IVT BlueSoleil™ Software Development Kit

Developer Guide

Version 2.1.3

IVT Corporation

4/F, Fazhan Plaza,
NO. 12, Xinxi Road,
Haidian District,
Beijing, 100085
P.R. China

Tel: +86 10 82898230
Fax: +86 10 62963059

www.ivtcorporation.com
www.bluesoleil.com

Revision History

Version	Date	Comments
1.0 Release	Jan. 18th, 2008	Initial version.
1.0.1	Jan. 24th, 2008	Added HFP/HSP Audio Gateway APIs and structures.
1.1.0 Alpha	Jan. 31st, 2008	Added Hands-free Unit/Headset APIs and structures.
1.1.1	Mar. 24th, 2008	Added HFP/HSP Audio Gateway sample. Added HFP/HSP Device sample. Added HSP/HFP specific event.
1.1.2	Apr. 2nd, 2008	Added call back functions of pairing and authentication. Added call back relevant macros.
1.1.3	Apr. 7th, 2008	Added pairing APIs and relevant macros.
1.1.4	Apr. 15th, 2008	Added call back functions of pairing and authentication. Added releavant call back events. Added Bluesoleil activating API. Added sample code of pairing and authentication.
1.1.5	May. 14th, 2008	Added Hands-free Unit/Headset APIs about wavein/waveout device configuration.
1.1.6	May. 23rd, 2008	Added Btsdk_AGAP_SetDialHandlerFlag function. Modified the description of the callback event BTSDK_APP_EV_AGAP_HF_LASTNUM_REDIAL_IND.
1.1.7	Jun. 5th, 2008	Added offline activation APIs for BlueSoleil 6.x.
2.0.0	Oct. 15th, 2008	Updated Hands-free/Headset profile from Version 1.1 to Version 1.5. Added APIs about setting and getting fixed pincode of local device. Modified the description of the API: Btsdk_GetAvailableExtSPPCOMPort
2.0.1	Feb. 26th, 2009	Added SPP Profile sample application code.
2.0.2	Apr. 10th, 2009	Added A2DP Profile APIs.
2.0.4	Sept. 22 nd, 2009	Modified the API: Btsdk_GetRemoteRSSI
2.0.5	Nov. 27 th, 2009	Added HID Profile structures and APIs.
2.0.6	Aug. 26th, 2010	Added local SPP service APIs
2.0.7	Sep. 8th, 2010	Added link key APIs.
2.0.8	Sep. 13th, 2010	Added PBAP Profile APIs and sample code

2.0.9	Oct. 13th, 2010	Added MAP profile APIs
2.1.0	Nov. 5th, 2012	Added BLE GATT APIs
2.1.1	Feb. 28th, 2013	Added AVRCP1.4 APIs Change Btsdk_EnumAVDriver to Btsdk_EnumAudioDriver
2.1.2	Mar. 11th, 2013	Modify AVRCP event changed notification from XXX_RSP to XXX_NOTIF. Added MAP Server APIs
2.1.3	Jun.8 th, 2013	Delete some AVRCP TG Sample codes.

Contents

1. Introduction

1.1 Purpose

This document is a developer guide to IVT BlueSoleil™ Software Development Kit, which describes the detailed information of IVT BlueSoleil™ APIs on Microsoft Windows platforms.

1.2 Audience

This document is intended for general VC++ developers involved in the production of VC++ applications for the IVT BlueSoleil™.

1.3 Reference

Reference	Link
Bluetooth Official	http://www.bluetooth.org
IVT BlueSoleil	http://www.ivtcorporation.com http://www.Bluesoleil.com
IVT BlueSoleil SDK Free Download	http://www.bluesoleil.com/products/index.asp?topic=bluesoleilapi

1.4 Abbreviations and Acronyms

Acronyms	Description
ACL	Asynchronous Connectionl-Less
AG	Audio Gateway
API	Application Program Interface
AVRCP	Audio/Video Remote Control Profile
BIP	Basic Imaging Profile
BPP	Basic Printing Profile
CTP	Cordless Telephony Profile
DUN	Dial-up Networking Profile
EC	Echo Canceling
FTP	File Transfer Profile
HSP	Headset Profile
HF	Hands-Free Unit
HFP	Hands-free Profile
HID	Human Interface Device
HS	Head Set
ICP	Intercom Profile

LAP	LAN Access Profile
LMP	Link Management Protocol
NR	Noise Reduction
OBEX	Object Exchange
OPP	Object Push Profile
OS	Operation System
PAN	Personal Area Networking Profile
REF	BIP Referenced Objects
RFCOMM	Radio Frequency Communication Protocol
SA	Service Atrribute
SCO	Synchronous Connection-Oriented
SDAP	Service Discovery Application Profile
SDK	Software Development Kit
SDM	Service Database Management
SDP	Service Discovery Profile
SNK	Audio Sink
SPP	Serial Port Profile
SRC	Audio Source
SS	Service Search
SSA	Service Search Attribute
UUID	Universally Unique Identifier

2. Claim

The contents contained in this developer guide are provided “AS IS”, except as required by applicable law, no warranties of any kind, either express or implied, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose, are made in relation to the accuracy, reliability, or contents of this guide. In no event, shall IVT be liable for direct, indirect, incidental and consequential damages, including but not limited to, losses of profits and/or data, in connection with or rising out of the use of SDK API, even if IVT has been advised of the possibility of such damages.

3. Overview

3.1 Supported Protocols and Profiles

IVT BlueSoleil™ SDK provides Bluetooth application developers with APIs allowing direct access to the protocols and profiles listed as follows:

- GAP
- SDP
- SPP
- OBEX
- FTP
- PBAP
- OPP
- DUN
- FAX
- PAN
- AVRCP
- A2DP
- HFP/HSP
- HID

3.2 SDK Components

The SDK consists of:

- C++ header files
- Library files
- Source and project files for sample applications
- Document of SDK developer guide (this document)
- Document of SDK sample application instruction

3.3 System Requirements

The operational context for the SDK is a standard Bluetooth PC platform on which IVT BlueSoleil is installed.

Recommended Hardware Requirements

- CPU: 600MHz or above
- RAM: 128M or above
- Free hard disk space: At least 20MB

Software Requirements

- OS: Windows2000 / Windows XP / Windows Vista

- IDE: Microsoft Visual C++ 6.0 / Visual Studio 6.0 / Visual Studio .net 2003 / Visual Studio .net 2005

Correlative Requirements

- To use this SDK IVT BlueSoleil version 6.4 or above is required
- Bluetooth radio device (Integrated or Bluetooth Dongle)

4. API Abstract

IVT BlueSoleil™ API is the interface exported by IVT BlueSoleil™. The intention of this SDK is to relieve the Application from managing the Bluetooth related components and make the Application light load. It is used to access the Bluetooth profiles from the application level software. It allows for:

- Standardized access to Bluetooth links.
- Supporting applications that implement different Bluetooth profiles.
- Writing portable applications to be used on different hardware and operating system platforms.
- Future expansions or hardware changes will not affect applications that use this interface.

To use the BlueSoleil™ API only a limited knowledge of Bluetooth basic principles and profile specifications is necessary. Therefore this document is not intended to be a Bluetooth profile tutorial.

The general structure of BlueSoleil SDK is shown in **Figure 1**. BlueSoleil SDK is between the Application and profile/stack. It wraps the various APIs of Bluetooth profiles protocol stack and provides the Application with clean APIs. The key component is a core manager and a profile manager with the following tasks:

- Store Bluetooth device information, including security-related information on devices.
- Store Bluetooth service information, including security-related information on devices.
- Store active connection information.
- Provide access to different Bluetooth profiles.

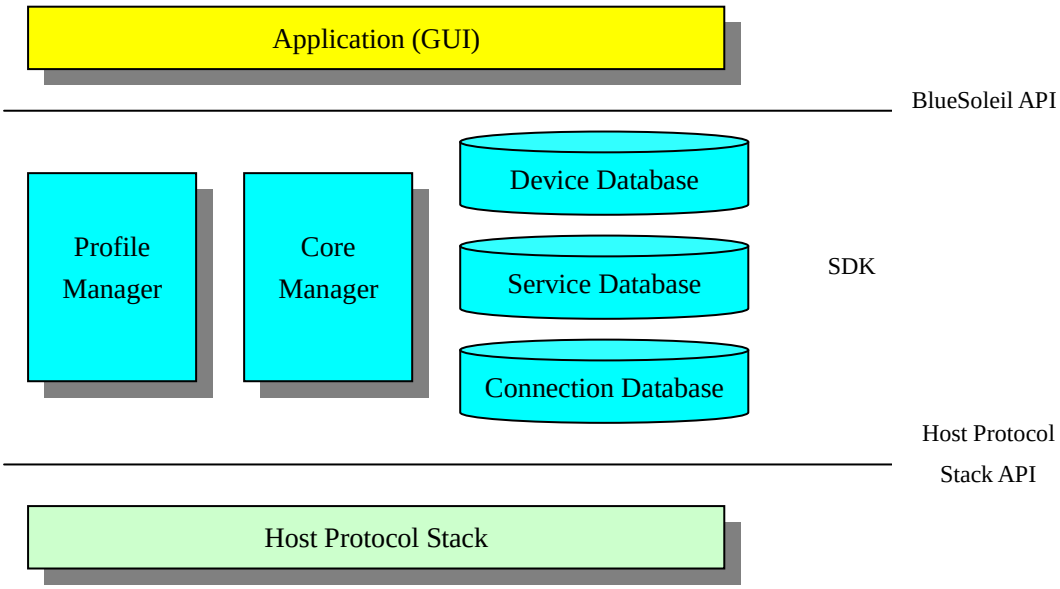

Figure 1: IVT BlueSoleil SDK Structure

The SDK maintains a list of remote devices, local services, remote services and active connections. Application can access these objects through a unique handle. The SDK can automatically store and recover information of these objects and security settings.

The SDK provides an abstraction of Bluetooth profiles that is independent of the underlying host stack used to provide Bluetooth services. Future expansions or hardware changes will not affect applications using SDK API.

The BlueSoleil SDK APIs are divided into two categories, General and Profile Specific.

The General part interface provides basic Bluetooth functions defined in General Access Profile (GAP) and Service Discovery Application Profile (SDAP). It also provides:

- Remote device management.
- Security Management.
- Connection Management.

The Profile Specific interface provides functions defined in different Bluetooth profiles except for General Access Profile and Service Discovery Application Profile.

These two categories of API are separately introduced in **Chapter 5** and **Chapter 6**.

5. General API Reference

5.1 BlueSoleil Data Types

The data types supported by IVT BlueSoleil are used to define function return values, function and message parameters, and structure members. They define the size and meaning of these elements.

Type	Definition
BTINT8	8-bit ANSI character.
BTUINT8	8-bit unsigned integer.
BTBOOL	Boolean variable (Should be BTSDK_TRUE or BTSDK_FALSE)
BTINT16	16-bit signed integer.
BTUINT16	16-bit unsigned integer.
BTINT32	32-bit signed integer.
BTUINT32	32-bit unsigned integer.
BTLPVOID	Pointer to any type.
BTDEVHDL	Handle to a device object.
BTSVCHDL	Handle to a service object.
BTCONNHDL	Handle to a connection object.
BTSHCHDL	Handle to a shortcut object.
BTSDKHANDLE	Handle to any object.
BTUCHAR	8-bit ANSI character.

5.2 Constant Reference

5.2.1 Error Codes

The following table provides a list of error codes. They are returned by many BlueSoleil functions when they fail.

Name	Value	Description
BTSDK_OK	0X0000	The operation completed successfully.
BTSDK_ER_SERVER_IS_ACTIVE	0X00C0	Local service is still active. When the application tries to remove or activate an active service, this error code is returned.
BTSDK_ER_NO_SERVICE	0X00C1	No service record with the specified search pattern is found on the remote device.
BTSDK_ER_SERVICE_RECORD_NOT_EXIST	0X00C2	The specified service record does not exist on the remote device.
BTSDK_ER_HANDLE_NOT_EXIST	0X0301	The object specified by the handle does not exist in local BlueSoleil SDK database.
BTSDK_ER_OPERATION_FAILURE	0X0302	The operation fails for an undefined reason.
BTSDK_ER_SDK_UNINIT	0X0303	BlueSoleil SDK has not been initialized.
BTSDK_ER_INVALID_PARAMETER	0X0304	The parameter value is invalid.
BTSDK_ER_NULL_POINTER	0X0305	The pointer value is NULL.
BTSDK_ER_NO_MEMORY	0X0306	Not enough storage is available to process this function.
BTSDK_ER_BUFFER_NOT_ENOUGH	0X0307	The specified buffer size is too small to hold the required information.
BTSDK_ER_FUNCTION_NOTSUPPORT	0X0308	The specified function is not supported by the BlueSoleil.
BTSDK_ER_NO_FIXED_PIN_CODE	0X0309	No fixed PIN code is available.
BTSDK_ER_CONNECTION_EXIST	0X030A	The specified service has been connected already.
BTSDK_ER_OPERATION_CONFLICT	0X030B	The request can’t be processed since a same request is being processed.
BTSDK_ER_NO_MORE_CONNECTION_ALLOWED	0X030C	The limit of connection number is reached.
BTSDK_ER_ITEM_EXIST	0X030D	An object with the specified attribute exists.

BTSDK_ER_ITEM_INUSE	0X030E	The specified object is accessed by other process. It can't be removed or modified.
BTSDK_ER_DEVICE_UNPAIRED	0X030F	The specified remote device is not paired.
BTSDK_ER_UNKNOWN_HCI_COMMAND	0X0401	HCI error "Unknown HCI Command (0X01)" is received.
BTSDK_ER_NO_CONNECTION	0X0402	HCI error "Unknown Connection Identifier (0X02)" is received.
BTSDK_ER_HARDWARE_FAILURE	0X0403	HCI error "Hardware Failure (0X03)" is received.
BTSDK_ER_PAGE_TIMEOUT	0X0404	HCI error "Page Timeout (0X04)" is received.
BTSDK_ER_AUTHENTICATION_FAILURE	0X0405	HCI error "Authentication Failure (0X05)" is received.
BTSDK_ER_KEY_MISSING	0X0406	HCI error "PIN or Key Missing (0X06)" is received.
BTSDK_ER_MEMORY_FULL	0X0407	HCI error "Memory Capacity Exceeded (0X07)" is received.
BTSDK_ER_CONNECTION_TIMEOUT	0X0408	HCI error "Connection Timeout (0X08)" is received.
BTSDK_ER_MAX_NUMBER_OF_CONNECTIONS	0X0409	HCI error "Connection Limit Exceeded (0X09)" is received.
BTSDK_ER_MAX_NUMBER_OF_SCO_CONNECTIONS	0X040A	HCI error "Synchronous Connection Limit to a Device Exceeded (0X0A)" is received.
BTSDK_ER_ACL_CONNECTION_ALREADY_EXISTS	0X040B	HCI error "ACL Connection Already Exists (0X0B)" is received.
BTSDK_ER_COMMAND_DISALLOWED	0X040C	HCI error "Command Disallowed (0X0C)" is received.
BTSDK_ER_HOST_REJECTED_LIMITED_RESOURCES	0X040D	HCI error "Connection Rejected due to Limited Resources (0X0D)" is received.
BTSDK_ER_HOST_REJECTED_SECURITY_REASONS	0X040E	HCI error "Connection Rejected due to Security Reasons (0X0E)" is received.
BTSDK_ER_HOST_REJECTED_PERSONAL_DEVICE	0X040F	HCI error "Connection Rejected due to Unacceptable BD_ADDR (0X0F)" is received.
BTSDK_ER_HOST_TIMEOUT	0X0410	HCI error "Connection Accept Timeout Exceeded (0X10)" is received.
BTSDK_ER_UNSUPPORTED_FEATURE	0X0411	HCI error "Unsupported Feature or

		Parameter Value (0X11)" is received.
BTSDK_ER_INVALID_HCI_COMMAND_PARAMETERS	0X0412	HCI error "Invalid HCI Command parameters (0X12)" is received.
BTSDK_ER_PEER_DISCONNECTION_USER_END	0X0413	HCI error "Remote User Terminated Connection (0X13)" is received.
BTSDK_ER_PEER_DISCONNECTION_LOW_RESOURCES	0X0414	HCI error "Remote Device Terminated Connection due to Low Resources (0X14)" is received.
BTSDK_ER_PEER_DISCONNECTION_TO_POWER_OFF	0X0415	HCI error "Remote Device Terminated Connection due to Power Off (0X15)" is received.
BTSDK_ER_LOCAL_DISCONNECTION	0X0416	HCI error "Connection Terminated by Local Host (0X16)" is received.
BTSDK_ER_REPEATED_ATTEMPTS	0X0417	HCI error "Repeated Attempts (0X17)" is received.
BTSDK_ER_PAIRING_NOT_ALLOWED	0X0418	HCI error "Pairing Not Allowed (0X18)" is received.
BTSDK_ER_UNKNOWN_LMP_PDU	0X0419	HCI error "Unknown LMP PDU (0X19)" is received.
BTSDK_ER_UNSUPPORTED_REMOTE_FEATURE	0X041A	HCI error "Unsupported Remote Feature / Unsupported LMP Feature (0X1A)" is received.
BTSDK_ER_SCO_OFFSET_REJECTED	0X041B	HCI error "SCO Offset Rejected (0X1B)" is received.
BTSDK_ER_SCO_INTERVAL_REJECTED	0X041C	HCI error "SCO Interval Rejected (0X1C)" is received.
BTSDK_ER_SCO_AIR_MODE_REJECTED	0X041D	HCI error "SCO Air Mode Rejected (0X1D)" is received.
BTSDK_ER_INVALID_LMP_PARAMETERS	0X041E	HCI error "Invalid LMP Parameters (0X1E)" is received.
BTSDK_ER_UNSPECIFIED_ERROR	0X041F	HCI error "Unspecified Error (0X1F)" is received.
BTSDK_ER_UNSUPPORTED_LMP_PARAMETER_VALUE	0X0420	HCI error "Unsupported LMP Parameter Value (0X20)" is received.
BTSDK_ER_ROLE_CHANGE_NOT_ALLOWED	0X0421	HCI error "Role Change Not Allowed (0X21)" is received.
BTSDK_ER_LMP_RESPONSE_TIMEOUT	0X0422	HCI error "LMP Response Timeout (0X22)" is received.
BTSDK_ER_LMP_ERROR_TRANSACTION_COLLISION	0X0423	HCI error "LMP Error Transaction Collision (0X23)" is received.
BTSDK_ER_LMP_PDU_NOT_ALLOWED	0X0424	HCI error "LMP PDU Not Allowed (0X24)" is received.
BTSDK_ER_ENCRYPTION_MODE_NOT_ACCEPTABLE	0X0425	HCI error "Encryption Mode Not Acceptable (0X25)" is received.

BTSDK_ER_UNIT_KEY_USED	0X0426	HCI error “Link Key Can not be Changed (0X26)” is received.
BTSDK_ER_QOS_IS_NOT_SUPPORTED	0X0427	HCI error “Requested QOS Not Supported (0X27)” is received.
BTSDK_ER_INSTANT_PASSED	0X0428	HCI error “Instant Passed (0X28)” is received.
BTSDK_ER_PAIRING_WITH_UNIT_KEY_NOT_SUPPORTED	0X0429	HCI error “Pairing with Unit Key Not Supported (0X29)” is received.
BTSDK_ER_DIFFERENT_TRANSACTION_COLLISION	0X042A	HCI error “Different Transaction Collision (0X2A)” is received.
BTSDK_ER_QOS_UNACCEPTABLE_PARAMETER	0X042C	HCI error “QOS Unacceptable Parameter (0X2C)” is received.
BTSDK_ER_QOS_REJECTED	0X042D	HCI error “QOS Rejected (0X2D)” is received.
BTSDK_ER_CHANNEL_CLASS_NOT_SUPPORTED	0X042E	HCI error “Channel Classification Not Supported (0X2E)” is received.
BTSDK_ER_INSUFFICIENT_SECURITY	0X042F	HCI error “Insufficient Security (0X2F)” is received.
BTSDK_ER_PARAMETER_OUT_OF_RANGE	0X0430	HCI error “Parameter Out of Mandatory Range (0X30)” is received.
BTSDK_ER_ROLE_SWITCH_PENDING	0X0432	HCI error “Role Switch Pending (0X32)” is received.
BTSDK_ER_RESERVED_SLOT_VIOLATION	0X0434	HCI error “Reserved Slot Violation (0X34)” is received.
BTSDK_ER_ROLE_SWITCH_FAILED	0X0435	HCI error “Role Switch Failed (0X35)” is received.

Table 1: BlueSoleil Error Codes.

5.2.2 Service Class Identifier

The following table provides a list of class identifiers of services supported by current version BlueSoleil. These service class identifiers are specified as 16-bit UUID. These values will be used when the service class is required as a parameter.

Name	UUID	Description
BTSDK_CLS_SERIAL_PORT	0X1101	Serial Port service.
BTSDK_CLS_LAN_ACCESS	0X1102	LAN Access service.
BTSDK_CLS_DIALUP_NET	0X1103	Dial-up Networking service.
BTSDK_CLS_IRMC_SYNC	0X1104	Synchronization service.
BTSDK_CLS_OBEX_OBJ_PUSH	0X1105	Object Push service.
BTSDK_CLS_OBEX_FILE_TRANS	0X1106	File Transfer service.
BTSDK_CLS_IRMC_SYNC_CMD	0X1107	IrMC Sync Command service.
BTSDK_CLS_HEADSET	0X1108	Headset service.
BTSDK_CLS_CORDLESS_TELE	0X1109	Cordless Telephony service.
BTSDK_CLS_AUDIO_SOURCE	0X110A	Audio Source service.
BTSDK_CLS_AUDIO_SINK	0X110B	Audio Sink service.
BTSDK_CLS_AVRCP_TG	0X110C	A/V Remote Control Target service.
BTSDK_CLS_ADV_AUDIO_DISTRIB	0X110D	Advanced Audio Distribution service.
BTSDK_CLS_AVRCP_CT	0X110E	A/V Remote Control service.
BTSDK_CLS_VIDEO_CONFERENCE	0X110F	Video conference service.
BTSDK_CLS_INTERCOM	0X1110	Intercom service.
BTSDK_CLS_FAX	0X1111	Fax service.
BTSDK_CLS_HEADSET_AG	0X1112	Headset Audio Gateway service.
BTSDK_CLS_WAP	0X1113	WAP service.
BTSDK_CLS_WAP_CLIENT	0X1114	WAP client service.
BTSDK_CLS_PAN_PANU	0X1115	PANU service.
BTSDK_CLS_PAN_NAP	0X1116	NAP service.
BTSDK_CLS_PAN_GN	0X1117	GN service.
BTSDK_CLS_DIRECT_PRINT	0X1118	Direct Print service.
BTSDK_CLS_REF_PRINT	0X1119	Referenced Print service.
BTSDK_CLS_IMAGING	0X111A	Imaging service.
BTSDK_CLS_IMAG_RESPONDER	0X111B	Imaging Responder service.
BTSDK_CLS_IMAG_AUTO_ARCH	0X111C	Imaging Automatic Archive service.
BTSDK_CLS_IMAG_REF_OBJ	0X111D	Imaging Referenced Objects service.
BTSDK_CLS_HANDSFREE	0X111E	Hands-free service.
BTSDK_CLS_HANDSFREE_AG	0X111F	Hands-free Audio Gateway service.
BTSDK_CLS_DPS_REF_OBJ	0X1120	DPS Referenced Objects service.
BTSDK_CLS_REFLECTED_UI	0X1121	Reflected UI service
BTSDK_CLS_BASIC_PRINT	0X1122	Basic Print service.
BTSDK_CLS_PRINT_STATUS	0X1123	Print Status service.

BTSDK_CLS_HID	0X1124	Human Interface Device service.
BTSDK_CLS_HCRP	0X1125	Hardcopy Cable Replacement service.
BTSDK_CLS_HCR_PRINT	0X1126	HCRP Print service.
BTSDK_CLS_HCR_SCAN	0X1127	HCRP Scan service.
BTSDK_CLS_SIM_ACCESS	0X112D	SIM Card Access service
BTSDK_CLS_PBAP_PCE	0X112E	PBAP Phonebook Client Equipment service.
BTSDK_CLS_PBAP_PSE	0X112F	PBAP Phonebook Server Equipment service.
BTSDK_CLS_PHONEBOOK_ACCESS	0X1130	Phonebook Access service.
BTSDK_CLS_PNP_INFO	0X1200	Bluetooth Device Identification.

Table 2: IVT BlueSoleil Service Class Identifiers.

5.2.3 Class of Device/Service Field

The following table provides a list of device class identifiers categorized by major device class. These device class identifiers are mapped to the device class field of the Class of Device/Service field (first format type).

