Handspring GSM/GPRS Phone Library Reference

Release 1.1

Document number: 80-0234-00

Handspring, Inc.

189 Bernardo Ave.

Mountain View, CA 94043-5203

Voice: +1-650-230-5000

Fax: +1-650-230-2100

www.handspring.com

Table of Contents

1. Introduction **8**

2. Using the Phone Library **8**

2.1 Loading the Phone Library *8*
2.2 Detecting the Device Type *9*
2.3 Message Database *9*
2.4 Launch Codes *9*
2.4.1 phnLibLaunchCmdEvent 9
2.4.2 phnLibLaunchCmdRegister 10
2.5 Registration *10*

3. API Reference **10**

3.1 Library Information *10*
3.2 Header files to include *10*
3.3 Constants, Types, Enumerations, and Structures *11*
3.3.1 Constants 11
3.3.1.1 Common 11
3.3.1.2 Lock Types 11
3.3.1.3 Phone slider switch 11
3.3.1.4 Phone Volume 11
3.3.1.5 Phone Call Status flags 11
3.3.1.6 Phone Provider Configuration 12
3.3.2 Types 12
3.3.3 Enumerations 12
3.3.3.1 GSMDialCLIRMode 12
3.3.3.2 GSMSIMMessagesDialogKind 12
3.3.3.3 GSMRegistrationMode 13
3.3.3.4 GSMSIMStatus 13
3.3.3.5 PhnAddressField 13
3.3.3.6 PhnCLIRStatus 13
3.3.3.7 PhnConnectionEnum 14
3.3.3.8 PhnConnectStateType 14
3.3.3.9 PhnEquipmentMode 14
3.3.3.10 PhnEventCode on GSM Products 15
3.3.3.11 PhnIndicationKind 16
3.3.3.12 PhnModuleButtonModifersType 17
3.3.3.13 PhnModuleButtonType 17
3.3.3.14 PhnMsgBoxType 17
3.3.3.15 PhnOperatorStatus 17
3.3.3.16 PhnPasswordType 18
3.3.3.17 PhnPowerType 18
3.3.3.18 PhnProgressType 19
3.3.3.19 PhnRegistrationStatus 19
3.3.3.20 PhnRingerVolumeType 19
3.3.3.21 PhnServiceType 19
3.3.3.22 PhnVibrateType 20
3.3.3.23 PhoneServiceClassType 20
3.3.4 Structures 20
3.3.4.1 PhnConnectionType 20
3.3.4.2 PhnEventPtr & PhnEventType 21
3.3.4.3 PhnIndicationType 22
3.3.4.4 PhnMovedMsgDescType 22
3.3.4.5 PhnMovedMsgsParamsType 22
3.3.4.6 PhnMsgBoxDataType 23
3.3.4.7 PhnNBSNotificationEventType 23

3.3.4.8 PhnOperatorListPtr 24
3.3.4.9 PhnOperatorListType 24
3.3.4.10 PhnOperatorType 24
3.3.4.11 PhnPasswordEventType 24
3.3.4.12 PhnPhoneBookInfoPtr & PhnPhoneBookInfoType 24
3.3.4.13 PhnPhoneEquipmentMode 24
3.3.4.14 PhnPowerEventType 25
3.3.4.15 PhnProgressEventType 25
3.3.4.16 PhnRegistrationType 25
3.3.4.17 PhnRingingInfoPtr & PhnRingingInfoType 25
3.3.4.18 PhnRingingProfileType 25
3.3.4.19 PhoneGlobalsType 25
3.4 Library API 27
3.4.1 PhnLibAddAddress 27
3.4.2 PhnLibBattery 27
3.4.3 PhnLibBatteryCharge 27
3.4.4 PhnLibBoxInformation 28
3.4.5 PhnLibCardInfo 28
3.4.6 PhnLibClose 29
3.4.7 PhnLibConnectionAvailable 29
3.4.8 PhnLibCount 29
3.4.9 PhnLibCurrentOperator 30
3.4.10 PhnLibCurrentOperatorID 30
3.4.11 PhnLibCurrentProvider 30
3.4.12 PhnLibErrorRate 31
3.4.13 PhnLibFirstAppForService 31
3.4.14 PhnLibGetBoxNumber 32
3.4.15 PhnLibGetCLIP 32
3.4.16 PhnLibGetCLIR 33
3.4.17 PhnLibGetEchoCancellation 33
3.4.18 PhnLibGetEquipmentMode 33
3.4.19 PhnLibGetErrorText 33
3.4.20 PhnLibGetField 34
3.4.21 PhnLibGetLibAPIVersion 34
3.4.22 PhnLibGetMicrophone 34
3.4.23 PhnLibGetNth 35
3.4.24 PhnLibGetOperatorList 35
3.4.25 PhnLibGetOperatorLock 35
3.4.26 PhnLibGetOwnNumbers 36
3.4.27 PhnLibGetPhoneBook 36
3.4.28 PhnLibGetPhoneBookIndex 37
3.4.29 PhnLibGetPhoneCallStatus 37
3.4.30 PhnLibGetRingingInfo 37
3.4.31 PhnLibGetSIMStatus 37
3.4.32 PhnLibGetSMSGateway 38
3.4.33 PhnLibGetSMSRingInfo 38
3.4.34 PhnLibGetToneIDs 39
3.4.35 PhnLibGetToneName 39
3.4.36 PhnLibGetVolume 39
3.4.37 PhnLibHomeOperatorID 40
3.4.38 PhnLibIsLegalCharacter 40
3.4.39 PhnLibLength 41
3.4.40 PhnLibModuleButtonDown 41
3.4.41 PhnLibModulePowered 41
3.4.42 PhnLibNetworkAvailable 42
3.4.43 PhnLibNewAddress 42
3.4.44 PhnLibNewAddressList 42
3.4.45 PhnLibOpen 42

3.4.46 PhnLibPlayDTMF 43
3.4.47 PhnLibPlayTone 43
3.4.48 PhnLibRegister 43
3.4.49 PhnLibRegistered 44
3.4.50 PhnLibRoaming 44
3.4.51 PhnLibSendDTMF 44
3.4.52 PhnLibSendSilentDTMF 45
3.4.53 PhnLibSetEchoCancellation 45
3.4.54 PhnLibSetEquipmentMode 45
3.4.55 PhnLibSetField 46
3.4.56 PhnLibSetMicrophone 46
3.4.57 PhnLibSetModulePower 46
3.4.58 PhnLibSetOperator 47
3.4.59 PhnLibSetOperatorLock 47
3.4.60 PhnLibSetPhoneBook 47
3.4.61 PhnLibSetRingingInfo 48
3.4.62 PhnLibSetSMSRingInfo 48
3.4.63 PhnLibSetVolume 48
3.4.64 PhnLibSignalQuality 49
3.4.65 PhnLibSIMInfo 49
3.4.66 PhnLibSleep 49
3.4.67 PhnLibStartVibrate 50
3.4.68 PhnLibStopTone 50
3.4.69 PhnLibStopVibrate 50
3.4.70 PhnLibUsableSignalStrengthThreshold 50
3.4.71 PhnLibWake 51
3.5 Error Codes 52
3.5.1 Phone Library Error Codes 52
3.5.2 General Error Codes 52
3.5.3 Errors defined in the GSM recommendations 53
3.5.4 Errors returned by the firmware (NO CARRIER) 55
3.5.5 Errors for SIM Application Toolkit 55
3.5.6 Library loading errors 55
3.5.7 Used internally 56
3.5.8 Firmware boot synchronization 56
3.5.9 Direct Radio Error Codes 56

Change History		
Date	**Revision**	**Description of Changes**
August 1, 2002	1.0	Initial Beta Version
November, 2002	1.1	Added category of number returned by PhnLibGetOwnNumbers() Added PhnLibGetSMSGateway info about Trap number change.

1. Introduction

This document provides details about the Phone Library included in the Treo GSM/GPRS Communicator Family (180, 180g, 270 and the GPRS Software Upgrade). Included in this document is a description of the APIs, enumerations, data types, constants, structures, and error codes related to the Phone Library.

The Treo 300 is a CDMA version of the Treo GSM/GPRS products. There are differences when running on the Treo 300. Although we cannot guarantee this, we are working to resolve these differences in future updates to the Treo Family. At this time, this API document and the header files **only** cover the GSM/GPRS version of the products.

If you want to submit a bug report send an email to DevBugs@handspring.com. Note that there will be no feedback from this email as it is only for bug reports and not for submitting technical questions. For technical support, see http://www.handspring.com/developers/tech_support.jhtml.

NOTE: Handspring Phone Library is not compatible with Palm OS Telephony API. Although we cannot guarantee this, we are working to resolve these differences in future updates of the library.

2. Using the Phone Library

The following section provides information on loading the Phone Library and details the launch codes associated with the library.

