UNIFY Harmonize your enterprise

OpenScape Business

Description

CallBridge Collection V2

Programmer's Guide

Supplement to the Microsoft Windows Telephony Application

Version 2.2

Definitions

HowTo

A HowTo describes the configuration of an feature within the administration of the system. It addresses primarily trained administrators.

Tutorial

Within the tutorials procedures for installation, administration and operation of specific devices, applications or 3rd party systems, which are connected to the system, are described. The tutorial addresses primarily trained administrators.

Description

A description shows the implementation of interfaces, protocols and APIs or the interworking of specific components of OpenScape Business

Table of Contents

Table of History

Date	Version	Changes
December 2001	Version 1.0	First Release
November 2004	Version 2.0	Add-On for HiPath 4000 V2.0
May 2011	Version 2.1	New copyright
December 2013	Version 2.2	Revision of preface and chapter 1, Unify Re-brand,

1 Preface

This guide is a supplement to the *Microsoft Windows Version 3.1 Telephony Programmer's Guide Version 1.0,* the *Implementation of Windows Telephony in Windows 95* specification and the *Win32 Implementation of Windows Telephony (TAPI 2.0)*.

The ***CallBridgeCollection*** **- CorNet TS TAPI Service Provider (CorNet TS TSP)** - follows the philosophy of providing open interfaces for Computer Telephony Integration (CTI) applications. It allows TAPI-compliant applications to run with the following switches:

- OpenScape Business, OpenScape Office MX/LX, HiPath 3000
- OpenScape 4000, HiPath 4000

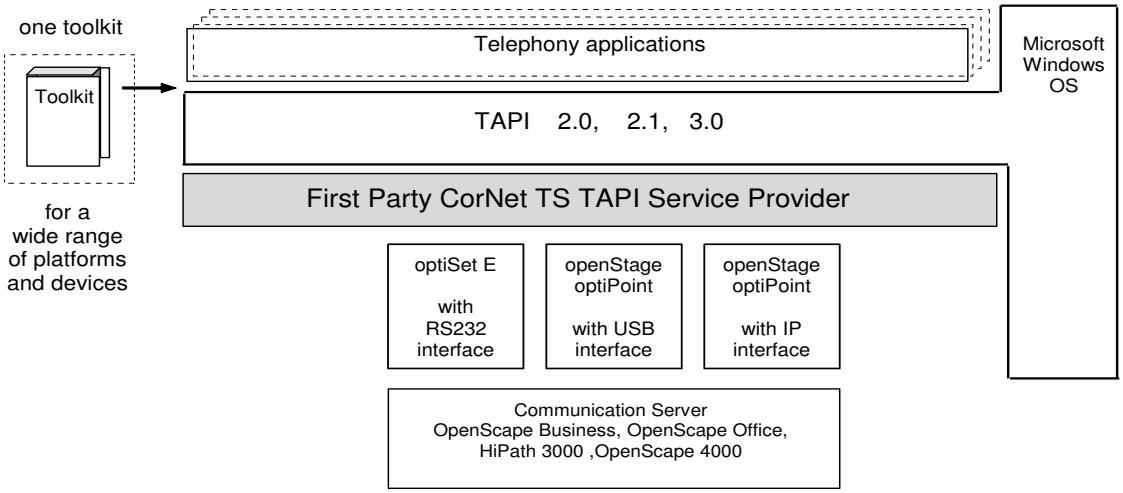
Please note:

Older switches and devices which have been replaced by newer versions or which have been phased out are still supported. Within this programmer´s guide in some tables the "old" systems and devices are referred. The successor systems and devices within the communication platform family offers the same feature set. For details have a look at the technical release note of CallBridge Collection V2.

CallBridge Collection resides on a user's personal computer and provides telephony service providers for the following different connection possibilities:

- CallBridge IP controls external phone devices of the openStage HFA telephone series via a IP connection.
- CallBridge TU controls external phone devices of the OpenStage 30/40 T series via a USB connection. The relevant USB drivers are content of the download package.
- CallBridge TA controls external phone devices of the *optiSet* telephone series with data/control adapter via a serial COM connection.

Figure P-1. CallBridge platforms



1.1 Who Should Read This Guide

This guide provides information that is needed to develop Microsoft TAPI applications for personal computers linked to a communication system of Unify. It is assumed that the application developer using this guide is familiar with TAPI and, to a certain extent, with features and functionality of the Unify communication system.

1.2 Sample Applications and Test Tools

For programmers who are still unfamiliar with programming on the Telephony Application Programming Interface, we suggest that they take a look at the simple dialer program *Dialer.exe,* which is available as source code in the Microsoft TAPI SDK.

As an all-purpose test tool for telephony applications we suggest the use of the TAPI Browser program *Tb20.exe*, which is available free of charge from Microsoft.

1.3 For More Information

- *Microsoft Windows Version 3.1 Telephony Programmer's Guide Version 1.0*. Describes TAPI functions and usage.
- *Implementation of Windows Telephony in Windows 95.* Describes TAPI enhancements for Windows 95.
- *Win32 Implementation of Windows Telephony (TAPI 2.0)*. Describes TAPI enhancements for Windows NT.

Consult your TAPI SDK for information on Microsoft's TAPI development support.

1.4 Requirements for Application Developers

All telephony application developers who's applications make use of Unify device specific extensions should have access to a Unify communication platform with OpenStage HFA, OpenStage T or *optiPoint 4xx, optiPoint 500* telephones. A current service contract for operational system support of Unify communication platform is also recommended.

1.5 Certification of Applications

Application developers may apply for the **"OpenScaoe Ready"** certification. Unify will issue this certificate under the following conditions:

- Certifications will be issued in the order in which applications for certification are received and are subject to the availability of certifying personnel.
- All required equipment must be supplied by the party applying for certification with the exception of Unify communication platform and telephones.
- The party applying for certification agrees to pay for the cost of the certification. Standard Unify rates apply. The applying party explicitely agrees to these terms. All charges will be billed at the end of the certification process.
- Unify reserves the right to end the certification process without a positive result, e.g. because of error conditions that are not caused by Unify components or lack of support by the party applying for certification.

- The *Product Test Report* will be issued following a successful test. The test report confirms the application's compliance with the interface and does not attempt to judge the overall quality of the application. Product management will report positive *Product Test Reports* to the Unify sales and marketing departments.

1.6 CallBridge Collection Product Management

For question about the handling of CallBridge in general, please contact:

OpenScapeBusiness@unify.com

1.7 CallBridge Collection developers support and certification

For developers support and certification you have to enter the Technology Partner Program of Unify. For details please contact:

technology.partners@unify.com

2 Product Overview

General CallBridge characteristics:

- Service provider for first party telephony applications.
- Voice-only service provider.
- Operates in a phone-centric connectivity model.
- Supports one phone device, which has to be connected via serial COM, USB or IP interface.
- Supports one line device.
- Supports device-specific extensions.
- Does not support media, digit, or tone monitoring.
- Does not support media streams.
- Runs with **TAPI Versions 2.x and 3.0**.
- Supports applications written for **TAPI Versions 1.3**, **1.4, 2.x and 3.0**.

2.1 Hardware and Software Requirements

For supported hard- and software have a look at the technical release note of the used CallBride collection SW-version.

2.1.1 CallBridge Collection and TAPI

The following section describes how CallBridge Collection approaches the implementation of some general concepts of TAPI.

2.1.2 Dialable Addresses

Applications may store directory numbers in canonical or dialable format. TAPI contains operations (address translation) to take addresses in the canonical format and convert them to dialable format. All addresses that appear at the TSP have been translated to dialable address format.

CallBridge Collection supports only the *DialableNumber* portion of a dialable address. The *Subaddress* and *Name* fields of the dialable address are ignored. If the **|, ^,** or **CRLF** characters are encountered in the dialable address, any information following these characters is ignored and not reported as invalid.

A *DialableNumber* can consist of the digits and modifiers *** , # 0-9**. A comma in the dialable number causes a pause in dialing. The duration of the pause is device specific and is reported in the LINEDEVCAPS data structure. Multiple commas may be used to provide longer pauses.

The characters **A-D ! ; P p T t W w @ $?** are ignored and will not cause an error if used.

2.1.3 Asynchronous Requests

CallBridge Collection allows multiple asynchronous requests for each line or phone to be outstanding at any time.

2.1.4 Device Availability

The ***open*** state of a device indicates the availability of that device to accept and process commands.

The *in service* state of that communication with the attached device is represented by the **LINEDEVSTATUS/dwDevStatusFlags/**LINEDEVSTATUSFLAGS_INSERVICE bit and the **PHONESTATUS/dwStatusFlags/**PHONESTATUSFLAGS_SUSPENDED bit.

When a **lineOpen** command causes the establishment of communication with the attached phone device to be initiated, the initialization and handshake procedure with the phone may take a few seconds. The application will receive a **LINE_LINEDEVSTATE** (LINEDEVSTATE_INSERVICE) and a **PHONE_STATE** (PHONESTATE_RESUME) message when the communication is successfully established. These messages are sent because the device states have changed from one state to another.

If the communication to the phone device could not be established after internal timeout, CallBridge Collection closes the opened line-/phoneDevice automatically. If the communication to the phone device is already established when the application issues the **lineOpen/phoneOpen** command, the application cannot expect a message reporting the *in service* state of the device since the device state did not change. After a successful **lineOpen/phoneOpen** command, the application should always query the device state via the **lineGetDevStatus** and/or **phoneGetStatus** command.

If **lineGetDevStatus** reports INSERVICE==*false* or **phoneGetStatus** reports SUSPENDED==*true*, the application should wait for the **LINE_LINEDEVSTATE/ PHONE_STATE** message which indicates *in service* before proceeding with commands that require communication with the phone device or line device.

> **Note:** some commands, such as **lineGetDevCaps** and **phoneGetDevCaps**, do not require the device to be *in service*. Other commands, such as those that control the device, do require the device to be *in service*.

If a device is not in the appropriate state to process a command when that command is received, the OPERATIONFAILED error code will be returned. Tables later in this chapter (refer to “TAPI Functions Supported by CallBridge Collection”) indicate for each function whether the *in service* state of the device is relevant or not.

2.1.5 Phone Buttons

If an application does not select an extension, no feature phone buttons may be accessed using **lineDevSpecificFeature.** An application that needs to access only the TAPI defined PHONEBUTTONFUNCTIONs must still successfully negotiate at least the lowest available extension version (currently 0x00010001).

An *optiPoint 500 Standard* for example which is configured for use with CallBridge TU on a Hicom 300/HiPath 4000 is always treated as having **four** *virtual* add-on units. CallBridge TU reports these to the CBX as if they were physically present. The buttons for add-on units 1 and 2 must be configured in the CBX and also during the CallBridge Collection installation. Add-on units 3 and 4 have fixed button sets and do not have to be configured.

This allows the application access to the 12 keypad buttons and the 17 feature buttons on the phone plus the 32 buttons on the add-on units. In addition, one or two *real* add-on units may be connected to the phone for a total of 65 or 81 feature buttons. Including the 12 keypad buttons, the total number of application controlable keys is 93.

The *virtual* add-on units are always configured as unit 3 and unit 4. The feature keys assigned to these units are fixed and may not be reconfigured by the Hicom/HiPath administrators. Starting with HiPath 4000 V2.0 it is not necessary to use add-on units to get access to more features. During the initialisation process between TAPI Service Provider and the CBX, all available features of this port (depending on the user level) are reported automatically.

The Hicom 100 E and Hicom 150 E do not support virtual add-on units. Starting with Hicom 150 E Office V2.2 it is not necessary to use add-on units to get access to more features. During the initialisation process between TAPI Service Provider and the CBX, all available features of this port (depending on the user level) are reported automatically.

2.2 Device-Specific Extensions of CallBridge Collection

The following paragraphs describe the device-specific features, messages, and data supported by CallBridge Collection.

2.2.1 Device-Specific Features

An application calls **lineDevSpecificFeature** with the PHONEBUTTONFUNCTION value of the feature it wants to invoke. (The PHONEBUTTONFUNCTION data type has been extended with Unify device-specific button functions).

An application must call the **phoneGetDevCaps** function to query the availability of features on the phone buttons. The **phoneGetDevCaps** function reports back all features and their respective button numbers on the phone.

The success of these feature invocations is highly context-specific. For many features, such as *transfer*, *conference* or *pickup,* the success of the activation will be reported back in a functional manner with an appropriate **LineEvent**.

For features which do not correspond to one of the TAPI *supplementary services*, CallBridge Collection does not provide an explicit indication of feature success or failure. The application must know how to interpret success or failure of the feature based on asynchronous information from the phone, such as display strings, LED indications, or state changes.

CallBridge Collection supports the function **lineDevSpecific** for different device, CBX and TSP specific functions and/or features, e.g. sending dialable strings to the phone at any time, as an addition to the functions **lineDial**, **lineMakeCall**, and **lineGenerateDigits**, which cannot be used in some call states. Please refer to “SIEMENS_LINE_DEV_SPECIFIC” in chapter 9 to get additional informations about the available functions/features.

The *capability* of a device to invoke a particular device-specific feature is given in the device-specific portion of the **LINEDEVCAPS** data structure. The current *availability* of a device-specific feature is given in the device-specific portion of the **LINEDEVSTATUS** data structure. The availability of the functions **lineDevSpecific** and **lineDevSpecificFeature** is given in the **dwLineFeatures** field of **LINEDEVSTATUS.**

2.2.2 Device-Specific Phone Button Extensions

CallBridge Collection contains a mechanism to invoke SIEMENS device-specific features.

In addition to the TAPI defined PHONEBUTTONFUNCTIONS, an application that selects an extension also has access to **all** buttons that are configured for a phone and its add-on units, including the Unify specific phone device buttons, e.g. the service menu and volume control

buttons. The Unify specific constants can be accessed by using **lineDevSpecificFeature** with the SIEMENSBUTTONFUNCTION constants for the selected button function.

2.2.3 Device-specific Data

The format of device-specific information follows the TAPI model. Fixed size fields are provided first, starting at **dwDeviceSpecificOffset** of the buffer. Variable length fields are defined via **dwSize** and **dwOffset** fields within the device-specific fixed area, and then the actual data is provided starting at the specified offset. The format of the device-specific data is defined for each data type.

CallBridge Collection defines the following device-specific data (see also chapter 9 *Device-Specific Data Structures and Constants*) :

Table 1-1. Device-specific Data

Data Structure	Device-specific Data
LINEDEVCAPS	• The types of requests that may be invoked via **lineDevSpecific**. • The features that may be invoked via **lineDevSpecificFeature**. Note: the dynamic availability of device-specific features is provided in the **LINEDEVSTATUS** data structure.
LINEDEVSTATUS	• Device-specific features that are currently available to be invoked via **lineDevSpecificFeature** (based on current device state).
PHONECAPS	• PHONEBUTTONFUNCTIONs and SIEMENSBUTTONFUNCTIONs that may be invoked via **lineDevSpecificFeature** are listed starting at **dwButtonFunctionOffset**. [1]
PHONESTATUS	• The programming mode of a phone (Service Menu is active/inactive)

[1] If no extension was selected, **PHONECAPS** reports only TAPI defined features of type PHONEBUTTONFUNCTION. All features of the type SIEMENSBUTTONFUNCTION will be reported as **PHONEBUTTONFUNCTION_UNKNOWN**

2.2.4 Service Menu Considerations

A phone can enter into *programming* mode when the *service menu* key is pressed.

- Because of the tree structure of the menu options, the menu selection cannot be functionally controlled. CallBridge Collection has no functional knowledge of the currently selected menu item.
- An application has the option of duplicating the phone's display on the PC screen and implementing the menu control buttons on the user interface. This gives the user the option to control all menu items from the PC user interface without ever touching a button on the phone.
- The menu keys are available as **SIEMENSBUTTONFUNCTION** key extensions.

The service menu active status is reported as a **PHONE_STATE** (PHONESTATE_DEVSPECIFIC) event and in the form of a steady lampmode state for the service menu key. After receiving the **PHONE_STATE** (PHONESTATE_DEVSPECIFIC) event, the application may call **phoneGetStatus** to query the current state of the *service menu*.

2.3 TAPI Functions Supported by CallBridge Collection

The following tables list the TAPI functions supported by CallBridge Collection and show the TSPI function that corresponds to each TAPI function.

The tables also indicate whether a device must be in service for the function to complete successfully.

An entry of ***none*** in the column **Corresponding TSPI Function** indicates that the function is supported by TAPI.

A number of TAPI functions are implemented in both Unicode (W) and ANSI versions. Applications may call either the W version or the generic version (no "W" suffix).

2.3.1 Basic Line Device Services

Table 1-2. Support of Basic Line Device Services

TAPI Function	Corresponding TSPI Function	Supported by CallBridge TU	Device must be in service
lineInitializeEx[1]	TSPI_lineNegotiateTSPIVersion TSPI_providerInit	**yes**	no
lineShutdown [1]	TSPI_providerShutdown	**yes**	no
lineNegotiateAPIVersion [1]	TSP_lineGetExtensionID	**yes**	no
lineOpen [1]	TSPI_lineOpen TSPI_lineConditionalMediaDetection TSPI_lineSetDefaultMediaDetection TSPI_lineSelectExtVersion	**yes**	no
lineClose	TSPI_lineClose	**yes**	no
lineAnswer	TSPI_lineAnswer	**yes**	yes
lineConfigDialog	TSPI_lineConfigDialog or TUISPI_lineConfigDialog	**yes**	no
lineConfigDialogEdit	TSPI_lineConfigDialogEdit or TUISPI_lineConfigDialogEdit	**yes**	no
lineDeallocateCall	TSPI_lineCloseCall	**yes**	yes
lineDial	TSPI_lineDial	**yes**	yes
lineDrop	TSPI_lineDrop	**yes**	yes
lineGetAddressCaps	TSPI_lineGetAddressCaps	**yes**	no
lineGetAddressID	TSPI_lineGetAddressID	**yes**	no
lineGetAddressStatus	TSPI_lineGetAddressStatus	**yes**	no
lineGetCallInfo	TSPI_lineGetCallInfo	**yes**	yes
lineGetCallStatus	TSPI_lineGetCallStatus	**yes**	yes
lineGetConfRelatedCalls	none	**n/a**	yes

lineGetCountry	none	**n/a**	no
lineGetDevCaps	TSPI_lineGetDevCaps	**yes**	no
lineGetDevConfig	TSPI_lineGetDevConfig	**no**	n/a
lineGetIcon	TSPI_lineGetIcon	**yes**	no
lineGetID	TSPI_lineGetID	**yes**	no
lineGetLineDevStatus	TSPI_lineGetLineDevStatus	**yes**	no
lineGetMessage	none	**n/a**	no
lineGetNewCalls	TSPI_lineGetCallAddressID TSPI_lineGetNumAddressIDs	**yes**	yes
lineGetNumRings [1]	TSPI_lineGetNumAddressIDs	**yes**	no
lineGetRequest	none	**n/a**	no
lineGetStatusMessages	none	**n/a**	no
lineGetTranslateCaps	none	**n/a**	no
lineHandoff	none	**n/a**	yes
lineMakeCall	TSPI_lineMakeCall	**yes**	yes
lineRegisterRequestRecipient	none	**n/a**	no
lineSetAppSpecific	TSPI_lineSetAppSpecific	**yes**	yes
lineSetCallPrivilege	none	**n/a**	yes
lineSetCurrentLocation	TSPI_lineSetCurrentLocation	**no** [2]	no
lineSetDevConfig	TSPI_lineSetDevConfig	**no**	n/a
lineSetNumRings [1]	TSPI_lineGetNumAddressIDs	**yes**	no
lineSetStatusMessages	TSPI_lineSetStatusMessages	**yes**	no
lineSetTollList	none	**n/a**	no
lineTranslateAddress	none	**n/a**	no
lineTranslateDialog	none	**n/a**	no

[1] The API function indirectly corresponds to one or more SPI functions.

[2] CallBridge Collection does not support **TSPI_lineSetCurrentLocation** but the API function, **lineSetCurrentLocation**, is still functional.

2.3.2 Supplementary Line Device Services.

Table 1-3. Support of Supplementary Line Device Services

TAPI Function	Corresponding TSPI Function	Supported by CallBridge TU	Device must be in service
lineAddProvider[1]	TSPI_providerInstall or TUISPI_providerInstall	**yes**	no

	TSPI_providerEnumDevices TSPI_providerInit		
lineConfigProvider[1]	TSPI_providerConfig or TUISPI_providerConfig	**yes**	no
lineRemoveProvider[1]	TSPI_providerRemove or TUISPI_providerRemove TSPI_providerShutdown	**yes**	no
lineGetProviderList	none	**n/a**	no
LineAccept	TSPI_lineAccept	**no**	n/a
lineAddToConference	TSPI_lineAddToConference	**yes**	yes
lineBlindTransfer	TSPI_lineBlindTransfer	**yes**	yes
lineCompleteCall	TSPI_lineCompleteCall	**yes**	yes
lineCompleteTransfer	TSPI_lineCompleteTransfer	**yes**	yes
LineForward	TSPI_lineForward	**yes**	yes
LineGatherDigits	TSPI_lineGatherDigits	**no**	n/a
lineGenerateDigits	TSPI_lineGenerateDigits	**yes**	yes
lineGenerateTone	TSPI_lineGenerateTone	**no**	n/a
lineGetAppPriority	none	**n/a**	no
LineHold	TSPI_lineHold	**yes**	yes
lineMonitorDigits	TSPI_lineMonitorDigits	**no**	n/a
lineMonitorMedia	TSPI_lineMonitorMedia	**no**	n/a
lineMonitorTones	TSPI_lineMonitorTones	**no**	n/a
LinePark	TSPI_linePark	**yes**	yes
LinePickup	TSPI_linePickup	**yes**	yes
linePrepareAddToConference	TSPI_linePrepareAddToConference	**yes**	yes
LineRedirect	TSPI_lineRedirect	**no**	n/a
lineReleaseUserUserInfo	TSPI_lineReleaseUserUserInfo	**no**	n/a
lineRemoveFromConference	TSPI_lineRemoveFromConference	**yes**	yes
LineSecureCall	TSPI_lineSecureCall	**no**	n/a
lineSendUserUserInfo	TSPI_lineSendUserUserInfo	**no**	n/a
lineSetAppPriority	none	**n/a**	no
LineSetCallData	TSPI_lineSetCallData	**yes**	yes
lineSetCallParams	TSPI_lineSetCallParams	**no**	n/a
lineSetCallQualityOfServive	TSPI_lineSetCallQualityOfServive	**no**	n/a
lineSetCallTreatment	TSPI_lineSetCallTreatment	**no**	n/a

lineSetLineDevStatus	TSPI_lineSetLineDevStatus	**no**	n/a
lineSetMediaControl	TSPI_lineSetMediaControl	**no**	n/a
lineSetMediaMode	TSPI_lineSetMediaMode	**no**	n/a
LineSetTerminal	TSPI_lineSetTerminal	**no**	n/a
lineSetupConference	TSPI_lineSetupConference	**yes**	yes
lineSetupTransfer	TSPI_lineSetupTransfer	**yes**	yes
LineSwapHold	TSPI_lineSwapHold	**yes**	yes
lineUncompleteCall	TSPI_lineUncompleteCall	**yes**	yes
LineUnhold	TSPI_lineUnhold	**yes**	yes
LineUnpark	TSPI_lineUnpark	**yes**	yes

[1] The API function indirectly corresponds to one or more SPI functions.

2.3.3 Phone Device Services

Table 1-4. Support of Phone Device Services

TAPI Function	Corresponding TSPI Function	Supported by CallBridge TU	Device must be in service
phoneInitializeEx [1]	TSPI_phoneNegotiateTSPIVersion TSPI_providerInit	**yes**	no
phoneShutdown [1]	TSPI_providerShutdown	**yes**	no
phoneNegotiateAPIVersion [1]	TSPI_phoneGetExtensionID	**yes**	no
phoneOpen [1]	TSPI_phoneOpen TSPI_phoneSelectExtVersion	**yes**	no
PhoneClose	TSPI_phoneClose	**yes**	no
PhoneConfigDialog	TSPI_phoneConfigDialog or TUISPI_phoneConfigDialog	**yes**	no
PhoneGetButtonInfo	TSPI_phoneGetButtonInfo	**yes**	yes
PhoneGetData	TSPI_phoneGetData	**yes**	yes
PhoneGetDevCaps	TSPI_phoneGetDevCaps	**yes**	no
PhoneGetDisplay	TSPI_phoneGetDisplay	**yes**	yes
PhoneGetGain	TSPI_phoneGetGain	**yes**	yes
PhoneGetHookSwitch	TSPI_phoneGetHookSwitch	**yes**	yes
PhoneGetIcon	TSPI_phoneGetIcon	**yes**	no
PhoneGetID	TSPI_phoneGetID	**yes**	no
PhoneGetLamp	TSPI_phoneGetLamp	**yes**	yes
PhoneGetRing	TSPI_phoneGetRing	**yes**	yes

PhoneGetStatus	TSPI_phoneGetStatus	**yes**	no
phoneGetStatusMessages	none	**n/a**	no
PhoneGetVolume	TSPI_phoneGetVolume	**yes**	yes [2]
PhoneSetButtonInfo	TSPI_phoneSetButtonInfo	**no**	n/a
PhoneSetData	TSPI_phoneSetData	**yes**	yes
PhoneSetDisplay	TSPI_phoneSetDisplay	**no**	n/a
PhoneSetGain	TSPI_phoneSetGain	**yes**	yes
PhoneSetHookSwitch	TSPI_phoneSetHookSwitch	**yes**	yes
PhoneSetLamp	TSPI_phoneSetLamp	**no**	n/a
PhoneSetRing	TSPI_phoneSetRing	**yes**	yes
phoneSetStatusMessages	TSPI_phoneSetStatusMessages	**yes**	no
PhoneSetVolume	TSPI_phoneSetVolume [3]	**yes**	yes

[1] The API function indirectly corresponds to one or more SPI functions.

