParam3 = (DWORD) 0;
```

Parameters

dwDevice

This parameter is not used.

dwCallbackInstance

Returns the callback instance for this application.

dwParam1

The request identifier for which this is the reply.

dwParam2

The success or error indication. The application should cast this parameter into a long integer:

- Zero indicates success.
- A negative number indicates an error.

dwParam3

This parameter is not used.

LINE_REQUEST

The TAPI LINE_REQUEST message reports the arrival of a new request from another application.

Function Details

```
LINE_REQUEST
hDevice = (DWORD) 0;
dwCallbackInstance = (DWORD) hRegistration;
dwParam1 = (DWORD) RequestMode;
dwParam2 = (DWORD) RequestModeDetail1;
dwParam3 = (DWORD) RequestModeDetail2;
```

Parameters

hDevice

This parameter is not used.

dwCallbackInstance

The registration instance of the application that is specified on lineRegisterRequestRecipient.

dwParam1

The request mode of the newly pending request. This parameter uses the LINEREQUESTMODE_constants.

dwParam2

If dwParam1 is set to LINEREQUESTMODE_DROP, dwParam2 contains the hWnd of the application that requests the drop. Otherwise, dwParam2 is not used.

dwParam3

If dwParam1 is set to LINEREQUESTMODE_DROP, the low-order word of dwParam3 contains the wRequestID as specified by the application requesting the drop. Otherwise, dwParam3 is not used.

TAPI Line Device Structures

The following table lists the TAPI line device structures that the Cisco Unified TSP supports. This section lists the possible values for the structure members as set by the TSP, and provides a cross reference to the functions that use them. If the value of a structure member is device, line, or call specific, the system notes the value for each condition.

Table 3: TAPI Line Device Structures

TAPI Line Device Structures
LINEADDRESSCAPS, on page 75
LINEADDRESSSTATUS, on page 86
LINEAPPINFO, on page 87
LINECALLINFO, on page 89

TAPI Line Device Structures
LINECALLLIST, on page 97
LINECALLPARAMS, on page 98
LINECALLSTATUS, on page 100
LINECARDENTRY, on page 106
LINECOUNTRYENTRY, on page 108
LINECOUNTRYLIST, on page 109
LINEDEVCAPS, on page 110
LINEDEVSTATUS, on page 115
LINEEXTENSIONID, on page 117
LINEFORWARD, on page 117
LINEFORWARDLIST, on page 121
LINEGENERATETONE, on page 121
LINEINITIALIZEEXPARAMS, on page 122
LINELOCATIONENTRY, on page 123
LINEMESSAGE, on page 125
LINEMONITORTONE, on page 126
LINEPROVIDERENTRY, on page 127
LINEPROVIDERLIST, on page 127
LINEREQMAKECALL, on page 128
LINETRANSLATECAPS, on page 129
LINETRANSLATEOUTPUT, on page 130

LINEADDRESSCAPS

Members

Members	Values
dwLineDeviceID	For All Devices:The device identifier of the line device with which this address is associated.

Members	Values
dwAddressSizedwAddressOffset	For All Devices:The size, in bytes, of the variably sized address field and the offset, in bytes, from the beginning of this data structure
dwDevSpecificSize dwDevSpecificOffset	For All Devices:0
dwAddressSharing	For All Devices:0
dwAddressStates	For All Devices (except Park DNs):LINEADDRESSSTATE_FORWARD
	For Park DNs:0

Members	Values
dwCallInfoStates	For All Devices (except Park DNs):LINECALLINFOSTATE_CALLEDID LINECALLINFOSTATE_CALLERID LINECALLINFOSTATE_CALLID LINECALLINFOSTATE_CONNECTEDID LINECALLINFOSTATE_MEDIAMODE LINECALLINFOSTATE_MONITORMODES LINECALLINFOSTATE_NUMMONITORS LINECALLINFOSTATE_NUMOWNERDECR LINECALLINFOSTATE_NUMOWNERINCR LINECALLINFOSTATE_ORIGIN LINECALLINFOSTATE_REASON LINECALLINFOSTATE_REDIRECTINGID LINECALLINFOSTATE_REDIRECTIONID
	For Park DNs: LINECALLINFOSTATE_CALLEDID LINECALLINFOSTATE_CALLERID LINECALLINFOSTATE_CALLID LINECALLINFOSTATE_CONNECTEDID LINECALLINFOSTATE_NUMMONITORS LINECALLINFOSTATE_NUMOWNERDECR LINECALLINFOSTATE_NUMOWNERINCR LINECALLINFOSTATE_ORIGIN LINECALLINFOSTATE_REASON LINECALLINFOSTATE_REDIRECTINGID LINECALLINFOSTATE_REDIRECTIONID
dwCallerIDFlags	For All Devices:LINECALLPARTYID_ADDRESS LINECALLPARTYID_NAME LINECALLPARTYID_UNKNOWN LINECALLPARTYID_BLOCKED

Members	Values
dwCalledIDFlags	For All Devices: LINECALLPARTYID_ADDRESS LINECALLPARTYID_NAME LINECALLPARTYID_UNKNOWN
dwConnectedIDFlags	For All Devices: LINECALLPARTYID_ADDRESS LINECALLPARTYID_NAME LINECALLPARTYID_UNKNOWN LINECALLPARTYID_BLOCKED
dwRedirectionIDFlags	For All Devices: LINECALLPARTYID_ADDRESS LINECALLPARTYID_NAME LINECALLPARTYID_UNKNOWN LINECALLPARTYID_BLOCKED
dwRedirectingIDFlags	For All Devices:LINECALLPARTYID_ADDRESS LINECALLPARTYID_NAME LINECALLPARTYID_UNKNOWN

Members	Values
dwCallStates	For IP Phones and CTI Ports:LINECALLSTATE_ACCEPTED LINECALLSTATE_CONFERENCED LINECALLSTATE_CONNECTED LINECALLSTATE_DIALING LINECALLSTATE_DIALTONE LINECALLSTATE_DISCONNECTED LINECALLSTATE_IDLE LINECALLSTATE_OFFERING LINECALLSTATE_ONHOLD LINECALLSTATE_ONHOLDPENDCONF LINECALLSTATE_ONHOLDPENDTRANSFER LINECALLSTATE_PROCEEDING LINECALLSTATE_RINGBACK LINECALLSTATE_UNKNOWN
	For CTI Route Points (without media):LINECALLSTATE_ACCEPTED LINECALLSTATE_DISCONNECTED LINECALLSTATE_IDLE LINECALLSTATE_OFFERING LINECALLSTATE_UNKNOWN For CTI Route Points (with media):LINECALLSTATE_ACCEPTED LINECALLSTATE_CONNECTED LINECALLSTATE_DISCONNECTED LINECALLSTATE_ONHOLD LINECALLSTATE_IDLE LINECALLSTATE_OFFERING LINECALLSTATE_UNKNOWN

Members	Values
	For Park DNs:LINECALLSTATE_ACCEPTED LINECALLSTATE_CONFERENCED LINECALLSTATE_CONNECTED LINECALLSTATE_DISCONNECTED LINECALLSTATE_IDLE LINECALLSTATE_OFFERING LINECALLSTATE_ONHOLD LINECALLSTATE_UNKNOWN
dwDialToneModes	For IP Phones and CTI Ports:LINEDIALTONEMODE_UNAVAIL
	For CTI Route Points and Park DNs:0
dwBusyModes	For All Devices:0
dwSpecialInfo	For All Devices:0
dwDisconnectModes	For All Devices:LINEDISCONNECTMODE_BADDADDRESS LINEDISCONNECTMODE_BUSY LINEDISCONNECTMODE_CONGESTION LINEDISCONNECTMODE_FORWARDED LINEDISCONNECTMODE_NOANSWER LINEDISCONNECTMODE_NORMAL LINEDISCONNECTMODE_REJECT LINEDISCONNECTMODE_TEMPFAILURE LINEDISCONNECTMODE_UNREACHABLE LINEDISCONNECTMODE_FACCMC (if negotiated extension version is 0x00050000 or greater)
dwMaxNumActiveCalls	For IP Phones, CTI Ports, and Park DNs:1
	For CTI Route Points (without media): 0 For CTI Route Points (with media):Cisco Unified Communications Manager Administration configuration

Members	Values
dwMaxNumOnHoldCalls	For IP Phones, CTI Ports:200
	For CTI Route Points:0 For CTI Route Points (with media):Cisco Unified Communications Manager Administration configuration (same configuration as dwMaxNumActiveCalls)
	For Park DNs: 1
dwMaxNumOnHoldPendingCalls	For IP Phones and CTI Ports:1
	For CTI Route Points and Park DNs:0
dwMaxNumConference	For IP Phones, CTI Ports, and Park DNs:16
	For CTI Route Points:0
dwMaxNumTransConf	For All Devices:0
dwAddrCapFlags	For IP Phones:LINEADDRCAPFLAGS_CONFERENCEHELD LINEADDRCAPFLAGS_DIALED LINEADDRCAPFLAGS_FWDSTATUSVALID LINEADDRCAPFLAGS_PARTIALDIAL LINEADDRCAPFLAGS_TRANSFERHELD
	For CTI Ports:LINEADDRCAPFLAGS_CONFERENCEHELD LINEADDRCAPFLAGS_DIALED LINEADDRCAPFLAGS_ACCEPTTOALERT LINEADDRCAPFLAGS_FWDSTATUSVALID LINEADDRCAPFLAGS_PARTIALDIAL LINEADDRCAPFLAGS_TRANSFERHELD
	For CTI Route Points:LINEADDRCAPFLAGS_ACCEPTTOALERT LINEADDRCAPFLAGS_FWDSTATUSVALID LINEADDRCAPFLAGS_ROUTEPOINT
	For Park DNs:LINEADDRCAPFLAGS_NOEXTERNALCALLS LINEADDRCAPFLAGS_NOINTERNALCALLS

Members	Values
dwCallFeatures	For IP Phones (except VG248 and ATA186) and CTI Ports:LINECALLFEATURE_ACCEPT LINECALLFEATURE_ADDTOCONF LINECALLFEATURE_ANSWER LINECALLFEATURE_BLINDTRANSFER LINECALLFEATURE_COMPLETETRANSF LINECALLFEATURE_DIAL LINECALLFEATURE_DROP LINECALLFEATURE_GATHERDIGITS LINECALLFEATURE_GENERATEDIGITS LINECALLFEATURE_GENERATETONE LINECALLFEATURE_HOLD LINECALLFEATURE_MONITORDIGITS LINECALLFEATURE_MONITORTONES LINECALLFEATURE_PARK LINECALLFEATURE_PREPAREADDTOCONF LINECALLFEATURE_REDIRECT LINECALLFEATURE_SETUPCONF LINECALLFEATURE_SETUPTRANSFER LINECALLFEATURE_UNHOLD LINECALLFEATURE_UNPARK

Members	Values
	For VG248 and ATA186 Devices:LINECALLFEATURE_ACCEPT LINECALLFEATURE_ADDTOCONF LINECALLFEATURE_BLINDTRANSFER LINECALLFEATURE_COMPLETETRANSF LINECALLFEATURE_DIAL LINECALLFEATURE_DROP LINECALLFEATURE_GATHERDIGITS LINECALLFEATURE_GENERATEDIGITS LINECALLFEATURE_GENERATETONE LINECALLFEATURE_HOLD LINECALLFEATURE_MONITORDIGITS LINECALLFEATURE_MONITORTONES LINECALLFEATURE_PARK LINECALLFEATURE_PREPAREADDTOCONF LINECALLFEATURE_REDIRECT LINECALLFEATURE_SETUPCONF LINECALLFEATURE_SETUPTRANSFER LINECALLFEATURE_UNHOLD INECALLFEATURE_UNPARK

Members	Values
dwCallFeatures (continued)	For CTI Route Points (without media):LINECALLFEATURE_ACCEPT LINECALLFEATURE_DROP LINECALLFEATURE_REDIRECT For CTI Route Points (with media):LINECALLFEATURE_ACCEPT LINECALLFEATURE_ANSWER LINECALLFEATURE_DIAL LINECALLFEATURE_DROP LINECALLFEATURE_GATHERDIGITS LINECALLFEATURE_GENERATEDIGITS LINECALLFEATURE_GENERATETONE LINECALLFEATURE_HOLD LINECALLFEATURE_MONITORDIGITS LINECALLFEATURE_MONITORTONES LINECALLFEATURE_REDIRECT LINECALLFEATURE_UNHOLD
	For Park DNs:0
dwRemoveFromConfCaps	For All Devices:0
dwRemoveFromConfState	For All Devices:0
dwTransferModes	For IP Phones and CTI Ports:LINETRANSFERMODE_TRANSFER LINETRANSFERMODE_CONFERENCE
	For CTI Route Points and Park DNs:0
dwParkModes	For IP Phones and CTI Ports:LINEPARKMODE_NONDIRECTED
	For CTI Route Points and Park DNs:0
dwForwardModes	For All Devices (except ParkDNs):LINEFORWARDMODE_UNCOND
	For Park DNs:0
dwMaxForwardEntries	For All Devices (except ParkDNs):1
	For Park DNs:0

Members	Values
dwMaxSpecificEntries	For All Devices:0
dwMinFwdNumRings	For All Devices:0
dwMaxFwdNumRings	For All Devices:0
dwMaxCallCompletions	For All Devices:0
dwCallCompletionConds	For All Devices:0
dwCallCompletionModes	For All Devices:0
dwNumCompletionMessages	For All Devices:0
dwCompletionMsgTextEntrySize	For All Devices:0
dwCompletionMsgTextSize dwCompletionMsgTextOffset	For All Devices:0
dwAddressFeatures	For IP Phones and CTI Ports:LINEADDRFEATURE_FORWARD LINEADDRFEATURE_FORWARDFWD LINEADDRFEATURE_MAKECALL
	For CTI Route Points:LINEADDRFEATURE_FORWARD LINEADDRFEATURE_FORWARDFWD
	For Park DNs:0
dwPredictiveAutoTransferStates	For All Devices:0
dwNumCallTreatments	For All Devices:0
dwCallTreatmentListSizedwCallTreatmentListOffset	For All Devices:0
dwDeviceClassesSize dwDeviceClassesOffset	For All Devices (except Park DNs): "tapi/line" "tapi/phone" "wave/in" "wave/out"
	For Park DNs : "tapi/line"
dwMaxCallDataSize	For All Devices:0

Members	Values
dwCallFeatures2	For IP Phones and CTI Ports:LINECALLFEATURE2_TRANSFERNORM LINECALLFEATURE2_TRANSFERCONF
	For CTI Route Points and Park DNs:0
dwMaxNoAnswerTimeout	For IP Phones and CTI Ports: 4294967295 (0xFFFFFFFF)
	For CTI Route Points and Park DNs:0
dwConnectedModes	For IP Phones, CTI Ports LINECONNECTEDMODE_ACTIVE LINECONNECTEDMODE_INACTIVE
	For Park DNs: LINECONNECTEDMODE_ACTIVE
	For CTI Route Points (without media):0 For CTI Route Points (with media)LINECONNECTEDMODE_ACTIVE
dwOfferingModes	For All Devices: LINEOFFERINGMODE_ACTIVE
dwAvailableMediaModes	For All Devices:0

LINEADDRESSSTATUS

Members

Members	Values
dwNumInUse	For All Devices:1
dwNumActiveCalls	For All Devices: The number of calls on the address that are in call states other than idle, onhold, onholdpendingtransfer, and onholdpendingconference.
dwNumOnHoldCalls	For All Devices: The number of calls on the address in the onhold state.

Members	Values
dwNumOnHoldPendCalls	For All Devices: The number of calls on the address in the onholdpendingtransfer or the onholdpendingconference state.
dwAddressFeatures	For IP Phones and CTI Ports:LINEADDRFEATURE_FORWARD LINEADDRFEATURE_FORWARDFWD LINEADDRFEATURE_MAKECALL
	For CTI Route Points:LINEADDRFEATURE_FORWARD LINEADDRFEATURE_FORWARDFWD
	For Park DNs:0
dwNumRingsNoAnswer	For All Devices:0
dwForwardNumEntries	For All Devices (except Park DNs): The number of entries in the array to which dwForwardSize and dwForwardOffset refer.
	For Park DNs:0
dwForwardSize dwForwardOffset	For All Devices (except Park DNs): The size, in bytes, and the offset, in bytes, from the beginning of this data structure of the variably sized field that describes the address forwarding information. This information appears as an array of dwForwardNumEntries elements, of type LINEFORWARD. Consider the offsets of the addresses in the array relative to the beginning of the LINEADDRESSSTATUS structure. The offsets dwCallerAddressOffset and dwDestAddressOffset in the variably sized field of type LINEFORWARD to which dwForwardSize and dwForwardOffset point are relative to the beginning of the LINEADDRESSSTATUS data structure (the root container).
	For Park DNs:0
dwTerminalModesSizedwTerminalModesOffset	For All Devices:0
dwDevSpecificSizedwDevSpecificOffset	For All Devices:0

LINEAPPINFO

The LINEAPPINFO structure contains information about the application that is currently running. The LINEDEVSTATUS structure can contain an array of LINEAPPINFO structures.

Structure Details