2.1 Loading the Phone Library

As with all Palm OS libraries, one must load and open a library before making calls to it. The code sample below demonstrates how to load and open the Phone Library. The reference ID returned from the `SysLibLoad` function is the reference ID required by all of the Phone Library calls.

```
#include <PhoneLib_GSM\PhoneGlobals.h>
LoadPhoneLibrary(UInt16* libRefP, Boolean* libLoadedP)
{
Err error = 0;
      // Routine is pointless without this parameter
      if (!libRefP)
            return memErrInvalidParam;
      // Get the Phone library
      error = SysLibFind(phnLibName, libRefP);
      if (error)
      {
            // It's not already here - have to load it ourselves
            error = SysLibLoad(phnLibDbType, phnLibDbCreator, libRefP);
            if (error)
                  return error;
            if (libLoadedP)
                  *libLoadedP = true;
      }
      return error;
}
```

Once the application has finished with the library, it should close it as shown below:

```
      PhnLibClose (libRefP);
```

The libRefP contains the reference number of the library as returned by the `SysLibLoad` function.

2.2 Detecting the Device Type

Application developers may wish to check what version of Communicator they are running on. The code sample shown below demonstrates how to check the phone type. Please keep in mind that different versions of the product may have different features. For example, the GSM products support Mobile Originated messages while the CDMA product does not.

```
UInt32 phoneType;
// Check what device we are on: GSM or CDMA
(void)HsAttrGet (hsAttrPhoneType, 0, &phoneType);
if (phoneType == hsAttrPhoneTypeGSM)
  {
    // We are on a GSM Product
  }
else if (phoneType == hsAttrPhoneTypeCDMA)
  {
    // We are on a CDMA Product
  }
else
  {
    // Default to the GSM (send/receive message) form.
    // Pre-Treo 300 devices do not support this attribute so default to GSM
phone type
    phoneType = hsAttrPhoneTypeGSM;
  }
```

2.3 Message Database

On the GSM Products, the SMS Message database is always open. On the CDMA product, the SMS Message database is normally closed. Third-party applications need to open the message database before performing operations on it. The following code segment demonstrates how to do this. This can be implemented for both GSM and CDMA products. There will not be a problem if the database is already opened.

```
DmOpenRef smsRefNum = 0;
smsRefNum = PhnLibGetDBRef(smsLibRef);
// SMS Message access code
…
PhnLibReleaseDBRef(smsLibRef, smsRefNum);
```

2.4 Launch Codes

The Phone Library can send different launch codes in connection with certain events on the system. Any application that is registered with the Phone Library should be able to handle these launch codes.

2.4.1 phnLibLaunchCmdEvent

This launch code is sent to registered applications when the Phone Library needs to send a specific event to the application, such as notification of an incoming SMS or other phone event. On the GSM products this is defined to be `0x2bad`. The pointer that is passed to the application (via the `cmdPBP` argument) is a pointer to a `PhnEventType`. This structure provides additional information about the Phone event.

2.4.2 phnLibLaunchCmdRegister

This launch code is associated with the insertion of the VisorPhone in Visor models or a reset on GSM Treo models. When an application receives this event, it will usually register itself with the Phone Library. This launch code is sent to all the applications to give them a chance to register themselves to the Phone Library. All other launch codes are sent only if the corresponding application has registered with the library.

2.5 Registration

The GSM Product will send out a `phnLibLaunchCmdRegister` to applications to let them know they can register with the system.

3. API Reference

The following section provides detailed information about the Phone Library API.

3.1 Library Information

The following table contains the attributes of the GSM version of the Phone Library and related information:

Constant	Value	Comment/Description
`PhnLibDbCreator`	`GSM!`	Creator ID
`PhnLibDbType`	`libr`	Type ID
`PhnLibDbName`	`GSM Library`	Library's Database Name
`PhnLibName`	`GSMLibrary.lib`	Library name
`PhoneGlobalsFeature`	`0x10`	Phone Globals Feature
`PhnRingsDbName`	`System Ring Tones`	Phone Ring Tone Database
`PhnLibRingsDBCreatorID`	`GSMr`	Phone Ring Tone Database Creator ID
`PhnVdrvDbType`	`Vdrv`	Virtual Serial Driver Type ID
`PhnVdrvDbCreator`	`Hpsa`	Virtual Serial Driver Creator ID
`PhnNBSEvent`	`Hnbs`	NBS Event
`PhnNotifySubscriber`	`CLIP`	Notify Subscriber Code
`PhnNotifyEquipMode`	`Heqp`	Notify Equipment Code

3.2 Header files to include

The following table contains the header files for the phone library. These files are located in the new Handspring Phone SDK for GSM/GPRS.

File name	Description
PhoneGlobals.h	
PhoneLib.h	Included by PhoneGlobals.h
PhoneLibErrors.h	Included by PhoneLib.h
PhoneLibTraps.h	Included by PhoneLib.h

3.3 Constants, Types, Enumerations, and Structures

3.3.1 Constants

The following section contains some of the constants used in the Phone Library

3.3.1.1 Common

Constant	Value	Comment/Description
`phnLibUnknownID`	`0xff000000`	
`kMaxPhoneNumberLen`	`16`	
`kMaxRingName`	`16`	Maximum length of a ringer name (string resource)
`minPasswordLen`	`4`	
`maxPasswordLen`	`8`	

3.3.1.2 Lock Types

Lock facility constants for PhnLibGetOperatorLock()

Constant	Value	Comment/Description
`GSMLockSelectorOperatorLock`	`'PN'`	
`GSMLockSelectorProviderLock`	`'PP'`	

3.3.1.3 Phone slider switch

Constant	Value	Comment/Description
`kSliderLow`	`0`	
`kSliderHigh`	`1`	
`kSliderPositions`	`2`	

3.3.1.4 Phone Volume

Constant	Value	Comment/Description
`phnVolumeMin`	`0`	
`phnVolumeMax`	`7`	

3.3.1.5 Phone Call Status flags

Constant	Value	Comment/Description
`phnVoiceCall1Active`	`0x0001`	There is a voice call active on line1
`phnVoiceCall2Active`	`0x0002`	There is a voice call active on line2
`phnCSDCallActive`	`0x0004`	There is a data call currently active (NOTE: Virtual modem has control but does not necessarily have an active data call)
`phnGPRSCallActive`	`0x0008`	There is a GPRS session active

3.3.1.6 Phone Provider Configuration

Constant	Value	Comment/Description
`phnConfigPowerOnPassword`	`0x0001`	Require password to turn on phone
`phnConfigCallWaiting`	`0x0002`	Call waiting disabling
`phnConfigCallerID`	`0x0004`	Caller ID blocking
`phnConfigBlockingPassword`	`0x0020`	Require password to block calls
`phnConfigVoicemail`	`0x0400`	Voicemail support
`phnConfigVoicemailEditable`	`0x0800`	Voicemail number is editable

3.3.2 Types

The following table lists some of the types that have been defined for the Phone Library

GSM Library Data Type	Defined As...
`PhnConnectionID`	`UInt16`
`PhnDatabaseID`	`UInt32`
`GSMOperatorID`	`UInt32`
`PhnAddressHandle`	`MemHandle`
`PhnAddressList`	`MemHandle`

3.3.3 Enumerations

The following list describes the various enumerations that are found in the Phone Library.

3.3.3.1 GSMDialCLIRMode

Enumeration	Values	Comment/Description
`GSMDialCLIRMode`		Calling Line Identification Restriction Mode
	`gsmDialCLIRDefault`	Default
	`gsmDialCLIRTemporaryInvocation`	Temporarily enable for current call
	`GsmDialCLIRTemporarySuppression`	Temporarily disable for current call

3.3.3.2 GSMSIMMessagesDialogKind

Enumeration	Values	Comment/Description
`GSMSIMMessagesDialogKind`		
	`gsmMessagesConfirmMove` `gsmMessagesCantReceive`	

3.3.3.3 GSMRegistrationMode

Enumeration	Values	Comment/Description
`GSMRegistrationMode`		Registration Mode

	`gsmRegModeAutomatic`
	`gsmRegModeManual`
	`gsmRegModeDeregister`
	`gsmRegModeFormat`
	`gsmRegModeManualAutomatic`	

3.3.3.4 GSMSIMStatus

Enumeration	Values	Comment/Description
`GSMSIMStatus`		Status of the SIM card on the device

	`simMissing`
	`simFailure`
	`simWrong`
	`simNotReady`
	`simReady`	

3.3.3.5 PhnAddressField

Enumeration	Values	Comment/Description
`PhnAddressField`		Different fields of address structure

	`phnAddrFldPhone`
	`phnAddrFldFirstName`
	`phnAddrFldLastName`	

3.3.3.6 PhnCLIRStatus

Enumeration	Values	Comment/Description
`PhnCLIRStatus`		Calling Line Identification Restriction Status
	`clirNotProvisioned`	sent: restricted presentation of the calling line
	`clirProvisioned`	not sent: don't restrict presentation of the calling line
	`clirUnknown`	status not available
	`clirTemporaryRestricted`	not sent, override allowed
	`clirTemporaryAllowed`	sent: override allowed

3.3.3.7 PhnConnectionEnum

Enumeration	Values	Comment/Description

`PhnConnectionEnum`	`voiceConnection`
	`csdConnection`
	`gprsConnection`
		Phone Connection Type