[2] Returns a default value if the device is not *in service*.

[3] Volume up and down a 'notch' is also supported via a device-specific extension (**lineDevSpecificFeature**).

Note: The CorNet WP TSP does not support **phoneSetVolume**, **lineDevSpecificFeature** must be used instead.

2.3.4 Extended Telephony Services

Table 1-5. Support of Extended Telephony Services

TAPI Function	Corresponding TSPI Function	Supported by CallBridge TU	Device must be in service
lineDevSpecific	TSPI_lineDevSpecific	**yes**	yes
lineDevSpecificFeature	TSPI_lineDevSpecificFeature	**yes**	yes
lineNegotiateExtVersion	TSPI_lineNegotiateExtVersion	**yes**	no
phoneNegotiateExtVersion	TSPI_phoneNegotiateExtVersion	**yes**	no

2.4 TAPI Messages Supported by CallBridge Collection

The following tables list the TAPI messages supported by CallBridge Collection and show the TSPI message that corresponds to each TAPI message.

2.4.1 Line Device Messages

Table 1-6. Support of TAPI Line Device Messages

TAPI Message	Corresponding TSPI Message	Supported by CallBridge

		Collection
LINE_ADDRESSSTATE	LINE_ADDRESSSTATE	**yes**
LINE_APPNEWCALL	None	**n/a**
LINE_CALLINFO	LINE_CALLINFO	**yes**
LINE_CALLSTATE	LINE_CALLSTATE	**yes**
LINE_CLOSE	LINE_CLOSE	**yes**
LINE_CREATE	LINE_CREATE	**no**
LINE_DEVSPECIFIC	LINE_DEVSPECIFIC LINE_CALLDEVSPECIFIC	**no**
LINE_DEVSPECIFICFEATURE	LINE_DEVSPECIFICFEATURE LINE_CALLDEVSPECIFICFEATURE	**no**
LINE_GATHERDIGITS	LINE_GATHERDIGITS	**no**
LINE_GENERATE	LINE_GENERATE	**yes**
LINE_LINEDEVSTATE	LINE_LINEDEVSTATE	**yes**
LINE_MONITORDIGITS	LINE_MONITORDIGITS	**no**
LINE_MONITORMEDIA	LINE_MONITORMEDIA	**no**
LINE_MONITORTONE	LINE_MONITORTONE	**no**
LINE_PROXYREQUEST	none	**n/a**
LINE_REMOVE	LINE_REMOVE	**no**
LINE_REPLY	none	**yes** [1]
LINE_REQUEST	none	**n/a**

[1] The service provider supports the ASYNC_COMPLETION callback function that TAPI maps to a **LINE_REPLY** message.

2.4.2 Phone Device Messages

Table 1-7. Support of TAPI Phone Device Messages

TAPI Message	Corresponding TSPI Message	Supported by CallBridge Collection
PHONE_BUTTON	PHONE_BUTTON	**no**
PHONE_CLOSE	PHONE_CLOSE	**yes**
PHONE_CREATE	PHONE_CREATE	**no**
PHONE_REMOVE	PHONE_REMOVE	**no**
PHONE_REPLY	none	**yes** [1]
PHONE_STATE	PHONE_STATE	**yes**

[1] The service provider supports the ASYNC_COMPLETION callback function that TAPI maps to a **PHONE_REPLY** message.

3 Line Device Functions

The following discusses line device functions.

3.1 lineAddToConference

- The **hConfCall** must be in the *onHoldPendingConference* or the *connected* state.
- The **hConsultCall** must be in the *connected* or the *onHoldPendingTransfer* state.
- The **hConfCall** may be a handle that was created via
  - **lineSetupTransfer** and **lineCompleteTransfer** (as *conference*)
  - **lineSetupConference**
  - **lineDevSpecificFeature** (PHONEBUTTONFUNCTION_CONFERENCE)
  - manually pressing buttons on the phone (transfer key, conference key)
- The conference call changes to the *connected* state, the consultation call changes to the *conferenced* state.
- Not available on Hicom 100 E and Hicom 150 E Modular.

3.2 lineAddProvider

- No CallBridge Collection-specific notes.

3.3 lineAnswer

- The **hCall** must be in the *offering* state and changes to the *connected* state.
- **lineAnswer** is available only on phones equipped with both a speaker and a microphone (handsfree capability). On phones lacking this capability, the handset must be manually taken off-hook to answer an incoming call.
- The **lpsUserUserInfo** and **dwSize** parameters are not supported. They are ignored.

3.4 lineBlindTransfer

- Transfers an existing two-party call which is in the *connected* state to the specified destination address.
- The call then changes to the *idle* state.
- The **dwCountryCode** parameter is not supported and is ignored.
- The created consultation call remains invisible to the application.
- Not available on Hicom 100 E and Hicom 150 E Modular.

3.5 lineClose

- The service provider closes the connection associated with the line (only if the phone device is not open). Active calls will not be physically disconnected, i.e. closing a telephony application will not unexpectedly end an existing call.

3.6 lineCompleteCall

- CallBridge Collection supports LINECALLCOMPLMODE_ CALLBACK, _CAMPON, _INTRUDE and _MESSAGE. The contents of lpdwCompletionID must be ignored. A callback causes the call to transition to the *idle* state.
- The completion requests could be cancelled by calling **lineUncompleteCall –** only available at Hicom 300/HiPath 4000 CBXs.

Note: LINECALLCOMPLMODE_CAMPON is only available on Hicom 300/HiPath 4000 CBXs and _MESSAGE is only available on Hicom 150 E Office and later.

3.7 lineCompleteTransfer

- The **hCall** must be in the *onHoldPendingTransfer* or *connected* state.
- This function is used for completing a transfer (complete as *transfer*) as well as for establishing a conference (complete as *conference*).
- Not available on Hicom 100 E and Hicom 150 E Modular.

3.8 lineConfigDialog

- "tapi/line", an empty string, or a NULL pointer are the only values permitted for the parameter **lpszDeviceClass**.
- When this function is used to change line configuration, an application receives a **LINEDEVSTATE_REINIT** message that tells the application to re-initialize.

3.9 lineConfigDialogEdit

- This function is implemented equivalent to **lineConfigDialog**.

3.10 lineConfigProvider

- No CallBridge Collection-specific notes.

3.11 lineDeallocateCall

- No CallBridge Collection-specific notes.

3.12 lineDevSpecific

- This function allows applications to invoke device, CBX and TSP specific functions and/or features, e.g. send a **dialable** string to the CBX, independent of the phone's state or the state of any line addresses or calls on the phone.
- The **dwAddressID** parameter is not used. It is ignored.
- The **hCall** parameter is not used. It is ignored.
- The parameter **lpParams** should point to a memory area whose contents are defined according to the data structure **SIEMENS_LINE_DEV_SPECIFIC** (Refer to "SIEMENS_LINE_DEV_SPECIFIC" in chapter 9).

- The **dwRequestType** field must be one of the values
  SIEMENS_LINE_DIAL_DIGITS
  SIEMENSLINE_GET_FEATURELIST
  SIEMENSLINE_SET_HLB_PROGRAMKEY
  SIEMENSLINE_SET_OPTISET_TYPE
  SIEMENSLINE_REQ_FEATURE_STATES
  SIEMENSLINE_PARK_DIRECT
  SIEMENSLINE_GET_PUSKEYINFO
  SIEMENSLINE_PRESS_PUSKEY
  SIEMENSLINE_SET_HEADSET
  SIEMENSLINE_GET_SYSPARKSLOTS
- The following data field in the data structure is dependent from the content of the **dwRequestType** field (Refer to “SIEMENS_LINE_DEV_SPECIFIC” in chapter 9).
- Invalid characters will be ignored (see chapter 1, *Dialable Addresses*).

- The **dwSize** parameter must be the size of the **SIEMENS_LINE_DEV_SPECIFIC** structure.

3.13 lineDevSpecificFeature

- CallBridge Collection invokes a device-specific feature by CBX dependent feature invocation. The success of the feature invocation is highly context-specific.

The service provider does not provide an explicit indication of feature success or failure. The application must know how to interpret success or failure of the feature based on asynchronous information from the phone, for example: display, LED indications, or state changes. For this reason the asynch completion of **lineDevSpecificFeature** always indicates the 'success' of the operation.

- There are three ways for an application to invoke the PHONE/SIEMENS-BUTTONFUNCTIONS.
  - The application sends the feature value in **dwFeature**. The TSP scans the phone's feature list and invokes it if the feature is found.
  - The application sends the feature value in **dwFeature** and lets **lpParams** point to the physical button number on the phone. This is a necessity for **indexed** buttons such as **line keys** (see *table 2-1* for a list of indexed features). The TSP invokes the button if the requested feature is found on that button.
  - The application sends **only** the physical button number on the phone (pointed to by **lpParams)** and leaves **dwFeature** empty. The TSP invokes the button if it exists and is not configured as PHONEBUTTONFUNCTION_UNKNOWN.
- The device-specific field of the **LINEDEVSTATUS** structure indicates which features are currently available to be invoked via the **lineDevSpecificFeature** function.
- The device-specific field of the **LINEDEVCAPS** structure indicates which features the device is capable of invoking, independent of current state.
- The **dwSize** parameter must be set correctly when **lpParams** is not NULL.

Table 2-1. lpParams for dwFeatures

DwFeature	Contents of lpParams

PHONEBUTTONFUNCTION_REPDIAL	The button ID of a repdial key
PHONEBUTTONFUNCTION_STATIONSPEED	The button ID of a speed dial key
PHONEBUTTONFUNCTION_SYSTEMSPEED	The button ID of a speed dial key
SIEMENSBUTTONFUNCTION_DSSKEY	The button ID of a direct station selection key
SIEMENSBUTTONFUNCTION_FIXEDCONFGROUP	The button ID of a conference group key
SIEMENSBUTTONFUNCTION_GENCALLKEY	The button ID of a general call key
SIEMENSBUTTONFUNCTION_INTERCOMGROUP	The button ID of an intercom group key
SIEMENSBUTTONFUNCTION_LINEKEY	The button ID of a line key on the phone
SIEMENSBUTTONFUNCTION_PARKINTERCOM	The button ID of a park with intercom key
SIEMENSBUTTONFUNCTION_PICKUPGROUP	The button ID of a pickup group key
SIEMENSBUTTONFUNCTION_RELAYCONTROL	The button ID of a relay control key
SIEMENSBUTTONFUNCTION_SECRINTERCEPT	The button ID of a secretarial intercept key
SIEMENSBUTTONFUNCTION_SECRPICKUP	The button ID of a secretarial pickup key
SIEMENSBUTTONFUNCTION_SECRXFER	The button ID of a secretarial transfer key
SIEMENSBUTTONFUNCTION_SPEEDSYSGROUP	The button ID of a speed dial group key
SIEMENSBUTTONFUNCTION_TAFAS	The button ID of a trunk answer key

3.14 lineDial

- The **hCall** must be in the state *dialtone, dialing, proceeding, ringback, busy* or *connected*.
- Dialing in other states may be done using the function **lineDevSpecific** and the structure SIEMENS_LINE_DEV_SPECIFIC (SIEMENS_LINE_DIAL_DIGITS) or the function **lineGenerateDigits.**
- CallBridge Collection does not support multiple destination addresses provided in a single dial string.
- The maximum dial string length is 100 digits.
- Invalid characters will be ignored (see chapter 1, *Dialable Addresses*).
- The **dwCountryCode** parameter is not supported. It is ignored.

3.15 lineDrop

- Changes a call from any state to *idle*, leaving the call handle valid. Other calls may be affected. Ony for the active foreground call valid.
- The **lpsUserUserInfo** and **dwSize** parameters are not supported. They are ignored.

- When a consultation call is dropped, the original call in the *onHoldPendingTransfer* state, changes back to the *connected* state.
- Dropping a conference call on a Hicom 300/HiPath 4000 CBX s(conference call handle as **hCall**) causes all conferenced calls to change to the *idle* state. On Hicom 150 CBXs the conferenced calls are still in a conference or a simple two party call.
- Dropping a conference member only on a Hicom 300/HiPath 4000 CBX valid

3.16 lineForward

- The parameter **bAllAddresses** must be set to 0. Only the primary address may be forwarded and the corresponding **dwAddressID** must be supplied.
- The availability of the LINEFORWARDMODES depends on the CBX model:

  Hicom 100 E, Hicom 150 H, HiPath AllServe 150, HiPath 3000:
  LINEFORWARDMODE_UNCOND
  LINEFORWARDMODE_UNCONDINTERNAL
  LINEFORWARDMODE_UNCONDEXTERNAL

  Hicom 150 E
  LINEFORWARDMODE_UNCOND
  LINEFORWARDMODE_BUSY
  LINEFORWARDMODE_NOANSW
  LINEFORWARDMODE_BUSYNA

  Hicom 300 E / HiPath 4000 V1.0:
  LINEFORWARDMODE_UNCOND

  HiPath 4000 V2.0:
  LINEFORWARDMODE_UNCOND
  LINEFORWARDMODE_UNCONDINTERNAL (with LINEFORWARDMODE _UNCONDEXTERNAL at the same time possible)
  LINEFORWARDMODE_UNCONDEXTERNAL (with LINEFORWARDMODE _UNCONDINTERNAL at the same time possible)
  LINEFORWARDMODE_BUSY
  LINEFORWARDMODE_NOANSW
  LINEFORWARDMODE_BUSYNA

Programming and activation of call forwarding may also be done by activating the appropriate feature using **lineDevSpecificFeature**.

3.17 lineGenerateDigits

- CallBridge Collection supports only the LINEDIGITMODE_DTMF value for the **dwDigitMode** parameter. The valid characters that the service provider supports for this mode are '**0**' through '**9**', '*****', '**#**' and '**,**' (comma).
- The **dwDuration** parameter is ignored. Default values are used for digit duration (100 ms) and inter-digit spacing (200 ms).
- The **hCall** must be in the *connected* state.
- The maximum number of digits supported in the digit string is 100.
- Invalid characters will be ignored.

- Digits are sent to the CBX, but there is no explicit CBX feedback regarding the success of this operation. Therefore **LINE_GENERATE** always reports LINEGENERATETERM_DONE (indicating success).
- All calls to **lineGenerateDigits** are queued in the TSP, and sent to the CBX one by one. Subsequent calls to **lineGenerateDigits** do **not** cancel any previous digit generation requests.
- The Hicom 300 / HiPath 4000 CBX allows DTMF digit generation only on external trunk calls.

  Note: since **lineGenerateDigits** should be used to control Hicom 300 / HiPath 4000 *VoiceMail* features, and the *VoiceMail* server does not accept inband tones from within the private network, it is not desirable to automatically switch to DTMF mode when **lineGenerateDigits** is called. The application or the user must 'manually' activate the DTMF mode by pressing the appropriate button(s) or activating the DTMF feature using **lineDevSpecificFeature**. After switching to DTMF mode, applications should observe a pause of 300ms before sending the first digit.

3.18 lineGetAddressCaps

- Refer to “LINEADDRESSCAPS” in chapter 6 for a description of which fields of the **LINEADDRESSCAPS** data structure are supported and which fields have constant values.
- The **LINEADDRESSCAPS** data structure is used to return the following device-specific information:
  - The button/lampID of the button on the phone which is a line key associated with this line address.

3.19 lineGetAddressID

- No CallBridge Collection-specific notes.

3.20 lineGetAddressStatus

- Refer to “LINEADDRESSSTATUS” in chapter 6 for a description of which fields of the **LINEADDRESSSTATUS** data structure are supported and which fields have constant values.
- If the line is not *in service*, all fields of **LINEADDRESSTATUS** will be zero.

3.21 lineGetCallInfo

- Refer to “LINECALLINFO” in chapter 6 for a description of which fields of the **LINECALLINFO** data structure are supported and which fields have constant values.

3.22 lineGetCallStatus

- Refer to “LINECALLSTATUS” in chapter 6 for a description of which fields of the **LINECALLSTATUS** data structure are supported and which fields have constant values.

3.23 lineGetDevCaps

- Refer to “LINEDEVCAPS” in chapter 6 for a description of which fields of the **LINEDEVCAPS** data structure are supported and which fields have constant values.
- The **LINEDEVCAPS** structure is used to return the following device-specific information:
  - The types of requests that may be invoked via **lineDevSpecific**.
  - The features that may be invoked via **lineDevSpecificFeature**.

    Note: the dynamic availability of device-specific features is provided in the **LINEDEVSTATUS** data structure.

3.24 lineGetIcon

- This function always returns the CallBridge Collection line icon.

3.25 lineGetID

- “tapi/line” and “tapi/phone” are the only strings permitted for the parameter **lpszDeviceClass**.
- When the “tapi/line” device class is specified, the integer line device ID of the line associated with **hLine**, **dwAddressID** or **hCall** is returned via **lpDeviceID**.
- When the “tapi/phone” device class is specified, the integer phone device ID of the phone associated with **hLine**, **dwAddressID**, or **hCall** is returned via **lpDeviceID**.

3.26 lineGetLineDevStatus

- Refer to “LINEDEVSTATUS” in chapter 6 for a description of which fields of the **LINEDEVSTATUS** data structure are supported and which fields have constant values.
- The **LINEDEVSTATUS** structure is used to return the following device-specific information:
  - The device-specific features that are currently available to be invoked via **lineDevSpecificFeature** (based on current device state)
- If the line device is not *in service*, all fields of the **LINEDEVSTATUS** structure except **dwDevStatusFlags** will be set to zero.

3.27 lineGetNewCalls

- No CallBridge Collection-specific notes.

3.28 lineGetNumRings

- No CallBridge Collection-specific notes.

3.29 lineHold

- Not available on Hicom 100 E, Hicom 300 and HiPath 4000.

3.30 lineMakeCall

- If no destination address is specified in the **lpszDestAddress** parameter, the call enters the *dialtone* state. Otherwise the call state changes to the *dialing/proceeding/ringback* state.
- If digits are not supplied by the **lineDial** function within a pre-defined time, the call times out and dialing is no longer possible. The timeout value depends on the CBX model.
- Only one address may be supplied as the destination address; CallBridge Collection does not support inverse multiplexing.
- The **dwCountryCode** parameter is not supported. It is ignored.
- Refer to “LINECALLPARAMS” in chapter 6 for a description of the values that may be specified in the **LINECALLPARAMS** structure pointed to by **lpCallParams**.
- The DESTINATION_OFFHOOK parameter is supported, but success depends on the class of service parameters of the called station (which may have activated a protection against this feature).
- If there is an active call on the specified line address at the time of the **lineMakeCall** request, the request fails with return value LINEERR_RESOURCEUNAVAIL.

3.31 lineNegotiateAPIVersion

- CallBridge Collection supports **TAPI Versions 1.3, 1.4** and **2.0/2.1**.

3.32 lineNegotiateExtVersion

- CallBridge Collection currently supports one extension version for all line devices. The current extension version is 0x00010001. Refer to “LINEEXTENSIONID” in chapter 6 for the value of the supported extension ID.

3.33 lineOpen

- Opens the connection and establishes the interaction with the *optiSet and optiPoint* telephones if the line/phone is not already open.
- CallBridge Collection supports only calls of media mode LINEMEDIAMODE_INTERACTIVEVOICE.
- Refer to “LINECALLPARAMS” in chapter 6 for a description of the values that may be specified in the **LINECALLPARAMS** structure pointed to by **lpCallParams**.

3.34 linePark

- Hicom 300 / HiPath 4000 V1 support only *non-directed* parking of calls.
- Hicom 150 E Office V2.2 / HiPath 3000 support only directed parking. The system provides 10 parking ports addressed between 0 and 9. But the TSP supports directed and non-directed parking. In case of non-directed parking the TSP decides itself if and which park slot can be reserved. The application gets then a LINE_DEVSPECIFIC event with the information which park slot was occupied with the given call. The call which is to be parked must be in CONNECTED state.
- HiPath 4000 V2.0 supports directed and nondirected parking. For directed parking, the system provides 10 parking ports addressed between 0 and 9. The call which is to be

parked must be in ONHOLDPENDTRANSFER state and the consultation call must be in DIALTONE state. For nondirected parking, the system provides also 10 parking ports, but the application do not have to declare a parking position. The system automatically selects the next free park slot. The call which is to be parked must be in CONNECTED state

- For *directed* parking, the **lpszDirAddress** parameter must specify the park position number where the call is to be parked.
- An empty string is always returned in **lpNonDirAddress**, since the *non-directed* park position is only known to the CBX.
- Not available on Hicom 100 E, Hicom 150 E Modular and Hicom 150 E Office.

3.35 linePickup

- If invoked with NULL as **lpszDestAddress** parameter, a *group pickup* is initiated.
- The **lpszGroupID** parameter is not needed and is ignored if supplied.
- CallBridge Collection allows the line address on which a call is picked to be either *idle,* or, depending on CBX model, to have one or more calls in different states.
- The LINEADDRCAPFLAGS_PICKUPCALLWAIT flag is set to TRUE for line addresses supported by CallBridge Collection.

3.36 linePrepareAddToConference

- Refer to“LINECALLPARAMS” in chapter 6 for a description of the values that may be specified in the **LINECALLPARAMS** structure pointed to by **lpCallParams**.
- The existing **hConfCall** changes to the *onHoldPendingConference* state.
- A new consultation call in the *dialtone* state is created.
- Not available on Hicom 100 E and Hicom 150 E Modular.

3.37 lineRemoveFromConference

- Any call (identified by **hCall**) may be removed from the conference.
- The removed call changes to the *idle* state.

  Note: if a call is removed from a 3-party conference, the conference call changes to the *idle* state, the remaining call changes to the *connected* state
- Only available on Hicom 300 E / HiPath 4000.

3.38 lineRemoveProvider

- No CallBridge Collection-specific notes.

3.39 lineSetAppSpecific

- No CallBridge Collection-specific notes.

3.40 lineSetNumRings

- No CallBridge Collection-specific notes.

3.41 lineSetStatusMessages

- Refer to “LINEADDRESSCAPS” and “LINEDEVCAPS” in chapter 6 for a list of which line states and address status changes are supported by CallBridge Collection.

3.42 lineSetupConference

- Creates a conference call in the *onHoldPendingConference* state and a consultation call in the *dialtone state*.
- Not available on Hicom 100 E and Hicom 150 E Modular.

  Note: when the consultation call reaches the *connected* state**, lineAddToConference** must be called to complete the conference.

3.43 lineSetupTransfer

- Refer to “LINECALLPARAMS” in chapter 6 for a description of the values that may be specified in the **LINECALLPARAMS** structure pointed to by **lpCallParams**.
- Transitions the original call to the *onHoldPendingTransfer* state and creates a consultation call in the *dialtone* state.
- If called during an active conference call, **hConf** transitions to the *onHoldPendingConference* state. A conference call may never be transfered to another party.
- When the consultation call has reached the *dialtone* state, a destination address must be dialed to complete the consultation call. CallBridge Collection does not support the unholding of an existing call in this state in order to complete a transfer.
- Not available on Hicom 100 E and Hicom 150 E Modular.

3.44 lineSwapHold

- No CallBridge Collection-specific notes.

3.45 lineUncompleteCall

- Hicom 300 / HiPath 4000 supports only completion mode CALLBACK. The line device must be in complete IDLE state to execute the CALLBACK completion mode.
- Not available on Hicom 150 CBXs.

3.46 lineUnhold

- No CallBridge Collection-specific notes.
- Not available on Hicom 100 E.

3.47 lineUnpark

- Hicom 300 / HiPath 4000 and Hicom 150 E Office V2.2 / HiPath 3000 supports only *directed* unparking of calls.