```
typedef struct lineappinfo_tag {
  DWORD  dwMachineNameSize;
  DWORD  dwMachineNameOffset;
  DWORD  dwUserNameSize;
  DWORD  dwUserNameOffset;
  DWORD  dwModuleFilenameSize;
  DWORD  dwModuleFilenameOffset;
  DWORD  dwFriendlyNameSize;
  DWORD  dwFriendlyNameOffset;
  DWORD  dwMediaModes;
  DWORD  dwAddressID;
} LINEAPPINFO, *LPLINEAPPINFO;
```

Members

Members	Values
dwMachineNameSize dwMachineNameOffset	Size (bytes) and offset from beginning of LINEDEVSTATUS of a string that specifies the name of the computer on which the application is executing.
dwUserNameSize dwUserNameOffset	Size (bytes) and offset from beginning of LINEDEVSTATUS of a string that specifies the user name under whose account the application is running.
dwModuleFilenameSize dwModuleFilenameOffset	Size (bytes) and offset from beginning of LINEDEVSTATUS of a string that specifies the module filename of the application. You can use this string in a call to lineHandoff to perform a directed handoff to the application.
dwFriendlyNameSize dwFriendlyNameOffset	Size (bytes) and offset from beginning of LINEDEVSTATUS of the string that the application provides to lineInitialize or lineInitializeEx, which should be used in any display of applications to the user.
dwMediaModes	The media types for which the application has requested ownership of new calls; zero if the line dwPrivileges did not include LINECALLPRIVILEGE_OWNER when it opened.
dwAddressID	If the line handle that was opened by using LINEOPENOPTION_SINGLEADDRESS contains the address identifier that is specified, set to 0xFFFFFFFF if the single address option was not used. An address identifier permanently associates with an address; the identifier remains constant across operating system upgrades.

LINECALLINFO

Members

Members	Values
hLine	For All Devices: The handle for the line device with which this call is associated.
dwLineDeviceID	For All Devices: The device identifier of the line device with which this call is associated.
dwAddressID	For All Devices:0
dwBearerMode	For All Devices: LINEBEARERMODE_SPEECH LINEBEARERMODE_VOICE
dwRate	For All Devices:0
dwMediaMode	For IP Phones and Park DNs:LINEMEDIAMODE_INTERACTIVEVOICE
	For CTI Ports and CTI Route Points:LINEMEDIAMODE_AUTOMATEDVOICE LINEMEDIAMODE_INTERACTIVEVOICE
dwAppSpecific	For All Devices: Not interpreted by the API implementation and service provider. Any owner application of this call can set it with the lineSetAppSpecific function.
dwCallID	For All Devices: In some telephony environments, the switch or service provider can assign a unique identifier to each call. This allows the call to be tracked across transfers, forwards, or other events. The domain of these call IDs and their scope is service provider-defined. The dwCallID member makes this unique identifier available to the applications. The Cisco Unified TSP uses dwCallID to store the “GlobalCallID” of the call. The “GlobalCallID” represents a unique identifier that allows applications to identify all call handles that are related to a call.
dwRelatedCallID	For All Devices:0
dwCallParamFlags	For All Devices:0

Members	Values
dwCallStates	For IP Phones and CTI Ports:LINECALLSTATE_ACCEPTED LINECALLSTATE_CONFERENCED LINECALLSTATE_CONNECTED LINECALLSTATE_DIALING LINECALLSTATE_DIALTONE LINECALLSTATE_DISCONNECTED LINECALLSTATE_IDLE LINECALLSTATE_OFFERING LINECALLSTATE_ONHOLD LINECALLSTATE_ONHOLDPENDCONF LINECALLSTATE_ONHOLDPENDTRANSFER LINECALLSTATE_PROCEEDING LINECALLSTATE_RINGBACK LINECALLSTATE_UNKNOWN

Members	Values
dwCallStates (continued)	For CTI Route Points (without media):LINECALLSTATE_ACCEPTED LINECALLSTATE_DISCONNECTED LINECALLSTATE_IDLE LINECALLSTATE_OFFERING LINECALLSTATE_UNKNOWN For CTI Route Points (with media):LINECALLSTATE_ACCEPTED LINECALLSTATE_BUSY LINECALLSTATE_CONNECTED LINECALLSTATE_DIALING LINECALLSTATE_DIALTONE LINECALLSTATE_DISCONNECTED LINECALLSTATE_IDLE LINECALLSTATE_OFFERING LINECALLSTATE_ONHOLD LINECALLSTATE_PROCEEDING LINECALLSTATE_RINGBACK LINECALLSTATE_UNKNOWN
	For Park DNs:LINECALLSTATE_ACCEPTED LINECALLSTATE_CONFERENCED LINECALLSTATE_CONNECTED LINECALLSTATE_DISCONNECTED LINECALLSTATE_IDLE LINECALLSTATE_OFFERING LINECALLSTATE_ONHOLD LINECALLSTATE_UNKNOWN
dwMonitorDigitModes	For IP Phones, CTI Ports, and CTI Route Points (with media): LINEDIGITMODE_DTMF
	For CTI Route Points and Park DNs:0

Members	Values
dwMonitorMediaModes	For IP Phones and Park DNs:LINEMEDIAMODE_INTERACTIVEVOICE
	For CTI Ports and CTI Route Points:LINEMEDIAMODE_AUTOMATEDVOICE LINEMEDIAMODE_INTERACTIVEVOICE
DialParams	For All Devices:0
dwOrigin	For All Devices:LINECALLORIGIN_CONFERENCE LINECALLORIGIN_EXTERNAL LINECALLORIGIN_INTERNAL LINECALLORIGIN_OUTBOUND LINECALLORIGIN_UNAVAIL LINECALLORIGIN_UNKNOWN
dwReason	For All Devices: LINECALLREASON_DIRECT LINECALLREASON_FWDBUSY LINECALLREASON_FWDNOANSWER LINECALLREASON_FWDUNCOND LINECALLREASON_PARKED LINECALLREASON_PICKUP LINECALLREASON_REDIRECT LINECALLREASON_REMINDER LINECALLREASON_TRANSFER LINECALLREASON_UNKNOWN LINECALLREASON_UNPARK
dwCompletionID	For All Devices:0
dwNumOwners	For All Devices: The number of application modules with different call handles with owner privilege for the call.
dwNumMonitors	For All Devices: The number of application modules with different call handles with monitor privilege for the call.
dwCountryCode	For All Devices:0
dwTrunk	For All Devices:0xFFFFFFFF

Members	Values
dwCallerIDFlags	For All Devices:LINECALLPARTYID_ADDRESS LINECALLPARTYID_NAME LINECALLPARTYID_UNKNOWN LINECALLPARTYID_BLOCKED
dwCallerIDSize dwCallerIDOffset	For All Devices: The size, in bytes, of the variably sized field that contains the caller party ID number information and the offset, in bytes, from the beginning of this data structure.
dwCallerIDNameSize dwCallerIDNameOffset	For All Devices: The size, in bytes, of the variably sized field that contains the caller party ID name information and the offset, in bytes, from the beginning of this data structure.
dwCalledIDFlags	For All Devices: LINECALLPARTYID_ADDRESS LINECALLPARTYID_NAME LINECALLPARTYID_UNKNOWN
dwCalledIDSize dwCalledIDOffset	For All Devices: The size, in bytes, of the variably sized field that contains the called-party ID number information and the offset, in bytes, from the beginning of this data structure.
dwCalledIDNameSize dwCalledIDNameOffset	For All Devices: The size, in bytes, of the variably sized field that contains the called-party ID name information and the offset, in bytes, from the beginning of this data structure.
dwConnectedIDFlags	For All Devices: LINECALLPARTYID_ADDRESS LINECALLPARTYID_NAME LINECALLPARTYID_UNKNOWN LINECALLPARTYID_BLOCKED
dwConnectedIDSize dwConnectedIDOffset	For All Devices: The size, in bytes, of the variably sized field that contains the connected party identifier number information and the offset, in bytes, from the beginning of this data structure.

Members	Values
dwConnectedIDNameSize dwConnectedIDNameOffset	For All Devices: The size, in bytes, of the variably sized field that contains the connected party identifier name information and the offset, in bytes, from the beginning of this data structure.
dwRedirectionIDFlags	For All Devices: LINECALLPARTYID_ADDRESS LINECALLPARTYID_NAME LINECALLPARTYID_UNKNOWN LINECALLPARTYID_BLOCKED
dwRedirectionIDSize dwRedirectionIDOffset	For All Devices: The size, in bytes, of the variably sized field that contains the redirection party identifier number information and the offset, in bytes, from the beginning of this data structure.
dwRedirectionIDNameSize dwRedirectionIDNameOffset	For All Devices: The size, in bytes, of the variably sized field that contains the redirection party identifier name information and the offset, in bytes, from the beginning of this data structure.
dwRedirectingIDFlags	For All Devices: LINECALLPARTYID_ADDRESS LINECALLPARTYID_NAME LINECALLPARTYID_UNKNOWN
dwRedirectingIDSize dwRedirectingIDOffset	For All Devices: The size, in bytes, of the variably sized field that contains the redirecting party identifier number information and the offset, in bytes, from the beginning of this data structure.
dwRedirectingIDNameSize dwRedirectingIDNameOffset	For All Devices: The size, in bytes, of the variably sized field that contains the redirecting party identifier name information and the offset, in bytes, from the beginning of this data structure.
dwAppNameSize dwAppNameOffset	For All Devices: The size, in bytes, and the offset, in bytes, from the beginning of this data structure of the variably sized field that holds the user-friendly application name of the application that first originated, accepted, or answered the call. This specifies the name that an application can specify in lineInitializeEx. If the application specifies no such name, the application module filename gets used instead.

Members	Values
dwDisplayableAddressSize dwDisplayableAddressOffset	For All Devices: 0
dwCalledPartySize dwCalledPartyOffset	For All Devices: 0
dwCommentSize dwCommentOffset	For All Devices: 0
dwDisplaySize dwDisplayOffset	For All Devices: 0
dwUserUserInfoSize dwUserUserInfoOffset	For All Devices: 0
dwHighLevelCompSize dwHighLevelCompOffset	For All Devices: 0
dwLowLevelCompSize dwLowLevelCompOffset	For All Devices: 0
dwChargingInfoSize dwChargingInfoOffset	For All Devices: 0
dwTerminalModesSize dwTerminalModesOffset	For All Devices: 0

Members	Values
dwDevSpecificSize dwDevSpecificOffset	For All Devices: If dwExtVersion > = 0x00060000 (6.0), this field will point to TSP_Unicode_Party_Names structure, If dwExtVersion > = 0x00070000 (7.0), this field will also point to a common structure that has a pointer to SRTP structure, DSCPValueForAudioCalls value, and Partition information. The LINECALLINFO defines the structure. The ExtendedCallInfo structure contains ExtendedCallReason that represents the last feature-related reason that caused a change in the callinfo/callstatus for this call. The ExtendedCallInfo will also provide SIP URL information for all call parties. If dwExtVersion > = 0x00080000 (8.0), this field will also point to common structure which has pointer to CallSecurityStatus structure. For IP Phones: If dwExtVersion > = 0x00080000 (8.0), this field will also point to common structure that has pointer to CallAtributeInfo and CCMCallID structure. The structures are defined below. If dwExtVersion > = 0x00080000 (8.0), this field will also point to common structure which has pointer to CallSecurityStatus structure.
	CallAttributeType: This field holds information about DN.Partition.DeviceName for regular calls, monitoring calls, monitored calls, and recording calls. PartyDNOffset, PartyDNSize, provides the size, in bytes, of the variably sized field that contains the Monitoring/Monitored/Recorder party DN information and the offset, in bytes, from the beginning of LINECALLINFO data structure. PartyPartitionOffset PartyPartitionSize, provides the size, in bytes, of the variably sized field that contains the Monitoring/Monitored/Recorder party Partition information and the offset, in bytes, from the beginning of LINECALLINFO data structure. DevcieNameSizeprovides the size, in bytes, of the variably sized field that contains the Monitoring/Monitored/Recorder party Device Name and the offset, in bytes, from the beginning of LINECALLINFO data structure. OverallCallSecurityStatus holds the security status of the call for two-party call as well for conference call. CCMCallID field holds the CCM call Id for each call leg.
dwCallTreatment	For All Devices: 0

Members	Values
dwCallDataSize dwCallDataOffset	For All Devices: 0
dwSendingFlowspecSize dwSendingFlowspecOffset	For All Devices: 0
dwReceivingFlowspecSize dwReceivingFlowspecOffset	For All Devices: 0

LINECALLLIST

The LINECALLLIST structure describes a list of call handles. The lineGetNewCalls and lineGetConfRelatedCalls functions return a structure of this type.

Note You must not extend this structure.

Structure Details