3.3.3.8 PhnConnectStateType

Enumeration	Values	Comment/Description
`PhnConnectStateType`		Phone Connection State

	`phnConnectionActive`
	`phnConnectionHeld`
	`phnConnectionDialing`
	`phnConnectionAlerting`
	`phnConnectionIncoming`
	`phnConnectionWaiting`
	`phnConnectionUnknown`	

3.3.3.9 PhnEquipmentMode

Enumeration	Values	Comment/Description

`PhnEquipmentMode`	`phnHandsetMode`
	`phnSpeakerPhoneMode`
	`phnCarKitMode`
	`phnHeadsetMode`
	`phnHandsetLidCloseMode`
		Phone Modes

3.3.3.10 PhnEventCode on GSM Products

Here are the different events that an application can receive when it registers with the Phone Library.

```
// Phone event types
typedef enum {
       phnEvtCardInsertion,
       phnEvtRegistration,
       phnEvtError,
       phnEvtKeyPress,
       phnEvtPower,
       phnEvtPassword,
       phnEvtProgress,
       phnEvtIndication,
       phnEvtConnectInd,
       phnEvtConnectConf,
       phnEvtSubscriber,
       phnEvtDisconnectInd,
       phnEvtDisconnectConf,
       phnEvtBusy,
       phnEvtUpdate,
       phnEvtConference,
       phnEvtVoiceMail,
       phnEvtMessageInd,
       phnEvtSegmentInd,
       phnEvtMessageStat,
       phnEvtMessageDel,
       phnEvtMessageMoved,
       phnEvtSATNotification,
       phnEvtUSSDInd,
       phnEvtPhoneEquipmentMode,
       phnEvtGPRSRegistration,
       phnEvtMMSInd
} PhnEventCode;
```

Values	Additional Comments
Phone specific events	
phnEvtCardInsertion	Used for VisorPhone or PhoneLib is installed.
phnEvtRegistration	Ask application to register with library or Phone able to find service
phnEvtError	Indicator of something important happens to the phone that needs to bring up alert
phnEvtKeyPress	VisorPhone SMS button pressed or Phone, data or power button pressed on Treo
phnEvtPower	Phone is at the end of Power up sequence or phone starts the power down process or is at the end of power off sequence

Values	Additional Comments
phnEvtPassword	
phnEvtProgress	Indicate that an outgoing call in dialing state needs to be created
phnEvtIndication	Network search banner or power save banner needs to be drawn
phnEvtConnectInd	Indicate that an incoming call in incoming state needs to be created
phnEvtConnectConf	The call in dialing or incoming state with specified ID is just connected
phnEvtSubscriber	Need to update the number and the name of the specific connection ID
phnEvtDisconnectInd	A specific connection ID is told to shut down suddenly
phnEvtDisconnectConf	An ACK for a disconnection command on a specific connection ID is received
phnEvtBusy	Network Busy condition is received.
phnEvtUpdate	PhoneUI and registered application need to update its view.
phnEvtConference	Modem is in 3-way call mode now.
phnEvtVoiceMail	Voicemail indicator has been received
SMS Related Events	**SMS Related Events**
phnEvtMessageInd	A new SMS message has just been received (CMT in IS-637 standard)
phnEvtSegmentInd	
phnEvtMessageStat	See SMSMessageStatus enum in SMS Library
phnEvtMessageDel	
phnEvtMessageMoved	
Events used by the SIM Application Toolkit phnEvtSATNotification	
Others	
phnEvtUSSDInd	
phnEvtPhoneEquipmentMode	
phnEvtGPRSRegistration	
PhnEvtMMSInd	

3.3.3.11 PhnIndicationKind

Enumeration	Values	Comment/Description
PhnIndicationKind		
	indicationSIMReady indicationSIMMessages indicationNetworkSearch indicationPasswordAccepted	

3.3.3.12 PhnModuleButtonModifersType

Enumeration	Values	Comment/Description
`PhnModuleButtonModifersType`		Phone button modifiers
	`phnButtonPowerOnMask`	Button causes device to power on
	`phnButtunUpMask`	The button is released
	`phnButtonHeld`	The button is held down

3.3.3.13 PhnModuleButtonType

Enumeration	Values	Comment/Description
`PhnModuleButtonType`		Phone buttons

	`phnButtonPhoneApp`
	`phnButtonDataApp`
	`phnButtonHeadset`	

3.3.3.14 PhnMsgBoxType

Enumeration	Values	Comment/Description
`PhnMsgBoxType`		Message Box Types

	`kBoxVoice`
	`kBoxTelefax`
	`kBoxEMail`
	`kBoxOther`
	`kBoxData`	

3.3.3.15 PhnOperatorStatus

Enumeration	Values	Comment/Description
`PhnOperatorStatus`		Phone operator status

	`phnOpUnknown`
	`phnOpAvailable`
	`phnOpCurrent`
	`phnOpForbidden`	

3.3.3.16 PhnPasswordType

Enumeration	Values	Comment/Description
`PhnPasswordType`		Phone password types
	`PhnPasswordUnknown`	FAULT or none of the strings below
	`PhnPasswordNone`	READY
	`PhnPasswordSIMPIN`	SIM PIN
	`PhnPasswordSIMPUK`	SIM PUK
	`PhnPasswordPhSIMPIN`	PH-SIM PIN
	`phnPasswordPh1SIMPIN`	PH-FSIM PIN
	`phnPasswordPh1SIMPUK`	PH-FSIM PUK
	`phnPasswordSIMPIN2`	SIM PIN2
	`phnPasswordSIMPUK2`	SIM PUK2
	`PhnPasswordNetworkPIN`	PH-NET PIN
	`PhnPasswordNetworkPUK`	PH-NET PUK
	`phnPasswordNetworkSubsetPIN`	PH-NETSUB PIN
	`PhnPasswordNetworkSubsetPUK`	PH-NETSUB PUK
	`PhnPasswordServiceProviderPIN`	PH-SP PIN
	`PhnPasswordServiceProviderPUK`	PH-SP PUK
	`PhnPasswordCorporatePIN`	PH-CORP PIN
	`PhnPasswordCorporatePUK`	PH-CORP PUK
	`PhnPasswordBarrAO`	all outgoing call
	`PhnPasswordBarrOI`	outgoing international calls
	`PhnPasswordBarrOX`	outgoing international calls except to home country
	`PhnPasswordBarrAI`	all incoming calls
	`PhnPasswordBarrIR`	incoming calls when roaming outside home country
	`PhnPasswordBarrAB`	all barring services
	`PhnPasswordBarrAG`	all outgoing barring services
	`PhnPasswordBarrAC`	all incoming barring services

3.3.3.17 PhnPowerType

Enumeration	Values	Comment/Description
`PhnPowerType`		

	`phnPowerOff`
	`phnPowerOn`
	`phnPowerStartCharging`
	`phnPowerStopCharging`
	`phnPowerLow`	

3.3.3.18 PhnProgressType

Enumeration	Values	Comment/Description
`PhnProgressType`		

	`kOpenDialog`
	`kCloseDialog`
	`kSetText`
	`kSetRecipient`
	`kShowSegment`	

3.3.3.19 PhnRegistrationStatus

Enumeration	Values	Comment/Description
`PhnRegistrationStatus`		

	`registrationNone`
	`registrationHome`
	`registrationSearch`
	`registrationDenied`
	`registrationUnknown`
	`registrationRoaming`	

3.3.3.20 PhnRingerVolumeType

Enumeration	Values	Comment/Description

`PhnRingerVolumeType`	`phnRingerLoud`
	`phnRingerSoft`
	`phnRingerOff`
		Volume of the phone ringer

3.3.3.21 PhnServiceType

Enumeration	Values	Comment/Description

`PhnServiceType`	`kVoice`
	`phnServeData`
	`phnServeTelefax`
		Type of GSM Services

3.3.3.22 PhnVibrateType

Enumeration	Values	Comment/Description
`PhnVibrateType`		Vibrate mode while ringing
	`phnVibrateOff` `phnVibrateOn`	

3.3.3.23 PhoneServiceClassType

Enumeration	Values	Comment/Description
`PhoneServiceClassType`		Phone service classes

	`phnServiceVoice`
	`phnServiceSMS`
	`phnServiceTelefax`
	`phnServiceData`
	`phnServiceMail`
	`phnServiceSIMToolkit`
	`phnServiceAll`	

3.3.4 Structures

The following list describes the various structures that are used by the Phone Library.