Name	Value	Description
DEVICE_CLASS_MASK	0x1FFC	Mask of device class
BTSDK_DEVCLS_COMPUTER	0x000100	Computer major device class.
BTSDK_COMPCLS_UNCLASSIFIED	0x000100	Uncategorized computer, code for device not assigned.
BTSDK_COMPCLS_DESKTOP	0X000104	Desktop workstation.
BTSDK_COMPCLS_SERVER	0X000108	Server-class computer.
BTSDK_COMPCLS_LAPTOP	0X00010C	Laptop computer.
BTSDK_COMPCLS_HANDHELD	0X000110	Handheld PC/PDA (clam shell).
BTSDK_COMPCLS_PALMSIZED	0X000114	Palm sized PC/PDA.
BTSDK_COMPCLS_WEARABLE	0X000118	Wearable computer (Watch sized).
BTSDK_DEVCLS_PHONE	0X000200	Phone major device class.
BTSDK_PHONECLS_UNCLASSIFIED	0X000200	Uncategorized phone, code for device not assigned.
BTSDK_PHONECLS_CELLULAR	0X000204	Cellular phone.
BTSDK_PHONECLS_CORDLESS	0X000208	Cordless phone.
BTSDK_PHONECLS_SMARTPHONE	0X00020C	Smart phone.
BTSDK_PHONECLS_WIREDMODEM	0X000210	Wired modem or voice gateway.
BTSDK_PHONECLS_COMMONISDNACCESS	0X000214	Common ISDN Access.
BTSDK_PHONECLS_SIMCARDREADER	0X000218	SIM card reader
BTSDK_DEVCLS_LAP	0X000300	LAN / Network Access Point major

		device class.
BTSDK_LAP_FULLY	0X000300	Fully available.
BTSDK_LAP_17	0X000320	1 - 17% utilized.
BTSDK_LAP_33	0X000340	17- 33% utilized.
BTSDK_LAP_50	0X000360	33 - 50% utilized.
BTSDK_LAP_67	0X000380	50 - 67% utilized.
BTSDK_LAP_83	0X0003A0	67 - 83% utilized.
BTSDK_LAP_99	0X0003C0	83 – 99% utilized.
BTSDK_LAP_NOSRV	0X0003E0	No service available.
BTSDK_DEVCLS_AUDIO	0X000400	Audio/Video major device class.
BTSDK_AV_UNCLASSIFIED	0X000400	Uncategorized A/V device, code for device not assigned.
BTSDK_AV_HEADSET	0X000404	Wearable headset device.
BTSDK_AV_HANDSFREE	0X000408	Hands-free device.
BTSDK_AV_MICROPHONE	0X000410	Microphone.
BTSDK_AV_LOUDSPEAKER	0X000414	Loudspeaker.
BTSDK_AV_HEADPHONES	0X000418	Headphones.
BTSDK_AV_PORTABLEAUDIO	0X00041C	Portable Audio.
BTSDK_AV_CARAUDIO	0X000420	Car Audio.
BTSDK_AV_SETTOPBOX	0X000424	Set-top box.
BTSDK_AV_HIFIAUDIO	0X000428	HiFi Audio device.
BTSDK_AV_VCR	0X00042C	Videocassette recorder
BTSDK_AV_VIDEOCAMERA	0X000430	Video camera
BTSDK_AV_CAMCORDER	0X000434	Camcorder
BTSDK_AV_VIDEOMONITOR	0X000438	Video monitor.
BTSDK_AV_VIDEODISPANDLOUDSPK	0X00043C	Video display and loudspeaker.
BTSDK_AV_VIDEOCONFERENCE	0X000440	Video conferencing.
BTSDK_AV_GAMEORTOY	0X000448	Gaming/Toy
BTSDK_DEVCLS_PERIPHERAL	0X000500	Peripheral major device class
BTSDK_PERIPHERAL_UNCLASSIFIED	0X000500	Uncategorized peripheral device, code for device not assigned.
BTSDK_PERIPHERAL_KEYBOARD	0X000540	Keyboard.
BTSDK_PERIPHERAL_POINT	0X000580	Pointing device.
BTSDK_PERIPHERAL_KEYORPOINT	0X0005C0	Combo keyboard/pointing device.
BTSDK_DEVCLS_IMAGE	0X000600	Imaging major device class.
BTSDK_IMAGE_DISPLAY	0X000610	Display.
BTSDK_IMAGE_CAMERA	0X000620	Camera.
BTSDK_IMAGE_SCANNER	0X000640	Scanner.
BTSDK_IMAGE_PRINTER	0X000680	Printer.
BTSDK_DEVCLS_WEARABLE	0x000700	Wearable major device class.
BTSDK_WERABLE_WATCH	0x000704	Wristwatch.
BTSDK_WERABLE_PAGER	0x000708	Pager.

BTSDK_WERABLE_JACKET	0x00070C	Jacket
BTSDK_WERABLE_HELMET	0x000710	Helmet.
BTSDK_WERABLE_GLASSES	0x000714	Glasses.

Table 3: BlueSoleil Device Class Filed Identifiers

The following table provides a list of major service class identifiers that are mapped to the service class field of the Class of Device/Service field (first format type).

Name	Value	Description
BTSDK_SRVCLS_LDM	0x002000	Limited Discoveralbe Mode
BTSDK_SRVCLS_POSITION	0x010000	Positioning (Location Identification).
BTSDK_SRVCLS_NETWORK	0x020000	Networking (LAN, AD hoc, …).
BTSDK_SRVCLS_RENDER	0x040000	Rendering (Printing, Speaker, …).
BTSDK_SRVCLS_CAPTURE	0x080000	Capturing (Scanner, Microphone, …).
BTSDK_SRVCLS_OBJECT	0x100000	Object Transfer (v-Inbox, v-Folder, …).
BTSDK_SRVCLS_AUDIO	0x200000	Audio (Speaker, Microphone, Headset service, …).
BTSDK_SRVCLS_TELEPHONE	0x400000	Telephony (Cordless telephony, Modem, Headset service, …).
BTSDK_SRVCLS_INFOR	0x800000	Information (WEB-server, WAP-server, …).

Table 4: IVT BlueSoleil Major Service Class Identifiers

A complete Class of Device/Service field (first format type) can be the combination of one device class identifier and multiple major service class identifiers.

5.2.4 Bluetooth Device Modes

The following table provides a list of flags that specify the Bluetooth device modes.

Name	Description
BTSDK_GENERAL_DISCOVERABLE	Sets the device into general discoverable mode. This is the default discoverable mode.
BTSDK_LIMITED_DISCOVERABLE	Sets the device into limited discoverable mode. If this value is specified, BTSDK_GENERAL_DISCOVERABLE mode value is ignored by BlueSoleil.
BTSDK_DISCOVERABLE	Makes the device discoverable. This is equivalent to BTSDK_GENERAL_DISCOVERABLE.
BTSDK_CONNECTABLE	Makes the device connectable. This is the default connectable mode.
BTSDK_PAIRABLE	Makes the device pairable. This is the default pairable mode.

Table 5: Bluetooth Device Modes

5.2.5 Messages from BlueSoleil to the Application

The following table provides a list of messages transferred from BlueSoleil to the application and the type of the callback functions to process these messages.

Message Name	Callback Function Type	Description
BTSDK_INQUIRY_RESULT_IND	Btsdk_Inquiry_Result_Ind_Func	This message indicates that a Bluetooth device has responded so far during the current inquiry process.
BTSDK_INQUIRY_COMPLETE_IND	Btsdk_Inquiry_Complete_Ind_Func	This message indicates that the inquiry is finished.
BTSDK_CONNECTION_EVENT_IND	Btsdk_Connection_Event_Ind_Func	This message indicates that a high-level protocol connection is created or disconnected.
BTSDK_PIN_CODE_IND	Btsdk_UserHandle_Pin_Req_Ind_Func	This message indicates the application to input PIN code for the specified device.
BTSDK_AUTHORIZATION_IND	Btsdk_UserHandle_Authorization_Req_Ind_Func	This message indicates that a remote device is trying to access a local service.
BTSDK_LINK_KEY_NOTIF_IND	Btsdk_Link_Key_Notif_Ind_Func	This message indicates that a new link key has been created for the specified device.
BTSDK_AUTHENTICATION_FAIL_IND	Btsdk_Authentication_Fail_Ind_Func	This message indicates that an error occurs when performing authentication with the specified device.
BTSDK_LINK_KEY_REQ_IND	Btsdk_Link_Key_Req_Ind_Func	This message indicates the

		application to input the link key for remote device.
BTSDK_DEVICE_FOUND_IND	Btsdk_DeviceFound_Func	This message indicates that a Bluetooth device information changes or a LE device enters the range of communication.

Table 6: Messages from BlueSoleil to the Application

5.2.6 Type of Device

The following table provides a list of device type.

Table Device type

Name	Value	Description
BTSDK_DEV_TYPE_LE_ONLY	0x01	LE device only
BTSDK_DEV_TYPE_BREDR_ONLY	0x10	BREDR device only
BTSDK_DEV_TYPE_BREDR_LE	0x11	BREDR LE device

5.3 Data Structures

5.3.1 BtSdkCallbackStru

Definition	typedef struct _BtSdkCallBackStru { BTUINT16 type; PVOID func } BtSdkCallbackStru, *PBtSdkCallbackStru;	
Description	The structure BtSdkCallbackStru contains information about a callback function.	
Members	*Type*	Specifies the message of the callback function to process. It also specifies the prototype of the callback function. It can be one of the values listed in Table 6.
	Func	Pointer to the callback function. If *func* is NULL, BlueSoleil will remove the callback.

Remarks

Detail about each callback function is discussed in the following section.

5.3.2 BtSdkLocalLMPInfoStru

Definition	typedef struct _BtSdkLocalLMPInfoStru { BTUINT8 lmp_feature[8]; BTUINT16 manuf_name; BTUINT16 lmp_subversion; BTUINT8 lmp_version; BTUINT8 hci_version; BTUINT16 hci_revision; BTUINT8 country_code; } BtSdkLocalLMPInfoStru, *PBtSdkLocalLMPInfoStru;	
Description	The structure BtSdkLocalLMPInfoStru contains information about local host controller.	
Members	*lmp_feature*	List of supported features for the local device.
	manuf_name	Integer specifies the manufacturer of the local device.
	lmp_subversion	Subversion of the current LMP in the local device.
	lmp_version	Version of the current LMP in the local device.
	hci_version	Version of the current HCI in the local device.
	hci_revision	Revision of the current HCI in the local device.
	Country_code	Integer defines which range of frequency band of the ISM 2.4GHz band is used by the local device. This member is for backwards compatibility with a prior version HCI (1.1 and 1.0A).

5.3.3 BtSdkVendorCmdStru

Definition	typedef struct _BtSdkVendorCmdStru { BTUINT16 ocf; BTUINT8 param_len; BTUINT8 param[1]; } BtSdkVendorCmdStru, *PBtSdkVendorCmdStru;	
Description	The structure BtSdkVendorCmdStru contains information about a vendor specific command.	
Members	Ocf	Specifies the OpCode Command Field value of this vendor specific command.
	param_len	Specifies the size in bytes of the content in the buffer pointer by the param element.
	Param	Pointer to the buffer containing the command parameters.

Remarks

The *param* element of this structure is a variable length array of octets. Contents in the buffer pointed to by the *param* element are copied to the final HCI command packet's parameter field directly. The core Bluetooth stack determines the number of octets to be copied by examining the value of the *param_len* element. The application must ensure the correctness and integrity of the parameters.

Example

```
/* This sample demonstrates how to set BtSdkVendorCmdStru for the vendor command:
    {0xFC, 0x01, 0x04, 0x00, 0x10, 0x3A, 0x33}. */
void AppVendorCommand (void)
{
    BTUINT8 param[] = {0x00, 0x10, 0x3A, 0x33};
    PBtSdkVendorCmdStru pCmd = (PBtSdkVendorCmdStru)malloc(szieof(BtSdkVendorCmdStru)+sizeof(param));
    pCmd->ocf = 0x01;
    pCmd->param_len = sizeof(param);
    memcpy(pCmd->param, param, pCmd->param_len);
    /* To Do: Processing the command. */
```

```
free(pCmd);
}
```

5.3.4 BtSdkEventParamStru

Definition	typedef struct _BtSdkEventParamStru { BTUINT8 ev_code; BTUINT8 param_len; BTUINT8 param[1]; } BtSdkEventParamStru, *PBtSdkEventParamStru;	
Description	The structure BtSdkEventParamStru contains information about a HCI event.	
Members	ev_code	Specifies the event code.
	param_len	On input, specifies the size in bytes of the param buffer. On output, receives the number of bytes required to receive the event parameters.
	Param	Pointer to the buffer receiving the raw event parameters copied from the HCI event packet's parameter field.

Remarks

BtSdkEventParamStru structure is usually used to receive the HCI event generated for a specific HCI command. The *param* element of this structure is a variable length array of octets. Contents in the buffer pointed to by the *param* element are copied from the HCI event packet's parameter field directly. The core Bluetooth stack determines the number of octets to be copied by examining the value of the *param_len* element and the actual size of the event parameter list.

The application shall allocate a buffer large enough to hold all the event parameters. Generally, if the buffer size specified by the *param_len* element is smaller than the number of bytes required, the BlueSoleil function call returns BTSDK_ER_BUFFER_NOT_ENOUGH and *param_len* is set to the actual size required by BlueSoleil.

A buffer of 257 bytes, which is the maximum length of an event packet, is suggested if the user doesn't know the actual size of the event parameter list.

Example

```
/* This sample demonstrates how to send a vendor specific command {0x01, 0xFC, 0x04, 0x00, 0x10, 0x3A, 0x33}
```

```
        and receive the created event {0x0E, 0x04, 0x01, 0x01, 0xFC, 0x02}.
        Command and event packet in this sample are used only for demonstration. Do NOT execute this sample function
on
        your platform unless you are sure they are really exported by the Bluetooth device you used.
*/
void AppVendorCommand (void)
{
        BTUINT8 param[] = {0x00, 0x10, 0x3A, 0x33};
        PBtSdkVendorCmdStru pCmd = (PBtSdkVendorCmdStru)malloc(szieof(BtSdkVendorCmdStru)+sizeof(param));
        PBtSdkEventParamStru pEv = (PBtSdkEventParamStru)malloc(257);

        pCmd->ocf = 0x01;
        pCmd->param_len = sizeof(param);
        memcpy(pCmd->param, param, pCmd->param_len);
        memset(pEv, 0, 257);
        pEv->param_len = 255;
        Btsdk_VendorCommand(0, pCmd, pEv);
        /* If the command is executed successfully, we shall find that:
              pEv->ev_code = 0x0E;       pEv->param_len = 0x04;
              pEv->param[0]  ＝  0x01; pEv->param[1] = 0x01; pEv->param[2] = 0xFC; pEv->param[3] = 0x02;
        */
        free(pCmd);
        free(pEv);
}
```

5.3.5 BtSdkRemoteLMPInfoStru

Definition	typedef struct _BtSdkRemoteLMPInfoStru { BTUINT8 lmp_feature[8]; BTUINT16 manuf_name; BTUINT16 lmp_subversion; BTUINT8 lmp_version; } BtSdkRemoteLMPInfoStru, *PBtSdkRemoteLMPInfoStru;	
Description	The structure BtSdkRemoteLMPInfoStru contains information about remote host controller.	
Members	*lmp_feature*	List of supported features for the remote device.
	manuf_name	Integer specifies the manufacturer of the local device.
	lmp_subversion	Subversion of the current LMP in the remote device.
	lmp_version	Version of the current LMP in the remote device.

5.3.6 BtSdkRemoteDevicePropertyStru

Definition	typedef struct _BtSdkRemoteDevicePropertyStru { BTUINT32 mask; BTDEVHDL dev_hdl; BTUINT8 bd_addr[BTSDK_BDADDR_LEN]; BTUINT8 name[BTSDK_DEVNAME_LEN]; BTUINT32 dev_class; BtSdkRemoteLMPInfoStru lmp_info; BTUINT8 link_key[BTSDK_LINKKEY_LEN]; } BtSdkRemoteDevicePropertyStru, *PBtSdkRemoteDevicePropertyStru;	
Description	The structure BtSdkRemoteDevicePropertyStru contains information about a remote device.	
Members	*mask*	Specifies which member is available.
	dev_hdl	Handle assigned to this device record.
	bd_addr	Bluetooth device address of this device record.
	name	User-friendly name of this device record. This string is coded in UTF-8 format.
	dev_class	The Class of Device/Service setting of this device record. It can be one of the device class identifiers listed in Table 3 combined with multiple major service class identifiers listed in Table 4.
	lmp_info	Information about the host controller of this device.
	link_key	Link key for this device.

The ***mask*** member can be one or more of these values.

Value	Description
BTSDK_RDPM_HANDLE	The value of the *dev_hdl* member is available.
BTSDK_RDPM_ADDRESS	The value of the *bd_addr* member is available.
BTSDK_RDPM_NAME	The value of the *name* member is available.
BTSDK_RDPM_CLASS	The value of the *dev_class* is available.
BTSDK_RDPM_LMPINFO	The value of the *lmp_info* is available.

BTSDK_RDPM_LINKKEY	The value of the *link_key* is available.

5.3.7 BtSdkHoldModeStru

Definition	typedef struct _BtSdkHoldModeStru { BTUINT16 conn_hdl; BTUINT16 max; BTUINT16 min; } BtSdkHoldModeStru, *PBtSdkHoldModeStru;	
Description	The structure BtSdkHoldModeStru contains hold mode parameters.	
Members	*conn_hdl*	Reserved for future extension. Set it to 0.
	max	Specifies the maximum acceptable number of Baseband slots (0.625msec) to wait in the Hold mode. Range: 0x0002 to 0xFFFE; only even values are valid.
	min	Specifies the minimum acceptable number of Baseband slots (0.625msec) to wait in the Hold mode. Range: 0x0002 to 0xFF00; only even values are valid.

5.3.8 BtSdkSniffModeStru

Definition	typedef struct _BtSdkSniffModeStru { BTUINT16 conn_hdl; BTUINT16 max; BTUINT16 min; BTUINT16 attempt; BTUINT16 timeout; } BtSdkSniffModeStru, *PBtSdkSniffModeStru;	
Description	The structure BtSdkSniffModeStru contains sniff mode parameters.	
Members	*conn_hdl*	Reserved for future extension. Set it to 0.
	max	Specifies the maximum acceptable periods, in number of Baseband slots (0.625msec), in the Sniff mode. Range: 0x0002 to 0xFFFE; only even values are valid.
	min	Specifies the minimum acceptable periods, in number of Baseband slots (0.625msec), in the Sniff mode. Range: 0x0002 to 0xFFFE; only even values are valid.
	attempt	Specifies the number of Baseband receive slots (0.625msec) for sniff attempt. Range: 0x0001 to 0x7FFF.
	timeout	Specifies the number of Baseband receive slots (0.625msec) for sniff timeout. Range: 0x0000 to 0x7FFF.

5.3.9 BtSdkParkModeStru

Definition	typedef struct _BtSdkParkModeStru { BTUINT16 conn_hdl; BTUINT16 max; BTUINT16 min; } BtSdkParkModeStru, *PBtSdkParkModeStru;	
Description	The structure BtSdkParkModeStru contains park mode parameters.	
Members	*conn_hdl*	Reserved for future extension. Set it to 0.
	max	Specifies the acceptable longest length of the interval, in number of Baseband slots (0.625msec), between beacons in the Park mode. Range: 0x000E to 0xFFFE; only even values are valid.
	min	Specifies the acceptable shortest length of the interval, in number of Baseband slots (0.625msec), between beacons in the Park mode. Range: 0x000E to 0xFFFE; only even values are valid.

5.3.10 BtSdkUUIDStru

Definition	typedef struct _BtSdkUUIDStru { BTUINT32 Data1; BTUINT16 Data2; BTUINT16 Data3; BTUINT8 Data4[8]; } BtSdkUUIDStru, *PBtSdkUUIDStru;	
Description	The structure BtSdkUUIDStru defines Universally Unique Identifier (UUID). UUID provides unique designations of service class.	
Members	*Data1*	Specifies the first 8 hexadecimal digits of the UUID.
	Data2	Specifies the first group of 4 hexadecimal digits of the UUID.
	Data3	Specifies the second group of 4 hexadecimal digits of the UUID.
	Data4	Specifies an array of eight elements. The first two elements contain the third group of 4 hexadecimal digits of the UUID. The remaining six elements contain the final 12 hexadecimal digits of the UUID.

Example

```
/*UUID value 0x00001234-0000-1000-8000-00805F9B34FB */
BtSdkUUIDStru uuid128 = {
                                                    0x00001234,
                                                    0x0000,
                                                    0x1000,
                                                    {0x80, 0x00, 0x00, 0x80, 0x5F, 0x9B,
0x34, 0xFB}
                                                };      /* Use BtSdkUUIDStru to represent a 128bit UUID
*/
```

5.3.11 BtSdkSDPSearchPatternStru

Definition	typedef struct _BtSdkSDPSearchPatternStru { BTUINT32 mask; BtSdkUUIDStru uuid; } BtSdkSDPSearchPatternStru, *PBtSdkSDPSearchPatternStru;	
Description	The structure BtSdkSDPSearchPatternStru contains information about a SDP search pattern.	
Members	*mask*	A set of flags which specify the valid bytes of the *uuid* member.
	uuid	A BtSdkUUIDStru type variable specifies the search pattern. A search pattern can be a 16bit, 32bit or 128bit UUID value according to the *mask* value.

The ***mask*** member can be one of these values.

Value	Description
BTSDK_SSPM_UUID16	The *uuid* member specifies a 16bit UUID value. That is, *uuid.Data1* contains the 16bit UUID value.
BTSDK_SSPM_UUID32	The *uuid* member specifies a 32bit UUID value. That is, *uuid.Data1* contains the 32bit UUID value.
BTSDK_SSPM_UUID128	The *uuid* member specifies a 128bit UUID value.

Example

```
/*Search pattern with UUID values 0x1002, 0x00112233 and 0x00001234-0000-1000-8000-00805F9B34FB */
BtSdkSDPSearchPatternStru ptn16 = {0}, ptn32 = {0}, ptn128 = {0};
BtSdkUUIDStru uuid128 = {
                                                    0x00001234,
                                                    0x0000,
                                                    0x1000,
                                                    {0x80, 0x00, 0x00, 0x80, 0x5F, 0x9B, 0x34, 0xFB}
                                                };          /* Use BtSdkUUIDStru to represent a 128bit UUID */
Ptn16.mask = BTSDK_SSPM_UUID16;
Ptn16.uuid.Data1 = 0x1002;
Ptn32.mask = BTSDK_SSPM_UUID32;
Ptn32.Data1 = 0x00112233;
```

```
Ptn128.mask = BTSDK_SSPM_UUID128;
memcpy(&ptn128.uuid, &uuid128, sizeof(BtSdkUUIDStru uuid128));
```

5.3.12 BtSdkRemoteServiceAttrStru

Definition	typedef struct _BtSdkRemoteServiceAttrStru { BTUINT32 mask; BTUINT16 service_class; BTDEVHDL dev_hdl; BTUINT8 svc_name[BTSDK_SERVICENAME_MAXLENGTH]; BTLPVOID ext_attributes; BTUINT16 status; } BtSdkRemoteServiceAttrStru, *PBtSdkRemoteServiceAttrStru;	
Description	The structure BtSdkRemoteServiceAttrStru contains information about a remote service record.	
Members	*mask*	A set of flags which specify members to retrieve.
	service_class	Type of the service record. It can be one of the values listed in the Table 2.
	dev_hdl	Handle to the remote device that exports this service record.
	svc_name	User-friendly name of this service record. This string is coded in UTF-8 format. Set *mask* to BTSDK_RSAM_SERVICENAME to use *svc_name*.
	ext_attributes	Profile specific attributes. It must be cast to a pointer to a structure decided by the service type. See following table. Set *mask* to BTSDK_RSAM_EXTATTRIBUTES to use *ext_attributes*. Always set it to NULL when input.
	status	Current status of this service record.

The ***mask*** member can be one or more of these values.

Value	Description
BTSDK_RSAM_SERVICENAME	Retrieves the *svc_name* member.
BTSDK_RSAM_EXTATTRIBUTES	Retrieves the *ext_attributes* member.

The ***ext_attributes*** member can be a pointer to one of these structures.

Value of ***service_class***	Type of ***ext_attributes***
BTSDK_CLS_SERIAL_PORT	PBtSdkRmtSPPSvcExtAttrStru
BTSDK_CLS_HID	PBtSdkRmtHIDSvcExtAttrStru
BTSDK_CLS_PNP_INFO	PBtSdkRmtDISvcExtAttrStru

Detail of these structures is specified in separate profile API documents.

The *ext_attributes* member is ignored and is set to NULL for profiles not listed in the upper table.