```
typedef struct linecalllist_tag {
  DWORD  dwTotalSize;
  DWORD  dwNeededSize;
  DWORD  dwUsedSize;
  DWORD  dwCallsNumEntries;
  DWORD  dwCallsSize;
  DWORD  dwCallsOffset;
} LINECALLLIST, FAR *LPLINECALLLIST;
```

Members

Members	Values
dwTotalSize	The total size, in bytes, that is allocated to this data structure.
dwNeededSize	The size, in bytes, for this data structure that is needed to hold all the returned information.
dwUsedSize	The size, in bytes, of the portion of this data structure that contains useful information.
dwCallsNumEntries	The number of handles in the hCalls array.
dwCallsSized wCallsOffset	The size, in bytes, and the offset, in bytes, from the beginning of this data structure of the variably sized field (which is an array of HCALL-sized handles).

LINECALLPARAMS

Members

Members	Values
dwBearerMode	not supported
dwMinRatedwMaxRate	not supported
dwMediaMode	not supported
dwCallParamFlags	not supported
dwAddressMode	not supported
dwAddressID	not supported
DialParams	not supported
dwOrigAddressSize dwOrigAddressOffset	not supported
dwDisplayableAddressSize dwDisplayableAddressOffset	not supported
dwCalledPartySize dwCalledPartyOffset	not supported
dwCommentSize dwCommentOffset	not supported
dwUserUserInfoSize dwUserUserInfoOffset	not supported
dwHighLevelCompSize dwHighLevelCompOffset	not supported
dwLowLevelCompSize dwLowLevelCompOffset	not supported
dwDevSpecificSize dwDevSpecificOffset	not supported
dwPredictiveAutoTransferStates	not supported
dwTargetAddressSize dwTargetAddressOffset	not supported

Members	Values
dwSendingFlowspecSize dwSendingFlowspecOffset	not supported
dwReceivingFlowspecSize dwReceivingFlowspecOffset	not supported
dwDeviceClassSize dwDeviceClassOffset	not supported
dwDeviceConfigSize dwDeviceConfigOffset	not supported
dwCallDataSize dwCallDataOffset	not supported
dwNoAnswerTimeout	For All Devices: The number of seconds, after the completion of dialing, that the call should be allowed to wait in the PROCEEDING or RINGBACK state before the service provider automatically abandons it with a LINECALLSTATE_DISCONNECTED and LINEDISCONNECTMODE_NOANSWER. A value of 0 indicates that the application does not want automatic call abandonment.
dwCallingPartyIDSize dwCallingPartyIDOffset	not supported

LINECALLSTATUS

Members

Members	Values
dwCallState	For IP Phones and CTI Ports:LINECALLSTATE_ACCEPTED LINECALLSTATE_CONFERENCED LINECALLSTATE_CONNECTED LINECALLSTATE_DIALING LINECALLSTATE_DIALTONE LINECALLSTATE_DISCONNECTED LINECALLSTATE_IDLE LINECALLSTATE_OFFERING LINECALLSTATE_ONHOLD LINECALLSTATE_ONHOLDPENDCONF LINECALLSTATE_ONHOLDPENDTRANSFER LINECALLSTATE_PROCEEDING LINECALLSTATE_RINGBACK LINECALLSTATE_UNKNOWN

Members	Values
	For CTI Route Points (without media):LINECALLSTATE_ACCEPTED LINECALLSTATE_DISCONNECTED LINECALLSTATE_IDLE LINECALLSTATE_OFFERING LINECALLSTATE_UNKNOWN For CTI Route Points (with media):LINECALLSTATE_ACCEPTED LINECALLSTATE_CONNECTED LINECALLSTATE_DIALING LINECALLSTATE_DIALTONE LINECALLSTATE_DISCONNECTED LINECALLSTATE_IDLE LINECALLSTATE_OFFERING LINECALLSTATE_ONHOLD LINECALLSTATE_PROCEEDING LINECALLSTATE_RINGBACK LINECALLSTATE_UNKNOWN
dwCallState (continued)	For Park DNs:LINECALLSTATE_ACCEPTED LINECALLSTATE_CONFERENCED LINECALLSTATE_CONNECTED LINECALLSTATE_DISCONNECTED LINECALLSTATE_IDLE LINECALLSTATE_OFFERING LINECALLSTATE_ONHOLD LINECALLSTATE_UNKNOWN

Members	Values
dwCallStateMode	For IP Phones, CTI Ports:LINECONNECTEDMODE_ACTIVE LINECONNECTEDMODE_INACTIVE LINEDIALTONEMODE_NORMAL LINEDIALTONEMODE_UNAVAIL LINEDISCONNECTMODE_BADADDRESS LINEDISCONNECTMODE_BUSY LINEDISCONNECTMODE_CONGESTION LINEDISCONNECTMODE_FORWARDED LINEDISCONNECTMODE_NOANSWER LINEDISCONNECTMODE_NORMAL LINEDISCONNECTMODE_REJECT LINEDISCONNECTMODE_TEMPFAILURE LINEDISCONNECTMODE_UNREACHABLE LINEDISCONNECTMODE_FACCMC (if negotiated extension version is 0x00050000 or greater)
	For CTI Route Points:LINEDISCONNECTMODE_BADADDRESS LINEDISCONNECTMODE_BUSY LINEDISCONNECTMODE_CONGESTION LINEDISCONNECTMODE_FORWARDED LINEDISCONNECTMODE_NOANSWER LINEDISCONNECTMODE_NORMAL LINEDISCONNECTMODE_REJECT LINEDISCONNECTMODE_TEMPFAILURE LINEDISCONNECTMODE_UNREACHABLE LINEDISCONNECTMODE_FACCMC (if negotiated extension version is 0x00050000 or greater)

Members	Values
	For Park DNs:LINECONNECTEDMODE_ACTIVE LINEDISCONNECTMODE_BADADDRESS LINEDISCONNECTMODE_BUSY LINEDISCONNECTMODE_CONGESTION LINEDISCONNECTMODE_FORWARDED LINEDISCONNECTMODE_NOANSWER LINEDISCONNECTMODE_NORMAL LINEDISCONNECTMODE_REJECT LINEDISCONNECTMODE_TEMPFAILURE LINEDISCONNECTMODE_UNREACHABLE
dwCallPrivilege	For All Devices LINECALLPRIVILEGE_MONITOR LINECALLPRIVILEGE_NONE LINECALLPRIVILEGE_OWNER

Members	Values
dwCallFeatures	For IP Phones (except VG248 and ATA186) and CTI Ports: LINECALLFEATURE_ACCEPT LINECALLFEATURE_ADDTOCONF LINECALLFEATURE_ANSWER LINECALLFEATURE_BLINDTRANSFER LINECALLFEATURE_COMPLETETRANSF LINECALLFEATURE_DIAL LINECALLFEATURE_DROP LINECALLFEATURE_GATHERDIGITS LINECALLFEATURE_GENERATEDIGITS LINECALLFEATURE_GENERATETONE LINECALLFEATURE_HOLD LINECALLFEATURE_MONITORDIGITS LINECALLFEATURE_MONITORTONES LINECALLFEATURE_PARK LINECALLFEATURE_PREPAREADDTOCONF LINECALLFEATURE_REDIRECT LINECALLFEATURE_SETUPCONF LINECALLFEATURE_SETUPTRANSFER LINECALLFEATURE_UNHOLD LINECALLFEATURE_UNPARK

Members	Values
	For VG248 and ATA186 Devices:LINECALLFEATURE_ACCEPT LINECALLFEATURE_ADDTOCONF LINECALLFEATURE_BLINDTRANSFER LINECALLFEATURE_COMPLETETRANSF LINECALLFEATURE_DIAL LINECALLFEATURE_DROP LINECALLFEATURE_GATHERDIGITS LINECALLFEATURE_GENERATEDIGITS LINECALLFEATURE_GENERATETONE LINECALLFEATURE_HOLD LINECALLFEATURE_MONITORDIGITS LINECALLFEATURE_MONITORTONES LINECALLFEATURE_PARK LINECALLFEATURE_PREPAREADDTOCONF LINECALLFEATURE_REDIRECT LINECALLFEATURE_SETUPCONF LINECALLFEATURE_SETUPTRANSFER LINECALLFEATURE_UNHOLD LINECALLFEATURE_UNPARK

Members	Values
dwCallFeatures (continued)	For CTI Route Points (without media):LINECALLFEATURE_ACCEPT LINECALLFEATURE_DROP LINECALLFEATURE_REDIRECT For CTI Route Points (with media):LINECALLFEATURE_ACCEPT LINECALLFEATURE_ANSWER LINECALLFEATURE_BLINDTRANSFER LINECALLFEATURE_DIA LLINECALLFEATURE_DROP LINECALLFEATURE_GATHERDIGITS LINECALLFEATURE_GENERATEDIGITS LINECALLFEATURE_GENERATETONE LINECALLFEATURE_HOLD LINECALLFEATURE_MONITORDIGITS LINECALLFEATURE_MONITORTONES LINECALLFEATURE_REDIRECT LINECALLFEATURE_UNHOLD
dwCallFeatures (continued)	For Park DNs:0
dwDevSpecificSizedwDevSpecificOffset	For All Devices:0
dwCallFeatures2	For IP Phones and CTI Ports:LINECALLFEATURE2_TRANSFERNORM LINECALLFEATURE2_TRANSFERCONF
	For CTI Route Points and Park DNs:0
tStateEntryTime	For All Devices: The Coordinated Universal Time at which the current call state was entered.

LINECARDENTRY

The LINECARDENTRY structure describes a calling card. The LINETRANSLATECAPS structure can contain an array of LINECARDENTRY structures.

Note You must not extend this structure.

Structure Details

```
typedef struct linecardentry_tag {
  DWORD  dwPermanentCardID;
  DWORD  dwCardNameSize;
  DWORD  dwCardNameOffset;
  DWORD  dwCardNumberDigits;
  DWORD  dwSameAreaRuleSize;
  DWORD  dwSameAreaRuleOffset;
  DWORD  dwLongDistanceRuleSize;
  DWORD  dwLongDistanceRuleOffset;
  DWORD  dwInternationalRuleSize;
  DWORD  dwInternationalRuleOffset;
  DWORD  dwOptions;
} LINECARDENTRY, FAR *LPLINECARDENTRY;
```

Members

Members	Values
dwPermanentCardID	The permanent identifier that identifies the card.
dwCardNameSize dwCardNameOffset	A null-terminated string (size includes the NULL) that describes the card in a user-friendly manner.
dwCardNumberDigits	The number of digits in the existing card number. The card number itself is not returned for security reasons (TAPI stores it in scrambled form). The application can use this parameter to insert filler bytes into a text control in “password” mode to show that a number exists.
dwSameAreaRuleSize dwSameAreaRuleOffset	The offset, in bytes, from the beginning of the LINETRANSLATECAPS structure and the total number of bytes in the dialing rule that is defined for calls to numbers in the same area code. The rule specifies a null-terminated string.
dwLongDistanceRuleSize dwLongDistanceRuleOffset	The offset, in bytes, from the beginning of the LINETRANSLATECAPS structure and the total number of bytes in the dialing rule that is defined for calls to numbers in the other areas in the same country or region. The rule specifies a null-terminated string.
dwInternationalRuleSize dwInternationalRuleOffset	The offset, in bytes, from the beginning of the LINETRANSLATECAPS structure and the total number of bytes in the dialing rule that is defined for calls to numbers in other countries/regions. The rule specifies a null-terminated string.
dwOptions	Indicates other settings that are associated with this calling card, by using the LINECARDOPTION_

LINECOUNTRYENTRY

The LINECOUNTRYENTRY structure provides the information for a single country entry. An array of one or more of these structures makes up part of the LINECOUNTRYLIST structure that the lineGetCountry function returns.

Note You must not extend this structure.

Structure Details

```
typedef struct linecountryentry_tag {
  DWORD  dwCountryID;
  DWORD  dwCountryCode;
  DWORD  dwNextCountryID;
  DWORD  dwCountryNameSize;
  DWORD  dwCountryNameOffset;
  DWORD  dwSameAreaRuleSize;
  DWORD  dwSameAreaRuleOffset;
  DWORD  dwLongDistanceRuleSize;
  DWORD  dwLongDistanceRuleOffset;
  DWORD  dwInternationalRuleSize;
  DWORD  dwInternationalRuleOffset;
} LINECOUNTRYENTRY, FAR *LPLINECOUNTRYENTRY;
```

Members

Members	Values
dwCountryID	The country or region identifier of the entry that specifies an internal identifier that allows multiple entries to exist in the country or region list with the same country code (for example, all countries in North America and the Caribbean share country code 1, but require separate entries in the list).
dwCountryCode	The actual country code of the country or region that the entry represents (that is, the digits that would be dialed in an international call). Display only this value to users (Country IDs should never display, as they could be confusing).
dwNextCountryID	The country identifier of the next entry in the country or region list. Because country codes and identifiers are not assigned in numeric sequence, the country or region list represents a single linked list, with each entry pointing to the next. The last country or region in the list includes a dwNextCountryID value of zero. When the LINECOUNTRYLIST structure is used to obtain the entire list, the entries in the list appear in sequence as linked by their dwNextCountryID members.
dwCountryNameSize dwCountryNameOffset	The size, in bytes, and the offset, in bytes, from the beginning of the LINECOUNTRYLIST structure of a null-terminated string that gives the name of the country or region.

Members	Values
dwSameAreaRuleSize dwSameAreaRuleOffset	The size, in bytes, and the offset, in bytes, from the beginning of the LINECOUNTRYLIST structure of a null-terminated string that contains the dialing rule for direct-dialed calls to the same area code.
dwLongDistanceRuleSize dwLongDistanceRuleOffset	The size, in bytes, and the offset, in bytes, from the beginning of the LINECOUNTRYLIST structure of a null-terminated string that contains the dialing rule for direct-dialed calls to other areas in the same country or region.
dwInternationalRuleSize dwInternationalRuleOffset	The size, in bytes, and the offset, in bytes, from the beginning of the LINECOUNTRYLIST structure of a null-terminated string that contains the dialing rule for direct-dialed calls to other countries/regions.

LINECOUNTRYLIST

The LINECOUNTRYLIST structure describes a list of countries/regions. This structure can contain an array of LINECOUNTRYENTRY structures. The lineGetCountry function returns LINECOUNTRYLIST.

Note You must not extend this structure.

Structure Details

```
typedef struct linecountrylist_tag {
 DWORD  dwTotalSize;
  DWORD  dwNeededSize;
  DWORD  dwUsedSize;
  DWORD  dwNumCountries;
  DWORD  dwCountryListSize;
  DWORD  dwCountryListOffset;
} LINECOUNTRYLIST, FAR *LPLINECOUNTRYLIST;
```

Members

Members	Values
dwTotalSize	The total size, in bytes, that are allocated to this data structure.
dwNeededSize	The size, in bytes, for this data structure that is needed to hold all the returned information.
dwUsedSize	The size, in bytes, of the portion of this data structure that contains useful information.

Members	Values
dwNumCountries	The number of LINECOUNTRYENTRY structures that are present in the array dwCountryListSize and dwCountryListOffset dominate.
dwCountryListSize dwCountryListOffset	The size, in bytes, and the offset, in bytes, from the beginning of this data structure of an array of LINECOUNTRYENTRY elements that provide information on each country or region.

LINEDEVCAPS

Members

Members	Values
dwProviderInfoSize dwProviderInfoOffset	For All Devices: The size, in bytes, of the variably sized field that contains service provider information and the offset, in bytes, from the beginning of this data structure. The dwProviderInfoSize/ Offset member provides information about the provider hardware and/or software. This information is useful when a user needs to call customer service with problems regarding the provider. The Cisco Unified TSP sets this field to "Cisco Unified TSPxxx.TSP: Cisco IP PBX Service Provider Ver. x.x(x.x)" where the text before the colon specifies the file name of the TSP and the text after "Ver." specifies the version of TSP.
dwSwitchInfoSize dwSwitchInfoOffset	For All Devices: The size, in bytes, of the variably sized device field that contains switch information and the offset, in bytes, from the beginning of this data structure. The dwSwitchInfoSize/Offset member provides information about the switch to which the line device connects, such as the switch manufacturer, the model name, the software version, and so on. This information is useful when a user needs to call customer service with problems regarding the switch. The Cisco Unified TSP sets this field to "Cisco Unified Communications Manager Ver. x.x(x.x), Cisco CTI Manager Ver x.x(x.x)" where the text after "Ver." specifies the version of the Cisco Unified Communications Manager and the version of the CTI Manager, respectively.

Members	Values
dwPermanentLineID	For All Devices: The permanent DWORD identifier by which the line device is known in the system configuration. This identifier specifies a permanent name for the line device. This permanent name (as opposed to dwDeviceID) does not change as lines are added or removed from the system and persists through operating system upgrades. You can therefore use it to link line-specific information in .ini files (or other files) in a way that is not affected by adding or removing other lines or by changing the operating system.
dwLineNameSize dwLineNameOffset	For All Devices: The size, in bytes, of the variably sized device field that contains a user-configurable name for this line device and the offset, in bytes, from the beginning of this data structure. You can configure this name when you configure the line device service provider, and the name gets provided for the convenience of the user. Cisco Unified TSP sets this field to “Cisco Line: [deviceName] (dirn)” where deviceName specifies the name of the device on which the line resides, and dirn specifies the directory number for the device.
dwStringFormat	For All Devices: STRINGFORMAT_ASCII
dwAddressModes	For All Devices: LINEADDRESSMODE_ADDRESSID
dwNumAddresses	For All Devices:1
dwBearerModes	For All Devices: LINEBEARERMODE_SPEECH LINEBEARERMODE_VOICE
dwMaxRate	For All Devices:0
dwMediaModes	For IP Phones and Park DNs: LINEMEDIAMODE_INTERACTIVEVOICE
	For CTI Ports and CTI Route Points: LINEMEDIAMODE_AUTOMATEDVOICE LINEMEDIAMODE_INTERACTIVEVOICE
dwGenerateToneModes	For IP Phones, CTI Ports, and CTI Route Points (with media): LINETONEMODE_BEEP
	For CTI Route Points (without media) and Park DNs:0
dwGenerateToneMaxNumFreq	For All Devices:0

Members	Values
dwGenerateDigitModes	For IP Phones, CTI Ports, and CTI Route Points (with media): LINETONEMODE_DTMF
	For CTI Route Points and Park DNs:0
dwMonitorToneMaxNumFreq	For All Devices:0
dwMonitorToneMaxNumEntries	For All Devices:0
dwMonitorDigitModes	For IP Phones, CTI Ports, and CTI Route Points (with media): LINETONEMODE_DTMF
	For CTI Route Points (without media) and Park DNs:0
dwGatherDigitsMinTimeout dwGatherDigitsMaxTimeout	For All Devices:0
dwMedCtlDigitMaxListSize dwMedCtlMediaMaxListSize dwMedCtlToneMaxListSize dwMedCtlCallStateMaxListSize	For All Devices:0
dwDevCapFlags	For IP Phones:0
	For All Other Devices: LINEDEVCAPFLAGS_CLOSEDROP
dwMaxNumActiveCalls	For All Devices:1
	For CTI Route Points (without media):0 For CTI Route Points (with media): Cisco Unified Communications Manager Administration configuration
dwAnswerMode	For IP Phones (except for VG248 and ATA186), CTI Route Points (with media) and CTI Ports: LINEANSWERMODE_HOLD
	For VG248 devices, ATA186 devices, CTI Route Points (without media), and Park DNs:0
dwRingModes	For All Devices:1

Members	Values
dwLineStates	For IP Phones, CTI Ports, and Route Points (with media): LINEDEVSTATE_CLOSE LINEDEVSTATE_DEVSPECIFIC LINEDEVSTATE_INSERVICE LINEDEVSTATE_MSGWAITOFF LINEDEVSTATE_MSGWAITON LINEDEVSTATE_NUMCALLS LINEDEVSTATE_OPEN LINEDEVSTATE_OUTOFSERVICE LINEDEVSTATE_REINIT LINEDEVSTATE_RINGING LINEDEVSTATE_TRANSLATECHANGE
	For CTI Route Points (without media): LINEDEVSTATE_CLOSE LINEDEVSTATE_INSERVICE LINEDEVSTATE_OPEN LINEDEVSTATE_OUTOFSERVICE LINEDEVSTATE_REINIT LINEDEVSTATE_RINGING LINEDEVSTATE_TRANSLATECHANGE
	For Park DNs:LINEDEVSTATE_CLOSE LINEDEVSTATE_DEVSPECIFIC LINEDEVSTATE_INSERVICE LINEDEVSTATE_NUMCALLS LINEDEVSTATE_OPEN LINEDEVSTATE_OUTOFSERVICE LINEDEVSTATE_REINIT LINEDEVSTATE_TRANSLATECHANGE
dwUUIAcceptSize	For All Devices:0
dwUUIAnswerSize	For All Devices:0
dwUUIMakeCallSize	For All Devices:0
dwUUIDropSize	For All Devices:0

Members	Values
dwUUISendUserUserInfoSize	For All Devices:0
dwUUICallInfoSize	For All Devices:0
MinDialParams MaxDialParams	For All Devices:0
DefaultDialParams	For All Devices:0
dwNumTerminals	For All Devices:0
dwTerminalCapsSize dwTerminalCapsOffset	For All Devices:0
dwTerminalTextEntrySize	For All Devices:0
dwTerminalTextSize dwTerminalTextOffset	For All Devices:0
dwDevSpecificSize dwDevSpecificOffset	For All Devices (except ParkDNs): If dwExtVersion > 0x00030000 (3.0):LINEDEVCAPS_DEV_SPECIFIC.m_DevSpecificFlags = 0
	For Park DNs: If dwExtVersion > 0x00030000 (3.0):LINEDEVCAPS_DEV_SPECIFIC.m_DevSpecificFlags = LINEDEVCAPSDEVSPECIFIC_PARKDN For Intercom DNs: LINEDEVCAPS_DEV_SPECIFIC. M_DevSpecificFlags = LINEDEVCAPSDEVSPECIFIC_INTERCOMDNLOCALE info PARTITION_INFO INTERCOM_SPEEDDIAL_INFO
dwLineFeatures	For IP Phones, CTI Ports, and CTI Route Points (with media): LINEFEATURE_DEVSPECIFIC LINEFEATURE_FORWARD LINEFEATURE_FORWARDFWD LINEFEATURE_MAKECALL
	For CTI Route Points (without media): LINEFEATURE_FORWARD LINEFEATURE_FORWARDFWD
	For Park DNs:0

Members	Values
dwSettableDevStatus	For All Devices:0
dwDeviceClassesSize dwDeviceClassesOffset	For IP Phones and CTI Route Points: "tapi/line" "tapi/phone"
	For CTI Ports: "tapi/line" "tapi/phone" "wave/in" "wave/out"
	For Park DNs: "tapi/line"
PermanentLineGuid	The GUID that is permanently associated with the line device.

LINEDEVSTATUS

Members

Members	Values
dwNumOpens	For All Devices: The number of active opens on the line device.
dwOpenMediaModes	For All Devices: Bit array that indicates for which media types the line device is currently open.
dwNumActiveCalls	For All Devices: The number of calls on the line in call states other than idle, onhold, onholdpendingtransfer, and onholdpendingconference.
dwNumOnHoldCalls	For All Devices: The number of calls on the line in the onhold state.
dwNumOnHoldPendCalls	For All Devices: The number of calls on the line in the onholdpendingtransfer or onholdpendingconference state.

Members	Values
dwLineFeatures	For IP Phones, CTI Ports, and CTI Route Points (with media): LINEFEATURE_DEVSPECIFIC LINEFEATURE_FORWARD LINEFEATURE_FORWARDFWD LINEFEATURE_MAKECALL
	For CTI Route Points (without media): LINEFEATURE_FORWARD LINEFEATURE_FORWARDFWD
	For Park DNs:0
dwNumCallCompletions	For All Devices:0
dwRingMode	For All Devices:0
dwSignalLevel	For All Devices:0
dwBatteryLevel	For All Devices:0
dwRoamMode	For All Devices:0
dwDevStatusFlags	For IP Phones and CTI Ports: LINEDEVSTATUSGLAGS_CONNECTED LINEDEVSTATUSGLAGS_INSERVICE LINEDEVSTATUSGLAGS_MSGWAIT
	For CTI Route Points and Park DNs: LINEDEVSTATUSGLAGS_CONNECTED LINEDEVSTATUSGLAGS_INSERVICE
dwTerminalModesSizedwTerminalModesOffset	For All Devices:0
dwDevSpecificSizedwDevSpecificOffset	For All Devices:0
dwAvailableMediaModes	For All Devices:0
dwAppInfoSizedwAppInfoOffset	For All Devices: Length, in bytes, and offset from the beginning of LINEDEVSTATUS of an array of LINEAPPINFO structures. The dwNumOpens member indicates the number of elements in the array. Each element in the array identifies an application that has the line open.

LINEEXTENSIONID

Members

Members	Values
dwExtensionID0	For All Devices: 0x8EBD6A50
dwExtensionID1	For All Devices: 0x128011D2
dwExtensionID2	For All Devices: 0x905B0060
dwExtensionID3	For All Devices: 0xB03DD275

LINEFORWARD

The LINEFORWARD structure describes an entry of the forwarding instructions.

Structure Details