3.3.4.1 PhnConnectionType

Structure	Data Type	Members	Description/Comment
`PhnConnectionType`			Phone connection info
	PhnConnectionID	id	
	PhnConnectStateType	state	
	PhnServiceType	service	
	Boolean	incoming	
	Boolean	multiparty	
	PhnAddressHandle	address	
	UInt32	owner	

3.3.4.2 PhnEventPtr & PhnEventType

Structure/Description	

`PhnEventType`
 	
`PhnEventPtr`
	Phone event record

```
Struct {
       Uint8  /*PhnEventCode*/       eventType;
       Boolean                       acknowledge;
       Uint16                        connectionID;
       Uint16                        launchCode;
       MemPtr                        launchParams;

       union Data {
              struct {
                     PhnAddressHandle      caller;        // Address handle
                     PhnServiceType        service;
              } info;

              struct {
                     PhnConnectionID       call1ID;
                     PhnConnectionID       call2ID;
                     PhnConnectionID       conferenceID;
              } conference;

              struct {
                     Err                   code;
                     UInt32                id;
              } error;

              struct {
                     UInt32                id;
                     char                  oldStatus;
                     char                  newStatus;
              } params;

              struct {
                     PhnModuleButtonType   key;
                     UInt16                modifiers;
              } keyPressed;

              PhnRegistrationType          registration;
              PhnMsgBoxDataType            msgBox;
              PhnPowerEventType            power;
              PhnPasswordEventType         password;
              PhnProgressEventType         progress;
              PhnIndicationType            indication;
              PhnMovedMsgsParamsType       moved;
              PhnUSSDEventType             ussd;
              PhnSATEventType              sat;
              PhnPhoneEquipmentMode        phoneEquipmentMode;
       } data;
};
```

3.3.4.3 PhnIndicationType

```
typedef struct   {
      PhnIndicationKind kind;
      char filler;
      union
            {
            struct
                  {
                  Boolean state;
                  } simReady;
            struct
                  {
                  GSMSIMMessagesDialogKind dialog;
                  Boolean moveMessages;
                  } simMessages;
            struct {
                  PhnPasswordType type;
                  } passwordAccepted;
            struct
                  {
                  Boolean state;
                  } networkAvailable;

      } data;
} PhnIndicationType;
```

3.3.4.4 PhnMovedMsgDescType

Structure	Data Type	Members	Description/Comment
`PhnMovedMsgDescType`			
	PhnDatabaseID	msgID	
	Uint32	msgOwner	
	Err	error	
	Uint8	event	

3.3.4.5 PhnMovedMsgsParamsType

Structure	Data Type	Members	Description/Comment
`PhnMovedMsgsParamsType`			
	UInt16	count	
	PhnMovedMsgDescType*	List	

3.3.4.6 PhnMsgBoxDataType

Structure	Data Type	Members	Description/Comment
`PhnMsgBoxDataType`			Message box indicator structure
	Boolean	indicatorOn	
	PhnMsgBoxType	type	
	Int16	messageCount	
	Int16	lineNumber	

3.3.4.7 PhnNBSNotificationEventType

Structure	Data Type	Members	Description/Comment
`PhnNBSNotificationEventType`			Structure passed to callbacks registered for incoming NBS notifications
	UInt16	version	Version number to provide future backwards compatibility
	Helper fields:		
	Boolean	NBSdatagram	Flag if it is an NBS datagram
	Boolean	binary	True if binary data
	void*	headerP	Pointer to raw header
	UInt8	headerLen	Length of `headerP`
	void*	dataP	Pointer to data body
	UInt8	dataLen	Length of `dataP`
	NBS datagram fields:		
	UInt8	refNum	NBS reference number
	UInt8	maxNum	Max segment number 1-255
	UInt8	seqNum	Sequence number 1-255, no more than `maxNum`
	Int8	reserved1	Padding
	UInt32	srcPort	Source port
	UInt32	dstPort	Destination port
	SMS related fields:		
	UInt32	msgID	ID into the SMS database to reference this message this ID is not guaranteed to be valid once the notification callback returns. Users should make a copy of the msg if they want to work on it after the callback returns.
	char*	senderP	Sender number - null terminated
	UInt32	datetime	Date/time stamp
	Int32	reserved2	Reserved
	Int32	reserved3	Reserved

3.3.4.8 PhnOperatorListPtr

3.3.4.9 PhnOperatorListType

Structure	Data Type	Members	Description/Comment
PhnOperatorListType PhnOperatorListPtr			Operator list
	short	count	
	PhnOperatorType	opData[1]	

3.3.4.10 PhnOperatorType

Structure	Data Type	Members	Description/Comment
PhnOperatorType			Operator info
	PhnOperatorStatus	Status;	
	GSMOperatorID	id	
	Char*	longname	
	Char*	shortname	

3.3.4.11 PhnPasswordEventType

Structure	Data Type	Members	Description/Comment
PhnPasswordEventType			
	PhnPasswordType	type	
	PhnPasswordType	prevType	
	Err	error	
	PhnPassword	pin	
	PhnPassword	Puk	

3.3.4.12 PhnPhoneBookInfoPtr & PhnPhoneBookInfoType

Structure	Data Type	Members	Description/Comment
PhnPhoneBookInfoType PhnPhoneBookInfoPtr			
	UInt16	firstEntry	
	UInt16	lastEntry	
	UInt16	maxNameLength	
	UInt16	maxNumberLength	

3.3.4.13 PhnPhoneEquipmentMode

Structure	Data Type	Members	Description/Comment
PhnPhoneEquipmentMode			
	long	mode	

3.3.4.14 PhnPowerEventType

Structure	Data Type	Members	Description/Comment
`PhnPowerEventType`			
	PhnPowerType	state	

3.3.4.15 PhnProgressEventType

Structure	Data Type	Members	Description/Comment
`PhnProgressEventType`			
	PhnProgressType	progress	
	PhnOpenDialogType	dialog	
	Uint32	data	Only for SMS progress

3.3.4.16 PhnRegistrationType

Structure	Data Type	Members	Description/Comment
`PhnRegistrationType`			
	PhnRegistrationStatus	status	

3.3.4.17 PhnRingingInfoPtr & PhnRingingInfoType

Structure	Data Type	Members	Description/Comment

`PhnRingingInfoType`
 `PhnRingingInfoPtr`			

	PhnRingingProfileType	Profiles [kSliderPositions]	

3.3.4.18 PhnRingingProfileType

Structure	Data Type	Members	Description/Comment
`PhnRingingProfileType`			Phone slider switch setting
	Uint32	ringerID	Unique ID of ringer record
	Uint16	volume	
	Boolean	vibrate	

3.3.4.19 PhoneGlobalsType

Structure	Data Type	Members	Description/Comment
`PhoneGlobalsType`	The structure contains phone state information that is shared between all phone applications. A pointer to this structure is saved to a system feature. The feature has a creator of: `hsFileTCardSetup` and a feature number of: `phoneGlobalsFeature`		
	Boolean	syncing	True if HotSync is running
	Boolean	activeCalls	Number of active voice call(??)

3.4 Library API

This section provides details about the various APIs that are found in the GSM/GPRS Library. Where appropriate, sample code is shown to demonstrate the use of the API. The application must have already loaded and opened the Library prior to making these calls.

3.4.1 PhnLibAddAddress

```
Err PhnLibAddAddress (Uint16 refNum, PhnAddressList list, PhnAddressHandle address)
```

Return Value:

Err – Error Code. 0 if no error

Parameters:

refNum – [IN] Library Reference Number

list – [IN] Address list

address – [IN] Address to add to the given list

Description:

Add the specified address to the address list.

3.4.2 PhnLibBattery

```
Err PhnLibBattery (UInt16 refNum, UInt16* battery)
```

Return Value:

Err – Error Code. 0 if no error

Parameters:

refNum – [IN] Library Reference Number

battery – [OUT] Battery level

Description:

Return the system battery level. Application developers should use the appropriate Palm OS routines to obtain the battery level.

3.4.3 PhnLibBatteryCharge

```
Err PhnLibBatteryCharge (Uint16 refNum, Uint16* charging)
```

Return Value:

Err – Error Code. 0 if no error

Parameters:

refNum – [IN] Library Reference Number

charging – [OUT] Charging indicator

Description:

Return the system charging state. Application developers should use the appropriate Palm OS routines to obtain the battery level.

3.4.4 PhnLibBoxInformation

```
Err PhnLibBoxInformation (Uint16 refNum, PhnMsgBoxDataType* data)
```

Return Value:

Err – Error Code. 0 if no error

Parameters:

refNum – [IN] Library Reference Number

data – [INOUT] Structure indicating the status of the message box.

Description:

Use this function to retrieve the status of a given message box.

Example:

```
PhnMsgBoxDataType msgBoxData;

// Check the status of the voice mailbox
msgBoxData.type = kBoxVoice;
msgBoxData.lineNumber = 0;
PhnLibBoxInformation (PhoneLibRefNum, &msgBoxData);
if (msgBoxData.indicatorOn)
  {
    // Display Voice Mail icon
  }
else
  {
    // Clear the voice mail icon
  }
```

3.4.5 PhnLibCardInfo

```
Boolean PhnLibCardInfo (UInt16 refNum, Char** manufacturer, Char** model, Char**
version, Char** serial)
```

Return Value:

Boolean – True if success. False if failure.

Parameters:

refNum – [IN] Library Reference Number

manufacturer – [OUT] Radio manufacture name. Set to NULL if not needed.

model – [OUT] Radio model name. Set to NULL if not needed.

version – [OUT] Radio Firmware revision. Set to NULL if not needed.

serial – [OUT] Radio serial number (IMEI) . Set to NULL if not needed.

Description:

Retrieve the various parameters of the radio.