5.3.13 BtSdkRmtSPPSvcExtAttrStru

Definition	typedef struct _BtSdkRmtSPPSvcExtAttrStru { BTUINT32 size; BTUINT8 server_channel; } BtSdkRmtSPPSvcExtAttrStru, *PBtSdkRmtSPPSvcExtAttrStru;	
Description	The structure **BtSdkRmtSPPSvcExtAttrStru** describes the server_chanel of remote 128bit SPP service.	
Members	*size*	Size of the structure, in bytes.
	server_channel	Server channel value of this SPP service record.

5.3.14 BtSdkConnectionPropertyStru

Definition	typedef struct _BtSdkConnectionPropertyStru { BTUINT32 role : 2; BTUINT32 result : 30; BTDEVHDL device_handle; BTSVCHDL service_handle; BTUINT16 service_class; BTUINT32 duration; BTUINT32 received_bytes; BTUINT32 sent_bytes; } BtSdkConnectionPropertyStru, *PBtSdkConnectionPropertyStru;	
Description	The structure BtSdkConnectionPropertyStru contains information about a high-level protocol connection.	
Members	*role*	Specifies the role that local BlueSoleil SDK performs in the connection. See following table.
	result	Result of the connecting procedure. It can be one of the values listed in the Table 1.
	device_handle	Handle to the remote device that is the peer side of this connection.
	service_handle	If the *role* is BTSDK_CONNROLE_INITIATOR, it specifies the handle to the remote service record that local device connects to. If the *role* is BTSDK_CONNROLE_ACCEPTOR, it specifies the local service record that the remote device connects to.
	service_class	Type of the service record specified by the *service_handle*. It can be one of the values listed in the Table 2.
	duration	Specifies the time in seconds elapsed since the connection is created.
	received_bytes	Specifies the number of bytes received on this connection since the connection is created.
	sent_bytes	Specifies the number of bytes sent on this connection since the connection is created.

The ***role*** member can be one of these values.

Value	Description
BTSDK_CONNROLE_INITIATOR	The local BlueSoleil SDK initiates the connection to the remote service.
BTSDK_CONNROLE_ACCEPTOR	The remote device initiates the connection to a local service.

5.3.15 BTSDK_GATT_DESCRIPTOR_TYPE

Definition	typedef enum _BTSDK_GATT_DESCRIPTOR_TYPE { CharacteristicExtendedProperties, CharacteristicUserDescription, ClientCharacteristicConfiguration, ServerCharacteristicConfiguration, CharacteristicFormat, CharacteristicAggregateFormat, CustomDescriptor } BTSDK_GATT_DESCRIPTOR_TYPE;	
Description	The BTSDK_GATT_DESCRIPTOR_TYPE enumeration describes the different types of Bluetooth LE generic attributes (GATT).	
Members	*CharacteristicExtendedProperties*	The characteristic value has additional properties that describe how it can be used, or how it can be accessed.
	CharacteristicUserDescription	The characteristic value contains a UTF-8 string of variable size that is a user textual description.
	ClientCharacteristicConfiguration	The characteristic value may be configured by the client.
	ServerCharacteristicConfiguration	The characteristic value may be configured for the server.
	CharacteristicFormat	The format of the characteristic value.
	CharacteristicAggregateFormat	The format of an aggregated characteristic value.
	CustomDescriptor	The characteristic value is customized.

5.3.16 BTSDK_GATT_EVENT_TYPE

Definition	typedef enum _BTSDK_GATT_EVENT_TYPE { CharacteristicValueChangedEvent } BTSDK_GATT_EVENT_TYPE;	
Description	The BTSDK_GATT_EVENT_TYPE enumeration describes the different types of Bluetooth Low Energy (LE) generic attribute (GATT) profile events.	
Members	*CharacteristicValueChangedEvent*	The characteristic value has changed.

5.3.17 BtsdkGATTUUIDStru

Definition	typedef struct _BtsdkGATTUUIDStru { BTINT32 IsShortUuid; BTUINT16 ShortUuid; BtSdkUUIDStru LongUuid; } BtsdkGATTUUIDStru, *PBtsdkGATTUUIDStru;	
Description	The BtsdkGATTUUIDStru structure contains information about a Bluetooth Low Energy (LE) Universally Unique Identifier (UUID).	
Members	*IsShortUuid*	Indicates if the Low Energy (LE) UUID a 16-bit shortened value, or if it is the long 128-bit value.
	ShortUuid	The short 16-bit value of the UUID. This member applies only if IsShortUuid is TRUE.
	LongUuid	The long 128-bit value of the UUID. This member applies only if IsShortUuid is FALSE.

5.3.18 BtsdkGATTServiceStru

Definition	typedef struct _BtsdkGATTServiceStru { BtsdkGATTUUIDStru ServiceUuid; BTUINT16 AttributeHandle; } BtsdkGATTServiceStru, *PBtsdkGATTServiceStru;	
Description	The BtsdkGATTServiceStru structure describes a Bluetooth Low Energy (LE) generic attribute (GATT) profile service.	
Members	*ServiceUuid*	The Universally Unique ID (UUID) of the Bluetooth LE GATT profile service.
	AttributeHandle	The handle to the Bluetooth LE GATT profile attributes.

5.3.19 BtsdkGATTCharacteristicStru

Definition	typedef struct _BtsdkGATTCharacteristicStru { BTUINT16 ServiceHandle; BtsdkGATTUUIDStru CharacteristicUuid; BTUINT16 AttributeHandle; BTUINT16 CharacteristicValueHandle; BTINT32 IsBroadcastable; BTINT32 IsReadable; BTINT32 IsWritable; BTINT32 IsWritableWithoutResponse; BTINT32 IsSignedWritable; BTINT32 IsNotifiable; BTINT32 IsIndicatable; BTINT32 HasExtendedProperties; } BtsdkGATTCharacteristicStru, *PBtsdkGATTCharacteristicStru;	
Description	The structure BtsdkGATTCharacteristicStru describes a Bluetooth Low Energy(LE) generic attribute(GATT) profile characteristic.	
Members	*ServiceHandle*	The handle to the Bluetooth LE GATT profile service.
	CharacteristicUuid	The unique id of the characteristic.
	AttributeHandle	The handle to the Bluetooth LE GATT profile attributes.
	CharacteristicValueHandle	The handle to the Bluetooth LE GATT profile characteristic value.
	IsBroadcastable	The characteristic can be broadcast.
	IsReadable	The characteristic can be read.
	IsWritable	The characteristic can be written to.
	IsWritableWithoutResponse	The characteristic can be written to without requiring a response.
	IsSignedWritable	The characteristic can be signed writable.
	IsNotifiable	The characteristic can be updated by the device through the handle value notification, and the new value will be returned through the callback function registered via Btsdk_GATTRegisterCallback.

	IsIndicatable	The characteristic can be updated by the device through handle value notification, and the new value will be returned through the callback function registered via Btsdk_GATTRegisterCallback.
	HasExtendedProperties	The characteristic has extended properties, which will be presented through the characteristic extended properties descriptor.

5.3.20 BtsdkGATTCharacteristicValueStru

Definition	typedef struct _BtsdkGATTCharacteristicValueStru { BTUINT32 DataSize; BTUCHAR Data[1]; }BtsdkGATTCharacteristicValueStru, *PBtsdkGATTCharacteristicValueStru;	
Description	The structure BtSdkRemoteServiceAttrStru contains information about a remote service record.	
Members	*DataSize*	The size, in bytes, of the Bluetooth LE GATT characteristic value.
	Data[1]	A pointer to the Bluetooth LE GATT characteristic value data.

5.3.21 BtsdkGATTDescriptorStru

Definition	typedef struct _BtsdkGATTDescriptorStru { BTUINT16 ServiceHandle; BTUINT16 CharacteristicHandle; BTSDK_GATT_DESCRIPTOR_TYPE DescriptorType; BtsdkGATTUUIDStru DescriptorUuid; BTUINT16 AttributeHandle; } BtsdkGATTDescriptorStru, *PBtsdkGATTDescriptorStru;	
Description	The structure BtSdkLEGATTDescriptorStru describes a Bluetooth Low Energy(LE) generic attribute(GATT) profile descriptor.	
Members	*ServiceHandle*	The handle to the Bluetooth LE GATT profile service.
	CharacteristicHandle	The handle to the Bluetooth LE GATT profile characteristic.
	DescriptorType	The type of the Bluetooth LE GATT descriptor.
	DescriptorUuid	The unique idof the Bluetooth LE GATT descriptor.
	AttributeHandle	The handle to the Bluetooth LE GATT profile attribute.

5.3.22 BtsdkGATTDescriptorValueStru

<table>
<tr><td>Definition</td><td colspan="2">typedef struct _BtsdkGATTDescriptorValueStru {
BTSDK_GATT_DESCRIPTOR_TYPE DescriptorType;
BtsdkGATTUUIDStru DescriptorUuid;
union {
struct {
BTINT32 IsReliableWriteEnabled;
BTINT32 IsAuxiliariesWritable;
} CharacteristicExtendedProperties;
struct {
BTINT32 IsSubscribeToNotification;
BTINT32 IsSubscribeToIndication;
} ClientCharacteristicConfiguration;
struct {
BTINT32 IsBroadcast;
} ServerCharacteristicConfiguration;
struct {
BTUCHAR Format;
BTUCHAR Exponent;
BtsdkGATTUUIDStru Unit;
BTUCHAR NameSpace;
BtsdkGATTUUIDStru Description;
} CharacteristicFormat;
};
BTUINT32 DataSize;
BTUCHAR Data[1];
} BtsdkGATTDescriptorValueStru, *PBtsdkGATTDescriptorValueStru;</td></tr>
<tr><td>Description</td><td colspan="2">The structure BtsdkGATTDescriptorValueStru describes a parent characteristic.</td></tr>
<tr><td rowspan="3">Members</td><td>DescriptorType</td><td>The type of the descriptor value.</td></tr>
<tr><td>DescriptorUuid</td><td>The unique id of the descriptor value.</td></tr>
<tr><td>CharacteristicExtendedProperties</td><td>Container structure for the different characteristic extended property members.
IsReliableWriteEnabled
The parent characteristic value is reliable write enable.
IsAuxiliariesWritable
The characteristic user description descriptor is writable.</td></tr>
</table>

	ClientCharacteristicConfiguration	Container structure for the different client characteristic configuration members. **IsSubscribeToNotification** Whether the characteristic has been registered with the device to receive handle value notifications. TRUE if the characteristic has been registered. Otherwise, FALSE. **IsSubscribeToIndication** Whether the characteristic has been registered with the device to receive handle value indications. TRUE if the characteristic has been registered. Otherwise, FALSE.
	ServerCharacteristicConfiguration	Container structure for the different server characteristic configuration members. **IsBroadcast** The parent characteristic value can be broadcast.
	CharacteristicFormat	Container structure for the different characteristic format members. ***Format*** The format of the parent characteristic value. ***Exponent*** The exponent value to user to determine how the value of the characteristic value is further formatted. **Unit** The unit of the characteristic value as defined in the Assigned Numbers specification. **NameSpace** The name-space where the unit is defined in the assigned Numbers specification. **Description** The unique id that describes the format of the parent characteristic value.
	DataSize	The size, in bytes, of the descriptor value.
	Data[1]	A pointer to the descriptor value data.

5.4 API Functions

5.4.1 Initialization/Termination

5.4.1.1 Btsdk_Init

Prototype	void Btsdk_Init (void);	
Description	The **Btsdk_Init** function initializes context for subsequent BTSDK function calls.	
Parameters		
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

This function MUST be called and the return value MUST be BTSDK_OK before any other functions (except for *Btsdk_IsSDKInitialized*, and *Btsdk_IsBluetoothReady*) can be called.

This function initializes resources required to run the BlueSoleil. But it DOES NOT enable Bluetooth device. Function *Btsdk_StartBluetooth* must be called to enable Bluetooth device after initializing BlueSoleil successfully. This allows the application to implement a clear "Turn On Bluetooth" function.

After BlueSoleil is initialized successfully, the application can call any functions that require no communication with Bluetooth device. For example, the application can get a list of pre-configured paired devices.

Each successful call to *Btsdk_Init* must be balanced by a corresponding call to *Btsdk_Done* after subsequent BTSDK function calls are finished and BTSDK is no longer required.

This function is highly recommended to be called only once for successful initialization in an application.

5.4.1.2 Btsdk_Done

Prototype	void Btsdk_Done (void);	
Description	The **Btsdk_Done** function releases the context created by *Btsdk_Init*.	
Parameters		
Return:		

Remarks

An application must call *Btsdk_Done* once for each successful call it has made to *Btsdk_Init.*

This function releases all resources allocated by BlueSoleil functions and disables Bluetooth device finally. If the application wants to disable Bluetooth device only, it shall call *Btsdk_StopBluetooth* separately. This allows the application to implement a clear "Turn off Bluetooth" function.

5.4.1.3 Btsdk_IsSDKInitialized

Prototype	BTBOOL Btsdk_IsSDKInitialized (void);	
Description	The **Btsdk_IsSDKInitialized** function indicates whether a successful call to *Btsdk_Init* is made.	
Parameters		
Return:	If BTSDK is initialized successfully, the return value is BTSDK_TRUE. If BTSDK is not initialized, the return value is BTSDK_FALSE.	

Remarks

An application can call this function at any time to check the state of BlueSoleil.

5.4.1.4 Btsdk_IsServerConnected

Prototype	BTBOOL Btsdk_IsServerConnected();	
Description	The **Btsdk_IsServerConnected** function checks whether client application can call **BlueSoleil** Server APIs. When this fuction returns BTSDK_TRUE, client application can call APIs normally, versa versit.	
Parameters	*None*	
Return:	BTSDK_FALSE: Server isn't connected. BTSDK_TRUE: Server is connected.	

Remarks

5.4.1.5 Btsdk_RegisterCallback4ThirdParty

Prototype	BTINT32 Btsdk_RegisterCallback4ThirdParty (PbtSdkCallbackStru call_back);	
Description	The **Btsdk_RegisterCallback4ThirdParty** function registers an application-defined callback function.	
Parameters	*call_back*	[in] Pointer to a BtSdkCallbackStru structure that contains information about the callback function to be registered.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

This is an specific BlueSoleil SDK API.

A message from BlueSoleil is transferred to the application using a callback function. Only one callback function is allowed for one message. That is, if the application calls this Btsdk_RegisterCallback4ThirdParty twice to register different callback functions for the same message type, the second callback function will replace the first one.

If *call_back->func* is NULL, the call to Btsdk_RegisterCallback4ThirdParty will remove the callback for the specified message from BlueSoleil.

Table 6 lists the possible messages and callback function prototypes.

Example

```
/* This sample demonstrates how to register a callback to process inquiry result indication. */
void AppInquiryResultInd(BTDEVHDL dev_hdl)
{
    /* Process the Indication. */
}

void AppRegisterCallback(void)
{
```

```
    BtSdkCallbackStru cb;
    cb.type = BTSDK_INQUIRY_RESULT_IND;
    cb.func = (PVOID) AppInquiryResultInd;
    Btsdk_RegisterCallback4ThirdParty (&cb);
}
```

5.4.1.6 Btsdk_RegisterGetStatusInfoCB4ThirdParty

Prototype	BTINT32 Btsdk_RegisterGetStatusInfoCB4ThirdParty(Func_ReceiveBluetoothStatusInfo* statusCBK)	
Description	The **Btsdk_RegisterGetStatusInfoCB4ThirdParty** function registers a client process callback function to **BlueSoleil** to receive Bluetooth Status changing event.	
Parameters	statusCBK	[in] pointer to Func_ReceiveBluetoothStatusInfo, whose prototype is defined in bssdk_ui.h
Return:	BTSDK_OK for success, other for error code.	

Remarks

We use **Btsdk_RegisterGetStatusInfoCB4ThirdParty** to register a callback function to deal with status change of **BlueSoleil**. If a user doesn't want to deal with the callback events any more, he should use Btsdk_RegisterGetStatusInfoCB4ThirdParty (NULL) to un-register the callback function.

5.4.1.7 Btsdk_SetStatusInfoFlag

Prototype	BTINT32 Btsdk_SetStatusInfoFlag(USHORT usMsgType);	
Description	The **Btsdk_SetStatusInfoFlag** function is used to set the status changing callback types which the user wants to receive.	
Parameters	*usMsgType*	Message type which user wants to receive.
Return:	BTSDK_OK for success, other for error code.	

Remarks

A client process can just register one flag to BlueSoleil server, That is, if a client process calls this Btsdk_SetStatusInfoFlag twice to register different flags, the second flag will replace the first one.

usMsgType can be one of the following value or their combination:

BTSDK_NTSERVICE_STATUS_FLAG	The status change of **BlueSoleil** server event or OS message event.
BTSDK_BLUETOOTH_STATUS_FLAG	Message event of the change of Bluetooth status.
BTSDK_REFRESH_STATUS_FLAG	Refresh event.

5.4.1.8 Func_ReceiveBluetoothStatusInfo

Prototype	typedef void Func_ReceiveBluetoothStatusInfo(ULONG usMsgType, ULONG pulData, ULONG param, BTUINT8 *arg);	
Description	The function prototype of the function to deal with change of **BlueSoleil**'s status.	
Parameters	*usMsgType*	Message type
	pulData	Message event relative to usMsgType.
	param	Be different according to the difference of usMsgType and pulData.
	arg	Be different according to the difference of usMsgType and pulData.
Return:		

Remarks

All the messages of **BlueSoleil** are dealt with by using callback function. If a user wants to deal with status changes of **BlueSoleil** server, a callback function using **Btsdk_RegisterGetStatusInfoCB4ThirdParty** should be registered.
The following table indicates the relationship of usMsgType , pulData, param and Arg.

usMsgType	pulData	Description	Param	Arg
BTSDK_BLUETOOTH_STATUS_FLAG	BTSDK_BTSTATUS_TURNON	Bluetooth is turned on	Not used	Not used
	BTSDK_BTSTATUS_TURNOFF	Bluetooth is turned off		
	BTSDK_BTSTATUS_HWPLUGGED	Bluetooth hardware is plugged.		
	BTSDK_BTSTATUS_HWPULLED	Bluetooth hardware is pulled.		

Example

```
/* This sample demonstrates how to set the flag and register a callback to process status change event. */
void BsStatusCBKFuc(ULONG usMsgType, ULONG pucData, ULONG param, BTUINT8 *arg)
```

```
{
    switch(usMsgType)
     {
    case BTSDK_REFRESH_STATUS_FLAG:
            {
              Switch(pucData)
             {
             case BTSDK_DEL_DEVICE:
               //do something
             break;
             case BTSDK_UNPAIR_DEVICE:
            //do something
            break;
            …..
            }
     case BTSDK_BLUETOOTH_STATUS_FLAG :
             {
               Switch(pucData)
              {
               case BTSDK_BTSTATUS_TURNON;
               // do something
               break;
                 ……
               }
            }
      case ….
    …..
      }
}

Btsdk_SetStatusInfoFlag(BTSDK_NTSERVICE_STATUS_FLAG|
                        BTSDK_BLUETOOTH_STATUS_FLAG|
                        BTSDK_REFRESH_STATUS_FLAG);
Btsdk_RegisterGetStatusInfoCB4ThirdParty(BsStatusCBKFuc);
```

5.4.2 Memory Management

5.4.2.1 Btsdk_MallocMemory

Prototype	void* Btsdk_MallocMemory (BTUINT32 size;);	
Description	The **Btsdk_MallocMemory** function allocates memory block, which will be passed to the BlueSoleil through BlueSoleil API and released by BlueSoleil module finally, for the upper application.	
Parameters	*size*	[in] Bytes to allocate.
Return:	The pointer to the allocated space, or **NULL** if there is insufficient memory available.	

5.4.2.2 Btsdk_FreeMemory

Prototype	void Btsdk_FreeMemory (void *memblock;);	
Description	The **Btsdk_FreeMemory** function is used for the upper application to free the memory allocated by Btsdk_MallocMemory.	
Parameters	*memblock*	[in] Memory block to be freed.
Return:	None.	

5.4.3 Local Bluetooth Device Management

5.4.3.1 Device Initialization

5.4.3.1.1 Btsdk_StartBluetooth

Prototype	BTINT32 Btsdk_StartBluetooth (void);	
Description	The **Btsdk_StartBluetooth** function enables the local device and initializes the device settings to values configured recently. This function also reads device features required by BlueSoleil Host Protocol Stack.	
Parameters		
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

This function should be called and the return value expected to be BTSDK_OK before any other functions that require communication with Bluetooth device can be called.

5.4.3.1.2 Btsdk_StopBluetooth

Prototype	BTINT32 Btsdk_StopBluetooth (void);	
Description	The **Btsdk_StopBluetooth** function disables the local device.	
Parameters		
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

This function only disables the local device. It doesn't release the resources allocated by other BlueSoleil functions.

After the application makes a successful call to *Btsdk_Init*, it can call *Btsdk_StartBluetooth* and *Btsdk_StopBluetooth* functions repeatedly to implement "Turn on Bluetooth" and "Turn off Bluetooth" functions.

5.4.3.1.3 Btsdk_IsBluetoothReady

Prototype	BTBOOL Btsdk_IsBluetoothReady (void);	
Description	The **Btsdk_IsBluetoothReady** function checks whether the local Bluetooth device is working.	
Parameters		
Return:	If Bluetooth device is enabled, the return value is BTSDK_TRUE. If Bluetooth device is disabled, the return value is BTSDK_FALSE.	

Remarks

An application can call this function at any time to check the working state of the current local device.

5.4.3.1.4 Btsdk_IsBluetoothHardwareExisted

Prototype	BTBOOL Btsdk_IsBluetoothHardwareExisted();	
Description	The **Btsdk_IsBluetoothHardwareExisted** function checks whether Bluetooth hardware exists.	
Parameters		
Return:	BTSDK_TRUE: Bluetooth Hardware exists. BTSDK_FALSE: Bluetooth Hardware not exists.	

Remarks

5.4.3.1.5 Btsdk_ SetSecurityMode

Prototype	BTUINT32 Btsdk_SetSecurityMode (BTUINT16 secu_mode);	
Description	The **Btsdk_SetSecurityMode** function sets the security mode for the local bluetooth device.	
Parameters	*security_mode*	[in] Specify the new security mode. BTSDK_SECURITY_LOW: non-secure BTSDK_SECURITY_MEDIUM: service level enforced security BTSDK_SECURITY_HIGH: link level enforced security without encryption. BTSDK_SECURITY_ENCRYPT_MODE1: For pre-2.1 dongle, it is link level enforced security with encryption. For 2.1 dongle, it is service level enforced security with Simple Pairing mode enabled.
Return:	BTSDK_OK for success other for error code	

Remarks

5.4.3.2 Device Modes

5.4.3.2.1 Btsdk_SetDiscoveryMode

Prototype	BTINT32 Btsdk_SetDiscoveryMode (BTUINT16 mode);	
Description	The **Btsdk_SetDiscoveryMode** function sets the accessibility modes of the local device.	
Parameters	*mode*	[in] Specifies the modes to be set. It can be one or more of the values listed in Table 5.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

Before calling *Btsdk_SetDiscoveryMode*, the local device must be enabled by a previous successful call to *Btsdk_StartBluetooth*. By default, the local device is in general discoverable mode, connectable mode and pairable mode.

If the application wants to make local device non-discoverable, it must call *Btsdk_SetDiscoveryMode* with none of BTSDK_GENERAL_DISCOVERABLE, BTSDK_DISCOVERABLE and BTSDK_LIMITED_DISCOVERABLE specified in *mode* parameter.

If BTSDK_CONNECTABLE is not specified in *mode* parameter, local device is set to non-connectable mode. If BTSDK_PAIRABLE is not specified in *mode* parameter, local device is set to non-pairable mode.

Example

```
/* This sample demonstrates how to set local device mode. */
void AppChangeMode (void)
{
    /* Make local device discoverable, connectable and non-pairable. */
    BTUINT16 mode = BTSDK_DISCOVERABLE | BTSDK_CONNECTABLE;
    Btsdk_SetDiscoveryMode(mode);
    /* To do: Add other operation. */
    /* Make local device non-discoverable, connectable and pairable. */
    mode = BTSDK_CONNECTABLE | BTSDK_PAIRABLE.
```

```
    Btsdk_SetDiscoveryMode(mode);
    /* To do: Add other operation. */
}
```

5.4.3.2.2 Btsdk_GetDiscoveryMode

Prototype	BTINT32 Btsdk_GetDiscoveryMode (BTUINT16* pmode);	
Description	The **Btsdk_GetDiscoveryMode** function gets the accessibility modes of the local device.	
Parameters	*pmode*	[out] Pointer to a variable that receives the modes of the local device. The return value can be one or more of the values listed in Table 5.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

Before calling *Btsdk_GetDiscoveryMode*, the local device must be enabled by a previous successful call to *Btsdk_StartBluetooth*.

If none of BTSDK_GENERAL_DISCOVERABLE, BTSDK_DISCOVERABLE and BTSDK_LIMITED_DISCOVERABLE values are specified in **pmode* parameter, local device is in non-discoverable mode.