```
typedef struct lineforward_tag {
    DWORD  dwForwardMode;
    DWORD  dwCallerAddressSize;
    DWORD  dwCallerAddressOffset;
    DWORD  dwDestCountryCode;
    DWORD  dwDestAddressSize;
    DWORD  dwDestAddressOffset;
} LINEFORWARD, FAR *LPLINEFORWARD;
```

Members

Members	Values
dwForwardMode	

Members	Values
	The types of forwarding. The dwForwardMode member can have only a single bit set. This member uses the following LINEFORWARDMODE_ constants: **LINEFORWARDMODE_UNCOND** Forward all calls unconditionally, irrespective of their origin. Use this value when unconditional forwarding for internal and external calls cannot be controlled separately. Unconditional forwarding overrides forwarding on busy and/or no-answer conditions. **Note** LINEFORWARDMODE_UNCOND is the only forward mode that Cisco Unified TSP supports. **LINEFORWARDMODE_UNCONDINTERNAL** Forward all internal calls unconditionally. Use this value when unconditional forwarding for internal and external calls can be controlled separately. **LINEFORWARDMODE_UNCONDEXTERNAL** Forward all external calls unconditionally. Use this value when unconditional forwarding for internal and external calls can be controlled separately. **LINEFORWARDMODE_UNCONDSPECIFIC** Unconditionally forward all calls that originated at a specified address (selective call forwarding). **LINEFORWARDMODE_BUSY** Forward all calls on busy, irrespective of their origin. Use this value when forwarding for internal and external calls both on busy and on no answer cannot be controlled separately. **LINEFORWARDMODE_BUSYINTERNAL** Forward all internal calls on busy. Use this value when forwarding for internal and external calls on busy and on no answer can be controlled separately. **LINEFORWARDMODE_BUSYEXTERNAL** Forward all external calls on busy. Use this value when forwarding for internal and external calls on busy and on no answer can be controlled separately. **LINEFORWARDMODE_BUSYSPECIFIC** Forward on busy all calls that originated at a specified address (selective call forwarding). **LINEFORWARDMODE_NOANSW**

Members	Values
	Forward all calls on no answer, irrespective of their origin. Use this value when call forwarding for internal and external calls on no answer cannot be controlled separately. **LINEFORWARDMODE_NOANSWINTERNAL** Forward all internal calls on no answer. Use this value when forwarding for internal and external calls on no answer can be controlled separately. **LINEFORWARDMODE_NOANSWEXTERNAL** Forward all external calls on no answer. Use this value when forwarding for internal and external calls on no answer can be controlled separately. **LINEFORWARDMODE_NOANSWSPECIFIC** Forward all calls that originated at a specified address on no answer (selective call forwarding). **LINEFORWARDMODE_BUSYNA** Forward all calls on busy or no answer, irrespective of their origin. Use this value when forwarding for internal and external calls on both busy and on no answer cannot be controlled separately. **LINEFORWARDMODE_BUSYNAINTERNAL** Forward all internal calls on busy or no answer. Use this value when call forwarding on busy and on no answer cannot be controlled separately for internal calls. **LINEFORWARDMODE_BUSYNAEXTERNAL** Forward all external calls on busy or no answer. Use this value when call forwarding on busy and on no answer cannot be controlled separately for internal calls. **LINEFORWARDMODE_BUSYNASPECIFIC** Forward on busy or no answer all calls that originated at a specified address (selective call forwarding). **LINEFORWARDMODE_UNKNOWN** Calls get forwarded, but the conditions under which forwarding occurs are not known at this time. **LINEFORWARDMODE_UNAVAIL** Calls are forwarded, but the conditions under which forwarding occurs are not known and are never known by the service provider.

Members	Values
dwCallerAddressSize dwCallerAddressOffset	The size in bytes of the variably sized address field that contains the address of a caller to be forwarded and the offset in bytes from the beginning of the containing data structure. The dwCallerAddressSize/Offset member gets set to zero if dwForwardMode is not one of the following choices:LINEFORWARDMODE_BUSYNASPECIFIC, LINEFORWARDMODE_NOANSWSPECIFIC, LINEFORWARDMODE_UNCONDSPECIFIC, or LINEFORWARDMODE_BUSYSPECIFIC.
dwDestCountryCode	The country code of the destination address to which the call is to be forwarded.
dwDestAddressSize dwDestAddressOffset	The size in bytes of the variably sized address field that contains the address where calls are to be forwarded and the offset in bytes from the beginning of the containing data structure.

LINEFORWARDLIST

The LINEFORWARDLIST structure describes a list of forwarding instructions.

Structure Details

```
typedef struct lineforwardlist_tag {
    DWORD  dwTotalSize;
    DWORD  dwNumEntries;
    LINEFORWARD  ForwardList[1];
} LINEFORWARDLIST, FAR *LPLINEFORWARDLIST;
```

Members

Members	Values
dwTotalSize	The total size in bytes of the data structure.
dwNumEntries	Number of entries in the array, specified as ForwardList[ ].
ForwardList[ ]	An array of forwarding instruction. The array entries specify type LINEFORWARD.

LINEGENERATETONE

The LINEGENERATETONE structure contains information about a tone to be generated. The lineGenerateTone and TSPI_lineGenerateTone functions use this structure.

Note You must not extend this structure.

This structure gets used only for the generation of tones; it is not used for tone monitoring.

Structure Details

```
typedef struct linegeneratetone_tag {
  DWORD  dwFrequency;
  DWORD  dwCadenceOn;
  DWORD  dwCadenceOff;
  DWORD  dwVolume;
} LINEGENERATETONE, FAR *LPLINEGENERATETONE;
```

Members

Members	Values
dwFrequency	The frequency, in hertz, of this tone component. A service provider may adjust (round up or down) the frequency that the application specified to fit its resolution.
dwCadenceOn	The “on” duration, in milliseconds, of the cadence of the custom tone to be generated. Zero means no tone gets generated.
dwCadenceOff	The “off” duration, in milliseconds, of the cadence of the custom tone to be generated. Zero means no off time, that is, a constant tone.
dwVolume	The volume level at which the tone gets generated. A value of 0x0000FFFF represents full volume, and a value of 0x00000000 means silence.

LINEINITIALIZEEXPARAMS

The LINEINITIZALIZEEXPARAMS structure describes parameters that are supplied when calls are made by using LINEINITIALIZEEX.

Structure Details

```
typedef struct lineinitializeexparams_tag {
  DWORD  dwTotalSize;
  DWORD  dwNeededSize;
  DWORD  dwUsedSize;
  DWORD  dwOptions;

 union
 {
  HANDLE  hEvent;
  HANDLE  hCompletionPort;
 } Handles;
```

```
  DWORD  dwCompletionKey;
} LINEINITIALIZEEXPARAMS, FAR *LPLINEINITIALIZEEXPARAMS;
```

Members

Members	Values
dwTotalSize	The total size, in bytes, that is allocated to this data structure.
dwNeededSize	The size, in bytes, for this data structure that is needed to hold all the returned information.
dwUsedSize	The size, in bytes, of the portion of this data structure that contains useful information.
dwOptions	One of the LINEINITIALIZEEXOPTION_constants. Specifies the event notification mechanism that the application wants to use.
hEvent	If dwOptions specifies LINEINITIALIZEEXOPTION_USEEVENT, TAPI returns the event handle in this field.
hCompletionPort	If dwOptions specifies LINEINITIALIZEEXOPTION_USECOMPLETIONPORT, the application must specify in this field the handle of an existing completion port that was opened by using CreateIoCompletionPort.
dwCompletionKey	If dwOptions specifies LINEINITIALIZEEXOPTION_USECOMPLETIONPORT, the application must specify in this field a value that is returned through the lpCompletionKey parameter of GetQueuedCompletionStatus to identify the completion message as a telephony message.

Further Details

See lineInitializeEx, on page 34 for further information on these options.

LINELOCATIONENTRY

The LINELOCATIONENTRY structure describes a location that is used to provide an address translation context. The LINETRANSLATECAPS structure can contain an array of LINELOCATIONENTRY structures.

Note You must not extend this structure.

Structure Details

```
typedef struct linelocationentry_tag {
  DWORD  dwPermanentLocationID;
  DWORD  dwLocationNameSize;
  DWORD  dwLocationNameOffset;

  DWORD  dwCityCodeSize;
  DWORD  dwCityCodeOffset;
  DWORD  dwPreferredCardID;
  DWORD  dwLocalAccessCodeSize;
  DWORD  dwLocalAccessCodeOffset;
  DWORD  dwLongDistanceAccessCodeSize;
  DWORD  dwLongDistanceAccessCodeOffset;
  DWORD  dwTollPrefixListSize;
  DWORD  dwTollPrefixListOffset;
  DWORD  dwCountryID;
  DWORD  dwOptions;
  DWORD  dwCancelCallWaitingSize;
  DWORD  dwCancelCallWaitingOffset;
} LINELOCATIONENTRY, FAR *LPLINELOCATIONENTRY;
```

Members

Members	Values
dwPermanentLocationID	The permanent identifier that identifies the location.
dwLocationNameSize dwLocationNameOffset	Contains a null-terminated string (size includes the NULL) that describes the location in a user-friendly manner.
dwCountryCode	The country code of the location.
dwPreferredCardID	The preferred calling card when dialing from this location.
dwCityCodeSize dwCityCodeOffset	Contains a null-terminated string that specifies the city or area code that is associated with the location (the size includes the NULL). Applications can use this information, along with the country code, to “default” entry fields for the user when you enter the phone numbers, to encourage the entry of proper canonical numbers.
dwLocalAccessCodeSize dwLocalAccessCodeOffset	The size, in bytes, and the offset, in bytes, from the beginning of the LINETRANSLATECAPS structure of a null-terminated string that contains the access code to be dialed before calls to addresses in the local calling area.
dwLongDistanceAccessCodeSize dwLongDistanceAccessCodeOffset	The size, in bytes, and the offset, in bytes, from the beginning of the LINETRANSLATECAPS structure of a null-terminated string that contains the access code to be dialed before calls to addresses outside the local calling area.

Members	Values
dwTollPrefixListSize dwTollPrefixListOffset	The size, in bytes, and the offset, in bytes, from the beginning of the LINETRANSLATECAPS structure of a null-terminated string that contains the toll prefix list for the location. The string contains only prefixes that consist of the digits “0” through “9” and are separated from each other by a single “,” (comma) character.
dwCountryID	The country identifier of the country or region that is selected for the location. Use this identifier with the lineGetCountry function to obtain additional information about the specific country or region, such as the country or region name (you cannot use the dwCountryCode member for this purpose because country codes are not unique).
dwOptions	Indicates options in effect for this location with values taken from the LINELOCATIONOPTION_Constants.
dwCancelCallWaitingSize dwCancelCallWaitingOffset	The size, in bytes, and the offset, in bytes, from the beginning of the LINETRANSLATECAPS structure of a null-terminated string that contains the dial digits and modifier characters that should be prefixed to the dialable string (after the pulse/tone character) when an application sets the LINETRANSLATEOPTION_CANCELCALLWAITING bit in the dwTranslateOptions parameter of lineTranslateAddress. If no prefix is defined, dwCancelCallWaitingSize set to zero may indicate this, or dwCancelCallWaitingSize set to 1 and dwCancelCallWaitingOffset pointing to an empty string (single NULL byte) may indicate this.

LINEMESSAGE

The LINEMESSAGE structure contains parameter values that specify a change in status of the line that the application currently has open. The lineGetMessage function returns the LINEMESSAGE structure.

Structure Details

```
typedef struct linemessage_tag {
  DWORD  hDevice;
  DWORD  dwMessageID;
  DWORD_PTR  dwCallbackInstance;
  DWORD_PTR  dwParam1;
  DWORD_PTR  dwParam2;
  DWORD_PTR  dwParam3;
} LINEMESSAGE, FAR *LPLINEMESSAGE;
```

Members

Members	Values
hDevice	A handle to either a line device or a call. The context that dwMessageID provides can determine the nature of this handle (line handle or call handle).
dwMessageID	A line or call device message.
dwCallbackInstance	Instance data passed back to the application, which the application in the dwCallBackInstance parameter of lineInitializeEx specified. TAPI does not interpret this DWORD.
dwParam1	A parameter for the message.
dwParam2	A parameter for the message.
dwParam3	A parameter for the message.

For details about the parameter values that are passed in this structure, see TAPI Line Messages, on page 57.

LINEMONITORTONE

The LINEMONITORTONE structure defines a tone for the purpose of detection. Use this as an entry in an array. An array of tones gets passed to the lineMonitorTones function that monitors these tones and sends a LINE_MONITORTONE message to the application when a detection is made.

A tone with all frequencies set to zero corresponds to silence. An application can thus monitor the call information stream for silence.

Note You must not extend this structure.

Structure Details

```
typedef struct linemonitortone_tag {   DWORD  dwAppSpecific;
  DWORD  dwDuration;
  DWORD  dwFrequency1;
  DWORD  dwFrequency2;
  DWORD  dwFrequency3;
} LINEMONITORTONE, FAR *LPLINEMONITORTONE;
```

Members

Members	Values
dwAppSpecific	Used by the application for tagging the tone. When this tone is detected, the value of the dwAppSpecific member gets passed back to the application.

Members	Values
dwDuration	The duration, in milliseconds, during which the tone should be present before a detection is made.
dwFrequency1	dwFrequency2
dwFrequency3	The frequency, in hertz, of a component of the tone. If fewer than three frequencies are needed in the tone, a value of 0 should be used for the unused frequencies. A tone with all three frequencies set to zero gets interpreted as silence and can be used for silence detection.

LINEPROVIDERENTRY

The LINEPROVIDERENTRY structure provides the information for a single service provider entry. An array of these structures gets returned as part of the LINEPROVIDERLIST structure that the function lineGetProviderList returns.

Note You cannot extend this structure.

Structure Details