3.4.6 PhnLibClose

```
Err PhnLibClose (UInt16 refNum)
```

Return Value:

Err – Error Code. 0 is no error

Parameters:

refNum – [IN] Library Reference Number

Description:

Close the Phone Library previously opened with PhnLibOpen

3.4.7 PhnLibConnectionAvailable

```
Boolean PhnLibConnectionAvailable (Uint16 refNum, PhnConnectionEnum connection)
```

Return Value:

Boolean – True if connection available

Parameters:

refNum – [IN] Library Reference Number

connection – [IN] Type of connection to check for (see PhnConnectionEnum)

Description:

Check if the specified connection type is available for use.

3.4.8 PhnLibCount

```
Err PhnLibCount (Uint16 refNum, PhnAddressList list, Uint16* count)
```

Return Value:

Err – Error Code. 0 if no error

Parameters:

refNum – [IN] Library Reference Number

list – [IN] Address list

count – [OUT] Length of list

Description:

Return the size of the address list.

3.4.9 PhnLibCurrentOperator

```
Err PhnLibCurrentOperator (Uint16 refNum, GSMOperatorID* id, Char** name,
GSMRegistrationMode* mode)
```

Return Value:

Err – Error Code. 0 if no error

Parameters:

refNum – [IN] Library Reference Number

id – [OUT] Operator ID

name – [OUT] Operator name. Returned string must be freed by the caller.

Mode – [OUT] Registration Mode (see GSMRegistrationMode).

Description:

Retrieve the operator ID and the network name that the user is registered to.

Example:

```
Char* CurrentOperator = NULL
GSMOperatorID id = 0;
PhnLibCurrentOperator (PhoneLibRefNum, &id, &CurrentOperator, NULL);
// Current Operator will contain the string of the network name. This can
be used for display purposes
// When done with the string free the memory
MemPtrFree (CurrentOperator);
CurrentOperator = NULL;
```

3.4.10 PhnLibCurrentOperatorID

```
Err PhnLibCurrentOperatorID (UInt16 refNum, char *buffer, Int16* bufferSizeP)
```

Return Value:

Err – Error Code. 0 if no error

Parameters:

refNum – [IN] Library Reference Number

buffer – [OUT] Buffer containing the result string

bufferSizeP – [OUT] Pointer to the buffer size

Description:

Retrieve the current Mobile Country Code (MCC) and Mobile Network Code (MNC) that the user is registered in. To determine if the user is roaming, compare this code with the Home ID.

3.4.11 PhnLibCurrentProvider

```
Err PhnLibCurrentProvider (UInt16 refNum, char** name)
```

Return Value:

Err – Error Code. 0 if no error

Parameters:

refNum – [IN] Library Reference Number

name – [OUT] Operator name. Returned string must be freed by the caller.

Description:

Return the string of the current service provider.

3.4.12 PhnLibErrorRate

```
Err PhnLibErrorRate (Uint16 refNum, Uint16* errorRate)
```

Return Value:

Err – Error Code. 0 if no error

Parameters:

refNum – [IN] Library Reference Number

errorRate – [OUT] Bit error rate

Description:

Return the bit error rate (BER) of the wireless channel. This value will range from 0 to 7. A value of 99 indicates an unknown or not detectable signal.

3.4.13 PhnLibFirstAppForService

```
UInt32 PhnLibFirstAppForService (UInt16 refNum, PhoneServiceClassType service)
```

Return Value:

UInt32 – Creator ID of the application. 0 if not found.

Parameters:

refNum – [IN] Library Reference Number

service – [OUT] Service to query for (see PhoneServiceClassType)

Description:

Return the Creator ID for the application registered for the given service.

3.4.14 PhnLibGetBoxNumber

`Err` **`PhnLibGetBoxNumber`** `(UInt16 refNum, PhnMsgBoxType type, UInt16 line, Char** number)`

Return Value:

Err – Error Code. 0 if no error

Parameters:

refNum – [IN] Library Reference Number

type – [IN] Message box to check

line – [IN] Line number to check

number – [OUT] Number of the given box.

Description:

Use this function to retrieve the phone number of the given message box. This can be used to retrieve the operator’s voice mail access number.

Example:

```
Char*          number = NULL;
// Get the voice mailbox number
PhnLibGetBoxNumber (PhoneLibRefNum, kBoxVoice, 0, &number);
// Display the phone number
// NOTE: Some SIMs will report a blank number as a ‘+’
…
// Once done, free the memory
MemPtrFree (number);
number = NULL;
```

3.4.15 PhnLibGetCLIP

`Err` **`PhnLibGetCLIP`** `(UInt16 refNum, Boolean* enabled)`

Return Value:

Err – Error Code. 0 if no error

Parameters:

refNum – [IN] Library Reference Number

enable – [OUT] True to enabled. False to disable

Description:

Retrieve the status of the CLIP mode.

3.4.16 PhnLibGetCLIR

```
Err PhnLibGetCLIR (UInt16 refNum, GSMDialCLIRMode* mode, PhnCLIRStatus* status)
```

Return Value:

Err – Error Code. 0 if no error

Parameters:

refNum – [IN] Library Reference Number

mode – [OUT] Current mode (see GSMDialCLIRMode)

status – [OUT] Current status (see PhnCLIRStatus)

Description:

Get the status of the CLIR mode.

3.4.17 PhnLibGetEchoCancellation

```
Err PhnLibGetEchoCancellation (Uint16 refNum, Boolean* echoCancellationOn)
```

Return Value:

Err – Error Code. 0 if no error

Parameters:

refNum – [IN] Library Reference Number

echoCancellationOn – [OUT] Echo cancellation mode

Description:

Get the echo cancellation mode.

3.4.18 PhnLibGetEquipmentMode

```
Err PhnLibGetEquipmentMode (Uint16 refNum, PhnEquipmentMode* equipmentMode)
```

Return Value:

Err – Error Code. 0 if no error

Parameters:

refNum – [IN] Library Reference Number

equipmentMode – [OUT] Equipment Mode (see PhnEquipmentMode)

Description:

Get the equipment mode of the device. This should be used to determine the state the hardware is in such as, lid opened, headset plugged in, etc.

3.4.19 PhnLibGetErrorText

```
Void PhnLibGetErrorText (Uint16 refNum, Err error, char* s, Uint16 sSize)
```

Return Value:

None.

Parameters:

refNum – [IN] Library Reference Number

error – [IN] Error code

s – [OUT] String

sSize – [IN] Size of String

Description:

Translate the given error code and return a text string containing the error message. Caller must allocate the message and specify the size of the string that is passed to the routine.

3.4.20 PhnLibGetField

```
Char* PhnLibGetField (Uint16 refNum, PhnAddressHandle address, PhnAddressField field)
```

Return Value:

Char* - Field value for the address

Parameters:

refNum – [IN] Library Reference Number

address – [IN] Handle to object query

field – [IN] Field to get. See PhnAddressField.

Description:

This function returns the field's value for a given address in a newly allocated block. The function returns 0 if there was an error while retrieving data. NOTE: The caller of this function must dispose of this block.

3.4.21 PhnLibGetLibAPIVersion

```
Err PhnLibGetLibAPIVersion (UInt16 refNum, UInt32* dwVerP)
```

Return Value:

Err – Error Code. 0 is no error

Parameters:

refNum – [IN] Library Reference Number

dwVerP – [OUT] The version number. Caller must allocate the storage

Description:

Returns the version number as a 32-bit unsigned integer.

3.4.22 PhnLibGetMicrophone

```
Err PhnLibGetMicrophone (Uint16 refNum, int* gain)
```

Return Value:

Err – Error Code. 0 if no error

Parameters:

refNum – [IN] Library Reference Number

gain – [OUT] Microphone gain

Description:

Get the microphone gain

3.4.23 PhnLibGetNth

```
Err PhnLibGetNth (UInt16 refNum, PhnAddressList list, int index, PhnAddressHandle*
address)
```

Return Value:

Err – Error Code. 0 if no error

Parameters:

refNum – [IN] Library Reference Number

list – [IN] Address list

index – [IN] Index to access

address – [OUT] Address to add to the given list

Description:

Extract the address from the address list at the specified index.

3.4.24 PhnLibGetOperatorList

```
Err PhnLibGetOperatorList (UInt16 refNum, PhnOperatorListPtr * list)
```

Return Value:

Err – Error Code. 0 if no error

Parameters:

refNum – [IN] Library Reference Number

list – [OUT] List of available operators

Description:

Return the list of all available network operators for a given area.

Example:

```
PhnOperatorListPtr   operatorList;
operatorList = NULL;
PhnLibGetOperatorList (PhoneLibRefNum, &operatorList);
// Using the returned list, we can now display the name of the operators
available to the user
…
// Once finished with the list, free the memory
MemPtrFree (operatorList);
operatorList = NULL;
```

3.4.25 PhnLibGetOperatorLock

```
Err PhnLibGetOperatorLock (UInt16 refNum, UInt16 facilityType, Boolean* enabled)
```

Return Value:

Err – Error Code. 0 if no error

Parameters:

refNum – [IN] Library Reference Number

facilityType– [IN] Lock mode. Can be either GSMLockSelectorOperatorLock or GSMLockSelectorProviderLock enabled – [OUT] True to lock. False to unlock

Description:

Query the lock status.