If BTSDK_CONNECTABLE value is not specified in **pmode* parameter, local device is in non-connectable mode.

If BTSDK_PAIRABLE value is not specified in **pmode* parameter, local device is in non-pairable mode.

5.4.3.3 Device Information

5.4.3.3.1 Btsdk_GetLocalDeviceAddress

Prototype	BTINT32 Btsdk_GetLocalDeviceAddress (BTUINT8* bd_addr,);	
Description	The **Btsdk_GetLocalDeviceAddress** function gets the Bluetooth device address of the local device.	
Parameters	*bd_addr*	[out] Pointer to the buffer that receives the device address. The size, in bytes, of this buffer must be large enough to hold the 6bytes address value.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

Before calling *Btsdk_GetLocalDeviceAddress*, the local device must be enabled by a previous successful call to *Btsdk_StartBluetooth*.

5.4.3.3.2 Btsdk_SetLocalName

Prototype	BTINT32 Btsdk_SetLocalName (BTUINT8* name, BTUINT16 len);	
Description	The **Btsdk_SetLocalName** function sets the name of the local device.	
Parameters	*name*	[in] Pointer to the buffer containing the string to be used as the device name. This string must be coded in UTF-8 format.
	len	[in] Specifies the size in bytes of the string pointed to by the *name* parameter. It must be no more than BTSDK_DEVNAME_LEN. The exceeding bytes are ignored by BTSDK.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

Before calling *Btsdk_SetLocalName*, the local device must be enabled by a previous successful call to *Btsdk_StartBluetooth*.

5.4.3.3.3 Btsdk_GetLocalName

Prototype	BTINT32 Btsdk_GetLocalName (BTUINT8* name, BTUINT16* plen);	
Description	The **Btsdk_GetLocalName** function gets the name of the local device.	
Parameters	*name*	[out] Pointer to the buffer that receives the device name. This parameter can be NULL.
	plen	[in/out] Pointer to a variable that, on input, specifies the size, in bytes, of the buffer pointed to by the *name* parameter, or it can be NULL if the buffer size is larger than BTSDK_DEVNAME_LEN. On output, This variable receives the number of bytes copied to the buffer pointed to by the *name* parameter. To determine the required buffer size, call this function with *name* set to NULL. This function returns the required buffer size in **plen*.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

Before calling *Btsdk_GetLocalName*, the local device must be enabled by a previous successful call to *Btsdk_StartBluetooth*.

The device name is a UTF-8 character string.

5.4.3.3.4 Btsdk_SetLocalDeviceClass

Prototype	BTINT32 Btsdk_SetLocalDeviceClass (BTUINT32 device_class);	
Description	The **Btsdk_SetLocalDeviceClass** function sets the Class of Device/Service field of the local device.	
Parameters	*device_class*	[in] Specifies the Class of Device/Service value to be set. It can be one of the device class identifiers listed in Table 3 combined with multiple major service class identifiers listed in Table 4.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

Before calling *Btsdk_SetLocalDeviceClass*, the local device must be enabled by a previous successful call to *Btsdk_StartBluetooth*.

The default Class of Device/Service value of the local device is un-specified. The application shall call this function at least once to specify a proper value according to the usage scenario.

Example

```
/* This sample demonstrates how to set Class of Device/Service value. */
void AppChangeCoD (void)
{
    /* Set local device as a desktop PC.
        Furthermore, specifies that services of Networking and Object Transfer type are available. */
    BTUINT32 dev_class = BTSDK_COMPCLS_DESKTOP | BTSDK_SRVCLS_NETWORK |
                         BTSDK_SRVCLS_OBJECT;
    Btsdk_SetLocalDeviceClass(dev_class);
}
```

5.4.3.3.5 Btsdk_GetLocalDeviceClass

Prototype	BTINT32 Btsdk_GetLocalDeviceClass (BTUINT32* pdevice_class);	
Description	The **Btsdk_GetLocalDeviceClass** function gets the Class of Device/Service field value of the local device.	
Parameters	*pdevice_class*	[out] Pointer to a variable that receives the Class of Device/Service value of the local device. The return value can be one of the device class identifiers listed in Table 3 combined with multiple major service class identifiers listed in Table 4.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

Before calling *Btsdk_GetLocalDeviceClass*, the local device must be enabled by a previous successful call to *Btsdk_StartBluetooth*.

5.4.3.3.6 Btsdk_GetLocalLMPInfo

Prototype	BTINT32 Btsdk_GetLocalLMPInfo (PBtSdkLocalLMPInfoStru plmp_info);	
Description	The **Btsdk_GetLocalLMPInfo** function gets information about the HCI and LMP in the local device.	
Parameters	*plmp_info*	[out] Pointer to a BtSdkLocalLMPInfoStru structure that receives the information about the HCI and LMP in the local device.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

Before calling *Btsdk_GetLocalLMPInfo*, the local device must be enabled by a previous successful call to *Btsdk_StartBluetooth*.

5.4.3.3.7 Btsdk_SetFixedPinCode

Prototype	BTINT32 Btsdk_SetFixedPincode (BTUINT8 *pin_code, BTUINT16 size);	
Description	The **Btsdk_SetFixedPinCode** function sets a fixed PIN code for the local device.	
Parameters	*pin_code*	[in] Pointer to the fixed PIN code.
	size	[in] The size of the fixed PIN code. If the size is bigger than BTSDK_PIN_CODE_LEN, the length of pin_code will be cut to BTSDK_PIN_CODE_LEN.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

5.4.3.3.8 Btsdk_GetFixedPinCode

Prototype	BTINT32 Btsdk_GetFixedPincode (BTUINT8 *pin_code, BTUINT16 *psize);	
Description	The **Btsdk_GetFixedPinCode** function gets a fixed PIN code of the local device.	
Parameters	*pin_code*	[out] Pointer to the fixed PIN code.
	psize	[in/out] The size of the fixed PIN code. If *psize* is not NULL, *psize shall specify the maximum length of the 'pin_code'. If psize is NULL, the length of 'pin_code' buffer should not be less than BTSDK_PIN_CODE_LEN. The varible *psize returns the actually copied number.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

5.4.3.4 Application Extension

5.4.3.4.1 Btsdk_VendorCommand

Prototype	BTINT32 Btsdk_VendorCommand (BTUINT32 ev_flag, PBtSdkVendorCmdStru in_cmd, PBtSdkEventParamStru out_ev);	
Description	The **Btsdk_VendorCommand** function is used to send a vendor specific HCI command to the local device and receives the corresponding event.	
Parameters	*ev_flag*	[in] Specifies the events generated for the specified command. It is reserved for future extension. Always set it to 0.
	in_cmd	[in] Pointer to a BtSdkVendorCmdStru structure specifies the vendor specific command to be sent to the local device.
	out_ev	[out] Pointer to a BtSdkEventParamStru structure to receive the event generated for the command specified by *in_cmd* parameter.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

Before calling *Btsdk_VendorCommand*, the local device must be enabled by a previous successful call to *Btsdk_StartBluetooth*.

Btsdk_VendorCommand can be used to issue a command that generates only a command complete event or a vendor specific event. If more than one event are generated for the specified command, the behavior of BlueSoleil is undefined currently.

The return value BTSDK_OK only confirms that the specified command has been sent to the Bluetooth device and, a command complete event for this command or a vendor specific event is generated. The application shall examine the output event for the actual result itself. For example, if the command generates a command complete event and a “Status” parameter in the return parameters specifying the result, the application shall check the value of “Status” parameter.

5.4.3.4.2 Btsdk_EnumAudioDriver

Prototype	BTUINT32 Btsdk_EnumAudioDriver ();	
Description	The **Btsdk_ EnumAudioDriver** function enumerates the audio card installed on local machine.	
Parameters		
Return:	The return value indicates whether the audio card is plugged in succesfully on local machine. The audio can either be Advanced Audio sound card or SCO sound card. 0 means operation failed. 1 means the audio card is plugged in successfully. 2 means the audio card has already been plugged.	

Remarks

.

5.4.3.4.3 Btsdk_DeEnumAudioDriver

Prototype	void Btsdk_DeEnumAudioDriver();	
Description	The **Btsdk_DeEnumAudioDriver** function unplugs the audio card installed on local machine.	
Parameters		
Return:		

Remarks

5.4.3.4.4 Btsdk_ActivateEx

Prototype	BTINT32 Btsdk_ActivateEx (const BTINT8 *pszSN, BTINT32 iSnlen);	
Description	The **Btsdk_ActivateEx** function activates BlueSoleil by Serial Number for third party.	
Parameters	*pszSN*	[in] Pointer to the buffer that receives character string of the serial number.
	iSnlen	[in] Length of character string of serial number.
Return:	If BlueSoleil is successfully activated, the return value is BTSDK_OK. If the serial number is not inputted correctly, the return value is BTSDK_ER_INVALID_PARAMETER. Other return value indicates there is a network malfunction or SDK is not initialized.	

Remarks

It will take several seconds for this function to return its value. Consequently, call this function in another thread in order not to block the main thread.

5.4.4 Remote Bluetooth Device Management

This section describes the interface functions used to:

- Discover other nearby Bluetooth devices.
- Retrieve information about other Bluetooth devices.
- Pair or un-pair other Bluetooth devices.
- Manage the link with other Bluetooth devices.
- Manage the Remote device database.

5.4.4.1 Device Discovery

5.4.4.1.1 Btsdk_StartDeviceDiscovery

Prototype	BTINT32 Btsdk_StartDeviceDiscovery (BTUINT32 device_class, BTUINT16 max_num, BTUINT16 max_durations);	
Description	The **Btsdk_StartDeviceDiscovery** function makes the Bluetooth device start an inquiry procedure. This procedure is used to discover other nearby Bluetooth devices. A remote device that responds during the inquiry procedure is reported to the application through a BTSDK_INQUIRY_RESULT_IND message. The message BTSDK_INQUIRY_COMPLETE_IND is reported to the application when the inquiry procedure has completed.	
Parameters	*device_class*	[in] Specifies the Class of Device of interest. That is, only a device with the Class of Device specified by *device_class* parameter will be reported to the application. The application can specify one of the device class identifiers listed in Table 3. If this value is set to 0, BlueSoleil reports all devices discovered to the application.
	max_num	[in] Specifies the maximum number of responses during the inquiry procedure. Range of this value is from 0x00 to 0xFF. If this value is set to 0, the number of responses is unlimited.
	max_durations	[in] Specifies the maximum amount of time before the inquiry is halted. The actual duration in seconds is (max_durations * 1.28). Range of this value is from 0x01 to 0x30. If this value is set to 0, BTSDK adopts a default value of 10 instead.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

Before calling *Btsdk_StartDeviceDiscovery*, the local device must be enabled by a previous successful call to *Btsdk_StartBluetooth*.

A device discovered during the inquiry procedure is automatically stored in the device database and marked as an "Inquired" device. The "Inquired" flag will be kept until the next time *Btsdk_StartDeviceDiscovery* or *Btsdk_Done* is called. The application can refer to all "Inquired" devices by calling *Btsdk_GetInquiredDevices* in the future.

The application shall register at least a callback function BlueSoleil to process BTSDK_INQUIRY_COMPLETE_IND message, which indicates that the inquiry procedure has completed. To refer to the devices discovered, the application can register a callback function to BlueSoleil to process BTSDK_INQUIRY_RESULT_IND message, or call *Btsdk_GetInquiredDevices* after the inquiry procedure terminates.

5.4.4.1.2 Btsdk_Inquiry_Result_Ind_Func

Prototype	typedef void (Btsdk_Inquiry_Result_Ind_Func) (BTDEVHDL device_handle);	
Description	The **Btsdk_Inquiry_Result_Ind_Func** function prototype is the prototype of application defined callback function used to process BTSDK_INQUIRY_RESULT_IND message.	
Parameters	*device_handle*	[in] Handle assigned to the remote device discovered during the inquiry procedure.
Return:		

Remarks

This callback function is called to report each device discovered separately.

All information of the device discovered is stored in the device database. Each device record in the database is represented by a unique 32bit unsigned integer named as device handle. The handle value is reported to the application through *device_handle* parameter. And the application can call functions *Btsdk_GetRemoteDeviceAddress*, *Btsdk_GetRemoteDeviceClass* and *Btsdk_GetRemoteDeviceName* to get device information from the device database in the future.

Device handle value returned by *device_handle* parameter is valid until the device record is removed by *Btsdk_DeleteRemoteDeviceByHandle*, *Btsdk_DeleteUnpairedDevicesByClass*, or until *Btsdk_Done* is called to terminate using the Bluesoleil.

DO NOT call inside this callback function any functions, e.g. function that waits for a semaphore or requires the user interference, which may block internal thread of BlueSoleil. DO NOT call inside this callback function any BTSDK functions that require communicating with a remote device, either, e.g. *Btsdk_PairDevice*, *Btsdk_Connect* and so on. Furthermore, current version BlueSoelil doesn't support pairing or connecting to a remote device before inquiry procedure is completed.

5.4.4.1.3 Btsdk_Inquiry_Complete_Ind_Func

Prototype	typedef void (Btsdk_Inquiry_Complete_Ind_Func) (void);	
Description	The **Btsdk_Inquiry_Complete_Ind_Func** function prototype is the prototype of application defined callback function used to process BTSDK_INQUIRY_COMPLETE_IND message.	
Parameters		
Return:		

Remarks

This callback function is called when the inquiry procedure has completed.

DO not call inside this callback function any functions, e.g. function that waits for a semaphore or requires the user interference, which may block internal thread of BlueSoleil. DO not call inside this callback function any BlueSoleil functions that require communicating with a remote device either, e.g. *Btsdk_PairDevice*, *Btsdk_Connect* and so on. If the application wants to pair or connect to remote device(s) soon after inquiry procedure finishes, it shall call related functions in another thread.

5.4.4.1.4 Btsdk_StopDeviceDiscovery

Prototype	BTINT32 Btsdk_StopDeviceDiscovery(void);	
Description	The **Btsdk_StopDeviceDiscovery** function stops the ongoing discovery procedure initiated by a previous call to *Btsdk_StartDeviceDiscovery* function.	
Parameters		
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

Before calling *Btsdk_StopDeviceDiscovery*, the local device must be enabled by a previous successful call to *Btsdk_StartBluetooth*.

After the device discovery procedure is terminated by the *Btsdk_StopDeviceDiscovery* function, no BTSDK_INQUIRY_ COMPLETE_IND message will be reported to the application.

5.4.4.1.5 Btsdk_UpdateRemoteDeviceName

Prototype	BTINT32 Btsdk_UpdateRemoteDeviceName (BTDEVHDL device_handle, BTUINT8* name, BTUINT16* plen);	
Description	The **Btsdk_UpdateRemoteDeviceName** function gets the current user-friendly name of the specified remote device.	
Parameters	*device_handle*	[in] Handle to the remote device object.
	name	[out] Pointer to the buffer that receives the device name. This parameter can be NULL.
	plen	[in/out] Pointer to a variable that, on input, specifies the size, in bytes, of the buffer pointed to by the *name* parameter, or it can be NULL if the buffer size is larger than BTSDK_DEVNAME_LEN. On output, This variable receives the number of bytes copied to the buffer pointed to by the *name* parameter. To determine the required buffer size, call this function with *name* set to NULL. This function returns the required buffer size in **plen*.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

Before calling *Btsdk_UpdateRemoteDeviceName*, the device database must be initialized by a previous successful call to *Btsdk_StartBluetooth*.

The user-friendly device name is a UTF-8 character string. The device name acquired by this command is stored automatically in the device database.

5.4.4.1.6 Btsdk_CancelUpdateRemoteDeviceName

Prototype	BTINT32 Btsdk_CancelUpdateRemoteDeviceName (BTDEVHDL device_handle,);	
Description	The **Btsdk_CancelUpdateRemoteDeviceName** function cancels ongoing remote device name update process initiated by the *Btsdk_UpdateRemoteDeviceName* function.	
Parameters	*device_handle*	[in] Handle to the remote device object. It must be the same value as that of *device_handle* parameter of *Btsdk_UpdateRemoteDeviceName*.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

Before calling *Btsdk_CancelUpdateRemoteDeviceName*, the device database must be initialized by a previous successful call to *Btsdk_StartBluetooth*.

If the cancellation is successful, Btsdk_UpdateRemoteDeviceName returns error code BTSDK_ER_NO_CONNECTION immediately.

The Btsdk_CancelUpdateRemoteDeviceName function returns error code BTSDK_ER_UNKNOWN_COMMAND immediately, if the local device does not support the cancellation of remote device name request process.

5.4.4.1.7 Btsdk_DeviceFound_Func

Prototype	typedef void (Btsdk_DeviceFound_Func)(BTDEVHDL dev_hdl);	
Description	The **Btsdk_DeviceFound_Func** function prototype is the prototype of application defined callback function used to process BTSDK_DEVICE_FOUND_IND message.	
Parameters	*device_handle*	[in] Handle assigned to the remote device discovered during the inquiry procedure.
Return:		

Remarks

This callback function is called when the device information changes or LE device enters the range of communication.

DO NOT call inside this callback function any functions, e.g. function that waits for a semaphore or requires the user interference, which may block internal thread of BlueSoleil. DO NOT call inside this callback function any BlueSoleil functions that require communicating with a remote device either, e.g. *Btsdk_Connect* and so on.

5.4.4.2 Device Pairing

5.4.4.2.1 Btsdk_IsDevicePaired

Prototype	BTINT32 Btsdk_IsDevicePaired (BTDEVHDL dev_hdl, BTBOOL *pis_paired);	
Description	The **Btsdk_IsDevicePaired** function checks if the remote device is paired or not.	
Parameters	*dev_hdl*	[in] Handle of the remote device.
	pis_paired	[out] Pointer to the variable of the condition, BTSDK_TRUE or BTSDK_FALSE.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code.	

Remarks

Before calling *Btsdk_IsDevicePaired*, the device database must be initialized by a previous successful call to *Btsdk_Init*.

5.4.4.2.2 Btsdk_PairDevice

Prototype	BTINT32 Btsdk_PairDevice (BTDEVHDL device_handle,);	
Description	The **Btsdk_PairDevice** function pairs the specified remote device.	
Parameters	*device_handle*	[in] Handle to the device to be paired.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

Before calling *Btsdk_PairDevice*, the local device must be enabled by a previous successful call to *Btsdk_StartBluetooth*.

After a successful pairing, the new link key is stored automatically in the device database, and the remote device is marked as a "Paired" device. The link key and the "Paired" flag will be kept until the next time *Btsdk_PairDevice* or *Btsdk_UnPairDevice* function is called, or the authentication process with this remote device fails for some reasons (e.g., the remote device deletes the link key.). The application can refer to all "Paired" devices by calling *Btsdk_GetPairedDevices* in the future.

Do not call Btsdk_PairDevice inside a window's SendMessage handler function, which may block message-processing thread and cause PINCODE dialog cannot pop up properly.

5.4.4.2.3 Btsdk_UnPairDevice

Prototype	BTINT32 Btsdk_UnPairDevice (BTDEVHDL device_handle,);	
Description	The **Btsdk_UnPairDevice** function removes the link key and the "Paired" flag of the specified device from the device database.	
Parameters	*device_handle*	[in] Handle to the device to be unpaired.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

Before calling *Btsdk_UnPairDevice*, the device database must be initialized by a previous successful call to *Btsdk_Init*.

After the application calls Btsdk_UnPairDevice to abolish the pair relation with a remote device, the remote device itself may still think of local device as a "Paired" device.

5.4.4.2.4 Btsdk_RegisterCallbackEx

Prototype	BTINT32 Btsdk_RegisterCallbackEx (BtSdkCallBackStru* call_back, DWORD priority);	
Description	The **Btsdk_RegisterCallbackEx** function is a extension callback function processing pairing and authentication events of the third party.	
Parameters	*call_back*	[in] Pointer to a BtSdkCallbackStru structure that contains information about the callback function to be registered.
	priority	[in] Specifies the priority of pairing proccessing.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

The default processing priority set by BlueSoleil is “low”. If the processing priority is handled by this call back function, it will not be handled by BlueSoleil. While if the processing priority is not handled by this call back function, it will be handled by BlueSoleil as “low”.

The **priority** parameter can be one of these value

Value	Description
BTSDK_CLIENTCBK_PRIORITY_HIGH	Indicates the priority is “high”
BTSDK_CLIENTCBK_PRIORITY_MEDIUM	Indicates the priority is “medium”

5.4.4.2.5 Btsdk_UserHandle_Pin_Req_Ind_Func

Prototype	typedef BTUINT8 (Btsdk_UserHandle_Pin_Req_Ind_Func) (BTDEVHDL dev_hdl);	
Description	The **Btsdk_UserHandle_Pin_Req_Ind_Func** function prototype is the prototype of application defined callback function used to process BTSDK_PIN_CODE_IND message	
Parameters	*dev_hdl*	[in] Handle to the remote device that a PIN code is required to create the new link key for.
Return:	Refer to the table below.	

Remarks

This callback function should return immediately, and the pairing should be handled through another thread. Otherwise BlueSoleil will be blocked.

The return value can be one of these:

Value	Description
BTSDK_CLIENTCBK_HANDLED	It indicates that the client callback is handled.
BTSDK_CLIENTCBK_NOTHANDLED	It indicates that the client callback is not handled.

5.4.4.2.6 Btsdk_UserHandle_Authorization_Req_Ind_Func

Prototype	typedef BTUINT8 (Btsdk_UserHandle_Authorization_Req_Ind_Func) (BTSVCHDL svc_hdl, BTDEVHDL dev_hdl);	
Description	The **Btsdk_UserHandle_Authorization_Req_Ind_Fun** function prototype is the prototype of application defined callback function used to process BTSDK_AUTHORIZATION_IND message	
Parameters	*svc_hdl*	[in] Handle to the local service record that the remote device specified by the *device_handle* tries to connect to.
	dev_hdl	[in] Handle to the remote device that tries to connect to the local service record specified by the *service_handle*.
Return:	Refer to the table below.	

Remarks

This callback function should return immediately, and the pairing processing should be handled through another thread. Otherwise BlueSoleil will be blocked.

The return value can be one of these:

Value	Description
BTSDK_CLIENTCBK_HANDLED	It indicates that the client callback is handled.
BTSDK_CLIENTCBK_NOTHANDLED	It indicates that the client callback is not handled.

5.4.4.2.7 Btsdk_PinCodeReply

Prototype	BTINT32 Btsdk_PinCodeReply (BTDEVHDL device_handle, BTUINT8* pin_code, BTUINT16 pin_len);	
Description	The **Btsdk_PinCodeReply** function is used to reply the PIN code request during the pair procedure.	
Parameters	*device_handle*	[in] Handle to the remote device to be paired.
	pin_code	[in] Pointer to the buffer contains the PIN code. If the *pin_code* parameter is set to NULL, BlueSoleil sends "HCI PIN Code Request Negative Reply Command" and the pair request fails.
	pin_len	[in] Specifies the length, in bytes, of the PIN code to be used. If the *pin_len* parameter is set to 0, BlueSoleil sends "HCI PIN Code Request Negative Reply Command" and the pair request fails.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

The application shall call the *Btsdk_PinCodeReply* function to reply the PIN code request after it receives the BTSDK_PIN_CODE_IND message.

5.4.4.2.8 Btsdk_AuthorizationResponse

Prototype	BTUINT32 Btsdk_AuthorizationResponse (BTSVCHDL service_handle, BTDEVHDL device_handle, BTUINT16 author_response);	
Description	The **Btsdk_AuthorizationResponse** function accepts or rejects the authorization request.	
Parameters	*service_handle*	[in] Handle to the local service record that the remote device specified by the *device_handle* tries to connect to.
	device_handle	[in] Handle to the remote device that tries to connect to the local service record specified by the *service_handle*.
	author_response	[in] BTSDK_AUTHORIZATION_GRANT to accept the authorization request, or BTSDK_AUTHORIZATION_DENY otherwise.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

The application shall call the *Btsdk_AuthorizationResponse* function to reply the authorization request after it receives the BTSDK_AUTHORIZATION_IND message.

5.4.4.2.9 Btsdk_Link_Key_Notif_Ind_Func

Prototype	typedef void (Btsdk_Link_Key_Notif_Ind_Func) (BTDEVHDL device_handle, BTUINT8* link_key);	
Description	The **Btsdk_Link_Key_Notif_Ind_Func** function prototype is the prototype of application defined callback function used to process BTSDK_LINK_KEY_NOTIF_IND message.	
Parameters	*device_handle*	[in] Handle to the remote device that a new link key is created for.
	link_key	[in] Pointer to the buffer contains the new link key created.
Return:		

Remarks

This callback function is always called when the pairing succeeds, no matter which side initiates the pairing procedure.

DO NOT call inside this callback function any functions, e.g. function that waits for a semaphore or requires the user interference, which may block internal thread of BlueSoleil. DO NOT call inside this callback function any BlueSoleil functions that require communicating with a remote device either, e.g. *Btsdk_Connect* and so on.