```
typedef struct lineproviderentry_tag {
  DWORD  dwPermanentProviderID;
  DWORD  dwProviderFilenameSize;
  DWORD  dwProviderFilenameOffset;
} LINEPROVIDERENTRY, FAR *LPLINEPROVIDERENTRY;
```

Members

Members	Values
dwPermanentProviderID	The permanent provider identifier of the entry.
dwProviderFilenameSizedwProviderFilenameOffset	The size, in bytes, and the offset, in bytes, from the beginning of the LINEPROVIDERLIST structure of a null-terminated string that contains the filename (path) of the service provider DLL (.TSP) file.

LINEPROVIDERLIST

The LINEPROVIDERLIST structure describes a list of service providers. The lineGetProviderList function returns a structure of this type. The LINEPROVIDERLIST structure can contain an array of LINEPROVIDERENTRY structures.

Note You must not extend this structure.

Structure Details

```
typedef struct lineproviderlist_tag {
  DWORD  dwTotalSize;
  DWORD  dwNeededSize;
  DWORD  dwUsedSize;
DWORD  dwNumProviders;
  DWORD  dwProviderListSize;
  DWORD  dwProviderListOffset;
} LINEPROVIDERLIST, FAR *LPLINEPROVIDERLIST;
```

Members

Members	Values
dwTotalSize	The total size, in bytes, that are allocated to this data structure.
dwNeededSize	The size, in bytes, for this data structure that is needed to hold all the returned information.
dwUsedSize	The size, in bytes, of the portion of this data structure that contains useful information.
dwNumProviders	The number of LINEPROVIDERENTRY structures that are present in the array that is denominated by dwProviderListSize and dwProviderListOffset.
dwProviderListSize dwProviderListOffset	The size, in bytes, and the offset, in bytes, from the beginning of this data structure of an array of LINEPROVIDERENTRY elements, which provide the information on each service provider.

LINEREQMAKECALL

The LINEREQMAKECALL structure describes a request that a call initiated to the lineGetRequest function.

Note You cannot extend this structure.

Structure Details

```
typedef struct linereqmakecall_tag {
  char  szDestAddress[TAPIMAXDESTADDRESSSIZE];
  char  szAppName[TAPIMAXAPPNAMESIZE];
  char  szCalledParty[TAPIMAXCALLEDPARTYSIZE];
```

```
  char  szComment[TAPIMAXCOMMENTSIZE];
} LINEREQMAKECALL, FAR *LPLINEREQMAKECALL;
```

Members

Members	Values
szDestAddress [TAPIMAXADDRESSSIZE]	The null-terminated destination address of the make-call request. The address uses the canonical address format or the dialable address format. The maximum length of the address specifies TAPIMAXDESTADDRESSSIZE characters, which include the NULL terminator. Longer strings get truncated.
szAppName [TAPIMAXAPPNAMESIZE]	The null-terminated, user-friendly application name or filename of the application that originated the request. The maximum length of the address specifies TAPIMAXAPPNAMESIZE characters, which include the NULL terminator.
szCalledParty [TAPIMAXCALLEDPARTYSIZE]	The null-terminated, user-friendly called-party name. The maximum length of the called-party information specifies TAPIMAXCALLEDPARTYSIZE characters, which include the NULL terminator.
szComment [TAPIMAXCOMMENTSIZE]	The null-terminated comment about the call request. The maximum length of the comment string specifies TAPIMAXCOMMENTSIZE characters, which include the NULL terminator.

LINETRANSLATECAPS

The LINETRANSLATECAPS structure describes the address translation capabilities. This structure can contain an array of LINELOCATIONENTRY structures and an array of LINECARDENTRY structures. the lineGetTranslateCaps function returns the LINETRANSLATECAPS structure.

Note You must not extend this structure.

Structure Details

```
typedef struct linetranslatecaps_tag {
  DWORD  dwTotalSize;
  DWORD  dwNeededSize;
  DWORD  dwUsedSize;
  DWORD  dwNumLocations;
  DWORD  dwLocationListSize;
  DWORD  dwLocationListOffset;
  DWORD  dwCurrentLocationID;
  DWORD  dwNumCards;
  DWORD  dwCardListSize;
  DWORD  dwCardListOffset;
```

```
  DWORD  dwCurrentPreferredCardID;
} LINETRANSLATECAPS, FAR *LPLINETRANSLATECAPS;
```

Members

Members	Values
dwTotalSize	The total size, in bytes, that is allocated to this data structure.
dwNeededSize	The size, in bytes, for this data structure that is needed to hold all the returned information.
dwUsedSize	The size, in bytes, of the portion of this data structure that contains useful information.
dwNumLocations	The number of entries in the location list. It includes all locations that are defined, including zero (default).
dwLocationListSize dwLocationListOffset	List of locations that are known to the address translation. The list comprises a sequence of LINELOCATIONENTRY structures. The dwLocationListOffset member points to the first byte of the first LINELOCATIONENTRY structure, and the dwLocationListSize member indicates the total number of bytes in the entire list.
dwCurrentLocationID	The dwPermanentLocationID member from the LINELOCATIONENTRY structure for the CurrentLocation.
dwNumCards	The number of entries in the CardList.
dwCardListSize dwCardListOffset	List of calling cards that are known to the address translation. It includes only non-hidden card entries and always includes card 0 (direct dial). The list comprises a sequence of LINECARDENTRY structures. The dwCardListOffset member points to the first byte of the first LINECARDENTRY structure, and the dwCardListSize member indicates the total number of bytes in the entire list.
dwCurrentPreferredCardID	The dwPreferredCardID member from the LINELOCATIONENTRY structure for the CurrentLocation.

LINETRANSLATEOUTPUT

The LINETRANSLATEOUTPUT structure describes the result of an address translation. The lineTranslateAddress function uses this structure.

Note You must not extend this structure.

Structure Details

```
typedef struct linetranslateoutput_tag {
  DWORD  dwTotalSize;
  DWORD  dwNeededSize;
  DWORD  dwUsedSize;
  DWORD  dwDialableStringSize;
  DWORD  dwDialableStringOffset;
  DWORD  dwDisplayableStringSize;
  DWORD  dwDisplayableStringOffset;
  DWORD  dwCurrentCountry;
  DWORD  dwDestCountry;
  DWORD  dwTranslateResults;
} LINETRANSLATEOUTPUT, FAR *LPLINETRANSLATEOUTPUT;
```

Members

Members	Values
dwTotalSize	The total size, in bytes, that is allocated to this data structure.
dwNeededSize	The size, in bytes, for this data structure that is needed to hold all the returned information.
dwUsedSize	The size, in bytes, of the portion of this data structure that contains useful information.
dwDialableStringSize dwDialableStringOffset	Contains the translated output that can be passed to the lineMakeCall, lineDial, or other function that requires a dialable string. The output always comprises a null-terminated string (NULL gets included in the count in dwDialableStringSize). This output string includes ancillary fields such as name and subaddress if they were in the input string. This string may contain private information such as calling card numbers. To prevent inadvertent visibility to unauthorized persons, it should not display to the user.
dwDisplayableStringSize dwDisplayableStringOffset	Contains the translated output that can display to the user for confirmation. Identical to DialableString, except the “friendly name” of the card enclosed within bracket characters (for example, “[AT&T Card]”) replaces calling card digits. The ancillary fields, such as name and subaddress, get removed. You can display this string in call-status dialog boxes without exposing private information to unauthorized persons. You can also include this information in call logs.
dwCurrentCountry	Contains the country code that is configured in CurrentLocation. Use this value to control the display by the application of certain user interface elements for local call progress tone detection and for other purposes.

Members	Values
dwDestCountry	Contains the destination country code of the translated address. This value may pass to the dwCountryCode parameter of lineMakeCall and other dialing functions (so the call progress tones of the destination country or region such as a busy signal are properly detected). This field gets set to zero if the destination address that is passed to lineTranslateAddress is not in canonical format.
dwTranslateResults	Indicates the information that is derived from the translation process, which may assist the application in presenting user-interface elements. This field uses one LINETRANSLATERESULT_.

TAPI Phone Functions

TAPI phone functions enable an application to control physical aspects of a phone

Table 4: TAPI Phone Functions

TAPI phone functions
phoneCallbackFunc, on page 133
phoneClose, on page 134
phoneDevSpecific, on page 134
phoneGetDevCaps, on page 134
phoneGetDisplay, on page 135
phoneGetLamp, on page 136
phoneGetMessage, on page 136
phoneGetRing, on page 137
phoneGetStatus, on page 138
phoneGetStatusMessages, on page 139
phoneInitialize, on page 140
phoneInitializeEx, on page 141
phoneNegotiateAPIVersion, on page 143
phoneOpen, on page 144
phoneSetDisplay, on page 145

TAPI phone functions
phoneSetStatusMessages, on page 146
phoneShutdown, on page 148

phoneCallbackFunc

The phoneCallbackFunc function provides a placeholder for the application-supplied function name.

All callbacks occur in the application context. The callback function must reside in a dynamic-link library (DLL) or application module and be exported in the module-definition file.

Function Details

```
VOID FAR PASCAL phoneCallbackFunc(
  HANDLE hDevice,
  DWORD dwMsg,
  DWORD dwCallbackInstance,
  DWORD dwParam1,
  DWORD dwParam2,
  DWORD dwParam3
);
```

Parameters

hDevice

A handle to a phone device that is associated with the callback.

dwMsg

A line or call device message.

dwCallbackInstance

Callback instance data that is passed to the application in the callback. TAPI does not interpret this DWORD.

dwParam1

A parameter for the message.

dwParam2

A parameter for the message.

dwParam3

A parameter for the message.

Further Details

For more information about the parameters that are passed to this callback function, see TAPI Line Messages, on page 57 and TAPI Phone Messages, on page 148.

phoneClose

The phoneClose function closes the specified open phone device.

Function Details

```
LONG phoneClose(
  HPHONE hPhone
);
```

Parameter

hPhone

A handle to the open phone device that is to be closed. If the function succeeds, this means that the handle is no longer valid.

phoneDevSpecific

The phoneDevSpecific function gets used as a general extension mechanism to enable a telephony API implementation to provide features that are not described in the other TAPI functions. The meanings of these extensions are device specific.

When used with the Cisco Unified TSP, you can use phoneDevSpecific to send device-specific data to a phone device.

Function Details

```
LONG WINAPI phoneDevSpecific (
HPHONE hPhone,
LPVOID lpParams,
DWORD dwSize
);
```

Parameters

hPhone

A handle to a phone device.

lpParams

A pointer to a memory area used to hold a parameter block. Its interpretation is device specific. TAPI passes the contents of the parameter block unchanged to or from the service provider.

dwSize

The size in bytes of the parameter block area.

phoneGetDevCaps

The phoneGetDevCaps function queries a specified phone device to determine its telephony capabilities.

Function Details

```
LONG phoneGetDevCaps(
  HPHONEAPP hPhoneApp,
  DWORD dwDeviceID,
  DWORD dwAPIVersion,
  DWORD dwExtVersion,
  LPPHONECAPS lpPhoneCaps
);
```

Parameters

hPhoneApp

The handle to the registration with TAPI for this application.

dwDeviceID

The phone device that is to be queried.

dwAPIVersion

The version number of the telephony API that is to be used. The high-order word contains the major version number; the low-order word contains the minor version number. You can obtain this number with the function phoneNegotiateAPIVersion.

dwExtVersion

The version number of the service provider-specific extensions to be used. This number is obtained with the function phoneNegotiateExtVersion. It can be left as zero if no device-specific extensions are to be used. Otherwise, the high-order word contains the major version number, the low-order word contains the minor version number.

lpPhoneCaps

A pointer to a variably sized structure of type PHONECAPS. Upon successful completion of the request, this structure is filled with phone device capabilities information.

phoneGetDisplay

The phoneGetDisplay function returns the current contents of the specified phone display.

Function Details

```
LONG phoneGetDisplay(
  HPHONE hPhone,
  LPVARSTRING lpDisplay
);
```

Parameters

hPhone

A handle to the open phone device.

lpDisplay

A pointer to the memory location where the display content is to be stored, of type VARSTRING.

phoneGetLamp

The phoneGetLamp function returns the current lamp mode of the specified lamp.

Note Cisco Unified IP Phones 79xx series do not support this function.

Function Details

```
LONG phoneGetLamp(
  HPHONE hPhone,
  DWORD dwButtonLampID,
  LPDWORD lpdwLampMode
);
```

Parameters

hPhone

A handle to the open phone device.

dwButtonLampID

The identifier of the lamp that is to be queried. See PHONE_BUTTON, on page 149 for lamp IDs.

lpdwLampMode

Note Cisco Unified IP Phones 79xx series do not support this function.

A pointer to a memory location that holds the lamp mode status of the given lamp. The lpdwLampMode parameter can have at most one bit set. This parameter uses the following PHONELAMPMODE_constants:

- PHONELAMPMODE_FLASH -Flash means slow on and off.
- PHONELAMPMODE_FLUTTER -Flutter means fast on and off.
- PHONELAMPMODE_OFF -The lamp is off.
- PHONELAMPMODE_STEADY -The lamp is continuously lit.
- PHONELAMPMODE_WINK -The lamp winks.
- PHONELAMPMODE_UNKNOWN -The lamp mode is currently unknown.
- PHONELAMPMODE_DUMMY -Use this value to describe a button/lamp position that has no corresponding lamp.

phoneGetMessage

The phoneGetMessage function returns the next TAPI message that is queued for delivery to an application that is using the Event Handle notification mechanism (see phoneInitializeEx for further details).

Function Details

```
LONG WINAPI phoneGetMessage(
  HPHONEAPP hPhoneApp,
  LPPHONEMESSAGE lpMessage,
  DWORD dwTimeout
);
```

Parameters

hPhoneApp

The handle that phoneInitializeEx returns. The application must have set the PHONEINITIALIZEEXOPTION_USEEVENT option in the dwOptions member of the PHONEINITIALIZEEXPARAMS structure.

lpMessage

A pointer to a PHONEMESSAGE structure. Upon successful return from this function, the structure contains the next message that had been queued for delivery to the application.

dwTimeout

The time-out interval, in milliseconds. The function returns if the interval elapses, even if no message can be returned. If dwTimeout is zero, the function checks for a queued message and returns immediately. If dwTimeout is INFINITE, the time-out interval never elapses.

Return Values

Returns zero if the request succeeds or a negative number if an error occurs. Possible return values follow:

PHONEERR_INVALAPPHANDLE, PHONEERR_OPERATIONFAILED, PHONEERR_INVALPOINTER, PHONEERR_NOMEM.

phoneGetRing

The phoneGetRing function enables an application to query the specified open phone device as to its current ring mode.

Function Details

```
LONG phoneGetRing(
  HPHONE hPhone,
  LPDWORD lpdwRingMode,
  LPDWORD lpdwVolume
);
```

Parameters

hPhone

A handle to the open phone device.

lpdwRingMode

The ringing pattern with which the phone is ringing. Zero indicates that the phone is not ringing.

The system supports four ring modes.

The following table lists the valid ring modes.

Table 5: Ring Modes

Ring Modes	Definition
0	Off
1	Inside Ring
2	Outside Ring
3	Feature Ring

lpdwVolume

The volume level with which the phone is ringing. This parameter has no meaning; the value 0x8000 always gets returned.

phoneGetStatus

The phoneGetStatus function enables an application to query the specified open phone device for its overall status.

Function Details

```
LONG WINAPI phoneGetStatusMessages(
HPHONE hPhone,
LPPHONESTATUS lpPhoneStatus
) ;
```

Parameters

hPhone

A handle to the open phone device to be queried.

lpPhoneStatus

A pointer to a variably sized data structure of type PHONESTATUS, which is loaded with the returned information about the phone status.

Return Values

Returns zero if the request succeeds or a negative number if an error occurs. Return values include the following:

PHONEERR_INVALPHONEHANDLE, PHONEERR_NOMEM PHONEERR_INVALPOINTER, PHONEERR_RESOURCEUNAVAIL PHONEERR_OPERATIONFAILED, PHONEERR_STRUCTURETOOSMALL PHONEERR_OPERATIONUNAVAIL, PHONEERR_UNINITIALIZED

phoneGetStatusMessages

The phoneGetStatusMessages function returns information about which phone-state changes on the specified phone device generate a callback to the application.

An application can use phoneGetStatusMessages to query the generation of the corresponding messages. The phoneSetStatusMessages can control Message generation. All phone status messages remain disabled by default.

Function Details