3.4.26 PhnLibGetOwnNumbers

```
Err PhnLibGetOwnNumbers (Uint16 refNum, PhnAddressList* ownNumbers)
```

Return Value:

Err – Error Code. 0 if no error

Parameters:

refNum – [IN] Library Reference Number

ownNumbers – [OUT] List of addresses

Description:

Retrieve the user's phone number from the SIM card. Note that not all the SIM cards store the user's number. The possible number returned are Voice1,Voice2, Data, and Fax.

Example:

```
PhnAddressList      list;
PhnAddressHandle    address;
Char*               number;

number = NULL;
// Get the list of addresses stored
PhnLibGetOwnNumbers (PhoneLibRefNum, &list);
// Extract the address from the address list
PhnLibGetNth (PhoneLibRefNum, list, 1, &address);
// Get the string containing the phone number
number = PhnLibGetField (PhoneLibRefNum, address, phnAddrFldPhone);
MemHandleFree (address);
// Number can be used for display
// Free the results
MemPtrFree (number);
number = NULL;
```

3.4.27 PhnLibGetPhoneBook

```
Err PhnLibGetPhoneBook (UInt16 refNum, PhnAddressList* numbers, PhnPhoneBookInfoPtr
info)
```

Return Value:

Err – Error Code. 0 if no error

Parameters:

refNum – [IN] Library Reference Number

numbers – [OUT] List of addresses

info – [OUT] Phone Book info

Description:

Retrieve the phone book that is stored on the SIM.

3.4.28 PhnLibGetPhoneBookIndex

```
Uint32 PhnLibGetPhoneBookIndex (Uint16 refNum, PhnAddressHandle address)
```

Return Value:

Uint32 – Index to the address entry

Parameters:

refNum – [IN] Library Reference Number

address – [OUT] Handle to the address structure

Description:

Get the index of the given address.

3.4.29 PhnLibGetPhoneCallStatus

```
Err PhnLibGetPhoneCallStatus (UInt16 refNum, UInt32* phnFlags)
```

Return Value:

Err – Error Code. 0 if no error

Parameters:

refNum – [IN] Library Reference Number

phnFlags – [OUT] Phone Status flags. Caller must allocate the storage.

Description:

Query the phone status. The flags that are returned can be one of the following values OR'd together:

phnVoiceCall1Active	There is a voice call active on line1
phnVoiceCall2Active	There is a voice call active on line2
phnCSDCallActive	There is a data call currently active
phnGPRSCallActive	There is a GPRS session active

3.4.30 PhnLibGetRingingInfo

```
Err PhnLibGetRingingInfo (UInt16 refNum, PhnRingingInfoPtr info)
```

Return Value:

Err – Error Code. 0 if no error

Parameters:

refNum – [IN] Library Reference Number

info – [OUT] The PhnRingingInfo Structure

Description:

Return the PhnRingingInfo structure from the library. This structure defines the alert information on the different ringer switch positions. The structure will contains the tone, volume, and vibrate mode information on each position.

3.4.31 PhnLibGetSIMStatus

```
GSMSIMStatus PhnLibGetSIMStatus (UInt16 refNum)
```

Return Value:

GSMSIMStatus – Status message. See reference to GSMSIMStatus

Parameters:

refNum – [IN] Library Reference Number

Description:

Get the SIM status of the device.

3.4.32 PhnLibGetSMSGateway

```
Err PhnLibGetSMSGateway (UInt16 refNum, char** smsGateway)
```

Return Value:

Err – Error Code. 0 if no error

Parameters:

refNum – [IN] Library Reference Number

smsGateway – [OUT] Handle to the number string

Description:

Retrieve the SMS Gateway number from the device. The return parameter is a handle to the string. Caller must free the pointer that is returned.

NOTE: The Trap number for this API has changed from (sysLibTrapCustom + 128) to (sysLibTrapCustom + 130) when GPRS and 1xRTT was introduced. So if you’re using the current header files, this call will only work if the communicator has been updated to GPRS.

Example:

The following code sample demonstrates how to retrieve the SMS Gateway number from the library.

```
Err    error = 0;
char** smsGateway = NULL;
// Assume that the library is loaded and reference number is stored in
smsLibRef
error = PhnLibGetSMSGateway (smsLibRef, smsGateway);
if (!error && *smsGateway)
  {
    // Copy the string to a local variable or other storage location
    StrNCopy(smsEMailNum, *smsGateway, kMaxSMSCSize);
    if (smsGateway)
      MemPtrFree (smsGateway);
```

3.4.33 PhnLibGetSMSRingInfo

```
Err PhnLibGetSMSRingInfo (UInt16 refNum, PhnRingingInfoPtr info)
```

Return Value:

Err – Error Code. 0 if no error

Parameters:

refNum – [IN] Library Reference Number’

info – [OUT]The PhnRingingInfo Structure

Description:

Return the PhnRingingInfo structure from the library for SMS messages. The structure will contains the tone, volume, and vibrate mode information on each position.

3.4.34 PhnLibGetToneIDs

```
Err PhnLibGetToneIDs (Uint16 refNum, Uint32** list, int* listLength)
```

Return Value:

Err – Error Code. 0 if no error

Parameters:

refNum – [IN] Library Reference Number

list – [OUT] Handle to the tone list

listLength – [OUT] Length of the list

Description:

Return an array of all the unique record Ids (UID) of all the tones in the global ring tone list.

3.4.35 PhnLibGetToneName

```
Err PhnLibGetToneName (UInt16 refNum, UInt16 toneIndex, char* name, short maxLength)
```

Return Value:

Err – Error Code. 0 if no error

Parameters:

refNum – [IN] Library Reference Number

name – [OUT] name of the tone

maxLength – [IN] Maximum length of the tone name

Description:

Return the name of the tone as specified by the toneIndex. The toneIndex is the index number of the tone in the array retrieved by PhnGetToneIDs. It is not the UID of the tone.

3.4.36 PhnLibGetVolume

```
Err PhnLibGetVolume (Uint16 refNum, int* volume)
```

Return Value:

Err – Error Code. 0 if no error

Parameters:

refNum – [IN] Library Reference Number

volume – [OUT] Speaker volume

Description:

Get the current volume level. Speaker volume ranges from phnVolumeMin to phnVolumeMax.

3.4.37 PhnLibHomeOperatorID

`Err` **`PhnLibHomeOperatorID`** `(Uint16 refNum, char` **`buffer, Int16`** `bufferSizeP)`

Return Value:

Err – Error Code. 0 if no error

Parameters:

refNum – [IN] Library Reference Number

buffer – [OUT] Buffer containing the result string

bufferSizeP – [OUT] Pointer to the buffer size

Description:

Retrieve the Mobile Country Code (MCC) and Mobile Network Code (MNC) for the user’s home country and network.

Example:

```
Int16        bufferSize;
Char         mccmncP[] = “cccnnn ”;
// Get the size of the string that will be returned
PhnLibHomeOperatorID (PhoneLibRefNum, NULL, &bufferSize);
// Take into account the NULL terminating character in the string
++bufferSize;
PhnLibHomeOperatorID (PhoneLibRefNum, mccmncP, &bufferSize);
// The string, mccmncP will contain the MCC/MNC pair. The first three
characters will be the MCC and the next two or three characters will be
the MNC.
```

3.4.38 PhnLibIsLegalCharacter

`Boolean` **`PhnLibIsLegalCharacter`** `(Uint16 refNum, char c)`

Return Value:

Boolean – True if character is legal. False if not.

Parameters:

refNum – [IN] Library Reference Number

c – [IN] Character to check

Description:

Check if the given character is legal for the GSM Character set.

3.4.39 PhnLibLength

```
int **PhnLibLength** (UInt16 refNum, const char* text, Boolean inMessages, Boolean
substitution)
```

Return Value:

Count of given string

Parameters:

refNum – [IN] Library Reference Number

text – [OUT] String to count

inMessage – [IN]

substitution – [IN]

Description:

Return the length of the string.

3.4.40 PhnLibModuleButtonDown

```
Boolean PhnLibModuleButtonDown (Uint16 refNum, PhnModuleButtonType button)
```

Return Value:

Boolean – True if pressed. False if not pressed

Parameters:

refNum – [IN]Library Reference Number

button – [IN] Button to check (see PhnModuleButtonType)

Description:

Check if given button is pressed.

3.4.41 PhnLibModulePowered

```
Boolean PhnLibModulePowered (UInt16 refNum)
```

Return Value:

```
Boolean                 True if on. False if off.
```

Parameters:

```
refNum      [IN]        Library Reference Number
```

Description:

Check if the radio is powered on or not.

3.4.42 PhnLibNetworkAvailable

```
Boolean PhnLibNetworkAvailable (UInt refNum)
```

Return Value:

Boolean – True if on. False if off.