- **lpszDestAddress** must not be empty. To unpark a call that was parked in a system park position using *directed* or *undirected* park, **lpszDestAddress** must contain the number of the park position.
- CallBridge Collection allows the line address on which a call is unparked to be either *idle,* or, depending on CBX model, to have one or more calls in different states.
- Not available on Hicom 100 E, Hicom 150 E Modular and Hicom 150 E Office.

3.48 lineSetCallData

- Stores informations from **lpCallData** parameter related to the call. This information will be displayed in the **LINECALLINFO** data structure.

4 Phone Device Functions

The following discusses phone device functions.

4.1 phoneClose

- The service provider closes the connection associated with the phone (only if the line device is not open). Active calls will not be physically disconnected, i.e. closing a telephony application will not unexpectedly end an existing call.

4.2 phoneConfigDialog

- "tapi/phone," an empty string or a NULL pointer are the only values permitted for the parameter **lpszDeviceClass**.
- When this function is used to change phone configuration, an application receives a **PHONESTATE_REINIT** message that advises the application to re-initialize.

4.3 phoneGetButtonInfo

- Refer to "PHONEBUTTONINFO" in chapter 7 for a description of which fields of the **PHONEBUTTONINFO** data structure are supported and which fields have constant values.
- The **dwButtonLampID** parameter must specify the ID of a physical phone button.

  Note: the buttons on virtual add-on units are considered *physical buttons*.
- Refer to "PHONEBUTTONFUNCTION Constants" in chapter 8 for a list of the PHONEBUTTONFUNCTION values that CallBridge Collection supports as well as the Siemens defined extensions for this data type.

4.4 phoneGetData

- Returns the phone device specific informations, wich are stored over **phoneSetData** function.

4.5 phoneGetDevCaps

- Refer to "PHONECAPS" in chapter 7 for a description of which fields of the **PHONECAPS** data structure are supported and which fields have constant values.
- The **PHONECAPS** structure is used to return the following device-specific information:
  - The button functions of type PHONEBUTTONFUNCTION and the device-specific SIEMENSBUTTONFUNCTIONs that are available via **lineDevSpecificFeature**.

4.6 phoneGetDisplay

- No CallBridge Collection-specific notes.

4.7 phoneGetGain

- No CallBridge Collection-specific notes.

4.8 phoneGetHookSwitch

- No CallBridge Collection-specific notes.

4.9 phoneGetIcon

- This function always returns the CallBridge Collection phone icon.

4.10 phoneGetID

- "tapi/phone" and "tapi/line" are the only strings permitted for the parameter **lpszDeviceClass**.
- When the "tapi/phone" device class is specified, the integer phone device ID of the phone associated with **hPhone** is returned via **lpDeviceID**.
- When the "tapi/line" device class is specified, the integer line device ID of the line device associated with **hPhone** is returned in **lpDeviceID**.

4.11 phoneGetLamp

- The **dwButtonLampID** parameter must specify the ID of a physical phone button.

  Note: the buttons on virtual add-on units are considered *physical buttons*.

4.12 phoneGetRing

- The ring volume level is reported in increments of 0x1000 at HiPath 3000 CBX's and in increments of 0x2000 at Hicom 300/HiPath 4000 CBX's.
- Refer to "SIEMENSRINGMODE Constants" in chapter 9 for the ring mode values that can be returned in the parameter **lpdwRingMode**.

4.13 phoneGetStatus

- Refer to "PHONESTATUS" in chapter 7 for a description of which fields of the **PHONESTATUS** data structure are supported and which fields have constant values.
- The **PHONESTATUS** structure is used to return the following device-specific information:
  - The status (LAMPMODE) of all physical buttons.

4.14 phoneGetVolume

- The volume levels for the handset, headset and speaker are reported in increments of 0x1000. A default midrange value is reported if the phone device is not *in service*.

4.15 phoneNegotiateAPIVersion

- CallBridge Collection supports **TAPI Versions 1.3, 1.4 and 2.0/2.1.**

4.16 phoneNegotiateExtVersion

- CallBridge Collection supports one extension version for all phone devices. Refer to "PHONEEXTENSIONID" in chapter 7 for the value of the supported extension ID.

4.17 phoneOpen

- Opens the connection and establishes the interaction with the *optiSet and optiPoint* telephones if the line/phone is not already open.

4.18 phoneSetData

- This function stored phone device specific informations, wich can requested by **phoneGetData** function.

4.19 phoneSetGain

- This function set's the microphone gain in handset, headset or speaker mode.
- **S**witches only the microphone on (**dwGain** parameter > 0) or off (**dwGain** parameter = 0).
- Not available on Hicom 100 E, Hicom 150 E Modular and Hicom 150 E Office.

4.20 phoneSetHookSwitch

Specifies the hookswitch modes that can be set for the various hookswitch devices:

Table 3-1. Hookswitch Devices and Modes

Hookswitch Device (PHONEHOOKSWITCHDEV_)	**Hookswitch Mode** (PHONEHOOKSWITCHMODE_)
HANDSET	SPEAKER, MICSPEAKER [1]
SPEAKER	ONHOOK, SPEAKER, MICSPEAKER
HEADSET	ONHOOK, SPEAKER, MICSPEAKER[2]

[1] The hookswitch modes for the handset can be set only if the handset is **not** on-hook.

[2] The headset may be explicitely controlled only when it is plugged into the headset option adapter of the phone. A headset plugged into the handset port cannot be distinguished from a standard handset, and is implicitely controlled via PHONEHOOKSWITCHDEV_HANDSET.

4.21 phoneSetRing

- Setting **dwVolume** to zero will not turn off the ringer on all CBX models.

- The ringing pattern of the phone cannot be changed using this function, but setting **dwRingMode** to 0 may be used to indicate that the phone should not be ring (silence). All other values for **dwRingMode** cancel ringer silence.

  Note: the ring volume level and ringing cadence may also be controlled via the **lineDevSpecificFeature** PHONEBUTTONFUNCTION_VOLUMEUP or _VOLUMEDOWN and the *service menu* buttons.

4.22 phoneSetStatusMessages

- Refer to “PHONECAPS” in chapter 7 for a list of which phone states are supported by CallBridge Collection .
- CallBridge Collection does not support the PHONE_BUTTON message; therefore, the **dwButtonModes** and **dwButtonStates** parameters are not supported. These parameters are ignored.

4.23 phoneSetVolume

- Controlling the volume levels for the handset, headset and speaker is possible only when the devices are **not** onhook.

  Note: the hookswitch device volume may also be controlled via the **lineDevSpecificFeature** PHONEBUTTONFUNCTION_VOLUMEUP or _VOLUMEDOWN and the service menu buttons.

5 Line Device Messages

The following discusses line device messages.

5.1 LINE_ADDRESSSTATE

- For a list of the supported **LINEADDRESSSTATE** values, refer to the **dwAddressStates** field of the "**LINEADDRESSCAPS**" data structure in chapter 6.
- The LINEADDRESSSTATE_OTHER value is sent when the **LINEADDRESSSTATUS/dwAddressFeatures** field has changed.

5.2 LINE_CALLINFO

- For a list of the supported **LINECALLINFOSTATE** values, refer to the dwCallInfoStates field of the "LINEADDRESSCAPS" data structure in chapter 6.

5.3 LINE_CALLSTATE

- For a list of the supported **LINECALLSTATE** values, refer to the **dwCallStates** field of the "**LINEADDRESSCAPS**" data structure in chapter 6.
- In addition to reporting call state changes, a **LINE_CALLSTATE** message is sent when the **LINECALLSTATUS/dwCallFeatures** field changes. Therefore, the application may receive several **LINE_CALLSTATE** messages reporting the same call state.
- If **dwParam1** is LINECALLSTATE_BUSY, **dwParam2** is specified as LINEBUSYMODE_UNAVAIL.
- If **dwParam1** is LINECALLSTATE_DIALTONE, **dwParam2** is specified as LINEDIALTONEMODE_UNAVAIL.
- If **dwParam1** is LINECALLSTATE_DISCONNECTED, **dwParam2** is specified as LINEDISCONNECTMODE_UNAVAIL.
- If **dwParam1** is LINECALLSTATE_CONFERENCED, **dwParam2** contains the **hConfCall** of the parent call of the conference.

5.4 LINE_CLOSE

- In case of IP connectivity with the devices optiPoint 600/410 and loss of networking, e.g. due to device goes out of service or unplugging the LAN cable, it can be that the LINE_CLOSE event has a delay up to 90 seconds. This is a permanent restriction based on the protocol and the driver.

5.5 LINE_GENERATE

- Sent as a response to **lineGenerateDigits.** The **dwParam1** always reports LINEGENERATETERM_DONE as the actual success or failure of the operation cannot be detected.

5.6 LINE_LINEDEVSTATE

- For a list of the supported **LINEDEVSTATE** values, refer to **dwLineStates** field of the **"LINEDEVCAPS"** data structure in chapter 6.
- The LINEDEVSTATE_INSERVICE value is set when CallBridge Collection has established communication with the phone for the first.
- The LINEDEVSTATE_RINGING value is sent when an incoming call is alerting a line device, even if the ringmode is *silence*. This would be the case if a call was incoming to a line address that was configured as non-ringing or if an incoming call camped on to a line address while another call was active.
- The LINEDEVSTATE_OTHER value is sent when the **LINEDEVSTATUS/dwLineFeatures** field has changed.
- The LINEDEVSTATE_DEVSPECIFIC value is sent when the device-specific information in the **LINEDEVSTATUS** data structure has changed.

5.7 LINE_REPLY

- No CallBridge Collection-specific notes.

6 Phone Device Messages

The following discusses phone device messages.

6.1 PHONE_CLOSE

- In case of IP connectivity with the devices optiPoint 600/410 and loss of networking, e.g. due to device goes out of service or unplugging the LAN cable, it can be that the PHONE_CLOSE event has a delay up to 90 seconds. This is a permanent restriction based on the protocol and the driver.

6.2 PHONE_REPLY

- No CallBridge Collection-specific notes.

6.3 PHONE_STATE

- For a list of the supported **PHONESTATE** values, refer to “PHONECAPS” in chapter 7.
- The PHONESTATE_RESUME value is set when CallBridge Collection has established communication with the phone for the first time.
- The PHONESTATE_DEVSPECIFIC value is set when CallBridge Collection detects that the *Service Menu* mode has changed.

7 Line Device Data Structures

The following chapter provides information about line device data structures. If device-specific data is included in a data structure, a reference is given to the section where the format of the device-specific data is described.

If a data structure is not included in this section, either it is not required by any line device function supported by CallBridge Collection, or is it used in a function supported exclusively by TAPI without any interaction with the service provider.

The notes in this section indicate which fields in the data structures are supported and which fields have constant values.

The tables in this chapter used to describe the fields of the data structures apply the following rules:

1. If a field is indicated as not supported ("n/s"), it will be initialized with a default value of 0.
2. If a field is static, its constant value is shown.
3. If a field is dynamic, the "value" field is left blank.
4. Fields that are filled in by TAPI and not by CallBridge Collection are marked as "TAPI supplies."
5. The **dwTotalSize**, **dwNeededSize**, and **dwUsedSize** fields are omitted from the tables since they provide no useful information for the purpose of this document.
6. Double lines in the tables mark the logical field groupings found in the Microsoft TAPI specifications.

7.1 LINEADDRESSCAPS

Table 6-1. Notes on LINEADDRESSCAPS

Field Name	**Constant Value ("n/s" if <u>not</u> supported)**
dwLineDeviceID	
dwAddressSize	
dwAddressOffset	
dwDevSpecificSize	If device-specific data is available and an extension version has been selected, these fields are set to the size and offset of the device-specific data. Refer to SIEMENS_LINEADDRCAPS_DEV_SPECIFIC in chapter 9 for the format and description of the device-specific data.
DwDevSpecificOffset	
DwAddressSharing	LINEADDRESSSHARING_PRIVATE
DwAddressStates	LINEADDRESSSTATE_OTHER [1] LINEADDRESSSTATE_INUSEONE LINEADDRESSSTATE_NUMCALLS LINEADDRESSSTATE_FORWARD [2] [1] Used to report changes in dwAddressFeatures field.

	[2] The forward mode is reported only for the *primary address* of the line device.
DwCallInfoStates	LINECALLINFOSTATE_APPSPECIFIC LINECALLINFOSTATE_ORIGIN LINECALLINFOSTATE_REASON LINECALLINFOSTATE_CALLERID LINECALLINFOSTATE_CALLEDID LINECALLINFOSTATE_CONNECTEDID LINECALLINFOSTATE_REDIRECTIONID LINECALLINFOSTATE_REDIRECTINGID LINECALLINFOSTATE_CALLID [1] LINECALLINFOSTATE_RELATEDCALLID [1] LINECALLINFOSTATE_CHARGINGINFO LINECALLINFOSTATE_DISPLAY LINECALLINFOSTATE_TRUNK [1] LINECALLINFOSTATE_CALLDATA [1] LINECALLINFOSTATE_OTHER [1] Other bits may also be set by TAPI. [1] For Hicom 150 CBXs only available starting with Hicom 150 E Office V2.2
dwCallerIDFlags	LINECALLPARTYID_ADDRESS LINECALLPARTYID_NAME [1] LINECALLPARTYID_UNKNOWN LINECALLPARTYID_UNAVAIL [1] For Hicom 150 CBXs only available starting with Hicom 150 E Office V2.2
dwCalledIDFlags	LINECALLPARTYID_ADDRESS LINECALLPARTYID_NAME [1] LINECALLPARTYID_UNKNOWN LINECALLPARTYID_UNAVAIL [1] For Hicom 150 CBXs only available starting with Hicom 150 E Office V2.2
dwConnectedIDFlags	LINECALLPARTYID_ADDRESS LINECALLPARTYID_NAME [1] LINECALLPARTYID_UNKNOWN LINECALLPARTYID_UNAVAIL [1] For Hicom 150 CBXs only available starting with Hicom 150 E Office V2.2
dwRedirectionIDFlags	LINECALLPARTYID_UNAVAIL LINECALLPARTYID_UNKNOWN [1] LINECALLPARTYID_NAME [1] LINECALLPARTYID_ADDRESS [1] [1] For Hicom 150 CBXs only available starting with Hicom 150 E Office V2.2
dwRedirectingIDFlags	LINECALLPARTYID_UNAVAIL LINECALLPARTYID_UNKNOWN [1] LINECALLPARTYID_NAME [1] LINECALLPARTYID_ADDRESS [1] [1] For Hicom 150 CBXs only available starting with Hicom 150 E Office V2.2
dwCallStates	LINECALLSTATE_IDLE LINECALLSTATE_OFFERING LINECALLSTATE_DIALTONE LINECALLSTATE_DIALING

	LINECALLSTATE_RINGBACK LINECALLSTATE_BUSY LINECALLSTATE_CONNECTED LINECALLSTATE_PROCEEDING [1] LINECALLSTATE_CONFERENCED LINECALLSTATE_ONHOLDPENDTRANSFER LINECALLSTATE_ONHOLDPENDCONF LINECALLSTATE_DISCONNECTED LINECALLSTATE_ONHOLD [1] [1] For Hicom 150 CBXs only available starting with Hicom 150 E Office
dwDialToneModes	LINEDIALTONEMODE_UNAVAIL
dwBusyModes	LINEBUSYMODE_UNAVAIL
dwSpecialInfo	n/s [1] LINESPECIALINFO_UNAVAIL [2] [1] at Hicom 300 / HiPath 4000 CBXs [2] at Hicom 100/150/HiPath 3000 CBXs
dwDisconnectModes	LINEDISCONNECTMODE_NORMAL [1] LINEDISCONNECTMODE_UNAVAIL [2] [1] at Hicom 300 / HiPath 4000 CBXs [2] at Hicom 100/150/HiPath 3000 CBXs
dwMaxNumActiveCalls	1 for Hicom 100E and Hicom 150 E Modular 8 for Hicom 300/HiPath 4000 CBXs 150 for Hicom 150/HiPath 3000 CBXs starting with Hicom 150 E Office
dwMaxNumOnHoldCalls	0 for Hicom 300/HiPath 4000 V1.0 CBXs 1 for Hicom 150 E Modular 2 for HiPath 4000 V2.0 CBXs 150 for Hicom 150/HiPath 3000 CBXs starting with Hicom 150 E Office
dwMaxNumOnHoldPendingCalls	0 1 for Hicom 300/HiPath 4000 CBXs and Hicom 150/HiPath 3000 CBXs starting with Hicom 150 E Office
dwMaxNumConference	0 5 for Hicom 150/HiPath 3000 CBXs starting with Hicom 150 E Office 8 for Hicom 300/HiPath 4000 CBXs
dwMaxNumTransConf	0 1 for Hicom 300/HiPath 4000 CBXs 5 for Hicom 150/HiPath 3000 CBXs starting with Hicom 150 E Office
dwAddrCapFlags	LINEADDRCAPFLAGS_DIALED LINEADDRCAPFLAGS_ORIGOFFHOOK [1] LINEADDRCAPFLAGS_DESTOFFHOOK [2/5] LINEADDRCAPFLAGS_AUTORECONNECT [3] LINEADDRCAPFLAGS_PARTIALDIAL [5] LINEADDRCAPFLAGS_PICKUPCALLWAIT LINEADDRCAPFLAGS_HOLDMAKESNEW [4] [1] Flag is present only if phone is configured as a speakerphone. [2] This value is dynamic based on Class of Service. [3] This capability depends on the CBX configuration. [4] Only for Hicom 150/HiPath 3000 CBXs starting with Hicom 150 E Office. [5] Only for Hicom 300/HiPath 4000 CBXs.

dwCallFeatures	LINECALLFEATURE_ADDTOCONF LINECALLFEATURE_ANSWER LINECALLFEATURE_BLINDTRANSFER LINECALLFEATURE_COMPLETECALL LINECALLFEATURE_COMPLETETRANSF LINECALLFEATURE_DIAL LINECALLFEATURE_DROP LINECALLFEATURE_HOLD LINECALLFEATURE_GENERATEDIGITS LINECALLFEATURE_PARK LINECALLFEATURE_PREPAREADDCONF LINECALLFEATURE_REMOVEFROMCONF LINECALLFEATURE_SETUPTRANSFER LINECALLFEATURE_SETUPCONF LINECALLFEATURE_SWAPHOLD LINECALLFEATURE_UNHOLD LINECALLFEATURE_SETCALLDATA
dwRemoveFromConfCaps	LINEREMOVEFROMCONF_ANY [1] [1] Hicom 300 / HiPath 4000 only.
dwRemoveFromConfState	LINECALLSTATE_IDLE [1] [1] Hicom 300 / HiPath 4000 only
dwTransferModes	LINETRANSFERMODE_TRANSFER LINETRANSFERMODE_CONFERENCE
dwParkModes	LINEPARKMODE_NONDIRECTED [1] LINEPARKMODE_DIRECTED [2] [1] only Hicom 300/HiPath 4000 CBXs [2] Starting with Hicom 150 E Office V2.2 and HiPath 4000 V2.0.
dwForwardModes	LINEFORWARDMODE_UNCOND [1] LINEFORWARDMODE_BUSY [2] LINEFORWARDMODE_NOANSW [2] LINEFORWARDMODE_BUSYNA [2] LINEFORWARDMODE_UNCONDINTERNAL [3] LINEFORWARDMODE_UNCONDEXTERNAL [3] [1] All CBXs [2] only Hicom 150 E Modular, HiPath 4000 V2.0 CBXs [3] only Hicom 100 E, Hicom 150 E, HiPath 3000, HiPath 4000 V2.0 CBXs
dwMaxForwardEntries	1 2 [1] [1] only HiPath 4000 V2.0 CBX
dwMaxSpecificEntries	0
dwMinFwdNumRings	0

dwMaxFwdNumRings	0
dwMaxCallCompletions	1 [1] 10 [2] [1] Hicom 300 CBXs:Once a callback request has been sent to the CBX, CallBridge Collection does not consider it an *outstanding* completion request, so more than one callback may be set for each line device. The actual number of settable *callbacks* is CBX dependent. [2] Hicom 100 E, Hicom 150 E, HiPath 3000 CBXs.
dwCallCompletionConds	LINECALLCOMPLCOND_BUSY LINECALLCOMPLCOND_NOANSWER
dwCallCompletionModes	LINECALLCOMPLMODE_CALLBACK LINECALLCOMPLMODE_CAMPON [1/3] LINECALLCOMPLMODE_INTRUDE [3] LINECALLCOMPLMODE_MESSAGE [2] [1] for Hicom 300 CBXs. [2] for Hicom 300 CBXs. [3] for HiPath 4000 V2.0
dwNumCompletionMessages	n/s for Hicom 300, HiPath 4000 CBXs 0 for Hicom 100 E and Hicom 150 E CBXs 10 for Hicom 150 E CBXs starting with Hicom 150 E Office V2.2
dwCompletionMsgTextEntrySize	n/s for Hicom 300, HiPath 4000 CBXs 0 for Hicom 100 E and Hicom 150 E CBXs 48 for Hicom 150 E CBXs starting with Hicom 150 E Office V2.2
dwCompletionMsgTextSize	n/s for Hicom 300 CBXs
dwCompletionMsgTextOffset	n/s for Hicom 300 CBXs
dwAddressFeatures	LINEADDRFEATURE_MAKECALL LINEADDRFEATURE_PICKUPGROUP LINEADDRFEATURE_FORWARD LINEADDRFEATURE_UNPARK [1] LINEADDRFEATURE_UNCOMPLETECALL [1] LINEADDRFEATURE_PICKUP LINEADDRFEATURE_FORWARDFWD [2] LINEADDRFEATURE_FORWARDDND [2] LINEADDRFEATURE_PICKUPWAITING [2] [1] only available for Hicom 300, HiPath 4000 CBXs [2] only available for Hicom 150, HiPath 3000 CBXs
dwPredictiveAutoTransferStates	n/s
dwNumCallTreatments	n/s
dwCallTreatmentListSize	n/s
dwCallTreatmentListOffset	n/s
dwDeviceClassesSize	
dwDeviceClassesOffset	
dwMaxCallDataSize	65536

dwCallFeatures2	n/s for Hicom 300 CBXs for Hicom 150 E CBXs starting with Hicom 150 E Office: LINECALLFEATURE2_TRANSFERCONF LINECALLFEATURE2_TRANSFERNORM LINECALLFEATURE2_COMPLCALLBACK LINECALLFEATURE2_COMPLINTRUDE LINECALLFEATURE2_COMPLIMESSAGE LINECALLFEATURE2_PARKDIRECT LINECALLFEATURE2_ONESTEPTRANSFER
dwMaxNoAnswerTimeout	n/s
dwConnectModes	n/s for Hicom 300, HiPath 4000 V1 CBXs for Hicom 150 E, HiPath 3000 CBXs starting with Hicom 150 E Office and HiPath 4000 V2.0: LINECONNECTEDMODE_ACTIVE LINECONNECTEDMODE_ACTIVEHELD LINECONNECTEDMODE_CONFIRMED
dwOfferingModes	n/s
dwAvailableMediaModes	LINEMEDIAMODE_INTERACTIVEVOICE

7.2 LINEADDRESSSTATUS

Note: all fields in this structure are set to zero if the line is not *in service*.