```
LONG WINAPI phoneGetStatusMessages(
  HPHONE hPhone,
  LPDWORD lpdwPhoneStates,
  LPDWORD lpdwButtonModes,
  LPDWORD lpdwButtonStates
);
```

Parameters

hPhone

A handle to the open phone device that is to be monitored.

lpdwPhoneStates

A pointer to a DWORD holding zero, one or more of the PHONESTATE_Constants. These flags specify the set of phone status changes and events for which the application can receive notification messages. You can enable or disable monitoring individually for the following states:

- PHONESTATE_OTHER
- PHONESTATE_CONNECTED
- PHONESTATE_DISCONNECTED
- PHONESTATE_OWNER
- PHONESTATE_MONITORS
- PHONESTATE_DISPLAY
- PHONESTATE_LAMP
- PHONESTATE_RINGMODE
- PHONESTATE_RINGVOLUME
- PHONESTATE_HANDSETHOOKSWITCH
- PHONESTATE_HANDSETVOLUME
- PHONESTATE_HANDSETGAIN
- PHONESTATE_SPEAKERHOOKSWITCH
- PHONESTATE_SPEAKERVOLUME
- PHONESTATE_SPEAKERGAIN

- PHONESTATE_HEADSETHOOKSWITCH
- PHONESTATE_HEADSETVOLUME
- PHONESTATE_HEADSETGAIN
- PHONESTATE_SUSPEND
- PHONESTATE_RESUMEF
- PHONESTATE_DEVSPECIFIC
- PHONESTATE_REINIT
- PHONESTATE_CAPSCHANGE
- PHONESTATE_REMOVED

lpdwButtonModes

A pointer to a DWORD that contains flags that specify the set of phone-button modes for which the application can receive notification messages. This parameter uses zero, one, or more of the PHONEBUTTONMODE_Constants.

lpdwButtonStates

A pointer to a DWORD that contains flags that specify the set of phone button state changes for which the application can receive notification messages. This parameter uses zero, one, or more of the PHONEBUTTONSTATE_ Constants.

Return Values

Returns zero if the request succeeds or a negative number if an error occurs. Possible return values follow:

PHONEERR_INVALPHONEHANDLE

PHONEERR_NOMEM

PHONEERR_INVALPOINTER

PHONEERR_RESOURCEUNAVAIL

PHONEERR_OPERATIONFAILED

PHONEERR_UNINITIALIZED.

phoneInitialize

Although the phoneInitialize function is obsolete, tapi.dll and tapi32.dll continue to export it for backward compatibility with applications that are using TAPI versions 1.3 and 1.4.

Function Details

```
LONG WINAPI phoneInitialize(
  LPHPHONEAPP lphPhoneApp,
  HINSTANCE hInstance,
  PHONECALLBACK lpfnCallback,
  LPCSTR lpszAppName,
```

```
  LPDWORD lpdwNumDevs
);
```

Parameters

lphPhoneApp

A pointer to a location that is filled with the application usage handle for TAPI.

hInstance

The instance handle of the client application or DLL.

lpfnCallback

The address of a callback function that is invoked to determine status and events on the phone device.

lpszAppName

A pointer to a null-terminated string that contains displayable characters. If this parameter is non-NULL, it contains an application-supplied name of the application. This name, which is provided in the PHONESTATUS structure, indicates, in a user-friendly way, which application is the current owner of the phone device. You can use this information for logging and status reporting purposes. If lpszAppName is NULL, the application filename gets used instead.

lpdwNumDevs

A pointer to DWORD. This location gets loaded with the number of phone devices that are available to the application.

Return Values

Returns zero if the request succeeds or a negative number if an error occurs. Possible return values follow:

PHONEERR_INVALAPPNAME

PHONEERR_INIFILECORRUPT

PHONEERR_INVALPOINTER

PHONEERR_NOMEM

PHONEERR_OPERATIONFAILED

PHONEERR_REINIT

PHONEERR_RESOURCEUNAVAIL

PHONEERR_NODEVICE

PHONEERR_NODRIVER

PHONEERR_INVALPARAM

phoneInitializeEx

The phoneInitializeEx function initializes the application use of TAPI for subsequent use of the phone abstraction. It registers the application specified notification mechanism and returns the number of phone devices that are available to the application. A phone device represents any device that provides an implementation for the phone-prefixed functions in the telephony API.

Function Details

```
LONG WINAPI phoneInitializeEx(
  LPHPHONEAPP lphPhoneApp,
  HINSTANCE hInstance,
  PHONECALLBACK lpfnCallback,
  LPCSTR lpszFriendlyAppName,
  LPDWORD lpdwNumDevs,
  LPDWORD lpdwAPIVersion,
  LPPHONEINITIALIZEEXPARAMS lpPhoneInitializeExParams
);
```

Parameters

lphPhoneApp

A pointer to a location that is filled with the application usage handle for TAPI.

hInstance

The instance handle of the client application or DLL. The application or DLL can pass NULL for this parameter, in which case TAPI uses the module handle of the root executable of the process.

lpfnCallback

The address of a callback function that is invoked to determine status and events on the line device, addresses, or calls, when the application is using the "hidden window" method of event notification (for more information, see phoneCallbackFunc). When the application chooses to use the event handle or completion port event notification mechanisms, this parameter gets ignored and should be set to NULL.

lpszFriendlyAppName

A pointer to a null-terminated string that contains only displayable characters. If this parameter is not NULL, it contains an application-supplied name for the application. This name, which is provided in the PHONESTATUS structure, indicates, in a user-friendly way, which application has ownership of the phone device. If lpszFriendlyAppName is NULL, the application module filename gets used instead (as returned by the Windows function GetModuleFileName).

lpdwNumDevs

A pointer to a DWORD. Upon successful completion of this request, the number of phone devices that are available to the application fills this location.

lpdwAPIVersion

A pointer to a DWORD. The application must initialize this DWORD, before calling this function, to the highest API version that it is designed to support (for example, the same value that it would pass into dwAPIHighVersion parameter of phoneNegotiateAPIVersion). Do no use artificially high values; ensure the values are accurately set. TAPI translates any newer messages or structures into values or formats that the application version supports. Upon successful completion of this request, the highest API version that TAPI supports fills this location, which allows the application to detect and adapt to being installed on a system with an older version of TAPI.

lpPhoneInitializeExParams

A pointer to a structure of type PHONEINITIALIZEEXPARAMS that contains additional parameters that are used to establish the association between the application and TAPI (specifically, the application-selected event notification mechanism and associated parameters).

Return Values

Returns zero if the request succeeds or a negative number if an error occurs. Possible return values follow:

PHONEERR_INVALAPPNAME

PHONEERR_OPERATIONFAILED

PHONEERR_INIFILECORRUPT

PHONEERR_INVALPOINTER

PHONEERR_REINIT

PHONEERR_NOMEM

PHONEERR_INVALPARAM

phoneNegotiateAPIVersion

Use the phoneNegotiateAPIVersion function to negotiate the API version number to be used with the specified phone device. It returns the extension identifier that the phone device supports, or zeros if no extensions are provided.

Function Details

```
LONG WINAPI phoneNegotiateAPIVersion(
  HPHONEAPP hPhoneApp,
  DWORD dwDeviceID,
  DWORD dwAPILowVersion,
  DWORD dwAPIHighVersion,
  LPDWORD lpdwAPIVersion,
  LPPHONEEXTENSIONID lpExtensionID
);
```

Parameters

hPhoneApp

The handle to the application registration with TAPI.

dwDeviceID

The phone device to be queried.

dwAPILowVersion

The least recent API version with which the application is compliant. The high-order word represents the major version number, and the low-order word represents the minor version number.

dwAPIHighVersion

The most recent API version with which the application is compliant. The high-order word represents the major version number, and the low-order word represents the minor version number.

lpdwAPIVersion

A pointer to a DWORD in which the API version number that was negotiated will be returned. If negotiation succeeds, this number ranges from dwAPILowVersion to dwAPIHighVersion.

lpExtensionID

A pointer to a structure of type PHONEEXTENSIONID. If the service provider for the specified dwDeviceID parameter supports provider-specific extensions, this structure gets filled with the extension identifier of these extensions when negotiation succeeds. This structure contains all zeros if the line provides no extensions. An application can ignore the returned parameter if it does not use extensions.

Return Values

Returns zero if the request succeeds or a negative number if an error occurs. Possible return values follow:

PHONEERR_INVALAPPHANDLE

PHONEERR_OPERATIONFAILED

PHONEERR_BADDEVICEID

PHONEERR_OPERATIONUNAVAIL

PHONEERR_NODRIVER

PHONEERR_NOMEM

PHONEERR_INVALPOINTER

PHONEERR_RESOURCEUNAVAIL,PHONEERR_INCOMPATIBLEAPIVERSION

PHONEERR_UNINITIALIZED

PHONEERR_NODEVICE

phoneOpen

The phoneOpen function opens the specified phone device. Open the device by using either owner privilege or monitor privilege. An application that opens the phone with owner privilege can control the lamps, display, ringer, and hookswitch or hookswitches that belong to the phone. An application that opens the phone device with monitor privilege receives notification only about events that occur at the phone, such as hookswitch changes or button presses. Because ownership of a phone device is exclusive, only one application at a time can have a phone device opened with owner privilege. The phone device can, however, be opened multiple times with monitor privilege.

Note To open a phone device on a CTI port, first ensure a corresponding line device is open.

Function Details

```
LONG phoneOpen(
  HPHONEAPP hPhoneApp,
  DWORD dwDeviceID,
  LPHPHONE lphPhone,
  DWORD dwAPIVersion,
  DWORD dwExtVersion,
  DWORD dwCallbackInstance,
  DWORD dwPrivilege
);
```

Parameters

hPhoneApp

A handle by which the application is registered with TAPI.

dwDeviceID

The phone device to be opened.

lphPhone

A pointer to an HPHONE handle that identifies the open phone device. Use this handle to identify the device when invoking other phone control functions.

dwAPIVersion

The API version number under which the application and telephony API agreed to operate. Obtain this number from phoneNegotiateAPIVersion.

dwExtVersion

The extension version number under which the application and the service provider agree to operate. This number is zero if the application does not use any extensions. Obtain this number from phoneNegotiateExtVersion.

dwCallbackInstance

User instance data that is passed back to the application with each message. The telephony API does not interpret this parameter.

dwPrivilege

The privilege requested. The dwPrivilege parameter can have only one bit set. This parameter uses the following PHONEPRIVILEGE_constants:

- PHONEPRIVILEGE_MONITOR -An application that opens a phone device with this privilege gets informed about events and state changes that occur on the phone. The application cannot invoke any operations on the phone device that would change its state.
- PHONEPRIVILEGE_OWNER -An application that opens a phone device in this mode can change the state of the lamps, ringer, display, and hookswitch devices of the phone. Having owner privilege to a phone device automatically includes monitor privilege as well.

phoneSetDisplay

The phoneSetDisplay function causes the specified string to display on the specified open phone device.

Note Prior to Release 4.0, Cisco Unified Communications Manager messages that were passed to the phone would automatically overwrite any messages sent to the phone by using phoneSetDisplay(). In Cisco Unified Communications Manager 4.0, the message sent to the phone in the phoneSetDisplay() API remains on the phone until the phone is rebooted. If the application wants to clear the text from the display and see the Cisco Unified Communications Manager messages again, a NULL string, not spaces, should be passed in the phoneSetDisplay() API. In other words, the lpsDisplay parameter should be NULL and the dwSize should be set to 0.

Function Details

```
LONG phoneSetDisplay(
  HPHONE hPhone,
  DWORD dwRow,
  DWORD dwColumn,
  LPCSTR lpsDisplay,
  DWORD dwSize
);
```

Parameters

hPhone

A handle to the open phone device. The application must be the owner of the phone.

dwRow

The row position on the display where the new text displays.

dwColumn

The column position on the display where the new text displays.

lpsDisplay

A pointer to the memory location where the display content is stored. The display information must follow the format that is specified in the dwStringFormat member of the device capabilities for this phone.

dwSize

The size in bytes of the information to which lpsDisplay points.

phoneSetStatusMessages

The phoneSetStatusMessages function enables an application to monitor the specified phone device for selected status events.

See TAPI Phone Messages, on page 148 for supported messages.

Function Details

```
LONG phoneSetStatusMessages(
  HPHONE hPhone,
  DWORD dwPhoneStates,
  DWORD dwButtonModes,
  DWORD dwButtonStates
);
```

Parameters

hPhone

A handle to the open phone device to be monitored.

dwPhoneStates

These flags specify the set of phone status changes and events for which the application can receive notification messages. This parameter can have zero, one, or more bits set. This parameter uses the following PHONESTATE_ constants:

- PHONESTATE_OTHER -Phone status items other than those in the following list changed. The application should check the current phone status to determine which items changed.
- PHONESTATE_OWNER -The number of owners for the phone device changed.
- PHONESTATE_MONITORS -The number of monitors for the phone device changed.
- PHONESTATE_DISPLAY -The display of the phone changed.
- PHONESTATE_LAMP -A lamp of the phone changed.
- PHONESTATE_RINGMODE -The ring mode of the phone changed.
- PHONESTATE_SPEAKERHOOKSWITCH -The hookswitch state changed for this speakerphone.
- PHONESTATE_REINIT -Items changed in the configuration of phone devices. To become aware of these changes (as with the appearance of new phone devices), the application should reinitialize its use of TAPI. New phoneInitialize, phoneInitializeEx, and phoneOpen requests get denied until applications have shut down their usage of TAPI. The hDevice parameter of the PHONE_STATE message stays NULL for this state change because it applies to any line in the system. Because of the critical nature of PHONESTATE_REINIT, you cannot mask such messages, so the setting of this bit gets ignored, and the messages always get delivered to the application.
- PHONESTATE_REMOVED -Indicates that the service provider is removing the device from the system (most likely through user action, through a control panel or similar utility). A PHONE_CLOSE message on the device immediately follows a PHONE_STATE message with this value. Subsequent attempts to access the device prior to TAPI being reinitialized result in PHONEERR_NODEVICE being returned to the application. If a service provider sends a PHONE_STATE message that contains this value to TAPI, TAPI passes it along to applications that negotiated TAPI version 1.4 or later; applications that negotiated a previous TAPI version do not receive any notification.

dwButtonModes

The set of phone-button modes for which the application can receive notification messages. This parameter can have zero, one, or more bits set. This parameter uses the following PHONEBUTTONMODE_constants:

- PHONEBUTTONMODE_CALL -The button is assigned to a call appearance.
- PHONEBUTTONMODE_FEATURE -The button is assigned to requesting features from the switch, such as hold, conference, and transfer.
- PHONEBUTTONMODE_KEYPAD -The button is one of the 12 keypad buttons, ‘0’ through ‘9’, ‘*’, and ‘#’.
- PHONEBUTTONMODE_DISPLAY -The button is a “soft” button that is associated with the phone display. A phone set can have zero or more display buttons.

dwButtonStates

The set of phone-button state changes for which the application can receive notification messages. If the dwButtonModes parameter is zero, the system ignores dwButtonStates. If dwButtonModes has one or

more bits set, this parameter also must have at least one bit set. This parameter uses the following PHONEBUTTONSTATE_constants:

- PHONEBUTTONSTATE_UP -The button is in the “up” state.
- PHONEBUTTONSTATE_DOWN -The button is in the “down” state (pressed down).
- PHONEBUTTONSTATE_UNKNOWN -The up or down state of the button is unknown at this time but may become known later.
- PHONEBUTTONSTATE_UNAVAIL -The service provider does not know the up or down state of the button, and the state will not become known.

phoneShutdown

The phoneShutdown function shuts down the application usage of the TAPI phone abstraction.

Note If this function is called when the application has open phone devices, these devices are closed.

Function Details

```
LONG WINAPI phoneShutdown(
  HPHONEAPP hPhoneApp
);
```

Parameter

hPhoneApp

The application usage handle for TAPI.

Return Values

Returns zero if the request succeeds or a negative number if an error occurs. Possible return values follow:

PHONEERR_INVALAPPHANDLE, PHONEERR_NOMEM, PHONEERR_UNINITIALIZED, PHONEERR_RESOURCEUNAVAIL.

TAPI Phone Messages

Messages notify the application of asynchronous events. All messages get sent to the application through the message notification mechanism that the application specified in lineInitializeEx. The message always contains a handle to the relevant object (phone, line, or call), of which the application can determine the type from the message type. The following table describes TAPI Phone messages.

Table 6: TAPI Phone Messages

TAPI Phone Messages
PHONE_BUTTON, on page 149

TAPI Phone Messages
PHONE_CLOSE, on page 152
PHONE_CREATE, on page 152
PHONE_REMOVE, on page 153
PHONE_REPLY, on page 154
PHONE_STATE, on page 154

PHONE_BUTTON

The PHONE_BUTTON message notifies the application that button press monitoring is enabled if it has detected a button press on the local phone.

Function Details