Parameters:

refNum – [IN] Library Reference Number

Description:

3.4.43 PhnLibNewAddress

```
PhnAddressHandle PhnLibNewAddress (UInt16 refNum, const char* number, PhnDatabaseID
id)
```

Return Value:

PhnAddressHandle – Handle to the new address that was allocated or 0 if there was an error

Parameters:

refNum – [IN] Library Reference Number

number – [IN] Address (phone number) to use

id – [IN] id

Description:

This function creates a new address and fills in the information given in number and id. In the case of an address for an SMS, the id should be set to `phnLibUnknownID`.

3.4.44 PhnLibNewAddressList

```
PhnAddressList PhnLibNewAddressList (UInt16 refNum)
```

Return Value:

PhnAddressList – Error Code. 0 if no error

Parameters:

refNum – [IN] Library Reference Number

Description:

Create a new address list

3.4.45 PhnLibOpen

```
Err PhnLibOpen (UInt16 refNum)
```

Return Value:

Err – Error Code 0 is no error

Parameters:

RefNum – Library Reference Number

Description:

Open the Phone Library. This call should be made after loading the library, as shown above.

3.4.46 PhnLibPlayDTMF

```
Err PhnLibPlayDTMF (Uint16 refNum, Char* sequence)
```

Return Value:

Err – Error Code. 0 if no error

Parameters:

refNum – [IN] Library Reference Number

sequence – [IN] Digits to play

Description:

Play the DTMF tones corresponding to the sequence passed in. The tone will be played through the phone speaker. The valid digits are from ‘0’ to ‘9’ and ‘A’ to ‘D’.

3.4.47 PhnLibPlayTone

```
Err PhnLibPlayTone (Uint16 refNum, Uint32 tone, int volume)
```

Return Value:

Err – Error Code. 0 if no error

Parameters:

refNum – [IN] Library Reference Number

tone – [IN] Unique Record ID (UID) of the tone

volume – [IN] Volume to play the tone (see PhnRingerVolumeType)

Description:

Play the tone as specified by the tone’s UID at the given volume.

3.4.48 PhnLibRegister

```
Err PhnLibRegister (Uint16 refNum, Uint32 creator, Uint16 services)
```

Return Value:

Err – Error Code. 0 is no error. GsmErrUnknownApp when the application cannot be found.

Parameters:

refNum – [IN] Library Reference Number

creator – [IN] The creator ID of the calling application

services – [IN] The services that the application is registering for. See GSMEventClass for more information.

Description:

Register an application with the GSM library for the specified set of services. After registration, the GSM library will send out the corresponding events via the specific launch code. If the application wishes to un-register, it should pass a 0 for the services.

3.4.49 PhnLibRegistered

```
Boolean PhnLibRegistered (Uint16 refNum)
```

Return Value:

Boolean – True if registered. False if not registered

Parameters:

refNum – [IN] Library Reference Number

Description:

Check if the phone is registered on a network (local or roaming).

3.4.50 PhnLibRoaming

```
Boolean PhnLibRoaming (UInt16 refNum)
```

Return Value:

Boolean – True if roaming. False if not roaming.

Parameters:

refNum – [IN] Library Reference Number

Description:

Check if the phone is roaming.

3.4.51 PhnLibSendDTMF

```
Err PhnLibSendDTMF (UInt16 refNum, Char* sequence)
```

Return Value:

Err – Error Code. 0 if no error.

Parameters:

refNum – [IN] Library Reference Number

sequence – [IN] Sequence to send

Description:

Send the DTMF tones corresponding to the sequence passed in to the receiving end. The tone will be played through the phone speaker. The valid digits are from ‘0’ to ‘9’ and ‘A’ to ‘D’. The function can be used to control DTMF based systems such as voice mail systems.

3.4.52 PhnLibSendSilentDTMF

```
Err PhnLibSendSilentDTMF (UInt16 refNum, Char* sequence)
```

Return Value:

Err – Error Code. 0 if no error.

Parameters:

refNum – [IN] Library Reference Number

sequence – [IN] Sequence to send

Description:

Send the DTMF tones corresponding to the sequence passed in to the receiving end. The tone will **not** be played through the phone speaker. The valid digits are from ‘0’ to ‘9’ and ‘A’ to ‘D’. The function can be used to control DTMF based systems such as voice mail systems.

3.4.53 PhnLibSetEchoCancellation

```
Err PhnLibSetEchoCancellation (UInt16 refNum, Boolean echoCancellationOn)
```

Return Value:

Err – Error Code. 0 if no error

Parameters:

refNum – [IN] Library Reference Number

echoCancellationOn – [IN] Echo cancellation mode

Description:

Set the echo cancellation mode.

3.4.54 PhnLibSetEquipmentMode

```
Err PhnLibSetEquipmentMode (UInt16 refNum, PhnEquipmentMode equipmentMode)
```

Return Value:

Err – Error Code. 0 if no error

Parameters:

refNum – [IN] Library Reference Number

equipmentMode – [IN] Equipment Mode (see PhnEquipmentMode)

Description:

Set the equipment mode of the device. This function is used to switch between speakerphone and headset mode.

Example:

```
// Turn on Speaker Phone
PhnLibSetEquipmentMode (PhoneLibRefNum, phnSpeakerPhoneMode);
// Turn off Speaker Phone.
// NOTE: Speakerphone mode must be disabled when all calls
// are disconnected.
PhnLibSetEquipmentMode (PhoneLibRefNum, phnHandsetMode);
```

3.4.55 PhnLibSetField

```
Err PhnLibSetField (UInt16 refNum, PhnAddressHandle address, PhnAddressField field,
Char* data)
```

Return Value:

Err – Error Code. 0 if no error

Parameters:

refNum – [IN] Library Reference Number

address – [IN] Handle to the address structure

field – [IN] Field. See PhnAddressField

data – [IN] Data to set

Description:

Set the given field, data, in the address structure.

3.4.56 PhnLibSetMicrophone

```
Err PhnLibSetMicrophone (UInt16 refNum, int gain)
```

Return Value:

Err – Error Code. 0 if no error

Parameters:

refNum – [IN] Library Reference Number

gain – [IN] Microphone gain

Description:

Set the microphone gain

3.4.57 PhnLibSetModulePower

```
Err PhnLibSetModulePower (UInt16 refNum, Boolean On)
```

Return Value:

Err – Error Code. 0 if no error

Parameters:

refNum – [IN] Library Reference Number

On – [IN] True to turn on the radio. False to turn off radio

Description:

This will control the radio power. Using this function is equivalent to pressing and holding the power button to toggle the radio power.

3.4.58 PhnLibSetOperator

```
Err PhnLibSetOperator      ( Uint                    refNum,
                             PhnOperatorType*        op,
                             GSMRegistrationMode     regMode)
```

Return Value:

Err – Error Code. 0 if no error

Parameters:

```
refNum       – [IN]     Library Reference Number
op           – [IN]     Selected operator to register to
regMode      – [IN]
```

Description:

Change the current registration to the given operator. An application would use the results from PhnLibGetOperatorList to select the new operator.

3.4.59 PhnLibSetOperatorLock

```
Err PhnLibSetOperatorLock (Uint16 refNum, Uint16 facilityType, Boolean enable, Char*
password)
```

Return Value:

Err – Error Code. 0 if no error

Parameters:

refNum – [IN] Library Reference Number

facilityType– [IN] Lock mode. Can be either GSMLockSelectorOperatorLock or GSMLockSelectorProviderLock

enable – [IN] True to lock. False to unlock

password – [IN] Password for lock/unlock operation

Description:

Set (lock or unlock) the phone with a lock facility.

3.4.60 PhnLibSetPhoneBook

```
Err PhnLibSetPhoneBook (Uint16 refnum, PhnAddressList numbers)
```

Return Value:

Err – Error Code. 0 if no error

Parameters:

refNum – [IN] Library Reference Number

numbers – [IN] List of addresses

Description:

Commit the given phonebook to the SIM.

3.4.61 PhnLibSetRingingInfo

`Err` **`PhnLibSetRingingInfo`** `(Uint16 refNum, const PhnRingingInfoPtr info)`

Return Value:

Err – Error Code. 0 if no error

Parameters:

refNum – [IN] Library Reference Number

info – [IN] The PhnRingingInfo Structure

Description:

Set the ring info parameters for the system. This is equivalent to using the Call Tone in the Ringer preference panel.

3.4.62 PhnLibSetSMSRingInfo

`Err` **`PhnLibSetSMSRingInfo`** `(Uint16 refNum, const PhnRingingInfoPtr info)`

Return Value:

Err – Error Code. 0 if no error

Parameters:

refNum – [IN] Library Reference Number

info – [IN] The PhnRingingInfo Structure

Description:

Set the ring info parameters for SMS messages. This is equivalent to using the SMS Tone in the Ringer preference panel.

3.4.63 PhnLibSetVolume

`Err` **`PhnLibSetVolume`** `(UInt16 refNum, int volume)`

Return Value:

Err – Error Code. 0 if no error

Parameters:

refNum – [IN] Library Reference Number

volume – [IN] Speaker volume

Description:

Set the current volume level. Speaker volume ranges from phnVolumeMin to phnVolumeMax.