Table 6-2. Notes on LINEADDRESSSTATUS

Field Name	**Constant Value ("n/s" if not supported)**
dwNumInUse	This field will always be 1.
dwNumActiveCalls	
dwNumOnHoldCalls	
dwNumOnHoldPendCalls	
dwAddressFeatures	LINEADDRFEATURE_MAKECALL LINEADDRFEATURE_FORWARD LINEADDRFEATURE_PICKUP [1,2] LINEADDRFEATURE_PICKUPWAITING [1,2] LINEADDRFEATURE_PICKUPGROUP [1,2] LINEADDRFEATURE_PICKUPDIRECT [1,2] LINEADDRFEATURE_FORWARDFWD [1] LINEADDRFEATURE_FORWARDDND [1] LINEADDRFEATURE_UNPARK [1,2] LINEADDRFEATURE_UNCOMPLETECALL [2] [1] Available for Hicom 100 E, Hicom 150 E, HiPath 3000 CBXs 2 Available for HiPath 4000 V2.0 CBXs
dwNumRingsNoAnswer	n/s
dwForwardNumEntries	n/s
dwForwardSize	n/s

dwForwardOffset	n/s
dwTerminalModesSize	n/s
dwTerminalModesOffset	n/s
dwDevSpecificSize	n/s
dwDevSpecificOffset	n/s

7.3 LINECALLINFO

Table 6-3. Notes on LINECALLINFO

Field Name	**Constant Value ("n/s" if not supported)**
hLine	TAPI supplies
dwLineDeviceID	
dwAddressID	
dwBearerMode	LINEBEARERMODE_VOICE
dwRate	0
dwMediaMode	LINEMEDIAMODE_INTERACTIVEVOICE
dwAppSpecific	
dwCallID	
dwRelatedCallID	
dwCallParamFlags	
dwCallStates	= dwCallStates from LINEADDRESSCAPS
dwMonitorDigitModes	TAPI supplies
dwMonitorMediaModes	TAPI supplies
DialParams	
dwOrigin	
dwReason	
dwCompletionID	n/s An activated call completion may not be *uncompleted*, and dwCompletionID is not needed.
dwNumOwners	TAPI supplies
dwNumMonitors	TAPI supplies
dwCountryCode	n/s
dwTrunk	n/s for Hicom 300, HiPath 4000 CBXs provided for Hicom 150 E, HiPath 3000 CBXs starting with Hicom 150 E Office Rel. 2.2
dwCallerIDFlags	
dwCallerIDSize	

dwCallerIDOffset	
dwCallerIDNameSize	
dwCallerIDNameOffset	
dwCalledIDFlags	
dwCalledIDSize	
dwCalledIDOffset	
dwCalledIDNameSize	
dwCalledIDNameOffset	
dwConnectedIDFlags	For an **outgoing** call, the connected number is the digits dialed, including access codes. For an **incoming** call, the connected number reflects the number of the station which originated the call.
dwConnectedIDSize	
dwConnectedIDOffset	
dwConnectedIDNameSize	
dwConnectedIDNameOffset	
dwRedirectionIDFlags	n/s
dwRedirectionIDSize	n/s for Hicom 300, HiPath 4000 V1.0 CBXs provided for Hicom 100 E, Hicom 150 E, HiPath 3000 and HiPath 4000 V2.0 CBXs
dwRedirectionIDOffset	n/s for Hicom 300, HiPath 4000 V1.0 CBXs provided for Hicom 100 E, Hicom 150 E, HiPath 3000 and HiPath 4000 V2.0 CBXs
dwRedirectionIDNameSize	n/s for Hicom 300, HiPath 4000 V1.0 CBXs provided for Hicom 100 E, Hicom 150, HiPath 3000 and HiPath 4000 V2.0 E CBXs
dwRedirectionIDNameOffset	n/s for Hicom 300, HiPath 4000 V1.0 CBXs provided for Hicom 100 E Hicom 150 E, HiPath 3000 and HiPath 4000 V2.0 CBXs
dwRedirectingIDFlags	n/s for Hicom 300, HiPath 4000 V1.0 CBXs Dynamically provided for Hicom 100 E and Hicom 150 E, HiPath 3000 and HiPath 4000 V2.0 CBXs
dwRedirectingIDSize	n/s for Hicom 300, HiPath 4000 V1.0 CBXs provided for Hicom 150 E CBXs starting with Hicom 150 H, HiPath 3000 and HiPath 4000 V2.0
dwRedirectingIDOffset	n/s for Hicom 300, HiPath 4000 V1.0 CBXs provided for Hicom 150 E CBXs starting with Hicom 150 H, HiPath 3000 and HiPath 4000 V2.0
dwRedirectingIDNameSize	n/s for Hicom 300, HiPath 4000 V1.0 CBXs provided for Hicom 150 E CBXs starting with Hicom 150 H, HiPath 3000 and HiPath 4000 V2.0

dwRedirectingIDNameOffset	n/s for Hicom 300, HiPath 4000 V1.0 CBXs provided for Hicom 150 E CBXs starting with Hicom 150 H, HiPath 3000 and HiPath 4000 V2.0
dwAppNameSize	TAPI supplies
dwAppNameOffset	TAPI supplies
dwDisplayableAddressSize	TAPI supplies
dwDisplayableAddressOffset	TAPI supplies
dwCalledPartySize	TAPI supplies
dwCalledPartyOffset	TAPI supplies
dwCommentSize	TAPI supplies
dwCommentOffset	TAPI supplies
dwDisplaySize	When display information is provided, the display size is 2*24 bytes. All phone types with display have a 2 line display.
dwDisplayOffset	
dwUserUserInfoSize	
dwUserUserInfoOffset	
dwHighLevelCompSize	
dwHighLevelCompOffset	
dwLowLevelCompSize	
dwLowLevelCompOffset	
dwChargingInfoSize	n/s for Hicom 300, HiPath 4000 CBXs provided for Hicom 100 E, Hicom 150 E and HiPath 3000 CBXs
dwChargingInfoOffset	n/s for Hicom 300, HiPath 4000 CBXs provided for Hicom 100 E, Hicom 150 E and HiPath 3000 CBXs
dwTerminalModesSize	n/s
dwTerminalModesOffset	n/s
dwDevSpecificSize	If device specific data is available and an extension version has been selected, these fields are set to the size and offset of the device specific data. No device specific fields are currently defined for CallBridge Collection.
dwDevSpecificOffset	
dwCallTreatment	n/s
dwCallDataSize	
dwCallDataOffset	
dwSendingFlowspecSize	n/s
dwSendingFlowspecOffset	n/s
dwReceivingFlowspecSize	n/s

dwReceivingFlowspecOffset	n/s

7.4 LINECALLPARAMS

Table 6-4. Notes on LINECALLPARAMS

Field Name	**Constant Value ("n/s" if not supported)**
dwBearerMode	LINEBEARERMODE_VOICE
dwMinRate	0
dwMaxRate	0
dwMediaMode	LINEMEDIAMODE_INTERACTIVEVOICE
dwCallParamFlags	LINECALLPARAMFLAGS_ORIGOFFHOOK [1] LINECALLPARAMFLAGS_IDLE [2] LINECALLPARAMFLAGS_BLOCKID LINECALLPARAMFLAGS_ONESTEPTRANSFER [3] [1] May only be TRUE if the LINEADDRCAPFLAGS ORIGOFFHOOK is TRUE. If ORIGOFFHOOK is FALSE, the *offhook* behavior of the originating phone is dependent only on whether or not the phone has a microphone. If the phone has a microphone, it will automatically be taken offhook when a call is initiated. If the phone does not have a microphone, the user must lift the handset in order to talk to the destination party. [2] **lineMakeCall** may only be used if there is no call on the selected address. In other function calls this flag is always ignored. [3] only for Hicom 100 E, Hicom 150 E, HiPath 3000 and HiPath 4000 V2.0 CBXs
dwAddressMode	LINEADDRESSMODE_ADDRESSID
dwAddressID	The address supplied in LINECALLPARAMS is ignored unless the line request is **lineMakeCall**.
DialParams	DefaultDialParams from LINEDEVCAPS are overwritten by these values.
dwOrigAddressSize	The address supplied in LINECALLPARAMS is ignored unless the line request is **lineMakeCall**.
dwOrigAddressOffset	
dwDisplayableAddressSize	TAPI supplies
dwDisplayableAddressOffset	TAPI supplies
dwCalledPartySize	TAPI supplies
dwCalledPartyOffset	TAPI supplies
dwCommentSize	TAPI supplies
dwCommentOffset	TAPI supplies
dwUserUserInfoSize	n/s
dwUserUserInfoOffset	n/s
dwHighLevelCompSize	n/s
dwHighLevelCompOffset	n/s

dwLowLevelCompSize	n/s
dwLowLevelCompOffset	n/s
dwDevSpecificSize	n/s
dwDevSpecificOffset	n/s
dwPredictiveAutoTransferStates	n/s
dwTargetAddressSize	n/s for Hicom 300, HiPath 4000 V1.0 CBXs provided for Hicom 150 E CBXs starting with Hicom 150 E Office and HiPath 4000 starting with V2.0
dwTargetAddressOffset	n/s for Hicom 300, HiPath 4000 V1.0 CBXs provided for Hicom 150 E CBXs starting with Hicom 150 E Office and HiPath 4000 starting with V2.0
dwSendingFlowspecSize	n/s
dwSendingFlowspecOffset	n/s
dwReceivingFlowspecSize	n/s
dwReceivingFlowspecOffset	n/s
dwDeviceClassSize	n/s
dwDeviceClassOffset	n/s
dwDeviceConfigSize	n/s
dwDeviceConfigOffset	n/s
dwCallDataSize	
dwCallDataOffset	
dwNoAnswerTimeout	n/s
dwCallingPartyIDSize	n/s
dwCallingPartyIDOffset	n/s

7.5 LINECALLSTATUS

Table 6-5. Notes on LINECALLSTATUS

Field Name	**Constant Value ("n/s" if not supported)**
dwCallState	
dwCallStateMode	If **dwCallState** is: - LINECALLSTATE_BUSY, this field is LINEBUSYMODE_UNAVAIL – LINECALLSTATE_DIALTONE, this field is LINEDIALTONEMODE_UNAVAIL – LINECALLSTATE_ DISCONNECTED, this field is LINEDISCONNECTMODE_UNAVAIL
dwCallPrivilege	TAPI supplies
dwCallFeatures	
dwDevSpecificSize	n/s

dwDevSpecificOffset	n/s
dwCallFeatures2	n/s for Hicom 300 CBXs for Hicom 150 E CBXs starting with Hicom 150 E Office: LINECALLFEATURE2_TRANSFERCONF LINECALLFEATURE2_TRANSFERNORM LINECALLFEATURE2_COMPLCALLBACK LINECALLFEATURE2_COMPLINTRUDE LINECALLFEATURE2_COMPLMESSAGE LINECALLFEATURE2_PARKDIRECT LINECALLFEATURE2_ONESTEPTRANSFER for HiPath 4000 starting with HiPath 4000 V2.0: LINECALLFEATURE2_TRANSFERCONF LINECALLFEATURE2_TRANSFERNORM LINECALLFEATURE2_COMPLCALLBACK LINECALLFEATURE2_COMPLINTRUDE LINECALLFEATURE2_COMPLCAMPON LINECALLFEATURE2_PARKDIRECT LINECALLFEATURE2_ONESTEPTRANSFER
tStateEntryTime	n/s for Hicom 300, HiPath 4000 V1.0 CBXs provided for Hicom 100 E, Hicom 150 E, HiPath 3000 amd HiPath 4000 V2.0 CBXs

7.6 LINEDEVCAPS

Table 6-6. Notes on LINEDEVCAPS

Field Name	**Constant Value ("n/s" if <u>not</u> supported)**
dwProviderInfoSize	A string containing the following: "CorNet-TS TAPI Service Provider x" where 'x' stands for 1 = Hicom 150 / HiPath 3000 TSP 2 = Hicom 300 / HiPath 4000 V1.0 TSP 4 = HiPath 4000 V2.0 TSP
dwProviderInfoOffset	
dwSwitchInfoSize	Contains one of the following strings: For Hicom 300, HiPath 4000 V1.0 CBXs: "Hicom 300 Rel. 3.4"; "Hicom 300 Rel. 3.5"; "Hicom 300 Rel. 3.6"; "Hicom 300 EV Rel. 1.0"; "Hicom 300 EV Rel. 2.0"; "Hicom 300 EV Rel. 3.0"; "HiPath 4000 >= Rel. 1.0" For HiPath 4000 (> V2.0) CBXs: "HiPath 4000 >= Rel. 2.0" For Hicom 100 E and Hicom 150 E CBXs: "Hicom 100 E"; "Hicom 150 E"; "Hicom 150 E Office"; "Hicom 150 E Office V2.2"; "Hicom 150 H"; "HiPath AllServe 150"; "HiPath 3000"
dwSwitchInfoOffset	
dwPermanentLineID	
dwLineNameSize	

dwLineNameOffset	
dwStringFormat	STRINGFORMAT_ASCII
dwAddressModes	LINEADDRESSMODE_ADDRESSID
dwNumAddresses	1
dwBearerModes	LINEBEARERMODE_VOICE
dwMaxRate	0
dwMediaModes	LINEMEDIAMODE_INTERACTIVEVOICE
dwGenerateToneModes	n/s
dwGenerateToneMaxNumFreq	n/s
dwGenerateDigitModes	LINEDIGITMODE_DTMF
dwMonitorToneMaxNumFreq	n/s
dwMonitorToneMaxNumEntries	n/s
dwMonitorDigitModes	n/s
dwGatherDigitsMinTimeout	n/s
dwGatherDigitsMaxTimeout	n/s
dwMedCtlDigitMaxListSize	n/s
dwMedCtlMediaMaxListSize	n/s
dwMedCtlToneMaxListSize	n/s
dwMedCtlCallStateMaxListSize	n/s
dwDevCapFlags	0
dwMaxNumActiveCalls	8 for Hicom 300, HiPath 4000 CBXs 1 for Hicom 100 E, Hicom 150 E, HiPath 3000 CBXs
dwAnswerMode	LINEANSWERMODE_HOLD
dwRingModes	12
dwLineStates	LINEDEVSTATE_OTHER [1] LINEDEVSTATE_RINGING LINEDEVSTATE_OPEN LINEDEVSTATE_CLOSE LINEDEVSTATE_CONNECTED LINEDEVSTATE_MSGWAITON LINEDEVSTATE_MSGWAITOFF LINEDEVSTATE_INSERVICE LINEDEVSTATE_OUTOFSERVICE LINEDEVSTATE_NUMCALLS LINEDEVSTATE_DEVSPECIFIC LINEDEVSTATE_DISCONNECTED [2] LINEDEVSTATE_LOCK [3] Other bits may also be set by TAPI. [1] used to report changes in the dwLineFeatures field. [2] only for Hicom 100 / 150, HiPath 3000, HiPath 4000 V2.0 CBXs [3] only for Hicom 100 / 150, HiPath 3000 CBXs

dwUUIAcceptSize	n/s
dwUUIAnswerSize	n/s
dwUUIMakeCallSize	n/s
dwUUIDropSize	n/s
dwUUISendUserUserInfoSize	n/s
dwUUICallInfoSize	n/s
MinDialParams	= min. DialParams
MaxDialParams	= max. DialParams
DefaultDialParams	= def. DialParams
dwNumTerminals	0
dwTerminalCapsSize	n/s
dwTerminalCapsOffset	n/s
dwTerminalTextEntrySize	n/s
dwTerminalTextSize	n/s
dwTerminalTextOffset	n/s
dwDevSpecificSize	If device specific data is available and an extension version has been selected, these fields are set to the size and offset of the device specific data. Refer to "SIEMENS_LINEDEVCAPS_DEV_SPECIFIC" in chapter 9 for the format and description of the device specific data.
DwDevSpecificOffset	
DwLineFeatures	LINEFEATURE_MAKECALL LINEFEATURE_FORWARD LINEFEATURE_DEVSPECIFIC LINEFEATURE_DEVSPECIFICFEAT LINEFEATURE_FORWARDFWD [1] LINEFEATURE_ FORWARDDND [1] LINEFEATURE_SETDEVSTATUS [2] [1] only available for Hicom 150 E Office, HiPath 3000, HiPath 4000 V2.0 CBXs [2] only available for Hicom 150 E Office , HiPath 3000 CBXs
DwSettableDevStatus	n/s for Hicom 300, HiPath 4000 CBXs For Hicom 150 E CBXs starting with Hicom 150 E Office V2.2: LINEDEVSTATUSFLAGS_MSGWAIT LINEDEVSTATUSFLAGS_LOCKED
DwDeviceClassesSize	
DwDeviceClassesOffset	

7.7 LINEDEVSTATUS

Note: all fields in this structure except **dwDevStatusFlags** are set to zero if the line is not *in service*.

Table 6-7. Notes on LINEDEVSTATUS

Field Name	Constant Value ("n/s" if not supported)
DwNumOpens	TAPI supplies
dwOpenMediaModes	TAPI supplies
DwNumActiveCalls	
dwNumOnHoldCalls	
dwNumOnHoldPendCalls	
DwLineFeatures	LINEFEATURE_DEVSPECIFIC LINEFEATURE_DEVSPECIFICFEAT LINEFEATURE_MAKECALL LINEFEATURE_FORWARD LINEFEATURE_ FORWARDFWD LINEFEATURE_ FORWARDDND LINEFEATURE_SETDEVSTATUS [1] [1] only for Hicom 150 E CBXs starting with Hicom 150 E Office V2.2
dwNumCallCompletions	0 CallBridge Collection cannot determine the number of pending callbacks.
DwRingMode	Refer to chapter 9: SIEMENSRINGMODE Constants
DwSignalLevel	n/s
DwBatteryLevel	n/s
DwRoamMode	LINEROAMMODE_UNAVAIL
DwDevStatusFlags	LINEDEVSTATUSFLAGS_CONNECTED LINEDEVSTATUSFLAGS_INSERVICE LINEDEVSTATUSFLAGS_MSGWAIT LINEDEVSTATUSFLAGS_LOCKED [1] [1] only for Hicom 150 E CBXs starting with Hicom 150 E Office V2.2
dwTerminalModesSize	n/s
dwTerminalModesOffset	n/s
DwDevSpecificSize	If device specific data is available and an extension version has been selected, these fields are set to the size and offset of the device specific data. Refer to "SIEMENS_LINEDEVSTATUS_DEV_SPECIFIC" in chapter 9 for the format and description of the device specific data.
dwDevSpecificOffset	
dwAvailableMediaModes	LINEMEDIAMODE_INTERACTIVEVOICE
dwAppInfoSize	TAPI supplies
dwAppInfoOffset	

7.8 LINEDIALPARAMS

Table 6-8. Notes on LINEDIALPARAMS

Field Name	Constant Value ("n/s" if not supported)
dwDialPause	in milliseconds [1] min. = 0 def. = 2000 max. = 255000
dwDialSpeed	in milliseconds [2] min. = 50 def. = 95 max. = 255
dwDigitDuration	in milliseconds [3] min. = 50 def. = 95 max. = 255
dwWaitForDialtone	in milliseconds [4] min. = 0 def. = 2000 max. = 255000

[1] The pause value for a comma in dialstring. Multiple commas may be used when longer pauses are required.

[2] The TSPs digit spacing when sending DTMF digits. Other dial strings are always sent in block dialing mode.

[3] A dummy value, the actual digit duration is controlled by the CBX.

[4] Always ignored – irrelevant on a digital phone.

7.9 LINEEXTENSIONID

Table 6-9. Notes on LINEEXTENSIONID

Field Name	Constant Value ("n/s" if not supported)
dwExtensionID0	SIEMENS_DW_EXTENSION_ID_0 (**0x3C224D5B**) for Hicom 300, HiPath 4000 V1.0 CBXs (**0x373AECF0**) for Hicom 100 E / 150 E CBXs (**0x9D7BD7DA**) for HiPath 4000 V2.0 CBXs
dwExtensionID1	SIEMENS _DW_EXTENSION_ID_1 (**0x6C2E4B7B**) for Hicom 300, HiPath 4000 V1.0 CBXs (**0xAB734C55**) for Hicom 100 E / 150 E CBXs (**0x51F94069**) for HiPath 4000 V2.0 CBXs
dwExtensionID2	SIEMENS _DW_EXTENSION_ID_2 (**0x911FA8DD**) for Hicom 300, HiPath 4000 V1.0 CBXs (**0x83492CEE**) for Hicom 100 E / 150 E CBXs (**0x94B1D7F0**) for HiPath 4000 V2.0 CBXs
dwExtensionID3	SIEMENS _DW_EXTENSION_ID_3 (**0x08707715**) for Hicom 300, HiPath 4000 V1.0 CBXs (**0xB8D0FA00**) for Hicom 100 E / 150 E CBXs (**0x5AA979A3**) for HiPath 4000 V2.0 CBXs

Note: these are the extension IDs that uniquely identify CallBridge Collection

7.10 LINEFORWARD

Table 6-10. Notes on LINEFORWARD

Field Name	Constant Value ("n/s" if not supported)
dwForwardMode	LINEFORWARDMODE_UNCOND [1] LINEFORWARDMODE_BUSY [2] LINEFORWARDMODE_NOANSW [2] LINEFORWARDMODE_BUSYNA [2] LINEFORWARDMODE_UNCONDINTERNAL [3] LINEFORWARDMODE_UNCONDEXTERNAL [3] [1] All CBXs [2] only Hicom 150 E Modular, HiPath 4000 V2.0 CBXs [3] Hicom 100 E, Hicom 150 E, HiPath 3000, HiPath 4000 V2.0 CBXs
dwCallerAddressSize	n/s
dwCallerAddressOffset	n/s
dwDestCountryCode	n/s
dwDestAddressSize	
dwDestAddressOffset	

7.11 LINEFORWARDLIST

Table 6-11. Notes on LINEFORWARDLIST

Field Name	Constant Value ("n/s" if not supported)
dwNumEntries	0 1 2 0: set DnD 1: set unique forward mode 2: set different forward modes at the same time [1] [1] only for HiPath 4000 V2.0 CBXs, works only with LFM_ UNCONDINTERNAL and LFM_ UNCONDEXTERNAL
ForwardList	

8 Phone Device Data Structures

The following chapter provides information about phone device data structures.If device-specific data is included in a data structure, a reference is given to the section where the format of the device-specific data is described.

If a data structure is not included in this section, either it is not required by any phone device function supported by CallBridge Collection , or is it used in a function supported exclusively by TAPI without any interaction with the service provider.

The notes in this section indicate which fields in the data structures are supported and which fields have constant values.

The tables in this chapter used to describe the fields of the data structures apply the following rules:

1. If a field is indicated as not supported ("n/s"), it will be initialized with a default value of 0.
2. If a field is static, its constant value is shown.
3. If a field is dynamic, the "value" field is left blank.
4. Fields which are filled in by TAPI and not by the Service Provider are marked as "TAPI supplies".
5. The **dwTotalSize**, **dwNeededSize**, and **dwUsedSize** fields are omitted from the tables since they provide no useful information for the purpose of this document.
6. Double lines in the tables mark the logical field groupings found in the Microsoft TAPI specification.

8.1 PHONEBUTTONINFO

Table 7-1. Notes on PHONEBUTTONINFO

Field Name	**Constant Value ("n/s" if not supported)**
dwButtonMode	PHONEBUTTONMODE_KEYPAD PHONEBUTTONMODE_LOCAL [1] PHONEBUTTONMODE_DISPLAY [2] PHONEBUTTONMODE_FEATURE [1] buttons for volume control [2] buttons for menu control
dwButtonFunction	PHONEBUTTONFUNCTION_UNKNOWN PHONEBUTTONFUNCTION_VOLUMEUP PHONEBUTTONFUNCTION_ VOLUMEDOWN
dwButtonTextSize	
dwButtonTextOffset	
dwDevSpecificSize	If device-specific data is available and an extension version has been selected, these fields are set to the size and offset of the device-specific data. No device specific extensions are currently defined for CallBridge Collection.
dwDevSpecificOffset	
dwButtonState	PHONEBUTTONSTATE_UNAVAIL

	Note: CallBridge Collection cannot determine whether a button is in the *up* or the *down* position.