5.4.4.2.10 Btsdk_Authentication_Fail_Ind_Func

Prototype	typedef void (Btsdk_Authentication_Fail_Ind_Func) (BTDEVHDL device_handle,);	
Description	The **Btsdk_Authentication_Fail_Ind_Func** function prototype is the prototype of application defined callback function used to process BTSDK_AUTHENTICATION_FAIL_IND message.	
Parameters	*device_handle*	[in] Handle to the remote device with which the pairing or authentication fails.
Return:		

Remarks

This callback function is always called when the pairing or authentication fails, no matter which side initiates the pairing or authentication procedure.

DO NOT call inside this callback function any functions, e.g. function that waits for a semaphore or requires the user interference, which may block internal thread of BlueSoleil. DO NOT call inside this callback function any BlueSoleil functions that require communicating with a remote device either, e.g. *Btsdk_Connect* and so on.

5.4.4.2.11 Btsdk_Link_Key_Req_Ind_Func

Prototype	typedef void (Btsdk_Link_Key_Req_Ind_Func) (BTDEVHDL device_handle,);	
Description	The **Btsdk_Link_Key_Req_Ind_Func** function prototype is the prototype of application defined callback function used to process BTSDK_LINK_KEY_REQ_IND message.	
Parameters	*device_handle*	[in] Handle to the remote device with which the application needs to give the link key.
Return:		

Remarks

This callback function will be called before the pairing process, only if the application had registerdthe callback and Bluesoleil had no link key for the remote device.
In other words, if the application had never registerd this call back, Bluesoleil will return OPNO_LINK_KEY_REQ_NEGREPLY to remote device. Then the remote device will start the pariing process and create new link key if it needs.
If the application had registerd this callback, when receiveing this indication, application must call Btsdk_LinkKeyReply in the call back function.

5.4.4.2.12 Btsdk_LinkKeyReply

Prototype	BTINT32 Btsdk_LinkKeyReply (BTDEVHDL device_handle, BTUINT8* link_key);	
Description	The **Btsdk_LinkKeyReply** function is used to reply the link key request before the connection established.	
Parameters	*device_handle*	[in] Handle to the remote device to be connected.
	link_key	[in] Pointer to the 16 bytes buffer contains the lin key. If the *link_key* parameter is set to NULL, BlueSoleil sends "Link key Request Negative Reply Command" and the remote device can start the pair procedure to create a new link key.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

The application shall call the Btsdk_LinkKeyReply function to reply the LinK key request after it receives the BTSDK_LINK_KEY_REQ_IND message.

5.4.4.3 Link Management

This section describes the interface functions used to acquire and modify the status of the ACL link.

5.4.4.3.1 Btsdk_IsDeviceConnected

Prototype	BTBOOL Btsdk_IsDeviceConnected (BTDEVHDL device_handle,);	
Description	The **Btsdk_IsDeviceConnected** function checks whether there exist connection between local device and the specified remote device.	
Parameters	*device_handle*	[in] Handle to the device to check role.
Return:	If a connection exists, the return value is BTSDK_TRUE. If no connection exists, the return value is BTSDK_FALSE.	

Remarks

Before calling *Btsdk_IsDeviceConnected*, the device database must be initialized by a previous successful call to *Btsdk_Init*.

5.4.4.3.2 Btsdk_GetRemoteDeviceRole

Prototype	BTINT32 Btsdk_GetRemoteDeviceRole (BTDEVHDL device_handle, BTUINT16* prole);	
Description	The **Btsdk_GetRemoteDeviceRole** function gets the current role that the specified device is performing for the ACL link with local device.	
Parameters	*device_handle*	[in] Handle to the device to check role.
	prole	[out] Pointer to a variable to receive the current role. The possible role value can be one of BTSDK_MASTER_ROLE (master role) and BTSDK_SLAVE_ROLE (slave role).
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

Before calling *Btsdk_GetRemoteDeviceRole*, a connection between local device and the specified remote device must be created first.

5.4.4.3.3 Btsdk_GetRemoteLMPInfo

Prototype	BTINT32 Btsdk_GetRemoteLMPInfo (BTDEVHDL device_handle, PBtSdkRemoteLMPInfoStru lmp_info);	
Description	The **Btsdk_GetRemoteLMPInfo** function gets information about the LMP in the specified remote device.	
Parameters	*device_handle*	[in] Handle to the remote device used to specify the connection.
	lmp_info	[out] Pointer to a BtSdkRemoteLMPInfoStru structure that receives the information about the LMP in the specified remote device.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

Before calling *Btsdk_GetRemoteLMPInfo*, a connection between local device and the specified remote device must be created first.

5.4.4.3.4 Btsdk_GetRemoteRSSI

Prototype	BTINT32 Btsdk_GetRemoteRSSI (BTDEVHDL device_handle, BTINT8* prssi);	
Description	The **Btsdk_GetRemoteRSSI** function gets the RSSI value of the specified remote device.	
Parameters	*device_handle*	[in] Handle to the specified remote device.
	prssi	[out] Pointer to a variable to receive the RSSI value. Range: -128 to 127 (dB).
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

Before calling *Btsdk_GetRemoteRSSI*, the specified remote device must be inquired or a connection between local device and the remote device must be created.

5.4.4.3.5 Btsdk_GetRemoteLinkQuality

Prototype	BTINT32 Btsdk_GetRemoteLinkQuality (BTDEVHDL device_handle, BTUINT16* plink_quality);	
Description	The **Btsdk_GetRemoteLinkQuality** function gets the current link quality value of the connection between local device and the specified remote device.	
Parameters	*device_handle*	[in] Handle to the remote device used to specify the connection.
	plink_quality	[out] Pointer to a variable to receive the current link quality value. The higher the value, the better the link quality is. Range: 0 to 0xFF.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

Before calling *Btsdk_GetRemoteLinkQuality*, a connection between local device and the specified remote device must be created first.

5.4.4.3.6 Btsdk_GetSupervisionTimeout

Prototype	BTINT32 Btsdk_GetSupervisionTimeout (BTDEVHDL device_handle, BTUINT16* ptimeout);	
Description	The **Btsdk_GetSupervisionTimeout** function gets the Link Supervision Timeout value for the connection between local device and the specified remote device.	
Parameters	*device_handle*	[in] Handle to the remote device used to specify the connection.
	ptimeout	[out] Pointer to a variable to receive the timeout value. The timeout value is measured in number of Bluetooth Baseband slots (0.625msec).
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

Before calling *Btsdk_GetSupervisionTimeout*, a connection between local device and the specified remote device must be created first.

5.4.4.3.7 Btsdk_SetSupervisionTimeout

Prototype	BTINT32 Btsdk_SetSupervisionTimeout (BTDEVHDL device_handle, BTUINT16 timeout);	
Description	The **Btsdk_SetSupervisionTimeout** function sets the Link Supervision Timeout value for the connection between local device and the specified remote device.	
Parameters	*device_handle*	[in] Handle to the remote device used to specify the connection.
	timeout	[in] Specifies the timeout value to be set. The timeout value is measured in number of Bluetooth Baseband slots (0.625msec).
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

Before calling Btsdk_SetSupervisionTimeout, a connection between local device and the specified Remote device must be created first.

5.4.4.3.8 Btsdk_ChangeConnectionPacketType

Prototype	BTINT32 Btsdk_ChangeConnectionPacketType (BTDEVHDL device_handle, BTUINT16 packet_type);	
Description	The **Btsdk_ChangeConnectionPacketType** function changes the packet types that can be used for the connection that is currently established with the specified remote device.	
Parameters	*device_handle*	[in] Handle to the remote device used to specify the ACL link.
	packet_type	[in] A set of flags which specify the packet types to be used.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

The ***packet_type*** parameter can be one or more of these values.

Value	Description
BTSDK_ACL_PKT_2DH1	2-DH1 is requested. Only supported by V2.0EDR Bluetooth device.
BTSDK_ACL_PKT_3DH1	3-DH1 is requested. Only supported by V2.0EDR Bluetooth device.
BTSDK_ACL_PKT_DM1	DM1 is requested
BTSDK_ACL_PKT_DH1	DH1 is requested.
BTSDK_ACL_PKT_2DH3	2-DH3 is requested. Only supported by V2.0EDR Bluetooth device.
BTSDK_ACL_PKT_3DH3	3-DH3 is requested. Only supported by V2.0EDR Bluetooth device.
BTSDK_ACL_PKT_DM3	DM3 is requested
BTSDK_ACL_PKT_DH3	DH3 is requested.
BTSDK_ACL_PKT_2DH5	2-DH5 is requested. Only supported by V2.0EDR Bluetooth device.
BTSDK_ACL_PKT_3DH5	3-DH5 is requested. Only supported by V2.0EDR Bluetooth device.
BTSDK_ACL_PKT_DM5	DM5 is requested.
BTSDK_ACL_PKT_DH5	DH5 is requested.

Remarks

Before calling *Btsdk_ChangeConnectionPacketType*, a connection between local device and the specified remote device must be created first.

5.4.4.4 Device Database Management

BlueSoleil stores all the remote devices discovered from the first time run in the device database. At run time, each device record in the database is represented by a unique 32bit unsigned integer named as device handle. The handle value can be used in any function that requires a handle to a remote device.

Btsdk_Init initializes the device database and recovers device records from backup file to the device database. *Btsdk_Done* releases the device database finally. A device handle is created automatically for each record added to the database. The device handle is closed when the device record is removed from the database or when *Btsdk_Done* is called.

The information of a device is added to the database automatically when it responds during the inquiry procedure or when it connects to the BlueSoleil local Bluetooth Host Stack. The application can also add a device record to the database by calling function *Btsdk_AddRemoteDevice*.

Currently, there is no limit on the number of device records stored in the device database. The application is responsible for determining which device is to be stored or removed.

5.4.4.4.1 Btsdk_GetRemoteDeviceHandle

Prototype	BTDEVHDL Btsdk_GetRemoteDeviceHandle (BTUINT8* bd_addr,);	
Description	The **Btsdk_GetRemoteDeviceHandle** function gets the handle to the remote device with the specified Bluetooth device address. If no device record matched the device address is found in the database, this function returns BTSDK_INVALID_HANDLE immediately.	
Parameters	*bd_addr*	[in] Pointer to the buffer contains the Bluetooth device address.
Return:	If the function succeeds, the return value is the handle to the specified remote device. If the function fails, the return value is BTSDK_INVALID_HANDLE.	

Remarks

Before calling *Btsdk_GetRemoteDeviceHandle*, the device database must be initialized by a previous successful call to *Btsdk_Init*.

5.4.4.4.2 Btsdk_AddRemoteDevice

Prototype	BTDEVHDL Btsdk_AddRemoteDevice (BTUINT8* bd_addr,);	
Description	The **Btsdk_AddRemoteDevice** function Adds a device record with the specified device address to the database. If a device record matched the device address is found in the database, this function returns the device handle directly.	
Parameters	*bd_addr*	[in] Pointer to the buffer contains the Bluetooth device address.
Return:	If the function succeeds, the return value is the handle to the specified remote device. If the function fails, the return value is BTSDK_INVALID_HANDLE.	

Remarks

Before calling *Btsdk_AddRemoteDevice*, the device database must be initialized by a previous successful call to *Btsdk_Init*.

5.4.4.4.3 Btsdk_DeleteRemoteDeviceByHandle

Prototype	BTINT32 Btsdk_DeleteRemoteDeviceByHandle (BTDEVHDL device_handle,);	
Description	The **Btsdk_DeleteRemoteDeviceByHandle** function removes a specified device record from the database. If a connection between the local device and the specified device exists, BlueSoleil returns the error code BTSDK_ER_ITEM_INUSE and the specified device record isn't removed from the database.	
Parameters	*device_handle*	[in] Device handle specified the device record to be removed from the database.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

Before calling *Btsdk_DeleteRemoteDeviceByHandle*, the device database must be initialized by a previous successful call to *Btsdk_Init*.

5.4.4.4.4 Btsdk_DeleteUnpairedDevicesByClass

Prototype	BTINT32 Btsdk_DeleteUnpairedDevicesByClass (BTUINT32 device_class,);	
Description	The **Btsdk_DeleteUnpairedDevicesByClass** function removes all unpaired devices with the specified Class of Device from the device database. If a connection exists between the local device and one of the devices that match the condition, this device record isn't removed from the database.	
Parameters	*device_class*	[in] Specifies the Class of Device of interest. That is, only unpaired devices with the Class of Device specified by *device_class* parameter will be removed from the database. The application can specify one of the device class identifiers listed in Table 3. If this value is set to 0, BlueSoleil removes all unpaired devices from the database.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

Before calling *Btsdk_DeleteUnpairedDevicesByClass*, the device database must be initialized by a previous successful call to *Btsdk_Init*.

5.4.4.4.5 Btsdk_GetStoredDevicesByClass

Prototype	BTUINT32 Btsdk_GetStoredDevicesByClass (BTUINT32 device_class, BTDEVHDL* pdevice_handles, BTUINT32 max_dev_num);	
Description	The **Btsdk_GetStoredDevicesByClass** function gets a list of handles to the device records with the specified Class of Device from the device database.	
Parameters	*device_class*	[in] Specifies the Class of Device of interest. That is, only devices with the Class of Device specified by *device_class* parameter will be reported to the application. The application can specify one of the device class identifiers listed in Table 3. If this value is set to 0, BlueSoleil reports all devices stored in the database to the application.
	pdevice_handles	[out] Pointer to the buffer to receive the device handles. If this parameter is set to NULL, the total number of available handles is returned.
	max_dev_num	[in] Specifies the maximum number of handles can be copied to the buffer pointed to by the *pdevice_handles* parameter. If *pdevice_handle* is set to NULL, the value of *max_dev_num* parameter is ignored.
Return:	If *pdevice_handle* is not NULL and *max_dev_num* is non-zero, the return value is the number of handles copied to the buffer pointed to by *pdevice_handles*. If pdevice_handle is NULL, the return value is the total number of available handles.	

Remarks

Before calling *Btsdk_GetStoredDevicesByClass*, the device database must be initialized by a previous successful call to *Btsdk_Init*.

5.4.4.4.6 Btsdk_GetInquiredDevices

Prototype	BTUINT32 Btsdk_GetInquiredDevices (BTDEVHDL* pdevice_handles, BTUINT32 max_dev_num);	
Description	The **Btsdk_GetInquiredDevices** function gets a list of handles to the device records that are marked as "Inquired" devices.	
Parameters	*pdevice_handles*	[out] Pointer to the buffer to receive the device handles. If this parameter is set to NULL, the total number of available handles is returned.
	max_dev_num	[in] Specifies the maximum number of handles can be copied to the buffer pointed to by the *pdevice_handles* parameter. If *pdevice_handles* is set to NULL, the value of *max_dev_num* parameter is ignored.
Return:	If *pdevice_handle* is not NULL and *max_dev_num* is nonzero, the return value is the number of handles copied to the buffer pointed to by *pdevice_handles*. If pdevice_handle is NULL, the return value is the total number of available handles.	

Remarks

Before calling *Btsdk_GetInquiredDevices*, the device database must be initialized by a previous successful call to *Btsdk_Init*.

A device discovered during the inquiry procedure is marked as an "Inquired" device. The "Inquired" flag will be kept until the next time *Btsdk_StartDeviceDiscovery* or *Btsdk_Done* is called.

5.4.4.4.7 Btsdk_GetPairedDevices

Prototype	BTUINT32 Btsdk_GetPairedDevices (BTDEVHDL* pdevice_handles, BTUINT32 max_dev_num);	
Description	The **Btsdk_GetPairedDevices** function gets a list of handles to the device records that are marked as "Paired" devices.	
Parameters	*pdevice_handles*	[out] Pointer to the buffer to receive the device handles. If this parameter is set to NULL, the total number of available handles is returned.
	max_dev_num	[in] Specifies the maximum number of handles can be copied to the buffer pointed to by the *pdevice_handles* parameter. If *pdevice_handles* is set to NULL, the value of *max_dev_num* parameter is ignored.
Return:	If *pdevice_handles* is not NULL and *max_dev_num* is nonzero, the return value is the number of handles copied to the buffer pointed to by *pdevice_handles*. If pdevice_handles is NULL, the return value is the total number of available handles.	

Remarks

Before calling *Btsdk_GetPairedDevices*, the device database must be initialized by a previous successful call to *Btsdk_Init*.

Both the local device and the other device may initiate a pairing procedure between them. After the pairing procedure with a remote device finishes successfully, BlueSoleil stores the link key in the device database and marks this remote device as a "Paired" device. The "Paired" flag of a remote device will be kept until *Btsdk_UnPairDevice* is called or an unsuccessful authentication procedure with this remote device occurs.

5.4.4.4.8 Btsdk_StartEnumRemoteDevice

Prototype	BTSDKHANDLE Btsdk_StartEnumRemoteDevice (BTUINT32 flag, BTUINT32 device_class);	
Description	The **Btsdk_StartEnumRemoteDevice** function starts to search the device database for devices that match the specified attributes.	
Parameters	*flag*	[in] Specified the attributes to be used in the search.
	device_class	[in] Specifies the Class of Device of interest. That is, only devices with the Class of Device specified by *device_class* parameter will be reported to the application. The application can specify one of the device class identifiers listed in Table 3. The *device_class* parameter is used only when the BTSDK_ERD_FLAG_DEVCLASS value is set in the *flag* parameter.
Return:	If the function succeeds, the return value is a search handle used in a subsequent call to *Btsdk_EnumRemoteDevice* and *Btsdk_EndEnumRemoteDevice*. If the function fails, the return value is BTSDK_INVALID_HANDLE.	

The ***flag*** parameter can be one or more of these values.

Value	Description
BTSDK_ERD_FLAG_NOLIMIT	Search for all devices stored in the database. This value must be used separately.
BTSDK_ERD_FLAG_PAIRED	Search for devices marked as “Paired” devices.
BTSDK_ERD_FLAG_CONNECTED	Search for devices that are connecting with local device currently.
BTSDK_ERD_FLAG_INQUIRED	Search for devices marked as “Inquired” devices.
BTSDK_ERD_FLAG_TRUSTED	Search for devices marked as “Trusted” devices.
BTSDK_ERD_FLAG_DEVCLASS	Search for devices with the Class of Device specified by the *device_class* parameter.

Remarks

Before calling *Btsdk_StartEnumRemoteDevice*, the device database must be initialized by a previous successful call to *Btsdk_Init*.

The *Btsdk_StartEnumRemoteDevice* function only opens a search handle. After the search handle has been established, use the *Btsdk_EnumRemoteDevice* function to search for device records that match the specified attributes.

5.4.4.4.9 Btsdk_EnumRemoteDevice

Prototype	BTDEVHDL Btsdk_EnumRemoteDevice (BTSDKHANDLE enum_handle, PBtSdkRemoteDevicePropertyStru rmt_dev_prop);	
Description	The **Btsdk_EnumRemoteDevice** function continues to search the device database for a device matches the specified attributes. The attributes are specified by a previous call to the *Btsdk_StartEnumRemoteDevice* function.	
Parameters	*enum_handle*	[in] Search handle returned by a previous call to the *Btsdk_StartEnumRemoteDevice* function.
	rmt_dev_prop	[out] Pointer to the BtSdkRemoteDevicePropertyStru structure that receives information about the found device record.
Return:	If the function succeeds, the return value is the handle specifies the found device. If no matching device can be found, the return value is BTSDK_INVALID_HANDLE.	

Remarks

Before calling *Btsdk_EnumRemoteDevice*, the device database must be initialized by a previous successful call to *Btsdk_Init*.

Example

```
/* This sample demonstrates how to obtain the collection of paired devices. */
void AppGetPairedDevices(void)
{
    BtSdkRemoteDevicePropertyStru DevProp = {0};
    BTSDKHANDLE hEnumDev = BTSDK_INVALID_HANDLE;
    BTDEVHDL hDevFound = BTSDK_INVALID_HANDLE;

    hEnumDev = Btsdk_StartEnumRemoteDevice(BTSDK_ERD_FLAG_PAIRED, 0);
    if (hEnumDev != BTSDK_INVALID_HANDLE)
    {
        while ((hDevFound = Btsdk_EnumRemoteDevice(hEnumDev, &DevProp)) != BTSDK_INVALID_HANDLE)
        {
```

```
                    /*To Do: Add additional processing here. */
                }
                Btsdk_EndEnumRemoteDevice(hEnumDev);
            }
        }
```

5.4.4.4.10 Btsdk_EndEnumRemoteDevice

Prototype	BTINT32 Btsdk_EndEnumRemoteDevice (BTSDKHANDLE enum_handle,);	
Description	The **Btsdk_EndEnumRemoteDevice** function closes the specified search handle.	
Parameters	*enum_handle*	[in] Search handle returned by a previous call to the *Btsdk_StartEnumRemoteDevice* function.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

Before calling *Btsdk_EndEnumRemoteDevice*, the device database must be initialized by a previous successful call to *Btsdk_Init*.

When *Btsdk_EnumRemoteDevice* returns BTSDK_INVALID_HANDLE, the application must close the search handle by calling the function *Btsdk_EndEnumRemoteDevice*.

5.4.4.4.11 Btsdk_GetRemoteDeviceAddress

Prototype	BTINT32 Btsdk_GetRemoteDeviceAddress (BTDEVHDL device_handle, BTUINT8* bd_addr,);	
Description	The **Btsdk_GetRemoteDeviceAddress** function gets the Bluetooth device address of the specified remote device.	
Parameters	*device_handle*	[in] Handle to the remote device object.
	bd_addr	[out] Pointer to the buffer to receive the Bluetooth device address. The buffer must be large enough to receive 6 bytes device address.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

Before calling *Btsdk_GetRemoteDeviceAddress*, the device database must be initialized by a previous successful call to *Btsdk_Init*.

5.4.4.4.12 Btsdk_GetRemoteDeviceName

Prototype	BTINT32 Btsdk_GetRemoteDeviceName (BTDEVHDL device_handle, BTUINT8* name, BTUINT16* plen);	
Description	The **Btsdk_GetRemoteDeviceName** function gets the user-friendly name of the specified remote device from the device database.	
Parameters	*device_handle*	[in] Handle to the remote device object.
	name	[out] Pointer to the buffer that receives the device name. This parameter can be NULL.
	plen	[in/out] Pointer to a variable that, on input, specifies the size, in bytes, of the buffer pointed to by the *name* parameter, or it can be NULL if the buffer size is larger than BTSDK_DEVNAME_LEN. On output, This variable receives the number of bytes copied to the buffer pointed to by the *name* parameter. To determine the required buffer size, call this function with *name* set to NULL. This function returns the required buffer size in **plen*.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

Before calling *Btsdk_GetRemoteDeviceName*, the device database must be initialized by a previous successful call to *Btsdk_Init*.

The user-friendly device name is a UTF-8 character string. The *Btsdk_GetRemoteDeviceName* function returns BTSDK_OPERATION_FAILURE immediately if the device name doesn't exist in the database. In this case, the application shall call *Btsdk_UpdateRemoteDeviceName* to acquire the name information directly from the remote device.
BlueSoleil will automatically update the device name when the local device connects to the specified remote device.

5.4.4.4.13 Btsdk_GetRemoteDeviceClass

Prototype	BTINT32 Btsdk_GetRemoteDeviceClass (BTDEVHDL device_handle, BTUINT32* pdevice_class,);	
Description	The **Btsdk_GetRemoteDeviceClass** function gets the Class of Device/Service field value of the specified remote device from the device database.	
Parameters	*device_handle*	[in] Handle to the remote device object.
	pdevice_class	[out] Pointer to a variable that receives the Class of Device/Service value of the local device. The return value can be one of the device class identifiers listed in Table 3 combined with multiple major service class identifiers listed in Table 4.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

Before calling *Btsdk_GetRemoteDeviceClass*, the device database must be initialized by a previous successful call to *Btsdk_Init*.

5.4.4.4.14 Btsdk_GetRemoteDeviceProperty

Prototype	BTINT32 Btsdk_GetRemoteDeviceProperty (BTDEVHDL device_handle, PBtSdkRemoteDevicePropertyStru rmt_dev_prop);	
Description	The **Btsdk_GetRemoteDeviceProperty** function gets the information about the specified remote device.	
Parameters	*device_handle*	[in] Handle to the remote device object.
	rmt_dev_prop	[out] Pointer to the BtSdkRemoteDevicePropertyStru structure that receives information about the specified device.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

Before calling *Btsdk_GetRemoteDeviceProperty*, the device database must be initialized by a previous successful call to *Btsdk_Init*.

The *rmt_dev_prop->bd_addr*, *rmt_dev_prop->dev_class* and *rmt_dev_prop->link_key* values are read from the device database directly.

If the local device doesn't connect to the remote device, the *rmt_dev_prop->name* value is read from the device database. Otherwise, the *rmt_dev_prop->name* value is read from the remote device.