3.4.64 PhnLibSignalQuality

```
Err PhnLibSignalQuality (Uint16 refNum, Uint16* quality)
```

Return Value:

Err – Error Code. 0 if no error

Parameters:

refNum – [IN] Library Reference Number

quality – [OUT] Signal quality

Description:

Return the numerical value of the signal quality. This value will range from 0 to 31. A value of 99 indicates an unknown signal strength or not detectable signal.

3.4.65 PhnLibSIMInfo

```
Boolean PhnLibSIMInfo (Uint16 refNum, Char** imsi)
```

Return Value:

Boolean – True if success. False if failure.

Parameters:

refNum – [IN] Library Reference Number

imsi – [OUT] IMSI number

Description:

Return the International Mobile Subscriber Identity (IMSI) number from the SIM card.

3.4.66 PhnLibSleep

```
Err PhnLibSleep (Uint16 refNum)
```

Return Value:

Err – Error Code. 0 is no error

Parameters:

refNum – [IN] Library Reference Number

Description:

Puts the Phone Library to sleep

3.4.67 PhnLibStartVibrate

```
Err PhnLibStartVibrate (Uint16 refNum, Boolean pulse, Boolean repeat)
```

Return Value:

Err – Error Code. 0 if no error

Parameters:

refNum – [IN] Library Reference Number

pulse – [IN] Pulse motor

repeat – [IN] Repeat sequence

Description:

Start the vibrate motor. NOTE: Developers should use the HsIndicator function call instead.

3.4.68 PhnLibStopTone

```
Err PhnLibStopTone (UInt16 refNum)
```

Return Value:

Err – Error Code. 0 if no error

Parameters:

refNum – [IN] Library Reference Number

Description:

Stop the current tone playback.

3.4.69 PhnLibStopVibrate

```
Err PhnLibStopVibrate (UInt16 refNum)
```

Return Value:

Err – Error Code. 0 if no error

Parameters:

refNum – [IN] Library Reference Number

Description:

Stop the vibrate motor. NOTE: Developers should use the HsIndicator function call instead.

3.4.70 PhnLibUsableSignalStrengthThreshold

```
Err PhnLibUsableSignalStrengthThreshold (UInt16 refNum, UInt16* threshold)
```

Return Value:

Err – Error Code. 0 if no error

Parameters:

refNum – [IN] Library Reference Number

threshold – [OUT] Signal threshold

Description:

Return the threshold level of signal quality at which the radio is capable of sending data.

3.4.71 PhnLibWake

```
Err PhnLibWake (UInt16 refNum)
```

Return Value:

Err – Error Code. 0 is no error

Parameters:

refNum – [IN] Library Reference Number

Description:

Wake the Phone Library

3.5 Error Codes

3.5.1 Phone Library Error Codes

Error Code	Definition
phnErrSerLibAlreadyOpen	
phnErrAlreadyOpen	
phnErrNotOpen	
phnErrStillOpen	
phnLibErrBufferTooSmall	
phnLibCardNotFound	
phnErrUnknownID	
phnErrParseError	
phnErrIntermediateResult	
phnErrIncorrectPassword	

3.5.2 General Error Codes

Error Code	Definition
gsmErrParam	
gsmErrUnknownError	
gsmErrNoResponse	
gsmErrNotOpen	
gsmErrStillOpen	
gsmErrMemory	
gsmErrUnknownID	
gsmErrNoPower	
gsmErrNoNetwork	
gsmErrNoConnection	
gsmErrNotAllowed	
gsmErrIllegalFacility	
gsmErrIllegalCondition	
gsmErrIllegalStatus	
gsmErrIllegalIndex	
gsmErrIllegalChars	
gsmErrIllegalMsg	
gsmErrIllegalType	
gsmErrIllegalNumber	
gsmErrTimeout	
gsmErrUnknownApp	
gsmErrUnknownNumber	
gsmErrBufferTooSmall	
gsmErrPasswordRequired	
gsmErrResponsePending	

Error Code	Definition
gsmErrCancelled	
gsmErrNoRecipient	

3.5.3 Errors defined in the GSM recommendations

Error Code	Definition
gsmErrPhoneFailure	
gsmErrPhoneNotConnected	
gsmErrPhoneAdaptorLinkReserved	
gsmErrNotSupported	
gsmErrPhPINRequired	
gsmErrPhFPINRequired	
gsmErrPhFPUKRequired	
gsmErrNoSIM	
gsmErrPINRequired	
gsmErrPUKRequired	
gsmErrSIMFailure	
gsmErrSIMBusy	
gsmErrSIMWrong	
gsmErrIncorrectPassword	
gsmErrPIN2Required	
gsmErrPUK2Required	
gsmErrMemoryFull	
gsmErrInvalidMemIndex	
gsmErrNotFound	
gsmErrMemFailure	
gsmErrStringTooLong	
gsmErrInvalidTextChars	
gsmErrDialStringTooLong	
gsmErrInvalidDialChars	
gsmErrNoNetworkService	
gsmErrNetworkTimeout	
gsmErrNetworkNotAllowed	
gsmErrNetPINRequired	
gsmErrNetPUKRequired	
gsmErrNetSubPINRequired	
gsmErrNetSubPUKRequired	
gsmErrSPPINRequired	
gsmErrSPPUKRequired	
gsmErrCorpPINRequired	
gsmErrCorpPUKRequired	
gsmErrIllegalMS	
gsmErrIllegalME	

Error Code	Definition
gsmErrGPRSNotAllowed	
gsmErrPLMNNotAllowed	
gsmErrLocAreaNotAllowed	
gsmErrRoamingNotAllowed	
gsmErrOptionNotSupported	
gsmErrReqOptionNotSubscribed	
gsmErrOptionTempOutOfOrder	
gsmErrUnspecifiedGPSRError	
gsmErrAuthenticationFailure	
gsmErrInvalidMobileClass	
gsmErrUnassignedNumber	
gsmErrOperDeterminedBarring	
gsmErrCallBarred	
gsmErrSMSXferRejected	
gsmErrDestOutOfService	
gsmErrUnidentifedSubscriber	
gsmErrFacRejected	
gsmErrUnknownSubscriber	
gsmErrNetworkOutOfOrder	
gsmErrTemporaryFailure	
gsmErrCongestion	
gsmErrResourcesUnavailable	
gsmErrReqFacNotSubscribed	
gsmErrReqFacNotImplemented	
gsmErrInvalidSMSReference	
gsmErrInvalidMsg	
gsmErrInvalidMandInfo	
gsmErrMsgTypeNonExistent	
gsmErrMsgNoCompatible	
gsmErrInfoElemNonExistent	
gsmErrProtocolError	
gsmErrInterworking	
gsmErrTelematicIWNotSupported	
gsmErrSMType0NotSupported	
gsmErrCannotReplaceMsg	
gsmErrUnspecifiedTPPIDError	
gsmErrAlphabetNotSupported	
gsmErrMsgClassNotSupported	
gsmErrUnspecifiedTPDCSError	
gsmErrCmdCannotBeActioned	
gsmErrCmdUnsupported	
gsmErrUnspecifiedTPCmdError	

Error Code	Definition
`gsmErrTPDUNotSupported`	
`gsmErrSCBusy`	
`gsmErrNoSCSubscription`	
`gsmErrSCSystemFailure`	
`gsmErrInvalidSMEAddr`	
`gsmErrDestSMEBarred`	
`gsmErrSMRejectedDuplicate`	
`gsmErrTPVPFNotSupported`	
`gsmErrTPVPNotSupported`	
`gsmErrSMSStorageFull`	
`gsmErrNoSMSStorage`	
`gsmErrErrorInMS`	
`gsmErrSIMApplToolkitBusy`	
`gsmErrMEFailure`	
`gsmErrSMSServReserved`	
`gsmErrInvalidParameter`	
`gsmErrFiller`	
`gsmErrFiller2`	
`gsmErrFiller3`	
`gsmErrMemoryFailure`	
`gsmErrSCAddrUnknown`	
`gsmErrNoCNMAAckExpected`	

3.5.4 Errors returned by the firmware (NO CARRIER)

Error Code	Definition
`gsmErrFDNMismatch`	
`gsmErrEmergencyCallsOnly`	
`gsmErrACMLimitExceeded`	
`gsmErrHoldError`	
`gsmErrNumberBlacklisted`	
`gsmErrLidClosed`	

3.5.5 Errors for SIM Application Toolkit

Error Code	Definition
`gsmErrSATUnavailable`	
`gsmErrSATInactive`	
`gsmErrUNUSED`	

3.5.6 Library loading errors

Error Code	Definition
`gsmErrRadioNotAvailable`	

3.5.7 Used internally

Error Code	Definition
gsmErrReserved_408b	
gsmErrReserved_408c	
gsmErrReserved_408d	

3.5.8 Firmware boot synchronization

Error Code	Definition
gsmErrFirmwareBootNotInprogress	
gsmErrFirmwareBootInprogress	

3.5.9 Direct Radio Error Codes

Error Code	Definition
gsmErrMMFailed	
gsmErrLowerLayer	
gsmErrCPError	
gsmErrCommandInProgress	
gsmErrSATNotSupported	
gsmErrSATNoInd	
gsmErrNeedResetModule	
gsmErrCOPSAbort	