8.2 PHONECAPS

Table 7-2. Notes on PHONECAPS

Field Name	**Constant Value ("n/s" if not supported)**
dwProviderInfoSize	A string containing the following: "CorNet-TS TAPI Service Provider x" where 'x' stands for 1 = Hicom 150 / HiPath 3000 TSP 2 = Hicom 300 / HiPath 4000 V1.0 TSP 4 = HiPath 4000 V2.0 TSP
dwProviderInfoOffset	
dwPhoneInfoSize	A string containing informations of the configured phones: „optiset E basic" „optiset E comfort / advance plus" „optiset E memory" „optiPoint 500 basic" „optiPoint 500 standard" „optiPoint 500 advance" „optiPoint 600 office" „optiPoint 410 entry" „optiPoint 410 economy" „optiPoint 410 standard" „optiPoint 410 advance"
dwPhoneInfoOffset	
dwPermanentPhoneID	
dwPhoneNameSize	A string containing the TAPI phone name, e.g. "CallBridge TU" (for optiPoint UP0E devices)
dwPhoneNameOffset	
dwStringFormat	STRINGFORMAT_ASCII (**TAPI Version 1.3, 1.4**)
dwPhoneStates	PHONESTATE_OTHER PHONESTATE_CONNECTED PHONESTATE_DISCONNECTED 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_DEVSPECIFIC
dwHookSwitchDevs	PHONEHOOKSWITCHDEV_HANDSET PHONEHOOKSWITCHDEV_SPEAKER PHONEHOOKSWITCHDEV_HEADSET
dwHandsetHookSwitchModes	PHONEHOOKSWITCHMODE_ONHOOK PHONEHOOKSWITCHMODE_SPEAKER PHONEHOOKSWITCHMODE_MICSPEAKER PHONEHOOKSWITCHMODE_UNKNOWN
dwSpeakerHookSwitchModes	PHONEHOOKSWITCHMODE_ONHOOK PHONEHOOKSWITCHMODE_SPEAKER PHONEHOOKSWITCHMODE_MICSPEAKER **[1]** PHONEHOOKSWITCHMODE_UNKNOWN **[1]** This field is set based on whether the phone has a microphone as well as a speaker.
dwHeadsetHookSwitchModes	PHONEHOOKSWITCHMODE_ONHOOK PHONEHOOKSWITCHMODE_SPEAKER PHONEHOOKSWITCHMODE_MICSPEAKER PHONEHOOKSWITCHMODE_UNKNOWN
dwVolumeFlags	PHONEHOOKSWITCHDEV_HANDSET PHONEHOOKSWITCHDEV_SPEAKER PHONEHOOKSWITCHDEV_HEADSET
dwGainFlags	PHONEHOOKSWITCHDEV_HANDSET PHONEHOOKSWITCHDEV_SPEAKER PHONEHOOKSWITCHDEV_HEADSET
dwDisplayNumRows	the display size is device dependent
dwDisplayNumColumns	the display size is device dependent
dwNumRingModes	12 [1] [1] Please refer to “SIEMENSRINGMODE Constants” in chapter 9. The ring mode cannot be controlled with **phoneSetRing**, only the volume can be set by the application.
dwNumButtonLamps	For Hicom 300; HiPath 4000 CBXs: Maximum number of buttons is **100** (phone unit with **12** keypad buttons plus **24** feature buttons, plus a maximum of 4 virtual or physical add-on units with **16** buttons each). Depending on the CBX type, buttons may be reported for add-on units that are not actually configured. These buttons will be reported as PHONEBUTTONFUNCTION_UNKNOWN For Hicom 100 E and Hicom 150 E CBXs: 161 (neither physical nor virtual add-on units are needed)
dwButtonModesSize	
dwButtonModesOffset	PHONEBUTTONMODE_KEYPAD PHONEBUTTONMODE_LOCAL **[1]**

	PHONEBUTTONMODE_DISPLAY [2] PHONEBUTTONMODE_FEATURE [1] buttons for volume control [2] buttons for menu control
dwButtonFunctionsSize	
dwButtonFunctionsOffset	PHONEBUTTONFUNCTION_ xyz [1] SIEMENSBUTTONFUNCTION_xyz [2] [1] please refer to “PHONEBUTTONFUNCTION Constants” in chapter 8 [2] please refer to “SIEMENSBUTTONFUNCTION Constants” in chapter 8
dwLampModesSize	
dwLampModesOffset	PHONELAMPMODE_DUMMY PHONELAMPMODE_FLASH (LED cadence 3 – 500/500 ms – and 6 – 750/750 ms) PHONELAMPMODE_FLUTTER (LED cadence 1 – 50/50 ms – and 4 – 50/100 ms) PHONELAMPMODE_OFF PHONELAMPMODE_STEADY PHONELAMPMODE_WINK (LED cadence 2 – 450/50 ms – and 5 – 250/250 ms)
dwNumSetData	
dwSetDataSize	
dwSetDataOffset	
dwNumGetData	
dwGetDataSize	
dwGetDataOffset	
dwDevSpecificSize	n/s
dwDevSpecificOffset	n/s
dwDeviceClassesSize	A string containing the following: "tapi/phone"
dwDeviceClassesOffset	
dwPhoneFeatures	PHONEFEATURE_GETBUTTONINFO PHONEFEATURE_GETDATA PHONEFEATURE_GETDISPLAY PHONEFEATURE_GETGAINHANDSET PHONEFEATURE_GETGAINSPEAKER PHONEFEATURE_GETGAINHEADSET PHONEFEATURE_GETHOOKSWITCHHANDSET PHONEFEATURE_GETHOOKSWITCHSPEAKER PHONEFEATURE_GETHOOKSWITCHHEADSET PHONEFEATURE_GETLAMP PHONEFEATURE_GETRING PHONEFEATURE_GETVOLUMEHANDSET PHONEFEATURE_GETVOLUMESPEAKER PHONEFEATURE_GETVOLUMEHEADSET PHONEFEATURE_SETDATA PHONEFEATURE_SETGAINHANDSET PHONEFEATURE_SETGAINSPEAKER PHONEFEATURE_SETGAINHEADSET PHONEFEATURE_SETHOOKSWITCHHANDSET

	PHONEFEATURE_SETHOOKSWITCHSPEAKER PHONEFEATURE_SETHOOKSWITCHHEADSET PHONEFEATURE_SETRING PHONEFEATURE_SETVOLUMEHANDSET PHONEFEATURE_SETVOLUMESPEAKER PHONEFEATURE_SETVOLUMEHEADSET
DwSettableHandsetHookSwitch Modes	PHONEHOOKSWITCHMODE_SPEAKER PHONEHOOKSWITCHMODE_MICSPEAKER
DwSettableSpeakerHookSwitch Modes	PHONEHOOKSWITCHMODE_ONHOOK PHONEHOOKSWITCHMODE_SPEAKER PHONEHOOKSWITCHMODE_MICSPEAKER
DwSettableHeadsetHookSwitch Modes	PHONEHOOKSWITCHMODE_ONHOOK PHONEHOOKSWITCHMODE_SPEAKER PHONEHOOKSWITCHMODE_MICSPEAKER
dwMonitoredHandsetHook SwitchModes	PHONEHOOKSWITCHMODE_ONHOOK PHONEHOOKSWITCHMODE_SPEAKER PHONEHOOKSWITCHMODE_MICSPEAKER PHONEHOOKSWITCHMODE_UNKNOWN
dwMonitoredSpeakerHook SwitchModes	PHONEHOOKSWITCHMODE_ONHOOK PHONEHOOKSWITCHMODE_SPEAKER PHONEHOOKSWITCHMODE_MICSPEAKER PHONEHOOKSWITCHMODE_UNKNOWN
dwMonitoredHeadsetHook SwitchModes	PHONEHOOKSWITCHMODE_ONHOOK PHONEHOOKSWITCHMODE_SPEAKER PHONEHOOKSWITCHMODE_MICSPEAKER PHONEHOOKSWITCHMODE_UNKNOWN

8.3 PHONEEXTENSIONID

Table 7-3. Notes on PHONEEXTENSIONID

Field Name	**Constant Value ("n/s" if not supported)**
dwExtensionID0	SIEMENS_DW_EXTENSION_ID_0 (**0x3C224D5B**) for Hicom 300, HiPath 4000 V1.0 CBXs (**0x373AECF0**) for Hicom 100 E / 150 E CBXs (**0x9D7BD7DA**) for HiPath 4000 V2.0 CBXs
dwExtensionID1	SIEMENS _DW_EXTENSION_ID_1 (**0x6C2E4B7B**) for Hicom 300, HiPath 4000 V1.0 CBXs (**0xAB734C55**) for Hicom 100 E / 150 E CBXs (**0x51F94069**) for HiPath 4000 V2.0 CBXs
dwExtensionID2	SIEMENS _DW_EXTENSION_ID_2 (**0x911FA8DD**) for Hicom 300, HiPath 4000 V1.0 CBXs (**0x83492CEE**) for Hicom 100 E / 150 E CBXs (**0x94B1D7F0**) for HiPath 4000 V2.0 CBXs
dwExtensionID3	SIEMENS _DW_EXTENSION_ID_3 (**0x08707715**) for Hicom 300, HiPath 4000 V1.0 CBXs (**0xB8D0FA00**) for Hicom 100 E / 150 E CBXs (**0x5AA979A3**) for HiPath 4000 V2.0 CBXs

Note: these are the extension IDs that uniquely identify CallBridge Collection.

8.4 PHONESTATUS

Note: if the phone is not *in service*, all fields in this structure are set to default values, except the **dwStatusFlags** field.

Table 7-4. Notes on PHONESTATUS

Field Name	**Constant Value ("n/s" if not supported)**
dwStatusFlags	PHONESTATUSFLAGS_CONNECTED
dwNumOwners	TAPI supplies
dwNumMonitors	TAPI supplies
dwRingMode	
dwRingVolume	
dwHandsetHookSwitchMode	PHONEHOOKSWITCHMODE_ONHOOK PHONEHOOKSWITCHMODE_SPEAKER PHONEHOOKSWITCHMODE_MICSPEAKER PHONEHOOKSWITCHMODE_UNKNOWN
dwHandsetVolume	
dwHandsetGain	
dwSpeakerHookSwitchMode	PHONEHOOKSWITCHMODE_ONHOOK PHONEHOOKSWITCHMODE_SPEAKER PHONEHOOKSWITCHMODE_MICSPEAKER PHONEHOOKSWITCHMODE_UNKNOWN
dwSpeakerVolume	Field is meaningful only when dwSpeakerHookSwitchMode indicates that the device's speaker is active.
dwSpeakerGain	0x00000000 [1] 0x00007FFF [1] [1] indicates only the ON or OFF state which can be changed with the mute feature
dwHeadsetHookSwitchMode	PHONEHOOKSWITCHMODE_ONHOOK PHONEHOOKSWITCHMODE_SPEAKER PHONEHOOKSWITCHMODE_MICSPEAKER PHONEHOOKSWITCHMODE_UNKNOWN
dwHeadsetVolume	Field is meaningful only when dwHeadsetHookSwitchMode indicates that the device's speaker is active.
dwHeadsetGain	0x00000000 [1] 0x00007FFF [1] [1] indicates only the ON or OFF state which can be changed with the mute feature
dwDisplaySize	When display information is provided, the display size is device dependent.

dwDisplayOffset	
dwLampModesSize	
dwLampModesOffset	PHONELAMPMODE_DUMMY PHONELAMPMODE_FLASH PHONELAMPMODE_FLUTTER PHONELAMPMODE_OFF PHONELAMPMODE_STEADY PHONELAMPMODE_WINK
dwOwnerNameSize	TAPI supplies
dwOwnerNameOffset	TAPI supplies
dwDevSpecificSize	If device specific data is available and an extension version has been selected, these fields are set to the size and offset of the device specific data. Refer to “SIEMENS_PHONESTATUS_DEV_SPECIFIC” in chapter 9 for the format and description of the device specific data.
dwDevSpecificOffset	
dwPhoneFeatures	PHONEFEATURE_GETBUTTONINFO PHONEFEATURE_GETDATA PHONEFEATURE_GETDISPLAY PHONEFEATURE_GETGAINHANDSET PHONEFEATURE_GETGAINSPEAKER PHONEFEATURE_GETGAINHEADSET PHONEFEATURE_GETHOOKSWITCHHANDSET PHONEFEATURE_GETHOOKSWITCHSPEAKER PHONEFEATURE_GETHOOKSWITCHHEADSET PHONEFEATURE_GETLAMP PHONEFEATURE_GETRING PHONEFEATURE_GETVOLUMEHANDSET PHONEFEATURE_GETVOLUMESPEAKER PHONEFEATURE_GETVOLUMEHEADSET PHONEFEATURE_SETDATA PHONEFEATURE_SETHOOKSWITCHSPEAKER PHONEFEATURE_SETHOOKSWITCHHEADSET

9 Line and Phone Device Constants

The following chapter provides information about TAPI constants which have been extended to include Siemens device-specific values. An '**X**' in the CBX columns denotes the availability of a feature on that CBX type.

9.1 PHONEBUTTONFUNCTION Constants

Table 8-1. PHONEBUTTONFUNCTION Constants

Function ID constants are prefixed with PHONEBUTTON FUNCTION (PBF)	Code	Default Display Name	CBX's: 1-1 Hicom 100E - opti_150.tsp - 1-2 Hicom 150 E - opti_150.tsp - 2 Hicom 300/HiPath 4000 V1.0 - opti_300.tsp - 4 HiPath 4000 V2.0 - optiH4V2.tsp -			
			CBX 4	CBX 2	CBX 1-2	CBX 1-1
PBF_UNKNOWN	0x00000000		X	X	X	X
PBF_CONFERENCE	0x00000001	Conference Konferenz	X	X	X	X
PBF_TRANSFER	0x00000002	Consultation/transfer Rückfrage (Signaltaste)	X	X	X	X
PBF_DROP	0x00000003	Drop Trennen			X	
PBF_HOLD	0x00000004	Hold Halten			X	X
PBF_RECALL	0x00000005	Unhold -			X	
PBF_DISCONNECT	0x00000006	Disconnect key -			X	
PBF_CONNECT	0x00000007	Connect key -			X	
PBF_MSGWAITON	0x00000008	Message waiting LED on -			X	
PBF_MSGWAITOFF	0x00000009	Message waiting LED off -			X	

Function ID constants are prefixed with PHONEBUTTON FUNCTION (PBF)	Code	Default Display Name	CBX's: 1-1 Hicom 100E - opti_150.tsp - 1-2 Hicom 150 E - opti_150.tsp - 2 Hicom 300/HiPath 4000 V1.0 - opti_300.tsp - 4 HiPath 4000 V2.0 - optiH4V2.tsp -			
			CBX 4	CBX 2	CBX 1-2	CBX 1-1
PBF_ABBREVDIAL	0x0000000B	Speed dialing Kurzwahl			X	
PBF_FORWARD	0x0000000C	Forward key Anruf Umleiten		X	X [1]	
PBF_PICKUP	0x0000000D	Pickup key Anrufübernahme			X	X
PBF_RINGAGAIN	0x0000000E	Complete call – callback Rückruf setzen			X	
PBF_PARK	0x0000000F	Park key (System Park) Parken (System)			X	X
PBF_MUTE	0x00000012	Mute key Stummschalten, Micro ein/aus		X	X	X
PBF_VOLUMEUP	0x00000013	Volume up Laut	X	X	X	X
PBF_VOLUMEDOWN	0x00000014	Volume down Leise	X	X	X	X
PBF_SPEAKERON	0x00000015	Speaker key Lautsprecher	X	X	X	X
PBF_SPEAKEROFF	0x00000016	Speaker key Lautsprecher	X	X	X	
PBF_FLASH	0x00000017	Flash Erden der Leitung			X	
PBF_DONOTDISTURB	0x0000001A	Do Not Disturb Anrufschutz		X	X	X
PBF_INTERCOM	0x0000001B	Intercom community speaker call Lautsprecherdurchsagen			X	
PBF_BUSY	0x0000001D	Do Not Disturb Anrufschutz			X	
PBF_DIRECTORY	0x00000020	Directory dialing Kurzwahl			X	
PBF_COVER	0x00000021	Call Forwarding Anruf umleiten			X	
PBF_CALLID	0x00000022	Malicious Call Identification Rufnummerunterdrückung			X	

Function ID constants are prefixed with PHONEBUTTON FUNCTION (PBF)	Code	Default Display Name	CBX's: 1-1 Hicom 100E - opti_150.tsp - 1-2 Hicom 150 E - opti_150.tsp - 2 Hicom 300/HiPath 4000 V1.0 - opti_300.tsp - 4 HiPath 4000 V2.0 - optiH4V2.tsp -			
			CBX 4	CBX 2	CBX 1-2	CBX 1-1
PBF_LASTNUM	0x00000023	Redial (last number) Wahlwiederholung (zuletzt gewählte)			X	X
PBF_NIGHTSRV	0x00000024	Universal night answer Nachtschaltung			X	X
PBF_SENDCALLS	0x00000025	Call Forwarding Anruf umleiten			X	
PBF_MSGINDICATOR	0x00000026	Set Message Indicator			X	
PBF_REPDIAL	0x00000027	Autodial / direct destination select key Namentaste			X	X
PBF_SETREPDIAL	0x00000028	Set repdial key -			X	
PBF_SYSTEMSPEED	0x00000029	System speed - group 1 Systemnamentaste - Gruppe 1			X	X
PBF_STATIONSPEED	0x0000002A	Individual system speed Systemnamentaste - individuell			X	X
PBF_CAMPON	0x0000002B	Camp-on Aufschalten/Anklopfen	X	X	X	X
PBF_SAVEREPEAT	0x0000002C	Save and repeat Wahlwiederholung (speichern/wählen)			X	
PBF_QUEUECALL	0x0000002D	HardHold (line specific) -			X	
PBF_NONE	0x0000002E		X	X	X	X

[1] Available only on Hicom 150 E Office V 2.2 and higher.

9.2 SIEMENSBUTTONFUNCTION Constants

Table 8-2 Siemens Extensions to the PHONEBUTTONFUNCTION Constants

Function ID constants are prefixed with SIEMENSBUTTON FUNCTION (SBF)	Code	Default Display Name	CBX's: 1-1 Hicom 100E - opti_150.tsp - 1-2 Hicom 150 E - opti_150.tsp - 2 Hicom 300/HiPath 4000 V1.0 - opti_300.tsp - 4 HiPath 4000 V2.0 - optiH4V2.tsp -			
			CBX 4	CBX 2	CBX 1-2	CBX 1-1
SBF_CALLWAITWTONEOFF	0x80000017	Call Waiting w/o signal tone off Anklopfen ohne Ton aus	-	-	X	-
SBF_CALLWAITWTONEON	0x80000018	Call Waiting w/o signal tone on Anklopfen ohne Ton an	-	-	X	-
SBF_CONFQUIT	0x80000019	Cancel Conference Call Konferenz beenden	-	-	X	-
SBF_MSGWAIT_CANCEL_TERMIN	0x8000001A	Cancel Message Waiting from terminator Message Waiting löschen vom Ziel	X	-	-	-
SBF_MSGWAIT_CANCEL_ORIGIN	0x8000001B	Cancel Message Waiting from originator Message Waiting löschen vom Ursprung	X	-	-	-
SBF_MSGWAIT_ACTIVATE	0x8000001C	Activate Message Waiting Message Waiting aktivieren	X	-	-	-
SBF_SECRSUBSTITUTEOFF	0x8000001D	Secretarial Substitute off Vertretung auf Sekretärin aus	X	-	-	-
SBF_SECRSUBSTITUTEON	0x8000001E	Secretarial Substitute on Vertretung auf Sekretärin an	X	-	-	-
SBF_SECRINTERCEPT4OFF	0x8000001F	Secretarial Interception for Executive 4 off Abwurf auf Sekretärin für Chef 4 aus	X	-	-	-
SBF_SECRINTERCEPT4ON	0x80000020	Secretarial Interception for Executive 4 on Abwurf auf Sekretärin für Chef 4 an	X	-	-	-
SBF_SECRINTERCEPT3OFF	0x80000021	Secretarial Interception for Executive 3 off Abwurf auf Sekretärin für Chef 3 aus	X	-	-	-
SBF_SECRINTERCEPT3ON	0x80000022	Secretarial Interception for Executive 3 on Abwurf auf Sekretärin für Chef 3 an	X	-	-	-
SBF_SECRINTERCEPT2OFF	0x80000023	Secretarial Interception for Executive 2 off Abwurf auf Sekretärin für Chef 2 aus	X	-	-	-
SBF_SECRINTERCEPT2ON	0x80000024	Secretarial Interception for Executive 2 on Abwurf auf Sekretärin für Chef 2 an	X	-	-	-

Function ID constants are prefixed with SIEMENSBUTTON FUNCTION (SBF)	Code	Default Display Name	CBX's: 1-1 Hicom 100E - opti_150.tsp - 1-2 Hicom 150 E - opti_150.tsp - 2 Hicom 300/HiPath 4000 V1.0 - opti_300.tsp - 4 HiPath 4000 V2.0 - optiH4V2.tsp -			
			CBX 4	CBX 2	CBX 1-2	CBX 1-1
SBF_SECRINTERCEPT1OFF	0x80000025	Secretarial Interception for Executive 1 off Abwurf auf Sekretärin für Chef 1 aus	X	-	-	-
SBF_SECRINTERCEPT1ON	0x80000026	Secretarial Interception for Executive 1 on Abwurf auf Sekretärin für Chef 1 an	X	-	-	-
SBF_CLIROFF	0x80000027	Calling Line Identification Restriction Anzeige der Rufnummer des Anrufers nicht unterdrückt (=CLIP Calling Line Identification Presentation	X	-	-	-
SBF_CLIRON	0x80000028	Calling Line Identification Restriction Anzeige der Rufnummer des Anrufers unterdrückt	X	-	-	-
SBF_MAINMENUOFF	0x80000029	Deactivate Main Menu Haupt-Menü deaktivieren	X	-	-	-
SBF_MAINMENUON	0x8000002A	Activate Main Menu Haupt-Menü aktivieren	X	-	-	-
SBF_ACDNIGHTSRVOFF	0x8000002B	ACD night service off ACD Nachtschaltung aus	-	-	X	-
SBF_ACDNIGHTSRVON	0x8000002C	ACD night service on ACD Nachtschaltung an	-	-	X	-
SBF_AGTMODERDY	0x8000002D	Agent ready	-	-	X	-
SBF_AGTLOGOFF	0x8000002E	Agent log off -	X	-	X	-
SBF_MESSAGEANSWEROFF	0x8000002F	Message Answer Text off Antworttext aus	-	-	X	-
SBF_MESSAGEANSWERON	0x80000030	Message Answer Text on Antworttext an	-	-	X	-
SBF_PRIVACYOFF	0x80000031	Privacy off Ruhe aus	-	-	X	-
SBF_PRIVACYON	0x80000032	Privacy on Ruhe ein	-	-	X	-
SBF_CLIPOFF	0x80000033	Calling line identification presentation off (CLIR) Rufnummerunterdrückung an	-	-	X	-

Function ID constants are prefixed with SIEMENSBUTTON FUNCTION (SBF)	Code	Default Display Name	CBX's: 1-1 Hicom 100E - opti_150.tsp - 1-2 Hicom 150 E - opti_150.tsp - 2 Hicom 300/HiPath 4000 V1.0 - opti_300.tsp - 4 HiPath 4000 V2.0 - optiH4V2.tsp -			
			CBX 4	CBX 2	CBX 1-2	CBX 1-1
SBF_CLIPON	0x80000034	Calling line identification presentation on Rufnummerunterdrückung aus	X	-	X	-
SBF_STARTHUNT	0x80000035	Start Hunt in Sammelanschluß einschalten	X	-	X	-
SBF_CODELOCKOFF	0x80000036	Codelock off Telefonschloß aus	-	-	X	-
SBF_CODELOCKON	0x80000037	Codelock on Telefonschloß ein	-	-	X	-
SBF_CALLBACKGET	0x80000038	Get Callbacks Rückrufe abfragen	X	-	X	-
SBF_CALLBACKSET	0x80000039	Set Callback Rückruf einleiten	X	-	X	-
SBF_NIGHTSRVOFF	0x8000003A	Night service off Nachtschaltung aus	X	-	X	-
SBF_NIGHTSRVON	0x8000003B	Night service on Nachtschaltung an	X	-	X	-
SBF_ACCOUNTCODE	0x8000003C	Account Code -	X	-	-	-
SBF_UNPARKGROUP	0x8000003D	Group Unpark -	X	-	-	-
SBF_PROGRAMSERVICEMODEOFF	0x8000003E	cancel Program-/Service mode Programm-/Service-Menü verlassen	X	-	X	-
SBF_PROGRAMSERVICEMODEON	0x8000003F	Activate Program-/Service mode Programm-/Service-Menü aktivieren	X	-	X	-
SBF_SYSUNPARK	0x80000040	System Unpark -	X	-	-	-
SBF_SYSPARK	0x80000041	System Park -	X	-	-	-
SBF_CONFREMP8	0x80000042	Remove party 8 from conference trenne TLN 8 aus Konferenz	X	-	-	-
SBF_CONFREMP7	0x80000043	Remove party 7 from conference trenne TLN 7 aus Konferenz	X	-	-	-

Function ID constants are prefixed with SIEMENSBUTTON FUNCTION (SBF)	Code	Default Display Name	CBX's: 1-1 Hicom 100E - opti_150.tsp - 1-2 Hicom 150 E - opti_150.tsp - 2 Hicom 300/HiPath 4000 V1.0 - opti_300.tsp - 4 HiPath 4000 V2.0 - optiH4V2.tsp -			
			CBX 4	CBX 2	CBX 1-2	CBX 1-1
SBF_CONFREMP6	0x80000044	Remove party 6 from conference trenne TLN 6 aus Konferenz	X	-	-	-
SBF_CONFREMP5	0x80000045	Remove party 5 from conference trenne TLN 5 aus Konferenz	X	-	-	-
SBF_CONFREMP4	0x80000046	Remove party 4 from conference trenne TLN 4 aus Konferenz	X	-	-	-
SBF_CONFREMP3	0x80000047	Remove party 3 from conference trenne TLN 3 aus Konferenz	X	-	-	-
SBF_CONFREMP2	0x80000048	Remove party 2 from conference trenne TLN 2 aus Konferenz	X	-	-	-
SBF_CONFREMP1	0x80000049	Remove party 1 from conference trenne TLN 1 aus Konferenz	X	-	-	-
SBF_CONSULTATION	0x8000004A	Consultation Rückfrage einleiten	X	-	X	-
SBF_HEADSETOFF	0x8000004B	Headset onhook Headset-Taste trennen	X	-	-	-
SBF_HEADSETON	0x8000004C	Headset offhook Headset-Taste Ruf annehmen	X	-	-	-
SBF_CALLWAITINGDEACT	0x8000004D	Enable call waiting Anklopfschutz aus	X	-	-	-
SBF_DNDOFF	0x8000004E	Do Not Disturb off Anrufschutz aus	X	-	X	-
SBF_DNDON	0x8000004F	Do Not Disturb on Anrufschutz an	X	-	X	-
SBF_MUTEOFF	0x80000050	Mute off Microphone an	X	-	X	-
SBF_MUTEON	0x80000051	Mute on Microphone aus	X	-	X	-
SBF_CDRACCOUNTCODE	0x80000052	CDR Account code	X	-	-	-
SBF_RINGERCUTOFF	0x80000053	Ringer cut off -	-	-	-	-
SBF_PRIVATE	0x80000054	Privat -	-	-	-	-