```
PHONE_BUTTON
hPhone = (HPHONE) hPhoneDevice;
dwCallbackInstance = (DWORD) hCallback;
dwParam1 = (DWORD) idButtonOrLamp;
dwParam2 = (DWORD) ButtonMode;
dwParam3 = (DWORD) ButtonState;
```

Parameters

hPhone

A handle to the phone device.

dwCallbackInstance

The callback instance that is provided when the phone device for this application is opened.

dwParam1

The button/lamp identifier of the button that was pressed. Button identifiers zero through 11 always represent the KEYPAD buttons, with ‘0’ being button identifier zero, ‘1’ being button identifier 1 (and so on through button identifier 9), and with ‘*’ being button identifier 10, and ‘#’ being button identifier 11. Find additional information about a button identifier with phoneGetDevCaps, on page 134.

dwParam2

The button mode of the button. The button mode for each button ID gets listed as shown in the table below.

The TAPI service provider cannot detect button down or button up state changes. To conform to the TAPI specification, two messages are sent to simulate a down state followed by an up state in dwparam3.

This parameter uses the following PHONEBUTTONMODE_constants:

- PHONEBUTTONMODE_CALL -The button is assigned to a call appearance.

- PHONEBUTTONMODE_FEATURE -The button is assigned to requesting features from the switch, such as hold, conference, and transfer.
- PHONEBUTTONMODE_KEYPAD -The button is one of the 12 keypad buttons, ‘0’ through ‘9’, ‘*’, and ‘#’.
- PHONEBUTTONMODE_DISPLAY -The button is a soft button that is associated with the phone display. A phone set can have zero or more display buttons.

dwParam3

Specifies whether this is a button-down event or a button-up event. This parameter uses the following PHONEBUTTONSTATE_constants:

- PHONEBUTTONSTATE_UP -The button is in the up state.
- PHONEBUTTONSTATE_DOWN -The button is in the down state (pressed down).
- PHONEBUTTONSTATE_UNKNOWN -The up or down state of the button is not known at this time and may be known later.
- PHONEBUTTONSTATE_UNAVAIL -The service provider does not know the up or down state of the button, and the state cannot become known at a future time.

Button ID values of zero through 11 map to the keypad buttons as defined by TAPI. Values above 11 map to line and feature buttons. The low-order part of the DWORD specifies the feature. The high-order part of the DWORD specifies the instance number of that feature. The following table lists all possible values for the low-order part of the DWORD that corresponds to the feature.

Use the following expression to make the button ID:

ButtonID = (instance << 16) | featureID

The following table lists the valid phone button values.

Table 7: Phone Button Values

Value	Feature	Has Instance	Button Mode
0	Keypad button 0	No	Keypad
1	Keypad button 1	No	Keypad
2	Keypad button 2	No	Keypad
3	Keypad button 3	No	Keypad
4	Keypad button 4	No	Keypad
5	Keypad button 5	No	Keypad
6	Keypad button 6	No	Keypad
7	Keypad button 7	No	Keypad
8	Keypad button 8	No	Keypad
9	Keypad button 9	No	Keypad

Value	Feature	Has Instance	Button Mode
10	Keypad button '*'	No	Keypad
11	Keypad button '#'	No	Keypad
12	Last Number Redial	No	Feature
13	Speed Dial	Yes	Feature
14	Hold	No	Feature
15	Transfer	No	Feature
16	Forward All (for line one)	No	Feature
17	Forward Busy (for line one)	No	Feature
18	Forward No Answer (for line one)	No	Feature
19	Display	No	Feature
20	Line	Yes	Call
21	Chat (for line one)	No	Feature
22	Whiteboard (for line one)	No	Feature
23	Application Sharing (for line one)	No	Feature
24	T120 File Transfer (for line one)	No	Feature
25	Video (for line one)	No	Feature
26	Voice Mail (for line one)	No	Feature
27	Answer Release	No	Feature
28	Auto-answer	No	Feature
44	Generic Custom Button 1	Yes	Feature
45	Generic Custom Button 2	Yes	Feature
46	Generic Custom Button 3	Yes	Feature
47	Generic Custom Button 4	Yes	Feature
48	Generic Custom Button 5	Yes	Feature

PHONE_CLOSE

The PHONE_CLOSE message gets sent when an open phone device is forcibly closed as part of resource reclamation. The device handle is no longer valid after this message is sent.

Function Details

```
PHONE_CLOSE
hPhone = (HPHONE) hPhoneDevice;
dwCallbackInstance = (DWORD) hCallback;
dwParam1 = (DWORD) 0;
dwParam2 = (DWORD) 0;
dwParam3 = (DWORD) 0;
```

Parameters

hPhone

A handle to the open phone device that was closed. The handle is no longer valid after this message is sent.

dwCallbackInstance

The callback instance of the application that is provided on an open phone device.

dwParam1
Not used.
dwParam2
Not used.
dwParam3
Not used.

PHONE_CREATE

The PHONE_CREATE message gets sent to inform applications of the creation of a new phone device.

Note CTI Manager cluster support, extension mobility, change notification, and user addition to the directory can generate PHONE_CREATE events.

Function Details

```
PHONE_CREATE
hPhone = (HPHONE) hPhoneDevice;
dwCallbackInstance = (DWORD) 0;
dwParam1 = (DWORD) idDevice;
dwParam2 = (DWORD) 0;
dwParam3 = (DWORD) 0;
```

Parameters

hPhone

Not used.

dwCallbackInstance
Not used.
dwParam1

The dwDeviceID of the newly created device.

dwParam2
Not used.
dwParam3
Not used.

PHONE_REMOVE

The PHONE_REMOVE message gets sent to inform an application of the removal (deletion from the system) of a phone device. Generally, this method is not used for temporary removals, such as extraction of PCMCIA devices, but only for permanent removals in which the service provider would no longer report the device, if TAPI were reinitialized.

Note CTI Manager cluster support, extension mobility, change notification, and user deletion from the directory can generate PHONE_REMOVE events.

Function Details

```
PHONE_REMOVE
dwDevice = (DWORD) 0;
dwCallbackInstance = (DWORD) 0;
dwParam1 = (DWORD) dwDeviceID;
dwParam2 = (DWORD) 0;
dwParam3 = (DWORD) 0;
```

Parameters

dwDevice
Reserved. Set to zero.
dwCallbackInstance
Reserved. Set to zero.
dwParam1

Identifier of the phone device that was removed.

dwParam2
Reserved. Set to zero.
dwParam3
Reserved. Set to zero.

PHONE_REPLY

The TAPI PHONE_REPLY message gets sent to an application to report the results of function call that completed asynchronously.

Function Details

```
PHONE_REPLY
hPhone = (HPHONE) 0;
dwCallbackInstance = (DWORD) hCallback;
dwParam1 = (DWORD) idRequest;
dwParam2 = (DWORD) Status;
dwParam3 = (DWORD) 0;
```

Parameters

hPhone
Not used.

dwCallbackInstance

Returns the application callback instance.

dwParam1

The request identifier for which this is the reply.

dwParam2

The success or error indication. The application should cast this parameter into a LONG. Zero indicates success; a negative number indicates an error.

dwParam3
Not used.

PHONE_STATE

TAPI sends the PHONE_STATE message to an application whenever the status of a phone device changes.

Function Details

```
PHONE_STATE
hPhone = (HPHONE) hPhoneDevice;
dwCallbackInstance = (DWORD) hCallback;
dwParam1 = (DWORD) PhoneState;
dwParam2 = (DWORD) PhoneStateDetails;
dwParam3 = (DWORD) 0;
```

Parameters

hPhone

A handle to the phone device.

dwCallbackInstance

The callback instance that is provided when the phone device is opened for this application.

dwParam1

The phone state that changed. This parameter uses the following PHONESTATE_constants:

- PHONESTATE_OTHER -Phone-status items other than the following ones changed. The application should check the current phone status to determine which items changed.
- PHONESTATE_CONNECTED -The connection between the phone device and TAPI was just made. This happens when TAPI is first invoked or when the wire that connects the phone to the computer is plugged in while TAPI is active.
- PHONESTATE_DISCONNECTED -The connection between the phone device and TAPI just broke. This happens when the wire that connects the phone set to the computer is unplugged while TAPI is active.
- PHONESTATE_OWNER -The number of owners for the phone device changed.
- PHONESTATE_MONITORS -The number of monitors for the phone device changed.
- PHONESTATE_DISPLAY -The display of the phone changed.
- PHONESTATE_LAMP -A lamp of the phone changed.
- PHONESTATE_RINGMODE -The ring mode of the phone changed.
- PHONESTATE_HANDSETHOOKSWITCH -The hookswitch state changed for this speakerphone.
- PHONESTATE_REINIT -Items changed in the configuration of phone devices. To become aware of these changes (as with the appearance of new phone devices), the application should reinitialize its use of TAPI. The hDevice parameter of the PHONE_STATE message stays NULL for this state change as it applies to any of the phones in the system.
- PHONESTATE_REMOVED -Indicates that the device is being removed from the system by the service provider (most likely through user action, through a control panel or similar utility). Normally, a PHONE_CLOSE message on the device immediately follows a PHONE_STATE message with this value. Subsequent attempts to access the device prior to TAPI being reinitialized result in PHONEERR_NODEVICE being returned to the application. If a service provider sends a PHONE_STATE message that contains this value to TAPI, TAPI passes it along to applications that negotiated TAPI version 1.4 or later; applications that negotiated a previous API version do not receive any notification.
- PHONESTATE_SUSPEND -Indicates the phone unregisters as it enters Energywise DeepSleep/PowersavePlus mode.

dwParam2

Phone state-dependent information that details the status change. This parameter is not used if multiple flags are set in dwParam1 because multiple status items get changed. The application should invoke phoneGetStatus to obtain a complete set of information.

Parameter dwparam2 can comprise one of PHONESTATE_LAMP, PHONESTATE_DISPLAY, PHONESTATE_HANDSETHOOKSWITCH, or PHONESTATE_RINGMODE. Because the Cisco Unified TSP cannot differentiate among hook switches for handsets, headsets, or speaker, the PHONESTATE_HANDSETHOOKSWITCH value always gets used for hook switches.

If dwparam2 is PHONESTATE_LAMP, dwparam2 is the button ID that the PHONE_BUTTON message defines.

If dwParam1 is PHONESTATE_OWNER, dwParam2 contains the new number of owners.

If dwParam1 is PHONESTATE_MONITORS, dwParam2 contains the new number of monitors.

If dwParam1 is PHONESTATE_LAMP, dwParam2 contains the button/lamp identifier of the lamp that changed.

If dwParam1 is PHONESTATE_RINGMODE, dwParam2 contains the new ring mode.

If dwParam1 is PHONESTATE_HANDSET, SPEAKER, or HEADSET, dwParam2 contains the new hookswitch mode of that hookswitch device. This parameter uses the following PHONEHOOKSWITCHMODE_constants:

- PHONEHOOKSWITCHMODE_ONHOOK -The microphone and speaker both remain on hook for this device.
- PHONEHOOKSWITCHMODE_MICSPEAKER -The microphone and speaker both remain active for this device. The Cisco Unified TSP cannot distinguish among handsets, headsets, or speakers, so this value gets sent when the device is off hook.

If dw Param1 is PHONESTATE_SUSPEND, dwParam2 contains the reason EnergyWisePowerSavePlus when the phone unregisters as it enters EnergywiseDeepSleep.

dwParam3

The TAPI specification specifies that dwparam3 is zero; however, the Cisco Unified TSP will send the new lamp state to the application in dwparam3 to avoid the call to phoneGetLamp to obtain the state when dwparam2 is PHONESTATE_LAMP.

TAPI Phone Structures

This section describes the TAPI phone structures that Cisco Unified TSP supports:

.

Table 8: TAPI Phone Structures

TAPI Phone Structure
PHONECAPS Structure, on page 156
PHONEINITIALIZEEXPARAMS, on page 158
PHONEMESSAGE, on page 159
PHONESTATUS, on page 160
VARSTRING, on page 162

PHONECAPS Structure

This section lists the Cisco-set attributes for each member of the PHONECAPS structure. If the value of a structure member is device, line, or call specific, the list gives the value for each condition.

Members

dwProviderInfoSize
dwProviderInfoOffset

"Cisco Unified TSPxxx.TSP: Cisco IP PBX Service Provider Ver. X.X(x.x)" where the text before the colon specifies the file name of the TSP, and the text after "Ver. " specifies the version of the TSP.

dwPhoneInfoSize
dwPhoneInfoOffset

"DeviceType:[type]" where type specifies the device type that is specified in the Cisco Unified Communications Manager database.

dwPermanentPhoneID
dwPhoneNameSize
dwPhoneNameOffset

"Cisco Phone: [deviceName]" where deviceName specifies the name of the device in the Cisco Unified Communications Manager database.

dwStringFormat

STRINGFORMAT_ASCII

dwPhoneStates

PHONESTATE_OWNER |

PHONESTATE_MONITORS |

PHONESTATE_DISPLAY | (Not set for CTI Route Points)

PHONESTATE_LAMP | (Not set for CTI Route Points)

PHONESTATE_RESUME |

PHONESTATE_REINIT |

PHONESTATE_SUSPEND

dwHookSwitchDevs

PHONEHOOKSWITCHDEV_HANDSET (Not set for CTI Route Points)

dwHandsetHookSwitchModes

PHONEHOOKSWITCHMODE_ONHOOK | (Not set for CTI Route Points)

PHONEHOOKSWITCHMODE_MICSPEAKER | (Not set for CTI Route Points)

PHONEHOOKSWITCHMODE_UNKNOWN (Not set for CTI Route Points)

dwDisplayNumRows (Not set for CTI Route Points)
1

dwDisplayNumColumns

20 (Not set for CTI Route Points)

dwNumRingModes

3 (Not set for CTI Route Points)

dwPhoneFeatures (Not set for CTI Route Points)

PHONEFEATURE_GETDISPLAY |

PHONEFEATURE_GETLAMP |

PHONEFEATURE_GETRING |

PHONEFEATURE_SETDISPLAY |

PHONEFEATURE_SETLAMP

dwMonitoredHandsetHookSwitchModes

PHONEHOOKSWITCHMODE_ONHOOK | (Not set for CTI Route Points)

PHONEHOOKSWITCHMODE_MICSPEAKER (Not set for CTI Route Points)

PHONEINITIALIZEEXPARAMS

The PHONEINITIALIZEEXPARAMS structure contains parameters that are used to establish the association between an application and TAPI; for example, the application selected event notification mechanism. The phoneInitializeEx function uses this structure.

Structure Details

```
typedef struct phoneinitializeexparams_tag {
  DWORD  dwTotalSize;
  DWORD  dwNeededSize;
  DWORD  dwUsedSize;
  DWORD  dwOptions;
  union
  {
    HANDLE hEvent;
    HANDLE hCompletionPort;
  } Handles;
  DWORD dwCompletionKey;
} PHONEINITIALIZEEXPARAMS, FAR *LPPHONEINITIALIZEEXPARAMS;
```

Members

dwTotalSize

The total size, in bytes, that is allocated to this data structure.

dwNeededSize

The size, in bytes, for this data structure that is needed to hold all the returned information.

dwUsedSize

The size, in bytes, of the portion of this data structure that contains useful information.

dwOptions

One of the PHONEINITIALIZEEXOPTION_Constants. Specifies the event notification mechanism that the application wants to use.

hEvent

If dwOptions specifies PHONEINITIALIZEEXOPTION_USEEVENT, TAPI returns the event handle in this member.

hCompletionPort

If dwOptions specifies PHONEINITIALIZEEXOPTION_USECOMPLETIONPORT, the application must specify, in this member, the handle of an existing completion port that is opened by using CreateIoCompletionPort.

dwCompletionKey

If dwOptions specifies PHONEINITIALIZEEXOPTION_USECOMPLETIONPORT, the application must specify in this field a value that is returned through the lpCompletionKey parameter of GetQueuedCompletionStatus to identify the completion message as a telephony message.

PHONEMESSAGE

The PHONEMESSAGE structure contains the next message that is queued for delivery to the application. The phoneGetMessage function returns the following structure.

Structure Details

```
typedef struct phonemessage_tag {
  DWORD  hDevice;
  DWORD  dwMessageID;
  DWORD_PTR  dwCallbackInstance;
  DWORD_PTR  dwParam1;
  DWORD_PTR  dwParam2;
  DWORD_PTR  dwParam3;
} PHONEMESSAGE, FAR *LPPHONEMESSAGE;
```

Members

hDevice

A handle to a phone device.

dwMessageID

A phone message.

dwCallbackInstance

Instance data that is passed back to the application, which the application specified in phoneInitializeEx. TAPI does not interpret DWORD.

dwParam1

A parameter for the message.

dwParam2

A parameter for the message.

dwParam3

A parameter for the message.

Further Details

For details on the parameter values that are passed in this structure, see "TAPI Phone Messages, on page 148."

PHONESTATUS

The PHONESTATUS structure describes the current status of a phone device. The phoneGetStatus and TSPI_phoneGetStatus functions return this structure.

Device-specific extensions should use the DevSpecific (dwDevSpecificSize and dwDevSpecificOffset) variably sized area of this data structure.

Note The dwPhoneFeatures member is available only to applications that open the phone device with an API version of 2.0 or later.

Structure Details