The value of rmt_dev_prop->lmp_info is available only when the local device connects to the specified remote device.

	device_handle	[in] Handle to the remote device to set the trust relation.
	bIsTrusted	[in] BTSDK_TRUE if the specified remote device is trusted to the specified local service record or BTSDK_FALSE otherwise.

Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.

5.4.4.4.15 Btsdk_RemoteDeviceFlowStatistic

Prototype	BTINT32 Btsdk_RemoteDeviceFlowStatistic (BTDEVHDL dev_hdl, BTUINT32* rx_bytes, BTUINT32* tx_bytes);	
Description	The **Btsdk_RemoteDeviceFlowStatistic** function gets the statistic of data sent to and received from the remote device.	
Parameters	*dev_hdl*	[in] Handle of the remote device. If dev_hdl is set to BTSDK_INVALID_HANDLE, the statistic of data sent and received by the local device is returned.
	rx_bytes	[in] Pointer to the 32bit integer to store how many bytes received.
	tx_bytes	[in] Pointer to the 32bit integer to store how many bytes sent.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code.	

Remarks

5.4.4.4.16 Btsdk_GetRemoteDeviceType

Prototype	BTINT32 Btsdk_GetRemoteDeviceType(BTDEVHDL dev_hdl);	
Description	The **Btsdk_GetRemoteDeviceType** function gets the type of the remote device.	
Parameters	*dev_hdl*	[in] Handle of the remote device.
Return:	If the function succeeds, the return value is device type.Refer to Table Device Type. If the function fails, the return value is an error code.	

Remarks

5.4.5 Connection Management

When “connection” is said in this section, it means a synchronized high-level protocol connection defined in the related profile specification.

5.4.5.1 Service Discovery

At run time, each remote service record in the device database is represented by a unique 32bit unsigned integer named as remote service handle. The handle value can be used in any function that requires a handle to a remote service record.

The **service handle** specified here has nothing to do with the service record handle defined in the SDP specification. To differentiate these two concepts, we use **SDP record handle** in this document to represent the service record handle defined in the SDP specification.

5.4.5.1.1 Btsdk_BrowseRemoteServicesEx

Prototype	BTINT32 Btsdk_BrowseRemoteServicesEx (BTDEVHDL device_handle, PBtSdkSDPSearchPatternStru psch_ptn, BTUINT32 ptn_number, BTSVCHDL* pservice_handles, BTUINT32* phandle_number);	
Description	The **Btsdk_BrowseRemoteServicesEx** function discovers the available service records, which matches the specified search patterns, on the remote device and queries each service record for its attributes.	
Parameters	*device_handle*	[in] Handle to the remote device to browse service.
	psch_ptn	[in] Pointer to an array of *BtSdkSDPSearchPatternStru* structures that contains *ptn_number* elements. If the *psch_ptn* is a NULL pointer, BTSDK uses the 16bit UUID value 0x0100 as the default search pattern.
	ptn_number	[in] Specifies the number of elements present in the array *psch_ptn*. This value must be less than BTSDK_MAX_SEARCH_PATTERNS, or the exceeding elements are ignored. If the *ptn_number* value is 0, BlueSoleil uses the 16bit UUID value 0x0100 as the default search pattern.
	pservice_handles	[out] Pointer to the buffer to receive the remote service handles. This parameter can be NULL.

	phandle_number	[in/out] Pointer to a variable that, on input, specifies the number of handles can be copied to the *pservice_handles* buffer. On output, This variable receives the number of handles copied to the *pservice_handles* buffer. To determine the required buffer size, call this function with *pservice_handles* set to NULL. This function returns the total number of available handles in **phandle_number*.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

Before calling *Btsdk_BrowseRemoteServicesEx*, the local device must be enabled by a previous successful call to *Btsdk_StartBluetooth*.

All the service records discovered are stored in local SDK device database until *Btsdk_Done* is called. You can access them later by calling *Btsdk_GetRemoteServicesEx* or *Btsdk_GetRemoteServices*.

5.4.5.1.2 Btsdk_BrowseRemoteServices

Prototype	BTINT32 Btsdk_BrowseRemoteServices (BTDEVHDL device_handle, BTSVCHDL* pservice_handles, BTUINT32* phandle_number);	
Description	The **Btsdk_BrowseRemoteServices** function discovers all the service records available on the remote device and queries each service record for its attributes.	
Parameters	*device_handle*	[in] Handle to the remote device to browse service.
	pservice_handles	[out] Pointer to the buffer to receive the remote service handles. This parameter can be NULL.
	phandle_number	[in/out] Pointer to a variable that, on input, specifies the number of handles can be copied to the *pservice_handles* buffer. On output, This variable receives the number of handles copied to the *pservice_handles* buffer. To determine the required buffer size, call this function with *pservice_handles* set to NULL. This function returns the total number of available handles in **phandle_number*.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

Before calling *Btsdk_BrowseRemoteServices*, the local device must be enabled by a previous successful call to *Btsdk_StartBluetooth*.

This function uses the 16bit UUID value 0x0100 as the search pattern.

All the service records discovered are stored in local SDK device database until *Btsdk_Done* is called. You can access them later by calling *Btsdk_GetRemoteServicesEx* or *Btsdk_GetRemoteServices*.

5.4.5.1.3 Btsdk_RefreshRemoteServiceAttributes

Prototype	BTINT32 Btsdk_RefreshRemoteServiceAttributes (BTSVCHDL service_handle, PBtSdkRemoteServiceAttrStru pservice_attributes);	
Description	The **Btsdk_RefreshRemoteServiceAttributes** function retrieves all the attribute values of a specified remote service record and returns the most useful attribute values to the application.	
Parameters	*service_handle*	[in] Handle to the remote service record.
	pservice_attributes	[out] Pointer to a BtSdkRemoteServiceAttrStru structure to receive the attribute values about the specified service record. This parameter can be NULL.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

Before calling *Btsdk_RefreshRemoteServiceAttributes*, the local device must be enabled by a previous successful call to *Btsdk_StartBluetooth*.

Use the *mask* member of the *pservice_attributes* parameter to specify the attributes to be retrieved. If *pservice_attributes->mask* includes BTSDK_RSAM_EXTATTRIBUTES, the function allocates a buffer using the *Btsdk_MallocMemory* function, and returns the pointer to the buffer through *pservice_attributes->ext_attributes*. The application should use the *Btsdk_FreeMemory* function to free the buffer when it is no longer needed.

All the attribute values retrieved are stored in local SDK device database. You can access them later by calling *Btsdk_GetRemoteServiceAttributes*.

5.4.5.1.4 Btsdk_GetRemoteServicesEx

Prototype	BTINT32 Btsdk_GetRemoteServicesEx (BTDEVHDL device_handle, PBtSdkSDPSearchPatternStru psch_ptn, BTUINT32 ptn_number, BTSVCHDL* pservice_handles, BTUINT32* phandle_number);	
Description	The **Btsdk_GetRemoteServicesEx** function gets the available service records, which matches the specified search patterns, from the device database.	
Parameters	*device_handle*	[in] Handle to the remote device to browse service.
	psch_ptn	[in] Pointer to an array of *BtSdkSDPSearchPatternStru* structures that contains *ptn_number* elements. If the *psch_ptn* is a NULL pointer, BlueSoleil uses the 16bit UUID value 0x0100 as the default search pattern.
	ptn_number	[in] Specifies the number of elements present in the array *psch_ptn*. This value must be less than BTSDK_MAX_SEARCH_PATTERNS, or the exceeding elements are ignored. If the *ptn_number* value is 0, BlueSoleil uses the 16bit UUID value 0x0100 as the default search pattern.
	pservice_handles	[out] Pointer to the buffer to receive the remote service handles. This parameter can be NULL.

	phandle_number	[in/out] Pointer to a variable that, on input, specifies the number of handles can be copied to the *pservice_handles* buffer. On output, This variable receives the number of handles copied to the *pservice_handles* buffer. To determine the required buffer size, call this function with *pservice_handles* set to NULL. This function returns the total number of available handles in **phandle_number*.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

Before calling *Btsdk_GetRemoteServicesEx*, the device database must be initialized by a previous successful call to *Btsdk_Init*.

The *Btsdk_GetRemoteServicesEx* function won't initiate any SDP transactions. The application shall call *Btsdk_BrowseRemoteServicesEx* first to find out how many service records are available on the remote device and create a service list in local device database. Then call this function to get the list.

5.4.5.1.5 Btsdk_GetRemoteServices

Prototype	BTINT32 Btsdk_BrowseRemoteServices (BTDEVHDL device_handle, BTSVCHDL* pservice_handles, BTUINT32* phandle_number);	
Description	The **Btsdk_GetRemoteServices** function gets all the service records available on the remote device from the device database.	
Parameters	*device_handle*	[in] Handle to the remote device to browse service.
	pservice_handles	[out] Pointer to the buffer to receive the remote service handles. This parameter can be NULL.
	phandle_number	[in/out] Pointer to a variable that, on input, specifies the number of handles can be copied to the *pservice_handles* buffer. On output, This variable receives the number of handles copied to the *pservice_handles* buffer. To determine the required buffer size, call this function with *pservice_handles* set to NULL. This function returns the total number of available handles in **phandle_number*.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

Before calling *Btsdk_GetRemoteServices*, the device database must be initialized by a previous successful call to *Btsdk_Init*.

The *Btsdk_GetRemoteServices* function won't initiate any SDP transactions. The application shall call *Btsdk_BrowseRemoteServicesEx* or *Btsdk_BrowseRemoteServices* first to find out how many service records are available on the remote device and create a service list in local device database. Then call this function to get the list.

5.4.5.1.6 Btsdk_GetRemoteServiceAttributes

Prototype	BTINT32 Btsdk_GetRemoteServiceAttributes (BTSVCHDL service_handle, PBtSdkRemoteServiceAttrStru pattributes);	
Description	The **Btsdk_GetRemoteServiceAttributes** function reads attribute values of a specified remote service record from BlueSoleil local SDK device database.	
Parameters	*service_handle*	[in] Handle to the remote service record.
	pattributes	[out] Pointer to a BtSdkRemoteServiceAttrStru structure to receive the attribute values about the specified service record. This parameter can't be NULL.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

Before calling *Btsdk_GetRemoteServiceAttributes*, the device database must be initialized by a previous successful call to *Btsdk_Init*.

Use the *mask* member of the *pservice_attributes* parameter to specify the attributes to be retrieved. If *pservice_attributes->mask* includes BTSDK_RSAM_EXTATTRIBUTES, the function allocates a buffer using the *Btsdk_MallocMemory* function, and returns the pointer to the buffer through *pservice_attributes->ext_attributes*. The application should use the *Btsdk_FreeMemory* function to free the buffer when it is no longer needed.

The *Btsdk_GetRemoteServiceAttributes* function won't initiate any SDP transactions. The application shall call *Btsdk_RefreshRemoteServiceAttributes* first to retrieve attribute values from the remote device and stored the values in local device database. Then call this function to read the values.

5.4.5.1.7 Btsdk_StartEnumRemoteService

Prototype	BTSDKHANDLE Btsdk_StartEnumRemoteService (void);	
Description	The **Btsdk_StartEnumRemoteService** function starts to search the device database for all service records available on the specified remote device.	
Parameters		
Return:	If the function succeeds, the return value is a search handle used in a subsequent call to *Btsdk_EnumRemoteService* and *Btsdk_EndEnumRemoteService*. If the function fails, the return value is BTSDK_INVALID_HANDLE.	

Remarks

Before calling *Btsdk_StartEnumRemoteService*, the device database must be initialized by a previous successful call to *Btsdk_Init*.

The *Btsdk_StartEnumRemoteService* function won't initiate any SDP transactions. The application shall call *Btsdk_BrowseRemoteServicesEx* first to find out how many service records are available on the remote device and create a service list in local device database. Then call this function to enumerate the list.

The *Btsdk_StartEnumRemoteService* function only opens a search handle. After the search handle has been established, use the *Btsdk_EnumRemoteService* function to search for available service records.

5.4.5.1.8 Btsdk_EnumRemoteService

Prototype	BTSVCHDL Btsdk_EnumRemoteService (BTSDKHANDLE enum_handle, PBtSdkRemoteServiceAttrStru pservice_attributes);	
Description	The **Btsdk_EnumRemoteService** function continues to search the device database for an available service record of a previous specified remote device.	
Parameters	*enum_handle*	[in] Search handle returned by a previous call to the *Btsdk_StartEnumRemoteService* function.
	pservice_attributes	[out] Pointer to the BtSdkRemoteServiceAttrStru structure that receives information about the found service record.
Return:	If the function succeeds, the return value is the handle specifies the found service record. If no more service can be found, the return value is BTSDK_INVALID_HANDLE.	

Remarks

Before calling *Btsdk_EnumRemoteService*, the device database must be initialized by a previous successful call to *Btsdk_Init*.

Use the *mask* member of the *pservice_attributes* parameter to specify the attributes to be retrieved. If *pservice_attributes->mask* includes BTSDK_RSAM_EXTATTRIBUTES, the function allocates a buffer using the *Btsdk_MallocMemory* function, and returns the pointer to the buffer through *pservice_attributes->ext_attributes*. The application should use the *Btsdk_FreeMemory* function to free the buffer when it is no longer needed.

Example

```
/* This sample demonstrates how to obtain the collection of service records. */
void AppGetRemoteServices(void)
{
    BtSdkRemoteServerAttrStru SvcAttr = {0};
    BTSDKHANDLE hEnumSvc = BTSDK_INVALID_HANDLE;
    BTSVCHDL hSvcFound = BTSDK_INVALID_HANDLE;
```

```
    hEnumSvc = Btsdk_StartEnumRemoteService();
    if (hEnumSvc != BTSDK_INVALID_HANDLE)
    {
        SvcAttr.mask = BTSDK_RSAM_SERVICENAME | BTSDK_RSAM_EXTATTRIBUTES;
        while ((hSvcFound = Btsdk_EnumRemoteService(hEnumSvc, &SvcAttr)) != BTSDK_INVALID_HANDLE)
        {
            // To Do: Process the service attribute values:
            // …
            // Free the buffer
            Btsdk_FreeMemory(SvcAttr.ext_attributes);
        }
        Btsdk_EndEnumRemoteService(hEnumSvc);
    }
}
```

5.4.5.1.9 Btsdk_EndEnumRemoteService

Prototype	BTINT32 Btsdk_EndEnumRemoteService (BTSDKHANDLE enum_handle,);	
Description	The **Btsdk_EndEnumRemoteService** function closes the specified search handle.	
Parameters	*enum_handle*	[in] Search handle returned by a previous call to the *Btsdk_StartEnumRemoteService* function.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

Before calling *Btsdk_EndEnumRemoteService*, the service database must be initialized by a previous successful call to *Btsdk_Init*.

When *Btsdk_EnumRemoteService* returns BTSDK_INVALID_HANDLE, the application must close the search handle by calling the function *Btsdk_EndEnumLocalServer*.

5.4.5.2 Application Extension

5.4.5.2.1 Btsdk_SetRemoteServiceParam

Prototype	BTINT32 Btsdk_SetRemoteServiceParam (BTSVCHDL service_handle, BTUINT32 app_param);	
Description	The **Btsdk_SetRemoteServiceParam** function attaches an application specific value to a remote service record.	
Parameters	*service_handle*	[in] Handle to the service that the value is attached to.
	app_param	[in] Parameter value to be attached to the remote device record.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

Before calling *Btsdk_SetRemoteServiceParam*, the device database must be initialized by a previous successful call to *Btsdk_Init*.

In current version, SDK stores this application specific value until *Btsdk_Done* is called. The application shall recover this value itself next time after it calls *Btsdk_Init*.

5.4.5.2.2 Btsdk_GetRemoteServiceParam

Prototype	BTINT32 Btsdk_GetRemoteServiceParam (BTDEVHDL service_handle, BTUINT32* papp_param);	
Description	The **Btsdk_GetRemoteServiceParam** function gets the application specific value attached to a remote device record.	
Parameters	*service_handle*	[in] Handle to the service that the value is attached to.
	papp_param	[out] Pointer to a variable to receive the application specific value attached to the remote service record.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

Before calling *Btsdk_GetRemoteServiceParam*, the device database must be initialized by a previous successful call to *Btsdk_Init*.

5.4.5.3 Connection Establishment

At run time, each connection in the connection database is represented by a unique 32bit unsigned integer named as connection handle. The handle value can be used in any function that requires a handle to an existing connection.

5.4.5.3.1 Btsdk_Connect

Prototype	BTINT32 Btsdk_Connect (BTSVCHDL service_handle, BTUINT32 lParam, BTCONNHDL* pconnection_handle,);	
Description	The **Btsdk_Connect** function establishes a connection to the specified remote service record.	
Parameters	*service_handle*	[in] Handle to the remote service record to connect.
	lParam	[in] Profile specific parameter. If "Mandatory" is not specified in this document, it can be set to 0.
	pconnection_handle	[out] Pointer to a buffer to receive the handle specified the new connection.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

Before calling *Btsdk_Connect*, the local device must be enabled by a previous successful call to *Btsdk_StartBluetooth*.

The ***lParam*** member can be a pointer to one of these structures.

Type of remote service	Type of ***lParam***	Mandatory
BTSDK_CLS_SERIAL_PORT	PBtSdkSPPConnParamStru	No
BTSDK_CLS_DIALUP_NET	PBtSdkDUNConnParamStru.	No
BTSDK_CLS_FAX	PBtSdkFAXConnParamStru	No

Detail of these structures is specified in separate profile API documents.

The *lParam* member is ignored and shall be set to 0 for profiles not listed in the upper table.

5.4.5.3.2 Btsdk_ConnectEx

Prototype	BTINT32 Btsdk_ConnectEx (BTDEVHDL device_handle, BTUINT16 service_class, BTUINT32 lParam, BTCONNHDL* pconnection_handle,);	
Description	The **Btsdk_ConnectEx** function establishes a connection to a service record of the specified type on the specified remote device.	
Parameters	*device_handle*	[in] Handle to the remote device to connect.
	service_class	[in] Type of the service record to connect. It can be one of the values listed in the Table 2.
	lParam	[in] Profile specific parameter. If "Mandatory" is not specified in this document, it can be set to 0.
	pconnection_handle	[out] Pointer to a buffer to receive the handle specified the new connection.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

The ***lParam*** member can be a pointer to one of these structures.

Value of ***service_class***	Type of ***lParam***	Mandatory
BTSDK_CLS_SERIAL_PORT	PBtSdkSPPConnParamStru	No
BTSDK_CLS_DIALUP_NET	PBtSdkDUNConnParamStru.	No
BTSDK_CLS_FAX	PBtSdkFAXConnParamStru	No

Detail of these structures is specified in separate profile API documents.

The *lParam* member is ignored and shall be set to 0 for profiles not listed in the upper table.

Remarks

Before calling *Btsdk_ConnectEx*, the local device must be enabled by a previous successful call to *Btsdk_StartBluetooth*.

If multiple service records of the specified type exist on the remote device, BlueSoleil SDK will automatically select the first accessible record to connect.

5.4.5.3.3 Btsdk_Connection_Event_Ind_Func

Prototype	typedef void (Btsdk_Connection_Event_Ind_Func) (BTCONNHDL connection_handle, BTUINT16 event, BTUINT8* arg);	
Description	The **Btsdk_Connection_Event_Ind_Func** function prototype is the prototype of application defined callback function used to process BTSDK_CONNECTION_EVENT_IND message.	
Parameters	*connection_handle*	[in] Handle to the new connection created or to the connection lost.
	event	[in] Specifies the event type. See following table.
	arg	[in] Event specific parameter. If not specified additionally, it is a pointer to the BtSdkConnectionPropertyStru structure contains the details about the connection.
Return:		

The ***event*** member can be one or more of these values.

Value	Description
BTSDK_APP_EV_CONN_IND	A remote device connects to a local service record.
BTSDK_APP_EV_DISC_IND	The remote device disconnects the connection, or the connection is lost due to radio communication problems, e.g. the remote device is out of communication range.
BTSDK_APP_EV_CONN_CFM	A local device connects to a remote service record.
BTSDK_APP_EV_DISC_CFM	The local device disconnects the connection from remote service.

Remarks

This callback function is called when a service level connection is created or lost.

> DO NOT call inside this callback function any functions, e.g. function that waits for a semaphore or requires the user interference, which may block internal thread of BlueSoleil. DO NOT call inside this callback function any BlueSoleil functions that require communicating with a remote device either, e.g. *Btsdk_Connect* and so on.

5.4.5.4 Connection Database Management

5.4.5.4.1 Btsdk_GetConnectionProperty

Prototype	BTINT32 Btsdk_GetConnectionProperty (BTCONNHDL connection_handle, PBtSdkConnectionPropertyStru pproperty,);	
Description	The **Btsdk_GetConnectionProperty** function gets information about the specified connection.	
Parameters	*connection_handle*	[in] Handle to the connection to be queried.
	pproperty	[out] Pointer to the BtSdkConnectionPropertyStru structure that receives information about the specified connection.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

Before calling *Btsdk_GetConnectionProperty*, the local device must be enabled by a previous successful call to *Btsdk_StartBluetooth*.

5.4.5.4.2 Btsdk_StartEnumConnection

Prototype	BTSDKHANDLE Btsdk_StartEnumConnection (void);	
Description	The **Btsdk_StartEnumConnection** function starts to search the connection database for all connections available.	
Parameters		
Return:	If the function succeeds, the return value is a search handle used in a subsequent call to *Btsdk_EnumConnection* and *Btsdk_EndEnumConnection*. If the function fails, the return value is BTSDK_INVALID_HANDLE.	

Remarks

Before calling *Btsdk_StartEnumConnection*, the local device must be enabled by a previous successful call to *Btsdk_StartBluetooth*.

The *Btsdk_StartEnumConnection* function only opens a search handle. After the search handle has been established, use the *Btsdk_EnumConnection* function to search for available connections.

5.4.5.4.3 Btsdk_EnumConnection

Prototype	BTCONNHDL Btsdk_EnumConnection (BTSDKHANDLE enum_handle, PBtSdkConnectionPropertyStru pproperty);	
Description	The **Btsdk_EnumConnection** function continues to search the connection database for an available connection.	
Parameters	*enum_handle*	[in] Search handle returned by a previous call to the *Btsdk_StartEnumConnection* function.
	pproperty	[out] Pointer to the BtSdkConnectionPropertyStru structure that receives information about the found connection.
Return:	If the function succeeds, the return value is the handle specifies the found connection. If no more service can be found, the return value is BTSDK_INVALID_HANDLE.	

Remarks

Before calling *Btsdk_EnumConnection*, the local device must be enabled by a previous successful call to *Btsdk_StartBluetooth*.

Example

```
/* This sample demonstrates how to obtain the collection of connections. */
void AppGetConnections(void)
{
    BtSdkConnectionPropertyStru prop = {0};
    BTSDKHANDLE hEnumConn = BTSDK_INVALID_HANDLE;
    BTCONNHDL hConn = BTSDK_INVALID_HANDLE;

    hEnumConn = Btsdk_StartEnumConnection();
    if (hEnumConn != BTSDK_INVALID_HANDLE)
    {
        while ((hConn = Btsdk_EnumConn(hEnumConn, &prop)) != BTSDK_INVALID_HANDLE)
        {
            // To Do: Process the connection property:
            // …
```

```
            }
            Btsdk_EndEnumConnection(hEnumConn);
        }
}
```

5.4.5.4.4 Btsdk_EndEnumConnection

Prototype	BTINT32 Btsdk_EndEnumConnection (BTSDKHANDLE enum_handle,);	
Description	The **Btsdk_EndEnumConnection** function closes the specified search handle.	
Parameters	*enum_handle*	[in] Search handle returned by a previous call to the *Btsdk_StartEnumConnection* function.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

Before calling *Btsdk_EndEnumConnection*, the local device must be enabled by a previous successful call to *Btsdk_StartBluetooth*.

When *Btsdk_EnumConnection* returns BTSDK_INVALID_HANDLE, the application must close the search handle by calling the function *Btsdk_EndEnumConnection*.

5.4.5.5 Connection Release

5.4.5.5.1 Btsdk_Disconnect

Prototype	BTUINT32 Btsdk_Disconnect (BTCONNHDL connection_handle);	
Description	The **Btsdk_GetAllIncomingConnections** function disconnects a connection.	
Parameters	*connection_handle*	[in] Handle to the connection to disconnect.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

Before calling *Btsdk_Disconnect*, the local device must be enabled by a previous successful call to *Btsdk_StartBluetooth*.

5.4.6 BlueSoleil Extend APIs

5.4.6.1 Btsdk_VDIInstallDev

Prototype	BTUINT32 Btsdk_VDIInstallDev(BTINT8 *HardwareID, BTINT8 *COMName);	
Description	The **Btsdk_VDIInstallDev** function is used to install a device specified by HardwareID.	
Parameters	*HardwareID*	[in] hardware ID could be
	COMName	[in/out] [in]: name of COM Port to install. [out]: name of COM Port actually installed.
Return:	BTSDK_OK for success other for error code	

Hardware ID can be one of the following values:

HARDWAREID_MDMDUN	Argument for installation of DUN modem.
HARDWAREID_MDMFAX	Argument for installation of FAX modem.