Function ID constants are prefixed with SIEMENSBUTTON FUNCTION (SBF)	Code	Default Display Name	CBX's: 1-1 Hicom 100E - opti_150.tsp - 1-2 Hicom 150 E - opti_150.tsp - 2 Hicom 300/HiPath 4000 V1.0 - opti_300.tsp - 4 HiPath 4000 V2.0 - optiH4V2.tsp -			
			CBX 4	CBX 2	CBX 1-2	CBX 1-1
SBF_ACDNIGHTANSWER	0x80000055	ACD night answer -	-	-	-	-
SBF_GENCALLKEY	0x80000056	Generate call key -	-	-	-	-
SBF_SELECTLANGUAGE	0x80000057	Select language -	-	-	-	-
SBF_REMOTEFLASH	0x80000058	Remote flash -	-	-	-	-
SBF_DIALPADKEYA	0x80000059	Dialpad key A Taste A	-	-	-	-
SBF_DIALPADKEYB	0x80000060	Dialpad key B Taste B	-	-	-	-
SBF_DIALPADKEYC	0x80000061	Dialpad key C Taste C	-	-	-	-
SBF_DIALPADKEYD	0x80000062	Dialpad key D Taste D	-	-	-	-
SBF_ABSENT	0x80000063	Absent Abwesenheit	-	-	-	-
SBF_HELP	0x80000064	Help Hinweis	-	-	-	-
SBF_RECORDCHARGES	0x80000065	Record Charges Vermitteln mit Gebühren	-	-	X	-
SBF_MAIL	0x80000066	Mail Mail	-	-	-	-
SBF_STORETEXT	0x80000067	Store Text Textspeicher	-	-	-	-
SBF_CALLINGNAMEID	0x80000068	Calling Party Name Identification Wer	-	-	-	-
SBF_ATNDTRANSFER	0x80000069	Attendant terminal functional transfer AFT Übergeben	-	-	-	-
SBF_ATNDNOTAVAIL	0x8000006A	Attendant not available AFT Abwesenheit	-	-	-	-
SBF_ALTCONTACT	0x8000006B	Alternate Contact Vertreter	-	-	-	-

Function ID constants are prefixed with SIEMENSBUTTON FUNCTION (SBF)	Code	Default Display Name	CBX's: 1-1 Hicom 100E - opti_150.tsp - 1-2 Hicom 150 E - opti_150.tsp - 2 Hicom 300/HiPath 4000 V1.0 - opti_300.tsp - 4 HiPath 4000 V2.0 - optiH4V2.tsp -			
			CBX 4	CBX 2	CBX 1-2	CBX 1-1
SBF_BUSYINDICATOR	0x8000006C	Busy Indicator Besetztlampenfeld	-	-	-	-
SBF_ATNDQUEUE	0x8000006D	Attendant Console Queue Queue (Vermittlungsplatz)	-	-	-	-
SBF_PLUSMINUS	0x8000006E	- Plus-Minus	-	-	-	-
SBF_ACDAVAILTOGGLE	0x8000006F	ACD Agent available/unavailable ACD Agent verfügbar/nicht verfügbar	-	-	X	-
SBF_MAILBOXMENU	0x80000070	Mailbox menu Briefkastenmenü	-	-	-	-
SBF_VARFWDPOSTDIAL	0x80000071	Variable forwarding with post-dialing Variable Umleitung mit Nachwahl	-	-	X	-
SBF_SYSTERMMODE	0x80000072	System terminal mode Systemterminalmode	-	-	-	-
SBF_FWDMENU	0x80000073	Call forwarding menu Umleitungsmenü	-	-	X	-
SBF_CAFLOGOFF	0x80000074	CAF - Logoff -	-	-	-	-
SBF_CAFGPLOGOFF	0x80000075	CAF - Group Logoff -	-	-	-	-
SBF_CAFAVAIL	0x80000076	CAF - Available -	-	-	-	-
SBF_CAFGPAVAIL	0x80000077	CAF - Group Available -	-	-	-	-
SBF_CAFUNAVAIL	0x80000078	CAF - Unavailable -	-	-	-	-
SBF_CAFGPUNAVAIL	0x80000079	CAF - Group Unavailable -	-	-	-	-
SBF_CAFFORWARD	0x8000007A	CAF - Forward -	-	-	-	-
SBF_CAFFORWARDCAN	0x8000007B	CAF - Forward cancel -	-	-	-	-
SBF_T_VARFWDALL	0x8000007C	Forwarding-variable-all (toggles) Anrufumleitung variabel	-	-	X	X

Function ID constants are prefixed with SIEMENSBUTTON FUNCTION (SBF)	Code	Default Display Name	CBX's: 1-1 Hicom 100E - opti_150.tsp - 1-2 Hicom 150 E - opti_150.tsp - 2 Hicom 300/HiPath 4000 V1.0 - opti_300.tsp - 4 HiPath 4000 V2.0 - optiH4V2.tsp -			
			CBX 4	CBX 2	CBX 1-2	CBX 1-1
SBF_T_VARFWDBUSY	0x8000007D	Forwarding-variable-busy (toggles) Anrufumleitung variabel besetzt	-	-	-	-
SBF_T_VARFWDNA	0x8000007E	Forwarding-var-noanswer (toggles) Anrufumleitung variabel frei	-	-	-	-
SBF_T_FIXFWDALL	0x8000007F	Forwarding-fixed-all (toggles) Anrufumleitung fest	-	X	X	-
SBF_T_FIXFWDBUSY	0x80000080	Forwarding-fixed-busy (toggles) Anrufumleitung fest besetzt	-	-	-	-
SBF_T_FIXFWDNA	0x80000081	Forward-fixed-noanswer (toggles) Anrufumleitung fest frei	-	-	-	-
SBF_CODECALL	0x80000082	Code Call Personensucheinrichtung	-	X	X	-
SBF_MSGRSUMMON	0x80000083	Messenger Summoning Botenruf	-	X	-	-
SBF_CALLRCDSILENT	0x80000084	Call Record Silent Zeugenzuschaltung	-	X	X [1]	X
SBF_STOPHUNT	0x80000085	Stop Hunt Sammelanschluß ein/aus	X	X	X	X
SBF_XFERPUSHPULL	0x80000086	Transfer by Push / Pull Übergeben / Übernehmen	X	X	-	-
SBF_PARKGROUP	0x80000087	Call Park - Group Parken (Gruppe)	X	X	-	-
SBF_VOICECALL	0x80000088	Voice Call Direktansprechen	-	X	X	X
SBF_SECRXFER	0x80000089	Secretarial Transfer Vertretung für Sekretariat	-	X	-	-
SBF_SECRINTERCEPT	0x8000008A	Secretarial Intercept Rufumschaltung für Sekretariat	-	X	-	-
SBF_SECRPICKUP	0x8000008B	Secretarial Pickup Anrufübernahme Sekretariat	-	X	-	-
SBF_TIMEDREMINDER	0x8000008C	Timed Reminder Termin	-	X	X [1]	-
SBF_FIXEDCONFGROUP	0x8000008D	Fixed conference groups 1-3 Feste Konferenz 1-3	-	-	-	-

Function ID constants are prefixed with SIEMENSBUTTON FUNCTION (SBF)	Code	Default Display Name	CBX's: 1-1 Hicom 100E - opti_150.tsp - 1-2 Hicom 150 E - opti_150.tsp - 2 Hicom 300/HiPath 4000 V1.0 - opti_300.tsp - 4 HiPath 4000 V2.0 - optiH4V2.tsp -			
			CBX 4	CBX 2	CBX 1-2	CBX 1-1
SBF_INTERCOMGROUP	0x8000008E	Intercom groups 1-8 Lautsprecherdurchsagen 1-8	-	-	-	-
SBF_SPEEDSYSGROUP	0x8000008F	Speed dial system groups 1-5 Rufnummerngeber zentral 1-5	-	-	-	-
SBF_TAFAS	0x80000090	Trunk answer from any station 1-4 Allgemeine Abfrage	-	-	-	-
SBF_PICKUPGROUP	0x80000091	Pickup groups 1-5 Anrufübernahme Gruppe 1-5	-	-	-	-
SBF_PARKINTERCOM	0x80000092	Park with intercom 1-8 Parken mit Durchsage 1-8	X	-	-	-
SBF_RELAYCONTROL	0x80000093	Relay control 1-22 Relaisteuerung, Schalter 1-22	-	-	X	X
SBF_PICKUPINTERCOM	0x80000094	Pickup after intercom announcement Übernehmen Gespräch nach Durchsage	-	-	-	-
SBF_ACDDONOTDISTURB	0x80000095	ACD Do not disturb ACD Ruhe	-	-	X	-
SBF_RELEASETRUNK	0x80000096	Release trunk Amtleitung auslösen	-	-	X	-
SBF_REJECTCAMPON	0x80000097	Reject camped on call Anklopfen Abwurf	-	-	-	-
SBF_PICKUPCAMPON	0x80000098	Pickup camped on call Anklopfer annehmen	X	-	X	X
SBF_CALLWAITINGACT	0x80000099	Enable call waiting Anklopfschutz aus	X	-	-	-
SBF_CAMPONMANUAL	0x8000009A	Manual camp-on Anklopfen manuell	-	-	-	-
SBF_DIRECTEDPICKUP	0x8000009B	Directed Pickup Anrufübernahme gezielt	-	-	X	X
SBF_CODELOCK	0x8000009C	Private change of COS Berchtigungsumschalt. individuell/Codeschloß	-	-	X	X
SBF_GETERRORS	0x8000009D	Get error message Fehlerabfrage	-	-	-	-
SBF_GETCHARGES	0x8000009E	Get charge info Gebührenabfrage	-	-	X	X

Function ID constants are prefixed with SIEMENSBUTTON FUNCTION (SBF)	Code	Default Display Name	CBX's: 1-1 Hicom 100E - opti_150.tsp - 1-2 Hicom 150 E - opti_150.tsp - 2 Hicom 300/HiPath 4000 V1.0 - opti_300.tsp - 4 HiPath 4000 V2.0 - optiH4V2.tsp -			
			CBX 4	CBX 2	CBX 1-2	CBX 1-1
SBF_CLIR	0x8000009F	Calling Line ID Restriction Rufnummerunterdrückung	-	-	X	X
SBF_MONITORRLINE	0x800000A0	Monitor calls on other line Ruf/Leitung ein/aus, Rufzuschaltung	-	-	X	X
SBF_SERVICE_MENU	0x800000A1	Service Menu Key Service Menü	-	X	X	X
SBF_MENU_SELECT	0x800000A2	Menu selection Menü auswählen (OK Taste)	X	X	X	X
SBF_MENU_FWD	0x800000A3	Menu >> Menü vorwärts	X	X	X	X
SBF_MENU_BWD	0x800000A4	Menu << Menü rückwärts	X	X	X	X
SBF_MALICIOUSCALLID	0x800000A5	Malicious call ID Fangen	-	-	X	X
SBF_LASTCALLS	0x800000A6	List of unsuccessful calls Anruferliste	-	-	X [1]	X
SBF_LEAVEMESSAGE	0x800000A7	Leave message on display Antworttext	-	-	X [1]	X
SBF_ FIXFWDTRUNK	0x800000A8	Forward to outside line Umleitung im Amt	-	-	X [1]	X
SBF_DIRECTORY	0x800000A9	Directory (stored in CBX) Telefonbuch	-	-	X [1]	X
SBF_DUPLEXVOICECALLPROTECTION	0x800000AA	Duplex Voice Calling prot. on/off Direktantwort ein/aus	-	-	-	-
SBF_RETRIEVECALL	0x800000AB	Retrieve call (from Phonemail) Gespräch abnehmen	-	-	-	-
SBF_DOOR	0x800000AC	Door box Türöffner	-	X	X	X
SBF_DTMF	0x800000AD	Send DTMF digits MFV-Wahl	-	-	X [1]	X
SBF_MONITORROOM	0x800000AE	Monitor room Babyphone	-	-	-	-
SBF_NET_A	0x800000AF	Net A (LCR UK) Netz A	-	-	-	-

Function ID constants are prefixed with SIEMENSBUTTON FUNCTION (SBF)	Code	Default Display Name	CBX's: 1-1 Hicom 100E - opti_150.tsp - 1-2 Hicom 150 E - opti_150.tsp - 2 Hicom 300/HiPath 4000 V1.0 - opti_300.tsp - 4 HiPath 4000 V2.0 - optiH4V2.tsp -			
			CBX 4	CBX 2	CBX 1-2	CBX 1-1
SBF_NET_B	0x800000B0	Net B (LCR UK) Netz B	-	-	-	-
SBF_ASSOCIATEDDIALING	0x800000B1	Associated dialing Assoziierte Wahl	-	-	-	-
SBF_ASSOCIATEDSERVICE	0x800000B2	Associated service Assoziierter Dienst	-	-	X [1]	X
SBF_SHOWWAITINGCALLS	0x800000B3	Show number of waiting calls Zahl der Anrufe anzeigen	-	-	X [1]	X
reserved	0x800000B4		-	-	-	-
reserved	0x800000B5		-	-	-	-
reserved	0x800000B6		-	-	-	-
reserved	0x800000B7		-	-	-	-
SBF_SPKRONEBCAST	0x800000B8	Speaker One Way Broadcast -	-	-	-	-
SBF_ADL1	0x800000B9	Application Definable LED 1 -	-	-	-	-
SBF_ADL2	0x800000BA	Application Definable LED 2 -	-	-	-	-
SBF_ADL3	0x800000BB	Application Definable LED 3 -	-	-	-	-
SBF_ADL4	0x800000BC	Application Definable LED 4 -	-	-	-	-
SBF_AUTOICOM	0x800000BD	Auto Intercom -	-	-	-	-
SBF_TMMS	0x800000BE	TMMS key -	-	-	-	-
SBF_CAMPMSGS	0x800000BF	Camp Messages -	-	-	-	-
SBF_GROUPPICK	0x800000C0	Group Pickup Anrufübernahme (Gruppe)	X	X	X	X
SBF_MSGWAITCTRL	0x800000C1	Send Message Info senden	-	-	X	X

Function ID constants are prefixed with SIEMENSBUTTON FUNCTION (SBF)	Code	Default Display Name	CBX's: 1-1 Hicom 100E - opti_150.tsp - 1-2 Hicom 150 E - opti_150.tsp - 2 Hicom 300/HiPath 4000 V1.0 - opti_300.tsp - 4 HiPath 4000 V2.0 - optiH4V2.tsp -			
			CBX 4	CBX 2	CBX 1-2	CBX 1-1
SBF_MSGWAIT	0x800000C2	Message Waiting Meldung empfangen	X	-	X	X
SBF_SPKRONEWAY	0x800000C3	Speaker One Way (Dial Call) -	-	-	-	-
SBF_ADK	0x800000C4	Application Defined Key -	-	-	-	-
SBF_DSSKEY	0x800000C5	Direct Station Select Key Direktruftaste	-	X	-	-
SBF_AGTMODEAVL	0x800000C6	ACD Agent mode available ACD Agent verfügbar	X	X	X	-
SBF_AGTMODEUNAV	0x800000C7	ACD Agent mode unavailable ACD Agent nicht verfügbar	X	X	X	-
SBF_AGTMODEWRK	0x800000C8	ACD Agent mode work ACD Agent arbeitet	X	X	X [1]	-
SBF_MANANS	0x800000C9	Automatic / manual answer Automatische Gesprächsannahme	-	X	-	-
SBF_ACDAGTMSG	0x800000CA	ACD Agent message ACD Nachricht zum Agent senden	-	X	-	-
SBF_ACDSUPMSG	0x800000CB	ACD Supervisor message ACD Nachricht zum Supervisor senden	-	X	-	-
SBF_ACDEMERMSG	0x800000CC	ACD Emergency message ACD Notfallnachricht senden	-	X	-	-
SBF_ACDMONSLNT	0x800000CD	ACD Silent monitor ACD Mithoeren ohne Ton	-	X	X	-
SBF_ACDMONTONE	0x800000CE	ACD Monitor with tone ACD Mithoeren mit Ton	-	X	-	-
SBF_ACDPRIGP	0x800000CF	ACD Display primary group status ACD Primärer Gruppenstatus	-	X	-	-
SBF_ACDSECGP	0x800000D0	ACD Display secondary group status ACD Sekundärer Gruppenstatus	-	X	-	-

Function ID constants are prefixed with SIEMENSBUTTON FUNCTION (SBF)	Code	Default Display Name	CBX's: 1-1 Hicom 100E - opti_150.tsp - 1-2 Hicom 150 E - opti_150.tsp - 2 Hicom 300/HiPath 4000 V1.0 - opti_300.tsp - 4 HiPath 4000 V2.0 - optiH4V2.tsp -			
			CBX 4	CBX 2	CBX 1-2	CBX 1-1
SBF_ACDPRIQUE	0x800000D1	ACD Display primary queue status ACD Primärer Queuestatus	-	X	X	-
SBF_ACDSECQUE	0x800000D2	ACD Display secondary queue stat ACD Sekundärer Queuestatus	-	X	-	-
SBF_PROGRAM	0x800000D3	Program key Check/Program/Speicher	-	-	X	-
SBF_MAILBOX	0x800000D4	Mailbox key Briefkasten	-	X	-	-
SBF_HEADSET	0x800000D5	Headset key Sprechgarnitur	-	X	-	-
SBF_CALLBACKKEY	0x800000D6	Callback key Rückruf	X	X	X	X
SBF_PHONEMAIL	0x800000D7	PhoneMail key -	X	-	-	-
SBF_PRIVACY	0x800000D8	Privacy key Ruhe (kein Ton/Display)	-	X	X [1]	X
SBF_TOGGLE	0x800000D9	Toggle key Makeln	X	X	X	X
SBF_DATAKEY	0x800000DA	Data key Datentaste/Nonvoice	-	X	-	-
SBF_DDSLOCAL	0x800000DB	Local repdial -	-	-	-	-
SBF_ACCTNUM	0x800000DC	Account code Gespräch markieren	-	X	X	X
SBF_AGTLOGON	0x800000DD	ACD Agent logon/logoff ACD Agent logon/logoff	X	X	X [1]	-
SBF_ACDMSGACK	0x800000DE	ACD Message acknowledgement ACD Nachricht quittieren	-	X	-	-
SBF_ACDMSGSCROLL	0x800000DF	ACD Message scroll ACD Nachricht blättern	-	X	-	-
SBF_BUZZPRESET	0x800000E0	Buzz preset destination -	-	-	-	-

Function ID constants are prefixed with SIEMENSBUTTON FUNCTION (SBF)	Code	Default Display Name	CBX's: 1-1 Hicom 100E - opti_150.tsp - 1-2 Hicom 150 E - opti_150.tsp - 2 Hicom 300/HiPath 4000 V1.0 - opti_300.tsp - 4 HiPath 4000 V2.0 - optiH4V2.tsp -			
			CBX 4	CBX 2	CBX 1-2	CBX 1-1
SBF_COSCHANGE	0x800000E1	Class of service changeover Berechtigungsumschaltung	-	-	-	-
SBF_VARFWDALLINT	0x800000E2	Forwarding-variable-all internal-activate Anrufumleitung variabel intern ein	X	-	X	-
SBF_VARFWDALLEXT	0x800000E3	Forwarding-variable-all external-activate Anrufumleitung variabel extern ein	X	-	X	-
SBF_VARFWDBOTH	0x800000E4	Forwarding-var.-all both-activate Anrufumleitung variabel intern/extern ein	X	-	X	-
SBF_VARFWDBUSY	0x800000E5	Forwarding-variable-busy-activate Anrufumleitung variabel besetzt ein	X	-	-	-
SBF_VARFWDNA	0x800000E6	Forwarding-var.-no answer-activate Anrufumleitung variabel frei ein	X	-	-	-
SBF_VARFWDBUSYNA	0x800000E7	Forwarding-variable-busy & no answer-activate Anrufumleitung variabel frei/besetzt ein	X	-	-	-
SBF_VARFWDCANCEL	0x800000E8	Forwarding-variable-cancel Anrufumleitung variabel aus	X	-	X	-
SBF_FIXFWDACT	0x800000E9	Forwarding-fixed-all-activate Anrufumleitung fest ein	X	-	-	-
SBF_FIXFWDCAN	0x800000EA	Forwarding-fixed-all-cancel Anrufumleitung fest aus	X	-	X	-
SBF_PVTHOLD	0x800000EB	Hold private (exclusive hold) Halten	-	X	X [1]	-
SBF_STORE	0x800000EC	Store key Eingeben	-	X	-	-
SBF_BADTRUNK	0x800000ED	Report bad trunk -	-	-	-	-
SBF_SPKRFIXONE	0x800000EE	Speaker call – fixed one way (Voice Call) -	-	-	-	-
SBF_SPKRTWOWAY	0x800000EF	Speaker call - two way -	-	-	-	-
SBF_SPKRREJECT	0x800000F0	Speaker call – rejection Direktansprechschutz	-	X	-	X

Function ID constants are prefixed with SIEMENSBUTTON FUNCTION (SBF)	Code	Default Display Name	CBX's: 1-1 Hicom 100E - opti_150.tsp - 1-2 Hicom 150 E - opti_150.tsp - 2 Hicom 300/HiPath 4000 V1.0 - opti_300.tsp - 4 HiPath 4000 V2.0 - optiH4V2.tsp -			
			CBX 4	CBX 2	CBX 1-2	CBX 1-1
SBF_START	0x800000F1	Start key Ausgeben	-	X	-	-
unused	0x800000F2		-	-	-	-
unused	0x800000F3		-	-	-	-
unused	0x800000F4		-	-	-	-
SBF_SYSSPEEDGP2	0x800000F5	System speed dialing - group 2 -	-	-	-	-
SBF_LINEKEY	0x800000F6	Line key Leitungstaste	-	X	-	-
SBF_OVERRIDE	0x800000F7	Busy override Aufschalten	-	-	X	X
SBF_CLEAR	0x800000F8	Clear key Löschen	X	X	-	-
SBF_CANCEL	0x800000F9	Cancel request Trennen	X	X	X	X
SBF_DTS	0x800000FA	Direct trunk select Gesprächstaste (Bündeltaste)	-	-	-	-
SBF_INUSEKEY	0x800000FB	In use key I-use	-	X	-	-
SBF_REPEATID	0x800000FC	Repeat source ID -	-	-	-	-
SBF_ROLMPARK	0x800000FD	Park – individual -	-	-	-	-
SBF_CONFREMLAST	0x800000FE	Remove last party from conference -	X	-	-	-
SBF_CALLWTG	0x800000FF	Call waiting LED Briefkastenlampe	-	-	X	-

[1] Available only on Hicom 150 E Office V 2.2 and higher.

10 Device-Specific Data Structures and Constants

The following describes the device-specific data structures and constants defined by the CallBridge Collection telephony service providers.

Note: all device-specific data structure definitions and constants are provided in the file Siemens.h on the tool kit disk.