```
typedef struct phonestatus_tag {
  DWORD  dwTotalSize;
  DWORD  dwNeededSize;
  DWORD  dwUsedSize;
  DWORD  dwStatusFlags;
  DWORD  dwNumOwners;
  DWORD  dwNumMonitors;
  DWORD  dwRingMode;
  DWORD  dwRingVolume;
  DWORD  dwHandsetHookSwitchMode;
  DWORD  dwHandsetVolume;
  DWORD  dwHandsetGain;
  DWORD  dwSpeakerHookSwitchMode;
  DWORD  dwSpeakerVolume;
  DWORD  dwSpeakerGain;
  DWORD  dwHeadsetHookSwitchMode;
  DWORD  dwHeadsetVolume;
  DWORD  dwHeadsetGain;
  DWORD  dwDisplaySize;
  DWORD  dwDisplayOffset;
  DWORD  dwLampModesSize;
  DWORD  dwLampModesOffset;
  DWORD  dwOwnerNameSize;
  DWORD  dwOwnerNameOffset;
  DWORD  dwDevSpecificSize;
  DWORD  dwDevSpecificOffset;
  DWORD  dwPhoneFeatures;
} PHONESTATUS, FAR *LPPHONESTATUS;
```

Members

dwTotalSize

The total size, in bytes, that is allocated to this data structure.

dwNeededSize

The size, in bytes, for this data structure that is needed to hold all the returned information.

dwUsedSize

The size, in bytes, of the portion of this data structure that contains useful information.

dwStatusFlags

Provides a set of status flags for this phone device. This member uses one of the PHONESTATUSFLAGS_Constants.

dwNumOwners

The number of application modules with owner privilege for the phone.

dwNumMonitors

The number of application modules with monitor privilege for the phone.

dwRingMode

The current ring mode of a phone device.

dwRingVolume
0x8000
dwHandsetHookSwitchMode

The current hookswitch mode of the phone handset. PHONEHOOKSWITCHMODE_UNKNOWN

dwHandsetVolume
0
dwHandsetGain
0
dwSpeakerHookSwitchMode

The current hookswitch mode of the phone speakerphone. PHONEHOOKSWITCHMODE_UNKNOWN

dwSpeakerVolume
0

dwSpeakerGain
0
dwHeadsetHookSwitchMode

The current hookswitch mode of the phone's headset. PHONEHOOKSWITCHMODE_UNKNOWN

dwHeadsetVolume
0
dwHeadsetGain
0
dwDisplaySize
dwDisplayOffset
0
dwLampModesSize
dwLampModesOffset
0
dwOwnerNameSize
dwOwnerNameOffset

The size, in bytes, of the variably sized field that contains the name of the application that is the current owner of the phone device and the offset, in bytes, from the beginning of this data structure. The name is the application name that the application provides when it invokes with phoneInitialize or

phoneInitializeEx. If no application name was supplied, the application's filename is used instead. If the phone currently has no owner, dwOwnerNameSize is zero.

dwDevSpecificSize
dwDevSpecificOffset

Application can send XSI data to phone by using DeviceDataPassThrough device-specific extension. Phone can pass back data to Application. The data is returned as part of this field. The format of the data is as follows:

```
struct PhoneDevSpecificData
{    DWORD m_DeviceDataSize ; // size of device data
    DWORD m_DeviceDataOffset ; // offset from PHONESTATUS
structure
    // this will follow the actual variable length device data.
}
```

dwPhoneFeatures

The application negotiates an extension version > = 0x00020000. The following features are supported:

- PHONEFEATURE_GETDISPLAY
- PHONEFEATURE_GETLAMP
- PHONEFEATURE_GETRING
- PHONEFEATURE_SETDISPLAY
- PHONEFEATURE_SETLAMP

VARSTRING

The VARSTRING structure returns variably sized strings. The line device class and the phone device class both use it.

Note No extensibility exists with VARSTRING.

Structure Details

```
typedef struct varstring_tag {
  DWORD  dwTotalSize;
  DWORD  dwNeededSize;
  DWORD  dwUsedSize;
  DWORD  dwStringFormat;
  DWORD  dwStringSize;
  DWORD  dwStringOffset;
} VARSTRING, FAR *LPVARSTRING;
```

Members

dwTotalSize

The total size, in bytes, that is allocated to this data structure.

dwNeededSize

The size, in bytes, for this data structure that is needed to hold all the returned information.

dwUsedSize

The size, in bytes, of the portion of this data structure that contains useful information.

dwStringFormat

The format of the string. This member uses one of the STRINGFORMAT_Constants.

dwStringSize
dwStringOffset

The size, in bytes, of the variably sized device field that contains the string information and the offset, in bytes, from the beginning of this data structure.

If a string cannot be returned in a variable structure, the dwStringSize and dwStringOffset members get set in one of the following ways:

- dwStringSize and dwStringOffset members both get set to zero.
- dwStringOffset gets set to nonzero and dwStringSize gets set to zero.
- dwStringOffset gets set to nonzero, dwStringSize gets set to 1, and the byte at the given offset gets set to zero.]

Wave Functions

The AVAudio32.dll implements the wave interfaces to the Cisco wave drivers. The system supports all APIs for input and output waveform devices.

Cisco TSP 8.0 includes Cisco Media Driver, a new and innovative way for TAPI-based applications, to provide media interaction. Cisco TSP 8.0(1) includes support for Cisco Media Driver and Cisco Wave Driver. Only one driver is active at any given time. For more information, see Cisco TSP Media Driver.

Table 9: Wave Functions

Wave functions
waveInAddBuffer, on page 164
waveInClose, on page 164
waveInGetID, on page 165
waveInGetPosition, on page 165
waveInOpen, on page 166
waveInPrepareHeader, on page 167
waveInReset, on page 168
waveInStart, on page 168

Wave functions
waveInUnprepareHeader, on page 168
waveOutClose, on page 169
waveOutGetDevCaps, on page 169
waveOutGetID, on page 170
waveOutGetPosition, on page 170
waveOutOpen, on page 171
waveOutPrepareHeader, on page 172
waveOutReset, on page 172
waveOutUnprepareHeader, on page 173
waveOutWrite, on page 173

waveInAddBuffer

The waveInAddBuffer function sends an input buffer to the given waveform-audio input device. When the buffer is filled, the application receives notification.

Function Details

```
MMRESULT waveInAddBuffer(
  HWAVEIN hwi,
  LPWAVEHDR pwh,
  UINT cbwh
);
```

Parameters

hwi

Handle of the waveform-audio input device.

pwh

Address of a WAVEHDR structure that identifies the buffer.

cbwh

Size, in bytes, of the WAVEHDR structure.

waveInClose

The waveInClose function closes the given waveform-audio input device.

Function Details

```
MMRESULT waveInClose(
  HWAVEIN hwi
);
```

Parameter

hwi

Handle of the waveform-audio input device. If the function succeeds, the handle no longer remains valid after this call.

waveInGetID

The waveInGetID function gets the device identifier for the given waveform-audio input device.

This function gets supported for backward compatibility. New applications can cast a handle of the device rather than retrieving the device identifier.

Function Details

```
MMRESULT waveInGetID(
  HWAVEIN hwi,
  LPUINT puDeviceID
);
```

Parameters

hwi

Handle of the waveform-audio input device.

puDeviceID

Address of a variable to be filled with the device identifier.

waveInGetPosition

The waveInGetPosition function retrieves the current input position of the given waveform-audio input device.

Function Details

```
MMRESULT waveInGetPosition(
  HWAVEIN hwi,
  LPMMTIME pmmt,
  UINT cbmmt
);
```

Parameters

hwi

Handle of the waveform-audio input device.

pmmt

Address of the MMTIME structure.

cbmmt

Size, in bytes, of the MMTIME structure.

waveInOpen

The waveInOpen function opens the given waveform-audio input device for recording.

Function Details

```
MMRESULT waveInOpen(
  LPHWAVEIN phwi,
  UINT uDeviceID,
  LPWAVEFORMATEX pwfx,
  DWORD dwCallback,
  DWORD dwCallbackInstance,
  DWORD fdwOpen
);
```

Parameters

phwi

Address that is filled with a handle that identifies the open waveform-audio input device. Use this handle to identify the device when calling other waveform-audio input functions. This parameter can be NULL if WAVE_FORMAT_QUERY is specified for fdwOpen.HDR structure.

uDeviceID

Identifier of the waveform-audio input device to open. It can be either a device identifier or a handle of an open waveform-audio input device. You can use the following flag instead of a device identifier:

WAVE_MAPPER -The function selects a waveform-audio input device that is capable of recording in the specified format.

pwfx

Address of a WAVEFORMATEX structure that identifies the desired format for recording waveform-audio data. You can free this structure immediately after waveInOpen returns.

Note The formats that the TAPI Wave Driver supports include a 16-bit PCM at 8000 Hz, 8-bit mulaw at 8000 Hz, and 8-bit alaw at 8000 Hz.

dwCallback

Address of a fixed callback function, an event handle, a handle to a window, or the identifier of a thread to be called during waveform-audio recording to process messages that are related to the progress of recording. If no callback function is required, this value can specify zero. For more information on the callback function, see waveInProc in the TAPI API.

dwCallbackInstance

User-instance data that is passed to the callback mechanism. This parameter is not used with the window callback mechanism.

fdwOpen

Flags for opening the device. The following values definitions apply:

- CALLBACK_EVENT -The dwCallback parameter specifies an event handle.
- CALLBACK_FUNCTION -The dwCallback parameter specifies a callback procedure address.
- CALLBACK_NULL -No callback mechanism. This represents the default setting.
- CALLBACK_THREAD -The dwCallback parameter specifies a thread identifier.
- CALLBACK_WINDOW -The dwCallback parameter specifies a window handle.
- WAVE_FORMAT_DIRECT -If this flag is specified, the A driver does not perform conversions on the audio data.
- WAVE_FORMAT_QUERY -The function queries the device to determine whether it supports the given format, but it does not open the device.
- WAVE_MAPPED -The uDeviceID parameter specifies a waveform-audio device to which the wave mapper maps.

waveInPrepareHeader

The waveInPrepareHeader function prepares a buffer for waveform-audio input.

Function Details

```
MMRESULT waveInPrepareHeader(
  HWAVEIN hwi,
  LPWAVEHDR pwh,
  UINT cbwh
);
```

Parameters

hwi

Handle of the waveform-audio input device.

pwh

Address of a WAVEHDR structure that identifies the buffer to be prepared.

cbwh

Size, in bytes, of the WAVEHDR structure.

waveInReset

The waveInReset function stops input on the given waveform-audio input device and resets the current position to zero. All pending buffers get marked as done and get returned to the application.

Function Details

```
MMRESULT waveInReset(
  HWAVEIN hwi
);
```

Parameter

hwi

Handle of the waveform-audio input device.

waveInStart

The waveInStart function starts input on the given waveform-audio input device.

Function Details

```
MMRESULT waveInStart(
  HWAVEIN hwi
);
```

Parameter

hwi

Handle of the waveform-audio input device.

waveInUnprepareHeader

The waveInUnprepareHeader function cleans up the preparation that the waveInPrepareHeader function performs. This function must be called after the device driver fills a buffer and returns it to the application. You must call this function before freeing the buffer.

Function Details

```
MMRESULT waveInUnprepareHeader(
  HWAVEIN hwi,
  LPWAVEHDR pwh,
  UINT cbwh
);
```

Parameters

hwi

Handle of the waveform-audio input device.

pwh

Address of a WAVEHDR structure that identifies the buffer to be cleaned up.

cbwh

Size, in bytes, of the WAVEHDR structure.

waveOutClose

The waveOutClose function closes the given waveform-audio output device.

Function Details

```
MMRESULT waveOutClose(
  HWAVEOUT hwo
);
```

Parameter

hwo

Handle of the waveform-audio output device. If the function succeeds, the handle no longer remains valid after this call.

waveOutGetDevCaps

The waveOutGetDevCaps function retrieves the capabilities of a given waveform-audio output device.

Function Details

```
MMRESULT waveOutGetDevCaps(
  UINT uDeviceID,
  LPWAVEOUTCAPS pwoc,
  UINT cbwoc
);
```

Parameters

uDeviceID

Identifier of the waveform-audio output device. It can be either a device identifier or a handle of an open waveform-audio output device.

pwoc

Address of a WAVEOUTCAPS structure that is to be filled with information about the capabilities of the device.

cbwoc

Size, in bytes, of the WAVEOUTCAPS structure.

waveOutGetID

The waveOutGetID function retrieves the device identifier for the given waveform-audio output device.

This function gets supported for backward compatibility. New applications can cast a handle of the device rather than retrieving the device identifier.

Function Details

```
MMRESULT waveOutGetID(
  HWAVEOUT hwo,
  LPUINT puDeviceID
);
```

Parameters

hwo

Handle of the waveform-audio output device.

puDeviceID

Address of a variable to be filled with the device identifier.

waveOutGetPosition

The waveOutGetPosition function retrieves the current playback position of the given waveform-audio output device.

Function Details

```
MMRESULT waveOutGetPosition(
  HWAVEOUT hwo,
  LPMMTIME pmmt,
  UINT cbmmt
);
```

Parameters

hwo

Handle of the waveform-audio output device.

pmmt

Address of an MMTIME structure.

cbmmt

Size, in bytes, of the MMTIME structure.

waveOutOpen

The waveOutOpen function opens the given waveform-audio output device for playback.

Function Details

```
MMRESULT waveOutOpen(
  LPHWAVEOUT phwo,
  UINT uDeviceID,
  LPWAVEFORMATEX pwfx,
  DWORD dwCallback,
  DWORD dwCallbackInstance,
  DWORD fdwOpen
);
```

Parameters

phwo

Address that is filled with a handle that identifies the open waveform-audio output device. Use the handle to identify the device when other waveform-audio output functions are called. This parameter might be NULL if the WAVE_FORMAT_QUERY flag is specified for fdwOpen.

uDeviceID

Identifier of the waveform-audio output device to open. It can be either a device identifier or a handle of an open waveform-audio input device. You can use the following flag instead of a device identifier:

WAVE_MAPPER -The function selects a waveform-audio output device that is capable of playing the given format.

pwfx

Address of a WAVEFORMATEX structure that identifies the format of the waveform-audio data to be sent to the device. You can free this structure immediately after passing it to waveOutOpen.

Note The formats that the TAPI Wave Driver supports include 16-bit PCM at 8000 Hz, 8-bit mulaw at 8000 Hz, and 8-bit alaw at 8000 Hz.

dwCallback

Address of a fixed callback function, an event handle, a handle to a window, or the identifier of a thread to be called during waveform-audio playback to process messages that are related to the progress of the playback. If no callback function is required, this value can specify zero. For more information on the callback function, see waveOutProc in the TAPI API.

dwCallbackInstance

User-instance data that is passed to the callback mechanism. This parameter is not used with the window callback mechanism.

fdwOpen

Flags for opening the device. The following value definitions apply:

- CALLBACK_EVENT -The dwCallback parameter represents an event handle.

- CALLBACK_FUNCTION -The dwCallback parameter specifies a callback procedure address.
- CALLBACK_NULL -No callback mechanism. This value specifies the default setting.
- CALLBACK_THREAD -The dwCallback parameter represents a thread identifier.
- CALLBACK_WINDOW -The dwCallback parameter specifies a window handle.
- WAVE_ALLOWSYNC -If this flag is specified, a synchronous waveform-audio device can be opened. If this flag is not specified while a synchronous driver is opened, the device will fail to open.
- WAVE_FORMAT_DIRECT -If this flag is specified, the ACM driver does not perform conversions on the audio data.
- WAVE_FORMAT_QUERY -If this flag is specified, waveOutOpen queries the device to determine whether it supports the given format, but the device does not actually open.
- WAVE_MAPPED -If this flag is specified, the uDeviceID parameter specifies a waveform-audio device to which the wave mapper maps.

waveOutPrepareHeader

The waveOutPrepareHeader function prepares a waveform-audio data block for playback.

Function Details

```
MMRESULT waveOutPrepareHeader(
  HWAVEOUT hwo,
  LPWAVEHDR pwh,
  UINT cbwh
);
```

Parameters

hwo

Handle of the waveform-audio output device.

pwh

Address of a WAVEHDR structure that identifies the data block to be prepared.

cbwh

Size, in bytes, of the WAVEHDR structure.

waveOutReset

The waveOutReset function stops playback on the given waveform-audio output device and resets the current position to zero. All pending playback buffers get marked as done and get returned to the application.

Function Details

```
MMRESULT waveOutReset(
  HWAVEOUT hwo
);
```

Parameter

hwo

Handle of the waveform-audio output device.

waveOutUnprepareHeader

The waveOutUnprepareHeader function cleans up the preparation that the waveOUtPrepareHeader function performs. Ensure this function is called after the device driver is finished with a data block. You must call this function before freeing the buffer.

Function Details

```
MMRESULT waveOutUnprepareHeader(
  HWAVEOUT hwo,
  LPWAVEHDR pwh,
  UINT cbwh
);
```

Parameters

hwo

Handle of the waveform-audio output device.

pwh

Address of a WAVEHDR structure that identifies the data block to be cleaned up.

cbwh

Size, in bytes, of the WAVEHDR structure.

waveOutWrite

The waveOutWrite function sends a data block to the given waveform-audio output device.

Function Details

```
MMRESULT waveOutWrite(
  HWAVEOUT hwo,
  LPWAVEHDR pwh,
  UINT cbwh
);
```

Parameters

hwo

Handle of the waveform-audio output device.

pwh

Address of a WAVEHDR structure that contains information about the data block.

cbwh

Size, in bytes, of the WAVEHDR structure.