Remarks

5.4.6.2 Btsdk_VDIDelModem

Prototype	BTUINT32 Btsdk_VDIDelModem(BTINT8 *COMName);	
Description	The **Btsdk_VDIDelModem** function deletes a modem which has been installed on the COM port specified by COMName.	
Parameters	*COMName*	[in] name of COM Port.
Return:	BTSDK_OK for success other for error code	

Remarks

5.4.6.3 Btsdk_GetActivationInformation

Prototype	BTUINT32 Btsdk_GetActivationInformation(BTINT8* SerialNumber, BTINT8* ActivateInformation, BTUINT32 ActiveInformationLen);	
Description	The **Btsdk_GetActivationInformation** function allows users to acquire the URL of activate information.	
Parameters	*SerialNumber*	[in] Pointer to the buffer contains the Serial Number for activation of BlueSoleil.
	ActivateInformation	[out] Pointer to the buffer contains URL for Serial Number.
	ActiveInformationLen	[in] Specifies the length, in bytes, of the URL information. The length should not be less than 500 bytes.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code.	

Remarks

This function is used for offline activation when BlueSoleil 6.x is installed to a platform without accessing network and cannot be automatically activated.

5.4.6.4 Btsdk_EnterUnlockCode

Prototype	BTUINT32 Btsdk_EnterUnlockCode (BTINT8* UnlockCode);	
Description	The **Btsdk_EnterUnlockCode** function allows users to activate BlueSoleil 6.x without network service on local device. Users may get the activate information (unlock code) through another PC with network service, using the URL get from Btsdk_GetActivationInformation function. Store the unlock code in memory pointed by the *UnlockCode* parameter on local device. Then call this **Btsdk_EnterUnlockCode** function to activate BlueSoleil 6.x.	
Parameters	*UnlockCode*	[in] Pointer to the buffer contains the Serial Number.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code.	

Remarks

6. Profile Specific API Reference

6.1 Constant Reference

6.1.1 Error Codes

The following table provides a list of profile specific error codes. They are returned by many BlueSoleil functions when they fail.

Name	Value	Description
BTSDK_ER_CTP_GW_EXIST	0X0500	CTP gateway instance exists already. Current version SDK only supports one CTP gateway at a time.
BTSDK_ER_CTP_GW_NONEXIST	0X0501	There is no CTP gateway instance.
BTSDK_ER_USER_HANGUP	0X0502	The call is hung up by the user.
BTSDK_ER_REMOTE_HANGUP	0X0503	The call is hung up by the remote part.
BTSDK_ER_CONTINUE	0X0690	OBEX response code “Continue (0x90)” is received.
BTSDK_ER_SUCCESS	0X06A0	OBEX response code “OK, Success (0xA0)” is received.
BTSDK_ER_CREATED	0X06A1	OBEX response code “Created (0xA1)” is received.
BTSDK_ER_ACCEPTED	0X06A2	OBEX response code “Accepted (0XA2” is received
BTSDK_ER_NON_AUTH_INFO	0X06A3	OBEX response code “Non-Authoritative Information (0XA3)” is received.
BTSDK_ER_NO_CONTENT	0X06A4	OBEX response code “No Content (0xA4)” is received.
BTSDK_ER_RESET_CONTENT	0X06A5	OBEX response code “Reset Content (0XA5)” is received.
BTSDK_ER_PARTIAL_CONTENT	0X06A6	OBEX response code “Partial Content (0XA6)” is received.
BTSDK_ER_MULT_CHOICES	0X06B0	OBEX response code “Multiple Choices (0XB0)” is received.
BTSDK_ER_MOVE_PERM	0X06B1	OBEX response code “Moved Permanently (0XB1)” is received.

BTSDK_ER_MOVE_TEMP	0X06B2	OBEX response code “Moved Temporarily” is received.
BTSDK_ER_SEE_OTHER	0X06B3	OBEX response code “See Other (0XB3)” is received.
BTSDK_ER_NOT_MODIFIED	0X06B4	OBEX response code “Not Modified (0XB4)” is received.
BTSDK_ER_USE_PROXY	0X06B5	OBEX response code “Use Proxy” is received.
BTSDK_ER_BAD_REQUEST	0X06C0	OBEX response code “Bad Request – server couldn’t understand request (0XC0)” is received.
BTSDK_ER_UNAUTHORIZED	0X06C1	OBEX response code “Unauthorized (0XC1)” is received.
BTSDK_ER_PAY_REQ	0X06C2	OBEX response code “Payment Required (0XC2)” is received.
BTSDK_ER_FORBIDDEN	0X06C3	OBEX response code “Forbidden – operation is understood but refused (0XC3)” is received.
BTSDK_ER_NOTFOUND	0X06C4	OBEX response code “Not Found (0XC4)” is received.
BTSDK_ER_METHOD_NOT_ALLOWED	0X06C5	OBEX response code “Method not allowed (0XC5)” is received.
BTSDK_ER_NOT_ACCEPTABLE	0X06C6	OBEX response code “Not Acceptable (0XC6)” is received.
BTSDK_ER_PROXY_AUTH_REQ	0X06C7	OBEX response code “Proxy Authentication required” is received.
BTSDK_ER_REQUEST_TIMEOUT	0X06C8	OBEX response code “Request Timeout (0xC8)” is received.
BTSDK_ER_CONFLICT	0X06C9	OBEX response code “Conflict (0XC7)” is received.
BTSDK_ER_GONE	0X06CA	OBEX response code “Gone (0xCA)” is received.
BTSDK_ER_LEN_REQ	0X06CB	OBEX response code “Length Required (0XCB)” is received.
BTSDK_ER_PREC_FAIL	0X06CC	OBEX response code “Precondition failed (0XCC)” is received.
BTSDK_ER_REQ_ENTITY_TOO_LARGE	0X06CD	OBEX response code “Requested entity too large (0XCD)” is received.
BTSDK_ER_URL_TOO_LARGE	0X06CE	OBEX response code “Request URL too large (0XCE)” is received.
BTSDK_ER_UNSUPPORTED_MEDIA_TYPE	0X06CF	OBEX response code “Unsupported media type (0XCF)” is received.
BTSDK_ER_SVR_ERR	0X06D0	OBEX response code “Internal server error (0XD0)” is received.

BTSDK_ER_NOTIMPLEMENTED	0X06D1	OBEX response code “Not Implemented (0XD1)” is received.
BTSDK_ER_BAD_GATEWAY	0X06D2	OBEX response code “Bad Gateway (0XD2)” is received.
BTSDK_ER_SERVICE_UNAVAILABLE	0X06D3	OBEX response code “Service Unavailable (0XD3)” is received.
BTSDK_ER_GATEWAY_TIMEOUT	0X06D4	OBEX response code “Gateway timeout (0XD4)” is received.
BTSDK_ER_HTTP_NOTSUPPORT	0X06D5	OBEX response code “HTTP version not supported (0XD5)” is received.
BTSDK_ER_DATABASE_FULL	0X06E0	OBEX response code “Database Full (0XE0)” is received.
BTSDK_ER_DATABASE_LOCK	0X06E1	OBEX response code “Database Locked (0XE1)” is received.

Table 10: Profile Specific Error Codes.

6.1.2 AVRCP Error Codes

id	Description	Valid for Commands
BTSDK_AVRCP_ERROR_SUCCESSFUL	Operation completed without error. This is the status that should be returned if the operation was successful.	All except where the response CType is AV/C REJECTED
BTSDK_AVRCP_ERROR_INVALID_COMMAND	Invalid command, sent if TG received a PDU that it did not understand.	All
BTSDK_AVRCP_ERROR_INVALID_PARAMETER	Invalid parameter, sent if the TG received a PDU with a parameter ID that it did not understand. Sent if there is only one parameter ID in the PDU.	All
BTSDK_AVRCP_ERROR_SPECIFIED_PARAMETER_NOTFOUND	Specified parameter not found, sent if the parameter ID is understood, but content is wrong or corrupted.	All
BTSDK_AVRCP_ERROR_INTERNAL_ERROR	Internal Error, sent if there are error conditions not covered by a more specific error code.	All
BTSDK_AVRCP_ERROR_UID_CHANGED	UID Changed – The UIDs on the device have changed	All
BTSDK_AVRCP_ERROR_RESERVED	Reserved	All
BTSDK_AVRCP_ERROR_INVALID_DIRECTION	Invalid Direction – The Direction parameter is invalid	Change Path
BTSDK_AVRCP_ERROR_NOTA_DIRECTORY	Not a Directory – The UID provided does not refer to a folder item	Change Path
BTSDK_AVRCP_ERROR_UID_DOESNOT_EXIST	Does Not Exist – The UID provided does not refer to any currently valid item	ChangePath, PlayItem, AddToNowPlaying, GetItemAttributes
BTSDK_AVRCP_ERROR_INVALID_SCOPE	Invalid Scope – The scope parameter is invalid	GetFolderItems, PlayItem, AddToNowPlayer, GetItemAttributes
BTSDK_AVRCP_ERROR_RANGE_OUTOF_BOUNDS	Range Out of Bounds – The start of range provided is not valid	GetFolderItems
BTSDK_AVRCP_ERROR_UID_IS	UID is a Directory – The UID	PlayItem,

A_DIRECTORY	provided refers to a directory, which cannot be handled by this media player	AddToNowPlaying
BTSDK_AVRCP_ERROR_MEDIA_INUSE	Media in Use – The media is not able to be used for this operation at this time	PlayItem, AddToNowPlaying
BTSDK_AVRCP_ERROR_NOWPLAYING_LISTFULL	Now Playing List Full – No more items can be added to the Now Playing List	AddToNowPlaying
BTSDK_AVRCP_ERROR_SEARCH_NOTSUPPORTED	Search Not Supported – The Browsed Media Player does not support search	Search
BTSDK_AVRCP_ERROR_SEARCH_INPROGRESS	Search in Progress – A search operation is already in progress	Search
BTSDK_AVRCP_ERROR_INVALID_PLAYERID	Invalid Player Id – The specified Player Id does not refer to a valid player	SetAddressedPlayer, SetBrowsedPlayer
BTSDK_AVRCP_ERROR_PLAYER_NOT_BROWSABLE	Player Not Browsable – The Player Id supplied refers to a Media Player which does not support browsing.	SetBrowsedPlayer
BTSDK_AVRCP_ERROR_PLAYER_NOT_ADDRESSED	Player Not Addressed. The Player Id supplied refers to a player which is not currently addressed, and the command is not able to be performed if the player is not set as addressed.	Search SetBrowsedPlayer
BTSDK_AVRCP_ERROR_NO_VALID_SEARCH_RESULTS	No valid Search Results – The Search result list does not contain valid entries, e.g. after being invalidated due to change of browsed player	GetFolderItems
BTSDK_AVRCP_ERROR_NO_AVAILABLE_PLAYERS	No available players	All
BTSDK_AVRCP_ERROR_ADDRESSED_PLAYER_CHANGED	Addressed Player Changed	Register Notification

Table11 List of Error Status Code

6.2 Data Structures

6.2.1 Service Registry Parameters

6.2.1.1 BtSdkFileTransferReqStru

Definition	typedef struct _BtSdkFileTransferReqStru { BTDEVHDL dev_hdl; BTUINT16 operation; BTUINT16 flag; BTUINT8 file_name[BTSDK_PATH_MAXLENGTH]; } BtSdkFileTransferReqStru, *PBtSdkFileTransferReqStru;	
Description	The structure BtSdkFileTransferReqStru contains information about a request of file transferring through FTP.	
Members	*dev_hdl*	Specifies the handle of the remote device which tries to upload /delete the files.
	operation	Specifies the operation on the file.
	flag	Specifies the current status of uploading /deleting.
	file_name	Specifies the name of the file uploaded /deleted or to be uploaded /deleted.

The ***operation*** member can be one of these values.

FTP specific event	
Value	Description
BTSDK_APP_EV_FTP_PUT	The remote device request to upload the file.
BTSDK_APP_EV_FTP_GET	The remote device request to download the file.
BTSDK_APP_EV_FTP_DEL_FILE	The remote device request to delete the file.

BTSDK_APP_EV_FTP_DEL_FOLDER	The remote device request to delete the folder. In this case, file_name specify the name of the folder to be deleted.

OPP specific event	
Value	Description
BTSDK_APP_EV_OPP_PULL	The remote device request to pull the object.
BTSDK_APP_EV_OPP_PUSH	The remote device request to push the object.
BTSDK_APP_EV_OPP_PUSH_CARD	The remote device request to push the card.
BTSDK_APP_EV_OPP_EXCHG	The remote device request to exchange the objects with local server.

The ***flag*** member can be one of these values.

Value	Description
BTSDK_ER_CONTINUE	The remote device request to upload /delete the file.
BTSDK_ER_SUCCESS	The remote device uploads /deletes the file successfully.
Other value	Error code specifies the reason of uploading /deleting failure.

6.2.1.2 BtSdkAppExtSPPAttrStru

Definition	typedef struct _BtSdkAppExtSPPAttrStru { BTUINT32 size; BTUINT32 sdp_record_handle; BtSdkUUIDStru service_class_128; BTUINT8 svc_name[BTSDK_SERVICENAME_MAXLENGTH]; BTUINT16 rf_svr_chnl; BTUINT8 com_index; } BtSdkAppExtSPPAttrStru, *PBtSdkAppExtSPPAttrStru;	
Description	The structure BtSdkAppExtSPPAttrStru contains additional features of a application defined service based on SPP. This service has its own class identifier, but its behavior is the same as that of a SPP service.	
Members	*size*	Size of the structure, in bytes.
	sdp_record_handle	32bit interger specifies the SDP service record handle.
	service_class_128	128bit UUID specifies the service class of this service record
	svc_name	Name of the service record. This string must be coded in UTF-8 format.
	rf_svr_chnl	RFCOMM server channel assigned to this service record.
	com_index	Integer that specifies the serial port on which the connection is connected. For example, in the Windows OS, set *com_index* to 5 when the connection is connected on the COM5.

Remarks

Currently, both SPP client and server connections are combined with Bluetooth virtual serial ports pre-installed in the OS. After SPP connection is created, the application can use the standard OS serial port I/O functions to transfer data over the SPP connection.

6.2.1.3 BtSdkLocalPSEServerAttrStru

Definition	typedef struct _BtSdkLocalPSEServerAttrStru { BTUINT32 size; BTUINT16 mask; BTUINT8 root_dir[BTSDK_PATH_MAXLENGTH + 1]; BTUINT8 path_delimiter[BTSDK_PBAP_MAX_DELIMITER + 1]; BTUINT8 repositories; } BtSdkLocalPSEServerAttrStru, *PBtSdkLocalPSEServerAttrStru;	
Description	The structure BtSdkLocalPSEServerAttrStru contains additional features of an application defined service based on PSE.	
Members	*size*	Size of the structure, in bytes.
	mask	A flag which specifies parameter read or set. Currently, it is reserved to 0.
	root_dir	A null-terminated ANSI string specifies the path of the PSE. If it is NULL, the default root directory is represented by a *path_delimiter*.
	path_delimiter	A null-terminated ANSI string specifies the delimiter of the path in the virtual folders architecture of the PSE. If it is NULL, then "/" is taken as default.
	repositories	Specifies the type of phone book memory which supported by PSE.

Remarks

The ***repositories*** member can be one of these values.

Value	Description
BTSDK_PBAP_REPO_LOCAL	Store in local memory.
BTSDK_PBAP_REPO_SIM	Store in SIM card.

6.2.1.4 BtSdkLocalMASServerAttrStru

Definition	typedef struct _BtSdkLocalMASServerAttrStru { BTUINT32 size; BTUINT16 mask; BTUINT8 root_dir[BTSDK_PATH_MAXLENGTH + 1]; BTUINT8 path_delimiter; BTUINT8 mas_inst_id; BTUINT8 sup_msg_types; } BtSdkLocalMASServerAttrStru, *PBtSdkLocalMASServerAttrStru;	
Description	The structure BtSdkLocalMASServerAttrStru contains additional features of an application defined service based on MSE.	
Members	*size*	Size of the structure, in bytes.
	mask	A flag which specifies parameter read or set. Currently, it is reserved to 0.
	root_dir	A null-terminated ANSI string specifies the path of the MSE. If it is NULL, the default root directory is represented by a *path_delimiter*.
	path_delimiter	An ANSI character specifies the delimiter of the path in the virtual folders architecture of the MSE. If it is NULL, then '/' is taken as default.
	mas_inst_id	MASInstanceID assigned to this service instance by the application.
	sup_msg_types	Specifies the message types supported by this MAS service instance.

Remarks

The ***sup_msg_types*** member can be one or more of these values.

Value	Description
BTSDK_MAP_SUP_MSG_EMAIL	email RFC2822 or MIME RFC2045 - 47
BTSDK_MAP_SUP_MSG_SMSGSM	GSM short message
BTSDK_MAP_SUP_MSG_SMSCDMA	CDMA short message
BTSDK_MAP_SUP_MSG_MMS	3GPP MMS

6.2.2 Connection Establishment Parameters

6.2.2.1 BtSdkSPPConnParamStru

Definition	typedef struct _BtSdkSPPConnParamStru{ BTUINT32 size; BTUINT16 mask; BTUINT8 com_index; } BtSdkSPPConnParamStru, *PBtSdkSPPConnParamStru;	
Description	The structure BtSdkSPPConnParamStru contains additional parameters required to establish a SPP connection to a SPP server.	
Members	*size*	Size of the structure, in bytes.
	mask	A set of flags which specify connection options. Currently, it is reserved and shall be set to 0.
	com_index	Integer that specifies the serial port on which the SPP connection is connected. For example, in the Windows OS, set *com_index* to 5 when the SPP connection initiated by local application is connected on the COM5.

Remarks

In current version BlueSoleil, both SPP client and server connections are combined with Bluetooth virtual serial ports pre-installed in the OS. After SPP connection is created, the application can use the standard OS serial port I/O functions to transfer data over the SPP connection.

If the application doesn't know which Bluetooth virtual serial port is available, just set *lParam* to 0 when it calls *Btsdk_Connect or Btsdk_ConnectEx* to connect to a SPP server. BlueSoleil will automatically select an idle COM port. The application can call *Btsdk_GetClientPort* to get the actual serial port assigned to this SPP connection in the future.

6.2.2.2 BtSdkOPPConnParamStru

Definition	typedef struct _BtSdkOPPConnParamStru { BTUINT32 size; BTUINT8 inbox_path[BTSDK_PATH_MAXLENGTH]; BTUINT8 outbox_path[BTSDK_PATH_MAXLENGTH]; BTUINT8 own_card[BTSDK_CARDNAME_MAXLENGTH]; } BtSdkOPPConnParamStru, *PBtSdkOPPConnParamStru;	
Description	The structure BtSdkOPPConnParamStru contains additional parameters required to establish an OPP connection to a remote OPP gateway.	
Members	*size*	Size of the structure, in bytes.
	inbox_path	[in] A null-terminated string that specifies the directory used to receive files pushed to the OPP server. It must be a valid path recognized by the OS that running the application.
	outbox_path	[in] A null-terminated string that specifies the directory used to store the files to be pulled from the OPP server. It must be a valid path recognized by the OS that running the application.
	own_card	[in] A null-terminated string that specifies the vCard type (*.vcf) file contains the owner's information. It must be a valid path recognized by the OS that running the application. The OPP server will transfer this file when the OPP client request to pull business card from the OPP server.

6.2.2.3 BtSdkDUNConnParamStru

Definition	typedef struct _BtSdkDUNConnParamStru{ BTUINT32 size; BTUINT16 mask; BTUINT8 com_index; } BtSdkDUNConnParamStru, *PBtSdkDUNConnParamStru;	
Description	The structure BtSdkDUNConnParamStru contains additional parameters required to establish a DUN connection to a remote DUN gateway.	
Members	*size*	Size of the structure, in bytes.
	mask	A set of flags which specify connection options. Currently, it is reserved and shall be set to 0.
	com_index	Integer that specifies the serial port on which the DUN connection is connected. For example, in the Windows OS, set *com_index* to 5 when the DUN connection initiated by local application is connected on the COM5.

Remarks

Currently, DUN Client (Data Terminal) connections are combined with a Bluetooth DUN modem pre-installed in the OS. Each Bluetooth DUN modem is connected to a pre-installed Bluetooth virtual serial port. After connection to a remote DUN gateway is created, the application can use the standard OS modem I/O functions to transfer data over the DUN connection.

If the application doesn't know which Bluetooth virtual serial port is available, just set *lParam* to 0 when it calls *Btsdk_Connect or Btsdk_ConnectEx* to connect to a DUN gateway. BlueSoleil will automatically select an idle COM port that is assigned to a Bluetooth DUN modem. The application can call *Btsdk_GetClientPort* to get the actual serial port assigned to this DUN connection in the future.

6.2.2.4 BtSdkFAXConnParamStru

Definition	typedef struct _BtSdkFAXConnParamStru{ BTUINT32 size; BTUINT16 mask; BTUINT8 com_index; } BtSdkFAXConnParamStru, *PBtSdkFAXConnParamStru;	
Description	The structure BtSdkFAXConnParamStru contains additional parameters required to establish a Fax connection to a remote Fax gateway.	
Members	*size*	Size of the structure, in bytes.
	mask	A set of flags which specify connection options. Currently, it is reserved and shall be set to 0.
	com_index	Integer that specifies the serial port on which the Fax connection is connected. For example, in the Windows OS, set *com_index* to 5 when the Fax connection initiated by local application is connected on the COM5.

Remarks

Currently, Fax Client (Data Terminal) connections are combined with a Bluetooth Fax modem pre-installed in the OS. Each Bluetooth Fax modem is connected to a pre-installed Bluetooth virtual serial port. After connection to a remote Fax gateway is created, the application can use the standard OS modem I/O functions to transfer data over the Fax connection.

If the application doesn't know which Bluetooth virtual serial port is available, just set *lParam* to 0 when it calls *Btsdk_Connect or Btsdk_ConnectEx* to connect to a Fax gateway. BlueSoleil will automatically select an idle COM port that is assigned to a Bluetooth Fax modem. The application can call *Btsdk_GetClientPort* to get the actual serial port assigned to this Fax connection in the future.

6.2.3 Message Parameters

6.2.3.1 Btsdk_HFP_COPSInfoStru

Definition	struct Btsdk_HFP_COPSInfoStru {
	BTUINT8 mode;
	BTUINT8 format;
	BTUINT8 operator_len;
	BTINT8 operator_name[1];
	};	

Description	The structure Btsdk_HFP_COPSInfoStru contains the information of network operator.	
Members	*mode*	Current mode and provides no information with regard to the name of the operator.
	format	The format of the operator parameter string.
	operator_len	The length of the operator name.
	operator_name[1]	the string in alphanumeric format representing the name of the network operator

6.2.3.2 Btsdk_HFP_PhoneInfoStru

Definition	struct Btsdk_HFP_PhoneInfoStru {
	BTUINT8 type;
	BTUINT8 service;
	BTUINT8 num_len;
	BTINT8 number[32];
	BTUINT8 name_len;
	BTINT8 alpha_str[1];
	}	

Description	The structure Btsdk_HFP_PhoneInfoStru contains the information of subscriber.	
Members	*type*	The format of the phone number provided.
	service	This member indicates which service this phone number relates to. It shall be either 4 (voice) or 5 (fax).
	num_len	The length of the phone number provided
	number[32]	Subscriber number, the length shall be 32
	name_len	Length of sub-address.
	alpha_str[1]	String type sub-address of format specified by <cli_validity>

6.2.3.3 Btsdk_HFP_CLCCInfoStru

Definition	struct Btsdk_HFP_CLCCInfoStru{
	BTUINT8 idx;
	BTUINT8 dir;
	BTUINT8 status;
	BTUINT8 mode;
	BTUINT8 mpty;
	BTUINT8 type;
	BTUINT8 num_len;
	BTINT8 number[1];
	}	

Description	The structure Btsdk_HFP_CLCCInfoStru contains the information of current call.	
Members	*idx*	The numbering (start with 1) of the call given by the sequence of setting up or receiving the calls.
	dir	The direction of the call. 0 = outgoing, 1 = incoming
	status	The status of current call. 0=active, 1=held, 2 = dialing (outgoing), 3 = alerting (outgoing), 4 = incoming (incoming), 5 = waiting (incoming)
	mode	Current calling's mode. 0 = voice, 1 = data, 2 = fax
	mpty	The flag of multi-party calling. 0 = no multi-party, 1= multi-party.
	type	The format of the phone number provided.
	num_len	The length of the phone number provided.
	number[1]	Phone number.