10.1 SIEMENS_LINE_DEV_SPECIFIC

The **SIEMENS_LINE_DEV_SPECIFIC** structure is passed to the **lineDevSpecific** function via parameter **lpParams**. It identifies the extended feature that is being requested and any other information that is required for that extended feature.

```
typedef struct _SIEMENS_LINE_DEV_SPECIFIC
{
        DWORD dwRequestType;
        union
        {
                //dwRequestType == SIEMENSLINE_DIAL_DIGITS
                BYTE szDialableString [SIEMENS_MAX_DIALABLE_STRING_BYTES + 1];

                //dwRequestType == SIEMENSLINE_GET_FEATURELIST
                struct
                {
                        BOOL bFeatureListAvailable;
                        char szStaticFeatureListGlobal[SIEMENS_MAX_FEATURES/8];
                        char szStaticFeatureListSpecific[SIEMENS_MAX_FEATURES/8];
                };

              //dwRequestType == SIEMENSLINE_SET_HLB_PROGRAMKEY
              struct
              {
                      INT  iKeyIndex;
                      char szIntNumber[SIEMENS_MAX_RUFNR_LEN];
              };

              //dwRequestType == SIEMENSLINE_SET_OPTISET_TYPE
              DWORD dwOptisetType;

              //dwRequestType == SIEMENSLINE_REQ_FEATURE_STATES
```

```
        BOOL bSwitch;

        //dwRequestType == SIEMENSLINE_PARK_DIRECT
        BYTE szParkSTNNo [SIEMENS_MAX_DIALABLE_STRING_BYTES + 1];

        //dwRequestType == SIEMENSLINE_GET_PUSKEYINFO
        struct
        {
            INT     iPUSKeyNo;
            BYTE    bDevice;
            BYTE    bKeyNo;
            char    szSTNNo[7];
            char    szSTNName[50];
            BYTE    bState;
        };

        //dwRequestType == SIEMENSLINE_PRESS_PUSKEY
        INT     iPUSKeyNoToPress;

        //dwRequestType == SIEMENSLINE_SET_HEADSET
        BOOL bHSavail;

        //dwRequestType == SIEMENSLINE_GET_SYSPARKSLOTS
        BOOL bPlaceholder;
    };
}
SIEMENS_LINE_DEV_SPECIFIC, FAR *LPSIEMENS_LINE_DEV_SPECIFIC;
```

10.1.1 Fields

The **SIEMENS_LINE_DEV_SPECIFIC** structure contains the following fields:

dwRequestType

This field identifies the extended line device feature that is being requested. The values supported for this field are:

SIEMENSLINE_DIAL_DIGITS 0x00000001

tells lineDevSpecific() to send the digits in szDialableString[] to the phone.

szDialableString []

a Null-terminated ASCII string of dialable digits in the range of (0..9, *, #), which should be sent to the phone.

SIEMENSLINE_GET_FEATURELIST 0x00000002

returns the availability of the provided features

bFeatureListAvailable

an indicator, if the initial feature state transmission was done and if the following strings are filled or not

szStaticFeatureListGlobal[]
provides the feature availability of global features

szStaticFeatureListSpecific[]
provides the feature availability of system specific features

SIEMENSLINE_SET_HLB_PROGRAMKEY 0x00000003
programs a virtual station keys on the HG1500 (HiPath 3000 specific)

iKeyIndex
index of the programable virtual key

szIntNumber[]
internal station (destination) number

SIEMENSLINE_SET_OPTISET_TYPE 0x00000004
changes the configured device type from the applications side

dwOptisetType
device type

SIEMENSLINE_REQ_FEATURESTATES 0x00000005
requests DEV_SPECIFIC events for every change on a feature state

bSwitch
switch, if functionality should be ON or OFF

SIEMENSLINE_PARK_DIRECT 0x00000006
parks a call to a specific station (HiPath 4000 V2.0 specific)

szParkSTNNo[]
a Null-terminated ASCII string of dialable digits in the range of (0..9, *, #), which should be sent to the phone as park destination no.

SIEMENSLINE_GET_PUSKEYINFO 0x00000007
gets infos about a PUS key (HiPath 4000 V2.0 specific)

iPUSKeyNo
index of the available PUS keys

bDevice
device identifier regarding the device on which the PUS key is configured

bKeyNo
key identifier regarding the physical key on which the PUS key is configured

szSTNNo[]
a Null-terminated ASCII string of the PUS key related station no.

szSTNName[]
a Null-terminated ASCII string of the PUS key related station name

bState
identifier regarding the status in which the PUS key is actually in

SIEMENSLINE_PRESS_PUSKEY 0x00000008
presses a specific PUS key (HiPath 4000 V2.0 specific)

iPUSKeyNoToPress
index of the available PUS key which is to pressed

SIEMENSLINE_SET_HEADSET 0x00000009
sets/unsets headset support

bHSavail
switch, headset support should be ON (TRUE) or OFF (FALSE)

SIEMENSLINE_GET_SYSPARKSLOTS 0x0000000A
get the occupancy of all system park slots

bPlaceholder
placeholder, should be ON (TRUE) or OFF (FALSE) w/o impact

10.2 SIEMENS_LINEDEVCAPS_DEV_SPECIFIC

The **SIEMENS_LINEDEVCAPS_DEV_SPECIFIC** data structure provides device-specific information for the **LINEDEVCAPS** data structure. It is appended to the **LINEDEVCAPS** structure when a **dwExtVersion** parameter has been specified in **lineGetDevCaps**.

```
typedef struct siemens_linedevcaps_dev_specific_tag {
    DWORD dwDevSpecificRequestsAvailable;
    DWORD dwInvokableButtonsBitmask [SIEMENS_NUM_BUTTONFCN_BITMASKS];
} SIEMENS_LINEDEVCAPS_DEV_SPECIFIC;
```

10.2.1 Fields

The **SIEMENS_LINEDEVCAPS_DEV_SPECIFIC** structure contains the following fields:

dwDevSpecificRequestsAvailable
A bitmask of available extended features that may be invoked via the **lineDevSpecific** function. The bit constant is passed to **lineDevSpecific** in the **dwRequestType** field of the **SIEMENS_LINE_DEV_SPECIFIC** structure. The possible bits that may be set are:

SIEMENSLINE_DIAL_DIGITS
Send a dialable string to the phone.

dwInvokableButtonsBitmask[ ]

Bitmasks that indicate the PHONEBUTTONFUNCTIONs that may be invoked via the **lineDevSpecificFeature** function.

Note: this is the **static availability** of the feature as opposed to the dynamic availability of the feature based on the current state of the device. The dynamic feature availability is found in the device-specific portion of the **LINEDEVSTATUS** structure.

The PHONEBUTTONFUNCTION value is the zero-based bit position in the group of bitmasks, with the 0 bit being the low-order bit in the 0 array entry.

If a bit is set in the bitmask, the corresponding PHONEBUTTONFUNCTION can be invoked via **lineDevSpecificFeature**:

dwInvokableButtonsBitmask[0] reports buttons in the range 0-31
dwInvokableButtonsBitmask[1-6] reports buttons in the range 32-223
dwInvokableButtonsBitmask[7] reports buttons in the range 224-255

Note: for the Siemens extensions to the PHONEBUTTONFUNCTION constants, the high-order bit is masked off before using the constant as an index into the bitmasks.

10.2.2 Comments

The device-specific field of the **LINEDEVCAPS** structure should be used to determine the availability of a feature via **lineDevSpecificFeature**. In principle, all features listed in chapter 8 are invokable if configured for the associated phone.

10.3 SIEMENS_LINEDEVSTATUS_DEV_SPECIFIC

The **SIEMENS_LINEDEVSTATUS_DEV_SPECIFIC** structure provides device-specific information for the **LINEDEVSTATUS** structure. It is appended to the **LINEDEVSTATUS** structure when the device-specific data is available and an extension version has been selected.

```
typedef struct siemens_linedevstatus_dev_specific_tag {
    DWORD dwInvokableButtonsBitmasks [SIEMENS_NUM_BUTTONFCN_BITMASKS];
} SIEMENS_LINEDEVSTATUS_DEV_SPECIFIC;
```

10.3.1 Fields

The **SIEMENS_LINEDEVSTATUS_DEV_SPECIFIC** structure contains these fields:

dwInvokableButtonsBitmask[ ]
Bitmasks that indicate the **dynamic availability** of PHONEBUTTONFUNCTIONs, i.e. features that may be invoked via the **lineDevSpecificFeature** function based on the current state of the device. The PHONEBUTTONFUNCTION value is the zero-based bit position in the group of bitmasks, with the 0 bit being the low-order bit in the 0 array entry.

dwInvokableButtonsBitmask[0] reports buttons in the range 0-31
dwInvokableButtonsBitmask[1-6] reports buttons in the range 32-223
dwInvokableButtonsBitmask[7] reports buttons in the range 224-255

If a bit is set in the bitmask, the corresponding PHONEBUTTONFUNCTION can currently be invoked via **lineDevSpecificFeature**.

Note: for the Siemens extensions to the PHONEBUTTONFUNCTION constants, the high-order bit is masked off before using the constant as an index into the bitmasks.

10.4 SIEMENS_PHONESTATUS_DEV_SPECIFIC

The **SIEMENS_PHONESTATUS_DEV_SPECIFIC** structure provides device-specific information for the **PHONESTATUS** structure. It is appended to the **PHONESTATUS** structure when the device-specific data is available and an extension version has been selected.

```
typedef struct siemens_phonestatus_dev_specific_tag {
        DWORD dwProgramState;
        DWORD dwVirtualButtonSize;      // not used by CallBridge Collection
        DWORD dwVirtualButtonOffset;    // not used by CallBridge Collection
} SIEMENS_PHONESTATUS_DEV_SPECIFIC;
```

10.4.1 Fields

The **SIEMENS _PHONESTATUS_DEV_SPECIFIC** structure contains these fields:

dwProgramState

The program state of the phone. The values supported for this field are:

SIEMENSPROGRAMSTATE_DEACTIVATE 0x00000000
The user has not activated the *Service Menu*.

SIEMENSPROGRAMSTATE_SYSTEMHOST 0x00000002
The application or the user has pressed the *Service Menu* key.

dwVirtualButtonSize not used, set to zero
dwVirtualButtonOffset not used, set to zero

10.5 SIEMENSRINGMODE Constants

The following values are used to report the ring mode of a phone via **PHONE_STATE**/PHONESTATE_RINGMODE. The SIEMENSRINGMODE is reported in **dwParam2**.

SIEMENSRINGMODE_SILENCE	0	silence
SIEMENSRINGMODE_SINGLE_RING_INTERN	1	single ring (internal calls)
SIEMENSRINGMODE_SINGLE_RING	1	single ring
SIEMENSRINGMODE_DBL_RING_EXTERNAL	2	double ring (external calls)
SIEMENSRINGMODE_DBL_RING	2	double ring
SIEMENSRINGMODE_TPL_RING_RECALL	3	triple ring (recalls)
SIEMENSRINGMODE_TPL_RING	3	triple ring
SIEMENSRINGMODE_SINGLE_BUZZ_ALERT	4	single buzz (alerting calls)
SIEMENSRINGMODE_SINGLE_BUZZ	4	single buzz
SIEMENSRINGMODE_COM_RING	5	Com ring
SIEMENSRINGMODE_TPL_BUZZ_DATA_CALL	6	triple buzz (data calls)
SIEMENSRINGMODE_TPL_BUZZ	6	triple buzz

SIEMENSRINGMODE_CONT_RING_EMERGCY	7	Cont ring (emergency calls)
SIEMENSRINGMODE_CONT_RING	7	Cont ring
SIEMENSRINGMODE_DOUBLE_BUZZ_NOTIFY	8	Double buzz (notification calls)
SIEMENSRINGMODE_DOUBLE_BUZZ	9	double buzz
SIEMENSRINGMODE_RESERVED	10	reserved
SIEMENSRINGMODE_UNKNOWN	11	unknown

Note: the actual meaning of each ringer cadence (e.g. internal/external call) is configurable at the customer site and is therefore unknown to CallBridge Collection. *Rings* denote multiple cycle rings, *buzzes* are single cycle rings.

11 Appendix: Header File

This chapter shows the header file that is provided with the developer's toolkit.

Attention:
To protect your TAPI project from byte alignment and definition problems,
- please do not modify the following file,
- please do not copy only parts of it in your project, but include the whole file into your project
- please use every time the latest version which is downloadable from the internet

Siemens.h

The following shows the current version of the file *Siemens.h which* contains the SIEMENS/ROLM BUTTONFUNCTION constants, additional constants and the SIEMENS/ROLM_ device specific extensions. This header file is common to all Siemens TAPI service providers. Note that not all TSPs use all of the device specific structures. CallBridge Collection uses only the extensions described in this document.

```
//        MH*************************************************
//        Copyright (c) 2004 Siemens AG - Com ESY HD 631 Witten
//        Copyright (c) 2003 Siemens AG - ICN EN HS D 631 Witten
//        Copyright (c) 2002 Siemens AG - ICN EN HO SE 5 Witten
//        Copyright (c) 2001 Siemens AG - ICN EN HO SE 5 Witten
//        Copyright (c) 2000 Siemens AG - ICN EN HO SE 5 Witten
//        Copyright (c) 2000 Siemens AG - ICN WN BS SE 5 Witten
//        Copyright (c) 1999 Siemens AG - ICN WN BS SE 5 Witten
//
//        Telephony Service Provider for Windows NT
//
//        Module Name:        Siemens.h
//
//        Archive Name:
//
//        Description:        This is a include file for all Siemens 1stParty TSP's and TAPI applications
//
//        Author:             various
//
//        HOC*************************************************************************
//        Date                | Comments
                              | Name
//        ------------+-------------------------------------------------------+-----
//        12.09.99  | initial creation                                      | mar
//                  | #define SIEMENSLINE_DIAL_DIGITS added           | mar
//                  | #define SIEMENS_MAX_DIALABLE_STRING_BYTES added | mar
//                  | struct SIEMENS_LINE_DEV_SPECIFIC created                | mar
//                  | #pragma pack                                            | mar
//        13.09.99  | struct SIEMENS_LINEDEVCAPS_DEV_SPECIFIC created | mar
//        22.09.99  | Add new feature values (53-57)                        | M. Doelle
//        23.09.99  | #defines for ring modes added                         | mar
//        08.10.99  | struct SIEMENS_PHONESTATUS_DEV_SPECIFIC created | mar
//        02.11.99  | #defines for SIEMENSBUTTONFUNCTION_xxx added          | mar
//        04.11.99  | struct SIEMENS_LINEDEVSTATUS_DEV_SPECIFIC created     | mar
//        23.07.01  | Add new feature value (52)                            | B. Farkas
//        xx.yy.zz  | Add new DEV_SPECIFIC                                  | B. Gronostay
//        22.12.03  | Add new feature values x80000051 / x80000050          | P. Halacz
//        05.01.04  | Add new feature values x8000004F / x8000004E          | P. Halacz
//        26.01.04  | Add new feature values x8000004D                      | P. Halacz
//        28.01.04  | Add new feature values x8000004C ... x8000003C        | P. Halacz
//        19.02.04  | Include-Guards added                                  | Jaraczewski
//        20.02.04  | Add new feature values x8000003B ... x8000002B        | P. Halacz
//        16.04.04  | Add new feature values x8000002A ... x8000001A        | P. Halacz
//        28.04.04  | Add new DEV_SPECIFIC                                  | P. Halacz
//        01.09.04  | Add new DEV_SPECIFIC & new feature value x80000019    | P. Halacz
//        28.09.04  | Add new feature values x80000018 ... x80000017  | P. Halacz
//                  |                                                       |
```

```
//        ************************************************************************
#ifndef __SIEMENS_H__
#define __SIEMENS_H__

#pragma pack(push, include_siemens)                                  // store previos packing alingment
#pragma pack(1)                                                      // packing alingment: 1 byte

//-------------------------------------------------
// Siemens Service Provider extensions to the
// PHONEBUTTONFUNCTION values.
//-------------------------------------------------

// Decrement this value whenever new features are added.
#define DB_FIRST_SIEMENSBUTTONFUNCTION                                                  0x80000017

#define SIEMENSBUTTONFUNCTION_CALLWAITWTONEOFF                                  0x80000017
#define SIEMENSBUTTONFUNCTION_CALLWAITWTONEON                                   0x80000018

#define SIEMENSBUTTONFUNCTION_CONFQUIT                                          0x80000019

#define SIEMENSBUTTONFUNCTION_MSGWAIT_CANCEL_TERMIN                             0x8000001A
#define SIEMENSBUTTONFUNCTION_MSGWAIT_CANCEL_ORIGIN                             0x8000001B
#define SIEMENSBUTTONFUNCTION_MSGWAIT_ACTIVATE                                  0x8000001C
#define SIEMENSBUTTONFUNCTION_SECRSUBSTITUTEOFF                                         0x8000001D
#define SIEMENSBUTTONFUNCTION_SECRSUBSTITUTEON                                          0x8000001E
#define SIEMENSBUTTONFUNCTION_SECRINTERCEPT4OFF                                         0x8000001F
#define SIEMENSBUTTONFUNCTION_SECRINTERCEPT4ON                                          0x80000020
#define SIEMENSBUTTONFUNCTION_SECRINTERCEPT3OFF                                         0x80000021
#define SIEMENSBUTTONFUNCTION_SECRINTERCEPT3ON                                          0x80000022
#define SIEMENSBUTTONFUNCTION_SECRINTERCEPT2OFF                                         0x80000023
#define SIEMENSBUTTONFUNCTION_SECRINTERCEPT2ON                                          0x80000024
#define SIEMENSBUTTONFUNCTION_SECRINTERCEPT1OFF                                         0x80000025
#define SIEMENSBUTTONFUNCTION_SECRINTERCEPT1ON                                          0x80000026
#define SIEMENSBUTTONFUNCTION_CLIROFF                                                   0x80000027
#define SIEMENSBUTTONFUNCTION_CLIRON                                                    0x80000028
#define SIEMENSBUTTONFUNCTION_MAINMENUOFF                                               0x80000029
#define SIEMENSBUTTONFUNCTION_MAINMENUON                                                0x8000002A

#define SIEMENSBUTTONFUNCTION_ACDNIGHTSRVOFF                                            0x8000002B
#define SIEMENSBUTTONFUNCTION_ACDNIGHTSRVON                                             0x8000002C
#define SIEMENSBUTTONFUNCTION_AGTMODERDY                                                0x8000002D
#define SIEMENSBUTTONFUNCTION_AGTLOGOFF                                         0x8000002E
#define SIEMENSBUTTONFUNCTION_MESSAGEANSWEROFF                                          0x8000002F
#define SIEMENSBUTTONFUNCTION_MESSAGEANSWERON                                           0x80000030
#define SIEMENSBUTTONFUNCTION_PRIVACYOFF                                                0x80000031
#define SIEMENSBUTTONFUNCTION_PRIVACYON                                         0x80000032
#define SIEMENSBUTTONFUNCTION_CLIPOFF                                                   0x80000033
#define SIEMENSBUTTONFUNCTION_CLIPON                                                    0x80000034
#define SIEMENSBUTTONFUNCTION_STARTHUNT                                         0x80000035
#define SIEMENSBUTTONFUNCTION_CODELOCKOFF                                               0x80000036
#define SIEMENSBUTTONFUNCTION_CODELOCKON                                                0x80000037
#define SIEMENSBUTTONFUNCTION_CALLBACKGET                                               0x80000038
#define SIEMENSBUTTONFUNCTION_CALLBACKSET                                               0x80000039
#define SIEMENSBUTTONFUNCTION_NIGHTSRVOFF                                               0x8000003A
#define SIEMENSBUTTONFUNCTION_NIGHTSRVON                                                0x8000003B

#define SIEMENSBUTTONFUNCTION_ACCOUNTCODE                                               0x8000003C
#define SIEMENSBUTTONFUNCTION_UNPARKGROUP                                               0x8000003D
#define SIEMENSBUTTONFUNCTION_PROGRAMSERVICEMODEOFF                             0x8000003E
#define SIEMENSBUTTONFUNCTION_PROGRAMSERVICEMODEON                              0x8000003F
#define SIEMENSBUTTONFUNCTION_SYSUNPARK                                         0x80000040
#define SIEMENSBUTTONFUNCTION_SYSPARK                                           0x80000041
#define SIEMENSBUTTONFUNCTION_CONFREMP8                                                 0x80000042
#define SIEMENSBUTTONFUNCTION_CONFREMP7                                                 0x80000043
#define SIEMENSBUTTONFUNCTION_CONFREMP6                                                 0x80000044
#define SIEMENSBUTTONFUNCTION_CONFREMP5                                                 0x80000045
#define SIEMENSBUTTONFUNCTION_CONFREMP4                                                 0x80000046
#define SIEMENSBUTTONFUNCTION_CONFREMP3                                                 0x80000047
#define SIEMENSBUTTONFUNCTION_CONFREMP2                                                 0x80000048
#define SIEMENSBUTTONFUNCTION_CONFREMP1                                                 0x80000049
#define SIEMENSBUTTONFUNCTION_CONSULTATION                                      0x8000004A
#define SIEMENSBUTTONFUNCTION_HEADSETOFF                                        0x8000004B
#define SIEMENSBUTTONFUNCTION_HEADSETON                                         0x8000004C

#define SIEMENSBUTTONFUNCTION_CALLWAITINGDEACT                                          0x8000004D

#define SIEMENSBUTTONFUNCTION_DNDOFF                                                    0x8000004E
#define SIEMENSBUTTONFUNCTION_DNDON                                                     0x8000004F

#define SIEMENSBUTTONFUNCTION_MUTEOFF                                                   0x80000050
#define SIEMENSBUTTONFUNCTION_MUTEON                                                    0x80000051

#define SIEMENSBUTTONFUNCTION_CDRACCOUNTCODE                                            0x80000052
#define SIEMENSBUTTONFUNCTION_RINGERCUTOFF                                              0x80000053
#define SIEMENSBUTTONFUNCTION_PRIVATE                                                   0x80000054
#define SIEMENSBUTTONFUNCTION_ACDNIGHTANSWER                                            0x80000055
#define SIEMENSBUTTONFUNCTION_GENCALLKEY                                                0x80000056
#define SIEMENSBUTTONFUNCTION_SELECTLANGUAGE                                            0x80000057
```

```
#define SIEMENSBUTTONFUNCTION_REMOTEFLASH                                0x80000058
#define SIEMENSBUTTONFUNCTION_DIALPADKEYA                                0x80000059
#define SIEMENSBUTTONFUNCTION_DIALPADKEYB                                0x80000060
#define SIEMENSBUTTONFUNCTION_DIALPADKEYC                                0x80000061
#define SIEMENSBUTTONFUNCTION_DIALPADKEYD                                0x80000062
#define SIEMENSBUTTONFUNCTION_ABSENT                                     0x80000063
#define SIEMENSBUTTONFUNCTION_HELP                                       0x80000064
#define SIEMENSBUTTONFUNCTION_RECORDCHARGES                              0x80000065
#define SIEMENSBUTTONFUNCTION_MAIL                                       0x80000066
#define SIEMENSBUTTONFUNCTION_STORETEXT                                0x80000067
#define SIEMENSBUTTONFUNCTION_CALLINGNAMEID                              0x80000068
#define SIEMENSBUTTONFUNCTION_ATNDTRANSFER                               0x80000069
#define SIEMENSBUTTONFUNCTION_ATNDNOTAVAIL                               0x8000006A
#define SIEMENSBUTTONFUNCTION_ALTCONTACT                                 0x8000006B
#define SIEMENSBUTTONFUNCTION_BUSYINDICATOR                              0x8000006C
#define SIEMENSBUTTONFUNCTION_ATNDQUEUE                                0x8000006D
#define SIEMENSBUTTONFUNCTION_PLUSMINUS                                0x8000006E
#define SIEMENSBUTTONFUNCTION_ACDAVAILTOGGLE                             0x8000006F
#define SIEMENSBUTTONFUNCTION_MAILBOXMENU                                0x80000070
#define SIEMENSBUTTONFUNCTION_VARFWDPOSTDIAL                             0x80000071
#define SIEMENSBUTTONFUNCTION_SYSTERMMODE                                0x80000072
#define SIEMENSBUTTONFUNCTION_FWDMENU                                    0x80000073
#define SIEMENSBUTTONFUNCTION_CAFLOGOFF                                0x80000074
#define SIEMENSBUTTONFUNCTION_CAFGPLOGOFF                                0x80000075
#define SIEMENSBUTTONFUNCTION_CAFAVAIL                                 0x80000076
#define SIEMENSBUTTONFUNCTION_CAFGPAVAIL                                 0x80000077
#define SIEMENSBUTTONFUNCTION_CAFUNAVAIL                                 0x80000078
#define SIEMENSBUTTONFUNCTION_CAFGPUNAVAIL                               0x80000079
#define SIEMENSBUTTONFUNCTION_CAFFORWARD                                 0x8000007A
#define SIEMENSBUTTONFUNCTION_CAFFORWARDCAN                              0x8000007B
#define SIEMENSBUTTONFUNCTION_T_VARFWDALL                                0x8000007C
#define SIEMENSBUTTONFUNCTION_T_VARFWDBUSY                               0x8000007D
#define SIEMENSBUTTONFUNCTION_T_VARFWDNA                                 0x8000007E
#define SIEMENSBUTTONFUNCTION_T_FIXFWDALL                                0x8000007F
#define SIEMENSBUTTONFUNCTION_T_FIXFWDBUSY                               0x80000080
#define SIEMENSBUTTONFUNCTION_T_FIXFWDNA                                 0x80000081
#define SIEMENSBUTTONFUNCTION_CODECALL                                 0x80000082
#define SIEMENSBUTTONFUNCTION_MSGRSUMMON                                 0x80000083
#define SIEMENSBUTTONFUNCTION_CALLRCDSILENT                              0x80000084
#define SIEMENSBUTTONFUNCTION_STOPHUNT                                 0x80000085
#define SIEMENSBUTTONFUNCTION_XFERPUSHPULL                               0x80000086
#define SIEMENSBUTTONFUNCTION_PARKGROUP                                0x80000087
#define SIEMENSBUTTONFUNCTION_VOICECALL                                0x80000088
#define SIEMENSBUTTONFUNCTION_SECRXFER                                 0x80000089
#define SIEMENSBUTTONFUNCTION_SECRINTERCEPT                              0x8000008A
#define SIEMENSBUTTONFUNCTION_SECRPICKUP                                 0x8000008B
#define SIEMENSBUTTONFUNCTION_TIMEDREMINDER                              0x8000008C
#define SIEMENSBUTTONFUNCTION_FIXEDCONFGROUP                             0x8000008D
#define SIEMENSBUTTONFUNCTION_INTERCOMGROUP                              0x8000008E
#define SIEMENSBUTTONFUNCTION_SPEEDSYSGROUP                              0x8000008F
#define SIEMENSBUTTONFUNCTION_TAFAS                                      0x80000090
#define SIEMENSBUTTONFUNCTION_PICKUPGROUP                                0x80000091
#define SIEMENSBUTTONFUNCTION_PARKINTERCOM                               0x80000092
#define SIEMENSBUTTONFUNCTION_RELAYCONTROL                               0x80000093
#define SIEMENSBUTTONFUNCTION_PICKUPINTERCOM                             0x80000094
#define SIEMENSBUTTONFUNCTION_ACDDONOTDISTURB                            0x80000095
#define SIEMENSBUTTONFUNCTION_RELEASETRUNK                               0x80000096
#define SIEMENSBUTTONFUNCTION_REJECTCAMPON                               0x80000097
#define SIEMENSBUTTONFUNCTION_PICKUPCAMPON                               0x80000098
#define SIEMENSBUTTONFUNCTION_CALLWAITINGACT                             0x80000099
#define SIEMENSBUTTONFUNCTION_CAMPONMANUAL                               0x8000009A
#define SIEMENSBUTTONFUNCTION_DIRECTEDPICKUP                             0x8000009B
#define SIEMENSBUTTONFUNCTION_CODELOCK                                 0x8000009C
#define SIEMENSBUTTONFUNCTION_GETERRORS                                0x8000009D
#define SIEMENSBUTTONFUNCTION_GETCHARGES                                 0x8000009E
#define SIEMENSBUTTONFUNCTION_CLIR                                       0x8000009F
#define SIEMENSBUTTONFUNCTION_MONITORRLINE                               0x800000A0
#define SIEMENSBUTTONFUNCTION_SERVICE_MENU                               0x800000A1
#define SIEMENSBUTTONFUNCTION_MENU_SELECT                                0x800000A2
#define SIEMENSBUTTONFUNCTION_MENU_FWD                                 0x800000A3
#define SIEMENSBUTTONFUNCTION_MENU_BWD                                 0x800000A4
#define SIEMENSBUTTONFUNCTION_MALICIOUSCALLID                            0x800000A5
#define SIEMENSBUTTONFUNCTION_LASTCALLS                                0x800000A6
#define SIEMENSBUTTONFUNCTION_LEAVEMESSAGE                               0x800000A7
#define SIEMENSBUTTONFUNCTION_FIXFWDTRUNK                                0x800000A8
#define SIEMENSBUTTONFUNCTION_DIRECTORY                                0x800000A9
#define SIEMENSBUTTONFUNCTION_DUPLEXVOICECALLPROTECTION                  0x800000AA
#define SIEMENSBUTTONFUNCTION_RETRIEVECALL                               0x800000AB
#define SIEMENSBUTTONFUNCTION_DOOR                                       0x800000AC
#define SIEMENSBUTTONFUNCTION_DTMF                                       0x800000AD
#define SIEMENSBUTTONFUNCTION_MONITORROOM                                0x800000AE
#define SIEMENSBUTTONFUNCTION_NET_A                                      0x800000AF
#define SIEMENSBUTTONFUNCTION_NET_B                                      0x800000B0
#define SIEMENSBUTTONFUNCTION_ASSOCIATEDDIALING                          0x800000B1
#define SIEMENSBUTTONFUNCTION_ASSOCIATEDSERVICE                          0x800000B2
#define SIEMENSBUTTONFUNCTION_SHOWWAITINGCALLS                           0x800000B3
// B4-B7 reserved
```

```
#define SIEMENSBUTTONFUNCTION_SPKRONEBCAST                                  0x800000B8
#define SIEMENSBUTTONFUNCTION_ADL1                                          0x800000B9
#define SIEMENSBUTTONFUNCTION_ADL2                                          0x800000BA
#define SIEMENSBUTTONFUNCTION_ADL3                                          0x800000BB
#define SIEMENSBUTTONFUNCTION_ADL4                                          0x800000BC
#define SIEMENSBUTTONFUNCTION_AUTOICOM                                      0x800000BD
#define SIEMENSBUTTONFUNCTION_TMMS                                          0x800000BE
#define SIEMENSBUTTONFUNCTION_CAMPMSGS                                      0x800000BF
#define SIEMENSBUTTONFUNCTION_GROUPPICK                                     0x800000C0
#define SIEMENSBUTTONFUNCTION_MSGWAITCTRL                                   0x800000C1
#define SIEMENSBUTTONFUNCTION_MSGWAIT                                       0x800000C2
#define SIEMENSBUTTONFUNCTION_SPKRONEWAY                                    0x800000C3
#define SIEMENSBUTTONFUNCTION_ADK                                           0x800000C4
#define SIEMENSBUTTONFUNCTION_DSSKEY                                        0x800000C5
#define SIEMENSBUTTONFUNCTION_AGTMODEAVL                                    0x800000C6
#define SIEMENSBUTTONFUNCTION_AGTMODEUNAV                                   0x800000C7
#define SIEMENSBUTTONFUNCTION_AGTMODEWRK                                    0x800000C8
#define SIEMENSBUTTONFUNCTION_MANANS                                        0x800000C9
#define SIEMENSBUTTONFUNCTION_ACDAGTMSG                                     0x800000CA
#define SIEMENSBUTTONFUNCTION_ACDSUPMSG                                     0x800000CB
#define SIEMENSBUTTONFUNCTION_ACDEMERMSG                                    0x800000CC
#define SIEMENSBUTTONFUNCTION_ACDMONSLNT                                    0x800000CD
#define SIEMENSBUTTONFUNCTION_ACDMONTONE                                    0x800000CE
#define SIEMENSBUTTONFUNCTION_ACDPRIGRP                                     0x800000CF
#define SIEMENSBUTTONFUNCTION_ACDSECGRP                                     0x800000D0
#define SIEMENSBUTTONFUNCTION_ACDPRIQUE                                     0x800000D1
#define SIEMENSBUTTONFUNCTION_ACDSECQUE                                     0x800000D2
#define SIEMENSBUTTONFUNCTION_PROGRAM                                       0x800000D3
#define SIEMENSBUTTONFUNCTION_MAILBOX                                       0x800000D4
#define SIEMENSBUTTONFUNCTION_HEADSET                                       0x800000D5
#define SIEMENSBUTTONFUNCTION_CALLBACKKEY                                   0x800000D6
#define SIEMENSBUTTONFUNCTION_PHONEMAIL                                     0x800000D7
#define SIEMENSBUTTONFUNCTION_PRIVACY                                       0x800000D8
#define SIEMENSBUTTONFUNCTION_TOGGLE                                        0x800000D9
#define SIEMENSBUTTONFUNCTION_DATAKEY                                       0x800000DA
#define SIEMENSBUTTONFUNCTION_DDSLOCAL                                      0x800000DB
#define SIEMENSBUTTONFUNCTION_ACCTNUM                                       0x800000DC
#define SIEMENSBUTTONFUNCTION_AGTLOGON                                      0x800000DD
#define SIEMENSBUTTONFUNCTION_ACDMSGACK                                     0x800000DE
#define SIEMENSBUTTONFUNCTION_ACDMSGSCROLL                                  0x800000DF
#define SIEMENSBUTTONFUNCTION_BUZZPRESET                                    0x800000E0
#define SIEMENSBUTTONFUNCTION_COSCHANGE                                     0x800000E1
#define SIEMENSBUTTONFUNCTION_VARFWDALLINT                                  0x800000E2
#define SIEMENSBUTTONFUNCTION_VARFWDALLEXT                                  0x800000E3
#define SIEMENSBUTTONFUNCTION_VARFWDBOTH                                    0x800000E4
#define SIEMENSBUTTONFUNCTION_VARFWDBUSY                                    0x800000E5
#define SIEMENSBUTTONFUNCTION_VARFWDNA                                0x800000E6
#define SIEMENSBUTTONFUNCTION_VARFWDBUSYNA                                  0x800000E7
#define SIEMENSBUTTONFUNCTION_VARFWDCANCEL                                  0x800000E8
#define SIEMENSBUTTONFUNCTION_FIXFWDACT                               0x800000E9
#define SIEMENSBUTTONFUNCTION_FIXFWDCAN                               0x800000EA
#define SIEMENSBUTTONFUNCTION_PVTHOLD                                       0x800000EB
#define SIEMENSBUTTONFUNCTION_STORE                                         0x800000EC
#define SIEMENSBUTTONFUNCTION_BADTRUNK                                0x800000ED
#define SIEMENSBUTTONFUNCTION_SPKRFIXONE                                    0x800000EE
#define SIEMENSBUTTONFUNCTION_SPKRTWOWAY                                    0x800000EF
#define SIEMENSBUTTONFUNCTION_SPKRREJECT                                    0x800000F0
#define SIEMENSBUTTONFUNCTION_START                                         0x800000F1
// unused:
//                                                                          0x800000F2
//                                                                          0x800000F3
//                                                                          0x800000F4
#define SIEMENSBUTTONFUNCTION_SYSSPEEDGP2                                   0x800000F5
#define SIEMENSBUTTONFUNCTION_LINEKEY                                       0x800000F6
#define SIEMENSBUTTONFUNCTION_OVERRIDE                                      0x800000F7
#define SIEMENSBUTTONFUNCTION_CLEAR                                   0x800000F8
#define SIEMENSBUTTONFUNCTION_CANCEL                                        0x800000F9
#define SIEMENSBUTTONFUNCTION_DTS                                           0x800000FA
#define SIEMENSBUTTONFUNCTION_INUSEKEY                                0x800000FB
#define SIEMENSBUTTONFUNCTION_REPEATID                                      0x800000FC
#define SIEMENSBUTTONFUNCTION_ROLMPARK                                0x800000FD
#define SIEMENSBUTTONFUNCTION_CONFREMLAST                                   0x800000FE
#define SIEMENSBUTTONFUNCTION_CALLWTG                                       0x800000FF


//------------------------------------------------------------------------
// Additional literals
//------------------------------------------------------------------------

// possible values for dwRequestType in SIEMENS_LINE_DEV_SPECIFIC
#define SIEMENSLINE_DIAL_DIGITS                     0x00000001  //dials digits
#define SIEMENSLINE_GET_FEATURELIST                 0x00000002  //Get previously got FeatureList
#define SIEMENSLINE_SET_HLB_PROGRAMKEY        0x00000003  //Set speedkey
#define SIEMENSLINE_SET_OPTISET_TYPE                0x00000004  //Set type of optiset
#define SIEMENSLINE_REQ_FEATURE_STATES              0x00000005  //requests the transmission of feature states
#define SIEMENSLINE_PARK_DIRECT                     0x00000006  //parks a call to a specific station
#define SIEMENSLINE_GET_PUSKEYINFO                  0x00000007  //Get infos about a PUS key
```

```
#define SIEMENSLINE_PRESS_PUSKEY                          0x00000008  //Press a specific PUS key
#define SIEMENSLINE_SET_HEADSET                           0x00000009  //set/unset headset support
#define SIEMENSLINE_GET_SYSPARKSLOTS                      0x0000000A  //get the occupancy of all system park slots

// possible values for dwParam1 in LINE_DEVSPECIFIC
#define SIEMENSLINE_FEATURELIST_COMPLETE                  0x00000000
#define SIEMENSLINE_LM_ACK                                0x00000001
#define SIEMENSLINE_LM_REJECT                             0x00000002
#define SIEMENSLINE_LED_STATE_CHANGED                     0x00000003
#define SIEMENSLINE_PARKINFO                              0x00000004  //provide the slot which was used to parked a call on it
#define SIEMENSLINE_PARKSLOTSTATE                         0x00000005  //provide the (occupancy) status of all system park slots
#define SIEMENSLINE_DFP_STATES_CHANGED                    0x8000FFFF

//constants for Optiset-Type
#define SIEMENSLINE_OPTISET_TYPE_BASIC                    1  //"optiset E basic"
#define SIEMENSLINE_OPTISET_TYPE_ADVANCE                  2  //Hicom 300 US specific "optiSet advance"
#define SIEMENSLINE_OPTISET_TYPE_COMFORT                  3  //also known as "optiSet advance plus"
#define SIEMENSLINE_OPTISET_TYPE_MEMORY                   4  //"optiset E memory"
#define SIEMENSLINE_OPTIPOINT500_TYPE_BASIC               5  //"optiPoint 500 basic"
#define SIEMENSLINE_OPTIPOINT500_TYPE_STANDARD            6  //"optiPoint 500 standard"
#define SIEMENSLINE_OPTIPOINT500_TYPE_ADVANCE             7  //"optiPoint 500 advance"
#define SIEMENSLINE_OPTIPOINT600_TYPE_OFFICE              8  //"optiPoint 600 office"
#define SIEMENSLINE_OPTIPOINT410_TYPE_ENTRY               11 //"optiPoint 410 entry"
#define SIEMENSLINE_OPTIPOINT410_TYPE_ECONOMY             12 //"optiPoint 410 economy"
#define SIEMENSLINE_OPTIPOINT410_TYPE_STANDARD            13 //"optiPoint 410 standard"
#define SIEMENSLINE_OPTIPOINT410_TYPE_ADVANCE             14 //"optiPoint 410 advance"

// the number of digits allowed in szDialableString[] in SIEMENS_LINE_DEV_SPECIFIC
#define SIEMENS_MAX_DIALABLE_STRING_BYTES                 100

//maximum number of features in feature-list
#define SIEMENS_MAX_FEATURES                              256
//"*123" + 0x00
#define SIEMENS_FEATURE_LENGTH                            5

#define SIEMENS_MAX_RUFNR_LEN                             9

// The number of bitmasks required in an array to represent each button function with a bit
#define SIEMENS_NUM_BUTTONFCN_BITMASKS           8

// possible values for dwProgramState in SIEMENS_PHONESTATUS_DEV_SPECIFIC
#define SIEMENSPROGRAMSTATE_DEACTIVATED                   0L
#define SIEMENSPROGRAMSTATE_ACTIVATED                     1L
#define SIEMENSPROGRAMSTATE_SYSTEMUSER                    1L
#define SIEMENSPROGRAMSTATE_SYSTEMHOST                    2L
#define SIEMENSPROGRAMSTATE_USER                          3L


//---------------------------------------------------------------------------
// Ring Modes
//---------------------------------------------------------------------------

#define SIEMENSRINGMODE_SILENCE                    0

#define SIEMENSRINGMODE_SINGLE_RING_INTERN         1
#define SIEMENSRINGMODE_SINGLE_RING                1

#define SIEMENSRINGMODE_DBL_RING_EXTERNAL          2
#define SIEMENSRINGMODE_DBL_RING                   2

#define SIEMENSRINGMODE_TPL_RING_RECALL            3
#define SIEMENSRINGMODE_TPL_RING                   3

#define SIEMENSRINGMODE_SINGLE_BUZZ_ALERT          4
#define SIEMENSRINGMODE_SINGLE_BUZZ                4

#define SIEMENSRINGMODE_COM_RING                   5

#define SIEMENSRINGMODE_TPL_BUZ_DATA_CALL          6
#define SIEMENSRINGMODE_TPL_BUZZ                   6

#define SIEMENSRINGMODE_CONT_RING_EMERGCY          7
#define SIEMENSRINGMODE_CONT_RING                  7

#define SIEMENSRINGMODE_DOUBLE_BUZZ_NOTIFY         8

#define SIEMENSRINGMODE_DOUBLE_BUZZ                9

#define SIEMENSRINGMODE_RESERVED                   10

#define SIEMENSRINGMODE_UNKNOWN                    11


//---------------------------------------------------------------------------
// Structure Definitions
//---------------------------------------------------------------------------
```

```
//--------------------------------------------------------------------
// SIEMENS_LINE_DEV_SPECIFIC:
//
// Applications pass this device-specific structure to the
// lineDevSpecific() routine.
//
// dwRequestType[] - possible values:
//      SIEMENSLINE_DIAL_DIGITS tells lineDevSpecific() to
//       send the digits in szDialableString[] to the phone.
//
//                           szDialableString [] - a Null-terminated ASCII string of dialable
//                            digits in the range of (0..9, *, #), which should be sent to the phone.
//
//      SIEMENSLINE_GET_FEATURELIST returns the availability
//       of the provided features
//
//                           bFeatureListAvailable - an indicator, if the initial feature state transmission
//                            was done and if the following strings are filled or not
//
//                           szStaticFeatureListGlobal[] - provides the feature availability of global features
//
//                           szStaticFeatureListSpecific[] - provides the feature availability of system specific features
//
//      SIEMENSLINE_SET_HLB_PROGRAMKEY programs a virtual
//       station keys on the HG1500 (HiPath 3000 specific)
//
//                           iKeyIndex - index of the programable virtual key
//
//                           szIntNumber - internal station (destination) number
//
//      SIEMENSLINE_SET_OPTISET_TYPE changes the configured
//       device type from the applications side
//
//                           dwOptisetType - device type
//
//      SIEMENSLINE_REQ_FEATURESTATES requests DEV_SPECIFIC
//       events for every change on a feature state
//
//                           bSwitch - switch, if functionality should be ON or OFF
//
//      SIEMENSLINE_PARK_DIRECT parks a call to a specific station
//
//                           szParkSTNNo [] - a Null-terminated ASCII string of dialable
//                            digits in the range of (0..9, *, #), which should be sent to the phone as park destination no.
//
//      SIEMENSLINE_GET_PUSKEYINFO gets infos about a PUS key
//
//                           iPUSKeyNo - index of the available PUS keys
//                           bDevice - device identifier regarding the device on which the PUS key is configured
//                           bKeyNo - key identifier regarding the physical key on which the PUS key is configured
//                           szSTNNo [] - a Null-terminated ASCII string of the PUS key related station no.
//                           szSTNName [] - a Null-terminated ASCII string of the PUS key related station name
//                           bState - identifier regarding the status in which the PUS key is actually in
//
//      SIEMENSLINE_PRESS_PUSKEY presses a specific PUS key
//
//                           iPUSKeyNoToPress - index of the available PUS key which is to pressed
//
//      SIEMENSLINE_SET_HEADSET sets/unsets headset support
//
//                           bHSavail - switch, headset support should be ON (TRUE) or OFF (FALSE)
//
//      SIEMENSLINE_GET_SYSPARKSLOTS get the occupancy of all system park slots
//
//                           bPlaceholder - placeholder, should be ON (TRUE) or OFF (FALSE) w/o impact
//
//
//      NOTE: The max string size may be limited by the service provider.
//            Please consult the TSP documentation for this information.
//--------------------------------------------------------------------
typedef struct _SIEMENS_LINE_DEV_SPECIFIC
{
   DWORD dwRequestType;

   union
   {
      //dwRequestType == SIEMENSLINE_DIAL_DIGITS
      BYTE szDialableString [SIEMENS_MAX_DIALABLE_STRING_BYTES + 1];

      //dwRequestType == SIEMENSLINE_GET_FEATURELIST
      struct
      {
         BOOL bFeatureListAvailable;
         char szStaticFeatureListGlobal[SIEMENS_MAX_FEATURES/8];
         char szStaticFeatureListSpecific[SIEMENS_MAX_FEATURES/8];
      };
```

```
      //dwRequestType == SIEMENSLINE_SET_HLB_PROGRAMKEY
      struct
      {
         INT  iKeyIndex;
         char szIntNumber[SIEMENS_MAX_RUFNR_LEN];
      };

      //dwRequestType == SIEMENSLINE_SET_OPTISET_TYPE
      DWORD dwOptisetType;

      //dwRequestType == SIEMENSLINE_REQ_FEATURE_STATES
      BOOL bSwitch;

      //dwRequestType == SIEMENSLINE_PARK_DIRECT
      BYTE szParkSTNNo [SIEMENS_MAX_DIALABLE_STRING_BYTES + 1];

      //dwRequestType == SIEMENSLINE_GET_PUSKEYINFO
      struct
      {
                     INT      iPUSKeyNo;
                     BYTE     bDevice;
                     BYTE     bKeyNo;
                     char     szSTNNo[7];
                     char     szSTNName[50];
                     BYTE     bState;
      };

      //dwRequestType == SIEMENSLINE_PRESS_PUSKEY
            INT     iPUSKeyNoToPress;

      //dwRequestType == SIEMENSLINE_SET_HEADSET
      BOOL bHSavail;

      //dwRequestType == SIEMENSLINE_GET_SYSPARKSLOTS
      BOOL bPlaceholder;
   };
}
SIEMENS_LINE_DEV_SPECIFIC, FAR *LPSIEMENS_LINE_DEV_SPECIFIC;


//--------------------------------------------------------------------
// SIEMENS_LINEDEVCAPS_DEV_SPECIFIC
//
// The device specific portion of the LINEDEVCAPS structure contains the
// following device specific data:
//
//   dwDevSpecificRequestsAvailable - A bitmask of available requests
//      that may be passed to lineDevSpecific() in the dwRequestType
//      field of the SIEMENS_LINE_DEV_SPECIFIC structure. The possible
//      bit(s) that may be set:
//          SIEMENSLINE_DIAL_DIGITS
//
//   dwInvokableButtonsBitmasks[] - An array of bitmasks indicating which
//      PHONEBUTTONFUNCTIONs may be invoked via lineDevSpecificFeature().
//      Note: this is the static availability of the feature as opposed
//      to the dynamic availability of the feature based on the current
//      state of the device. The dynamic feature availability is found
//      in the device specific portion of the LINEDEVSTATUS structure.
//      The PHONEBUTTONFUNCTION value is the zero-based bit position in
//      the bitmask, and the bit is set if the function may be invoked.
//
//--------------------------------------------------------------------
typedef struct _SIEMENS_LINEDEVCAPS_DEV_SPECIFIC
{
          DWORD dwDevSpecificRequestsAvailable;
          DWORD dwInvokableButtonsBitmasks [SIEMENS_NUM_BUTTONFCN_BITMASKS];
}
SIEMENS_LINEDEVCAPS_DEV_SPECIFIC, FAR *LPSIEMENS_LINEDEVCAPS_DEV_SPECIFIC;


//--------------------------------------------------------------------
// SIEMENS_LINEDEVSTATUS_DEV_SPECIFIC
//
// The device specific portion of the LINEDEVSTATUS structure contains the
// following device specific data:
//
//   dwInvokableButtonsBitmasks[] - An array of bitmasks indicating which
//      PHONEBUTTONFUNCTIONs may currently be invoked by
//      lineDevSpecificFeature() based on the current state of the device.
//      The PHONEBUTTONFUNCTION value is the zero-based bit position in
//      the bitmask, and the bit is set if the function may be invoked.
//
//--------------------------------------------------------------------
typedef struct _SIEMENS_LINEDEVSTATUS_DEV_SPECIFIC
{
    DWORD       dwInvokableButtonsBitmasks [SIEMENS_NUM_BUTTONFCN_BITMASKS];
```

```
} SIEMENS_LINEDEVSTATUS_DEV_SPECIFIC, FAR* LPSIEMENS_LINEDEVSTATUS_DEV_SPECIFIC;


//--------------------------------------------------------------------------
// SIEMENS_PHONESTATUS_DEV_SPECIFIC:
//
// Structure for Device Specific extension to the PHONESTATUS structure.
//
//--------------------------------------------------------------------------
typedef struct _SIEMENS_PHONESTATUS_DEV_SPECIFIC
{
    DWORD       dwProgramState;
    DWORD       dwVirtualButtonSize;
    DWORD       dwVirtualButtonOffset;
}
SIEMENS_PHONESTATUS_DEV_SPECIFIC, FAR *LPSIEMENS_PHONESTATUS_DEV_SPECIFIC;



//--------------------------------------------------------------------------

#pragma pack(pop, include_siemens)      // restore previos packing alingment

#endif // __SIEMENS_H__
```

About Unify

Unify is one of the world's leading communications software and services firms, providing integrated communications solutions for approximately 75 percent of the Fortune Global 500. Our solutions unify multiple networks, devices and applications into one easy-to-use platform that allows teams to engage in rich and meaningful conversations. The result is a transformation of how the enterprise communicates and collaborates that amplifies collective effort, energizes the business, and enhances business performance. Unify has a strong heritage of product reliability, innovation, open standards and security.

Unify.com

UNIFY Harmonize your enterprise