6.2.3.4 Btsdk_HFP_CINDInfoStru

Definition	struct Btsdk_HFP_CINDInfoStru {
	BTUINT8 service;
	BTUINT8 call;
	BTUINT8 callsetup;
	BTUINT8 callheld;
	BTUINT8 signal;
	BTUINT8 roam;
	BTUINT8 battchg;
	};	

Description	The structure Btsdk_HFP_CINDInfoStru contains current state mask code for function BtSDK_AGAP_SetCurIndicatorVal.	
Members	*service*	Indicates the status of service. 0 = unavailable, 1 = available
	call	Indicates the status of active call. 0 = no active call, 1 = on an active call
	callsetup	Indicates the status of callsetup. 0 = no callsetup, 1 = incoming, 2 = outgoing, 3 = outalert
	callheld	Indicates the status of callheld. 0 = no callheld, 1 = active-hold, 2 = onhold
	signal	The strength of signal. 0~5
	roam	Indicates the status of roam. 0 = no roam, 1 = roam
	battchg	The strength of signal. The range is 0~5

6.2.3.5 Btsdk_HFP_ConnInfo

Definition	struct Btsdk_HFP_ConnInfo {
	BTUINT16 role;
	BTDEVHDL dev_hdl;
	}	

Description	The structure Btsdk_HFP_ConnInfo contains the information of HFP connection.	
Members	*role*	Specifies the role of the local device of the connection.
	dev_hdl	The handle of remote device.

Remarks

This structure is a parameter of the BTSDK_HFP_EV_SLC_ESTABLISHED_IND and BTSDK_HFP_EV_SLC_RELEASED_IND events.

The ***role*** parameter can be one of these values

Value	Description
BTSDK_CLS_HANDSFREE	Local device acts as a Hands-free device
BTSDK_CLS_HANDSFREE_AG	Local device acts as a Hands-free AG.
BTSDK_CLS_HEADSET	Local device acts as a Handset.
BTSDK_CLS_HEADSET_AG	Local device acts as a Headset AG.

6.2.3.6 Btsdk_HFP_ATCmdResult

Definition	struct Btsdk_HFP_ATCmdResult {
	BTUINT16 cmd_code;
	BTUINT8 result_code;
	}	

Description	The structure Btsdk_HFP_ATCmdResult contains the result of AT command.	
Members	*cmd_code*	Specify the AT command code.
	result_code	Result of the AT command *cmd_code*, it might be BTSDK_HFP_APPERR_TIMEOUT, CME Error Code or standard error result code.

Remarks

This structure is a parameter of the BTSDK_HFP_EV_ATCMD_RESULT events.

6.2.3.7 BtSdkHFPUIParam

Definition	struct BtSdkHFPUIParam {
	BTUINT32 size;
	BTUINT16 mask;
	BTUINT16 features;
	}	

Description	The structure BtSdkHFPUIParam contains the supported feature of local device.	
Members	*size*	The size of the structure BtSdkHFPUIParam
	mask	The mask is reserved and it should be set to 0.
	features	Supported features of local device.

Remarks

1) For HSP, it shall be 0.

2) For HFP-HF, it can be binary combination of the following values:

Value	Description
BTSDK_HF_BRSF_NREC	EC and/or NR function
BTSDK_HF_BRSF_3WAYCALL	Call waiting and 3-way calling
BTSDK_HF_BRSF_CLIP	CLI presentation capability
BTSDK_HF_BRSF_BVRA	Voice recognition activation
BTSDK_HF_BRSF_RMTVOLCTRL	Remote volume control
BTSDK_HF_BRSF_ENHANCED_CALLSTATUS	Enhanced call status
BTSDK_HF_BRSF_ENHANCED_CALLCONTROL	Enhanced call control

3) For HFP-AG, it can be binary combination of the following values:

Value	Description
BTSDK_AG_BRSF_3WAYCALL	Three-way calling
BTSDK_AG_BRSF_NREC	EC and/or NR function
BTSDK_AG_BRSF_BVRA	Voice recognition function
BTSDK_AG_BRSF_INBANDRING	In-band ring tone capability
BTSDK_AG_BRSF_BINP	Attach a number to a voice tag
BTSDK_AG_BRSF_REJECT_CALL	Ability to reject a call
BTSDK_AG_BRSF_ENHANCED_CALLSTATUS	Enhanced call status
BTSDK_AG_BRSF_ENHANCED_CALLCONTROL	Enhanced call control
BTSDK_AG_BRSF_EXTENDED_ERRORRESULT	Extended Error Result Codes

6.2.3.8 BtSdkRmtPSESvcAttrStru

Definition	typedef struct _BtSdkRmtPSESvcAttrStru { BTUINT32 size; BTUINT16 mask; BTUINT8 repositories; } BtSdkRmtPSESvcAttrStru, *PBtSdkRmtPSESvcAttrStru;	
Description	The structure BtSdkRmtPSESvcAttrStru contains the attribute of PSE, to specify ext_attributes of BtSdkRemoteServiceAttrStru.	
Members	*size*	Size of the structure, in bytes.
	mask	A flag which specifies parameter read or set. Currently, it is reserved.
	repositories	Specifies the type of phone book memory which supported by PSE. It can be one of: BTSDK_PBAP_REPO_LOCAL, BTSDK_PBAP_REPO_SIM.

6.2.3.9 BtSdkPassThrReqStru

Definition	typedef struct _BtSdkPassThrReqStru { BTDEVHDL dev_hdl; BTUINT8 state_flag; BTUINT8 op_id; BTUINT8 length; BTUINT8 op_data[1]; } BtSdkPassThrReqStru, *PBtSdkPassThrReqStru;	
Description	The structure BtSdkPassThrReqStru contains information about PASS THROUGH command that will be transferred from CT to TG.	
Members	*dev_hdl*	Handle to the peer device
	state_flag	Button state(0: pressed 1: released)
	op_id	Pass through command ID
	length	Length of op_data, Always 0
	op_data[1]	Additional parameter data, ignored

The ***op_id*** parameter can be one of these values,

Value	Description
BTSDK_AVRCP_OPID_AVC_PANEL_POWER	Power operation.
BTSDK_AVRCP_OPID_AVC_PANEL_VOLUME_UP	Volume Up operation.
BTSDK_AVRCP_OPID_AVC_PANEL_VOLUME_DOWN	Volume Down operation.
BTSDK_AVRCP_OPID_AVC_PANEL_MUTE	Mute operation.
BTSDK_AVRCP_OPID_AVC_PANEL_PLAY	Play operation.
BTSDK_AVRCP_OPID_AVC_PANEL_STOP	Stop operation.
BTSDK_AVRCP_OPID_AVC_PANEL_PAUSE	Pause operation.
BTSDK_AVRCP_OPID_AVC_PANEL_RECORD	Record operation.
BTSDK_AVRCP_OPID_AVC_PANEL_REWIND	Rewind operation.
BTSDK_AVRCP_OPID_AVC_PANEL_FAST_FORWARD	Fast Forward operation.
BTSDK_AVRCP_OPID_AVC_PANEL_EJECT	Reject operation.
BTSDK_AVRCP_OPID_AVC_PANEL_FORWARD	Forward operation.
BTSDK_AVRCP_OPID_AVC_PANEL_BACKWARD	Backward operation.

Remarks

6.2.3.10 BtSdkPassThroughStru

Definition	typedef struct _BtSdkPassThroughStru { BTUINT32 size; BTUINT8 state_flag; BTUINT8 op_id; BTUINT16 vendor_unique_id; } BtSdkPassThroughStru, *P BtSdkPassThroughStru;	
Description	The structure BtSdkPassThroughStru contains information about PASS THROUGH command that will be transferred from CT to TG(Only used be Btsdk_AVRCP_PassThroughReqEx and Btsdk_AVRCP_PassThroughRspEx).	
Members	*size*	Size if this structure.
	state_flag	Button state(BTSDK_AVRCP_BUTTON_STATE_PRESSED: pressed BTSDK_AVRCP_UTTON_STATE_RELEASED: released)
	op_id	Pass through command ID
	Vendor_unique_id	Vendor unique command id. When the op_id == BTSDK_AVRCP_OPID_VENDORUNIQUE, should be: BTSDK_AVRCP_BGN_NEXTGROUP: Next Group BTSDK_AVRCP_BGN_PREVIOUSGROUP: Previoud Group

6.2.3.11 BtSdkListPlayerAppSetValReqStru

Definition	typedef struct _ BtSdkListPlayerAppSetValReqStru { BTUINT32 size; BTUINT8 id; }BtSdkListPlayerAppSetValReqStru, *PBtSdkListPlayerAppSetValReqStru;	
Description	This structure is used in the input parameter of function **错误！未找到引用源。**	
Members	*size*	Size of the stucture, in bytes. Should be *size* >= sizeof(BtSdkListPlayerAppSetValReqStru).
	id	Specific the player application setting attribute IDS, see the following table List of player application setting attributes ID

id	Description
BTSDK_AVRCP_PASA_EQUALIZER_ONOFF_STATUS	Equalizer ON/OFF status
BTSDK_AVRCP_PASA_REPEAT_MODE_STATUS	Repeat Mode status
BTSDK_AVRCP_PASA_SHUFFLE_ONOFF_STATUS	Shuffle ON/OFF status
BTSDK_AVRCP_PASA_SCAN_ONOFF_STATUS	Scan ON/OFF status

6.2.3.12 BtSdkInformBattStatusReqStru

Definition	typedef struct _ BtSdkInformBattStatusReqStru { BTUINT32 size; BTUINT8 id; } BtSdkInformBattStatusReqStru, *PBtSdkInformBattStatusReqStru;	
Description	This structure is used in the input parameter of function **错误！未找到引用源。**	
Members	*size*	Size of the stucture, in bytes. Should be *size* >= sizeof(BtSdkInformBattStatusReqStru).
	id	Battery status, see the following table.

id	Description
BTSDK_AVRCP_BATTERYSTATUS_NORMAL	Battery operation is in normal state
BTSDK_AVRCP_BATTERYSTATUS_WARNING	Unable to operate soon. Specified when battery going down
BTSDK_AVRCP_BATTERYSTATUS_CRITICAL	Can not operate any more. Specified when battery going down.
BTSDK_AVRCP_BATTERYSTATUS_EXTERNAL	Connecting to external power supply
BTSDK_AVRCP_BATTERYSTATUS_FULL_CHARGE	When the device is completely charged.

6.2.3.13 BtSdkRegisterNotifReqStru

Definition	typedef struct _BtSdkRegisterNotifReqStru{ BTUINT32 size; BTUINT8 event_id; BTUINT32 playback_interval; } BtSdkRegisterNotifReqStru, *PBtSdkRegisterNotifReqStru；	
Description	This structure is used in the input parameter of function **错误！未找到引用源。**	
Members	*size*	Size of this structure
	event_id	ID of the event requires notification, e.g. BTSDK_AVRCP_EVENT_PLAYBACK_STATUS_CHANGED
	playback_interval	Playback interval in seconds

6.2.3.14 BtsdkIDStringStru

Definition	typedef struct _BtSdkIDStringStru{ BTUINT8 id; BTUINT16 characterset_id; BTUINT8 len; BTUINT8 string[1]; } BtSdkIDStringStru, *P BtSdkIDStringStru；	
Description	This structure is used to specify the displayable text of the id, such as the displayable text of player application setting attributes.	
Members	*id*	Attribute id or value id.
	characterset_id	Character set ID. Currently only support BTSDK_AVRCP_CHARACTERSETID_UTF8
	len	Length of the player application setting's attribute string.
	String	Player application setting attribute string in specified character set.

The *id* element of this structure is a variable length array of octets. Each *id* is 1 octet long. The application must ensure the correctness and integrity of the parameters.

错误！未找到引用源。.

id	**Description**
BTSDK_AVRCP_PASA_EQUALIZER_ONOFF_STATUS	Equalizer ON/OFF status
BTSDK_AVRCP_PASA_REPEAT_MODE_STATUS	Repeat Mode status
BTSDK_AVRCP_PASA_SHUFFLE_ONOFF_STATUS	Shuffle ON/OFF status
BTSDK_AVRCP_PASA_SCAN_ONOFF_STATUS	Scan ON/OFF status

6.2.3.15 BtSdkGetPlayerAppSetAttrTxtRspStru

Definition	typedef struct _BtSdkGetPlayerSetTxtRsp {
	BTUINT32 size;
	BTUINT32 subpacket_type;
	union {
	BTUINT8 id_num;
	BtSdkIDStringStru id_string;
	};
	}BtSdkGetPlayerAppSetAttrTxtRspStru,
	*PBtSdkGetPlayerAppSetAttrTxtRspStru,
	BtSdkGetPlayerAppSettingAttrTxtRspStru,
	*PBtSdkGetPlayerAppSettingAttrTxtRspStru;	

Description	This structure is used in the input parameter of function 错误！未找到引用源。.	
Members	*size*	Size of this structure, in bytes.
	subpacket_type	Subpacket type When subpacket_type==BTSDK_AVRCP_PACKET_HEAD use the id_num element to specify the attributes number. The member size=2*sizeof(BTUINT32)+sizeof(BTUINT8). When subpacket_type==BTSDK_AVRCP_SUBPACKET, use the id_string element to specify an attribute. The member size=sizeof(BtSdkGetPlayerAppSetAttrTxtRspStru)+(len-1)*sizeof(BTUINT8) , len is the text length.
	id_num	Number of attributes.
	id_string	The BtsdkIDstringStru structure that specifies the player application setting value displayable attribute text.

6.2.3.16 BtSdkGetPlayerAppSettingValTxtRspStru

Definition	typedef struct _BtSdkGetPlayerSetTxtRsp {
	BTUINT32 size;
	BTUINT32 subpacket_type;
	union {
	BTUINT8 id_num;
	BtSdkIDStringStru id_string;
	};
	}BtSdkGetPlayerAppSettingValTxtRspStru,
	*PBtSdkGetPlayerAppSettingValTxtRspStru;
	BtSdkGetPlayerAppSetValTxtRspStru,
	*PBtSdkGetPlayerAppSetValTxtRspStru;	

Description	This structure is used in the input parameter of function **错误！未找到引用源。**.	
Members	*size*	Size of this structure, in bytes.
	subpacket_type	Subpacket type When subpacket_type==BTSDK_AVRCP_PACKET_HEAD use the id_num element to specify the attributes number. The member size=2*sizeof(BTUINT32)+sizeof(BTUINT8). When subpacket_type==BTSDK_AVRCP_SUBPACKET, use the id_string element to specify an attribute. The member size=sizeof(BtSdkGetPlayerAppSetValTxtRspStru)+(len-1)*sizeof(BTUINT8) , len is the text length.
	id_num	Number of attributes.
	id_string	The BtsdkIDstringStru structure that specifies the player application setting value displayable attribute text.

6.2.3.17 BtSdkGetCurPlayerAppSetValReqStru

Definition	typedef struct _ BtSdkGetCurPlayerAppSetValReqStru { BTUINT32 size; BTUINT8 num; BTUINT8 id[1]; }BtSdkGetCurPlayerAppSetValReqStru, *PBtSdkGetCurPlayerAppSetValReqStru;	
Description	This structure is used in the input parameter of function **错误！未找到引用源。**	
Members	*size*	Size of the stucture, in bytes. Should be *size* >= sizeof(BtSdkGetCurPlayerAppSetValReqStru).
	num	Number of player application setting values.
	id	Pointer to the buffer contains the player application setting value ID.

Remarks

The *id* element of this structure is a variable length array of octets. Each *id* is 1 octet long. The application must ensure the correctness and integrity of the parameters.

See **错误！未找到引用源。**.

id	Description
BTSDK_AVRCP_PASA_EQUALIZER_ONOFF_STATUS	Equalizer ON/OFF status
BTSDK_AVRCP_PASA_REPEAT_MODE_STATUS	Repeat Mode status
BTSDK_AVRCP_PASA_SHUFFLE_ONOFF_STATUS	Shuffle ON/OFF status
BTSDK_AVRCP_PASA_SCAN_ONOFF_STATUS	Scan ON/OFF status

6.2.3.18 BtSdkTrackChangedStru

Definition	typedef struct _BtSdkTrackChangedStru { BTUINT32 size; BTUINT8 rsp_code; BTUINT8 identifier[8]; } BtSdkTrackChangedStru, *PBtSdkTrackChangedStru;	
Description	This structure is used in the input parameter of function Btsdk_AVRCP_EventTrackChanged.	
Members	*size*	Size of the structure, in bytes.
	rsp_code	Response message mark.
	identifier[8]	Unique Identifier to identify an element on TG, as is used for the GetElementAttributes command.

Response Code	Description
BTSDK_AVRCP_RSP_INTERIM	Interim Rsponse. The initial response to RegisterNotification command shall be this notification type.
BTSDK_AVRCP_RSP_CHANGED	Changed Notificion. When the status of the events specified has been changed, should notify the CT with this notification type.

6.2.3.19 BtSdkTrackReachEndStru

Definition	typedef struct _ BtSdkTrackReachEndStru { BTUINT32 size; BTUINT8 rsp_code; } BtSdkTrackReachEndStru, *PBtSdkTrackReachEndStru,	
Description	This structure is used in the input parameter of function Btsdk_AVRCP_EventTrackReachEnd.	
Members	*size*	Size of the strcture, in bytes. Should be size >=sizeof(BtsdkTrackReachEndStru)
	rsp_code	Response message mark

Response Code	Description
BTSDK_AVRCP_RSP_INTERIM	Interim Rsponse. The initial response to RegisterNotification command shall be this notification type.
BTSDK_AVRCP_RSP_CHANGED	Changed Notificion. When the status of the events specified has been changed, should notify the CT with this notification type.

6.2.3.20 BtSdkTrackReachStartStru

Definition	typedef struct _ BtSdkTrackReachStartStru { BTUINT32 size; BTUINT8 rsp_code; } BtSdkTrackReachStartStru, *PBtSdkTrackReachStartStru;	
Description	This structure is used in the input parameter of function Btsdk_AVRCP_EventTrackReachStart.	
Members	*size*	Size of the strcture, in bytes. Should be size >=sizeof(BtSdkTrackReachStartStru)
	rsp_code	Response message mark

Response Code	Description
BTSDK_AVRCP_RSP_INTERIM	Interim Rsponse. The initial response to RegisterNotification command shall be this notification type.
BTSDK_AVRCP_RSP_CHANGED	Changed Notificion. When the status of the events specified has been changed, should notify the CT with this notification type.

6.2.3.21 BtSdkNowPlayingContentChangedStru

Definition	typedef struct _ BtSdkNowPlayingContentChangedStru { BTUINT32 size; BTUINT8 rsp_code; }BtSdkNowPlayingContentChangedStru, *PBtSdkNowPlayingContentChangedStru;	
Description	This structure is used in the input parameter of function Btsdk_AVRCP_EventNowPlayingContentChangedStru.	
Members	*size*	Size of the strcture, in bytes. Should be size >=sizeof(BtSdkNowPlayingContentChangedStru)
	rsp_code	Response message mark

Response Code	Description
BTSDK_AVRCP_RSP_INTERIM	Interim Rsponse. The initial response to RegisterNotification command shall be this notification type.
BTSDK_AVRCP_RSP_CHANGED	Changed Notificion. When the status of the events specified has been changed, should notify the CT with this notification type.

6.2.3.22 BtSdkAvailablePlayerChangedStru

Definition	typedef struct _ BtSdkAvailablePlayerChangedStru { BTUINT32 size; BTUINT8 rsp_code; }BtSdkAvailablePlayerChangedStru, *PBtSdkAvailablePlayerChangedStru;	
Description	This structure is used in the input parameter of function Btsdk_AVRCP_EventAvailablePlayerChanged.	
Members	*size*	Size of the strcture, in bytes. Should be size >=sizeof(BtSdkAvailablePlayerChangedStru)
	rsp_code	Response message mark

Response Code	Description
BTSDK_AVRCP_RSP_INTERIM	Interim Rsponse. The initial response to RegisterNotification command shall be this notification type.
BTSDK_AVRCP_RSP_CHANGED	Changed Notificion. When the status of the events specified has been changed, should notify the CT with this notification type.

6.2.3.23 BtSdkPlayPosChangedStru

Definition	typedef struct _ BtSdkPlayPosChangedStru { BTUINT32 size; BTUINT8 rsp_code; BTUINT32 pos; } BtSdkPlayPosChangedStru , *PBtSdkPlayPosChangedStru, BtSdkPlayBackPosChangedStru, *P BtSdkPlayBackPosChangedStru;	
Description	The structure is used in the input parameter of function Btsdk_AVRCP_EventPlayPosChanged.	
Members	*size*	Size of this structure, in bytes. Should be size >= sizeof(BtsdkPlayPosChangedStru).
	rsp_code	Response message mark
	pos	Current playback position in millisecond. If no track currently selected, then return 0xFFFFFFFF in the INTERIM response.

Response Code	Description
BTSDK_AVRCP_RSP_INTERIM	Interim Rsponse. The initial response to RegisterNotification command shall be this notification type.
BTSDK_AVRCP_RSP_CHANGED	Changed Notificion. When the status of the events specified has been changed, should notify the CT with this notification type.

6.2.3.24 BtSdkGetCapabilitiesReqStru

Definition	typedef struct _ BtSdkGetCapabilitiesReqStru { BTUINT32 size; BTUINT8 id; } BtSdkGetCapabilitiesReqStru, *PBtSdkGetCapabilitiesReqStru;	
Description	This structure is used in the input parameter of function 错误！未找到引用源。	
Members	*size*	Size of the stucture, in bytes. Should be *size* >= sizeof(BtSdkGetCapabilitiesReqStru).
	id	The capabilitiy requested.

id	Description
BTSDK_AVRCP_CAPABILITYID_COMPANY_ID	Requests the list of CompanyID supported by TG.
BTSDK_AVRCP_CAPABILITYID_EVENTS_SUPPORTE	Requests the list of events supported by the TG.

6.2.3.25 BtSdkPlayerAppSetChangedStru

Definition	typedef struct _BtSdkPlayerAppSetChangedStru { BTUINT32 size; BTUINT8 rsp_code; BTUINT8 num; BtSdkIDPairStru id[1]; }BtSdkPlayerAppSetChangedStru, *PBtSdkPlayerAppSetChangedStru;	
Description	The structure is used in the input parameter of function Btsdk_AVRCP_EventPlayerAppSetChanged.	
Members	*size*	Size if the structure, in bytes. Should be size>=sieof(BtsdkPlayerAppSetChangedStru).
	rsp_code	Response message mark
	num	Number of player application setting attributes that follow.
	id	The BtsdkIDPairStru structure that specifies the setting values for the provided player application setting attributes list.

6.2.3.26 BtSdkGetElementAttrReqStru

Definition	typedef struct _ BtSdkGetElementAttrReqStru { BTUINT32 size; BTUINT8 identifier[8]; BTUINT8 num; BTUINT32 attr_id[1]; } BtSdkGetElementAttrReqStru, *PBtSdkGetElementAttrReqStru;	
Description	This structure is used in the input parameter of function 错误！未找到引用源。.	
Members	*size*	Size of the stucture, in bytes. Should be *size* >= sizeof(BtSdkGetElementAttrReqStru).
	identifier	Unique identifier to identify an element on TG. PLAYING(0x0): This should return attribute information for the element which is current track in the TG device. All other values other than 0x0 are currently reserved.
	num	Number of Attributes provided. If *num* is set to zero, all attribute information shall be returned, else attribute information for the specified attribute IDs shall be returned by TG.
	attr_id	Specifies the attribute ID for the attributes to be retrieved.

attr_id	Description
BTSDK_AVRCP_MA_TITLEOF_MEDIA	Title of the media. Any text encoded in specified character set.
BTSDK_AVRCP_MA_NAMEOF_ARTIST	Name of the artist. Any text encoded in specified character set.
BTSDK_AVRCP_MA_NAMEOF_ALBUM	Name of the album. Any text encoded in specified character set.
BTSDK_AVRCP_MA_NUMBEROF_MEDIA	Number of the media (ex. Track number of the CD). Numeric ASCII text with zero suppresses.
BTSDK_AVRCP_MA_TOTALNUMBEROF_MEDIA	Total number of the media (ex. Total track number of the CD). Numeric ASCII text with zero suppresses.
BTSDK_AVRCP_MA_GENRE	Genre. Any text encoded in specified character set.
BTSDK_AVRCP_MA_PLAYING_TIME	Playing time in millisecond. Numeric ASCII text with zero suppresses. (Ex.

	2min30sec -> 150000)

6.2.3.27 BtSdkBattStatusChangedStru

Definition	typedef struct _BtSdkBattStatusChangedStru { BTUINT32 size; BTUINT8 rsp_code; BTUINT8 id; }BtSdkBattStatusChangedStru, *PBtSdkBattStatusChangedStru;	
Description	This structure is used in the input parameter of function 错误！未找到引用源。.	
Members	*size*	Size of the stucture, in bytes. Should be *size* >= sizeof(BtSdkBattStatusChangedStru).
	rsp_code	Response code.
	id	Battery status.

id	Description
BTSDK_AVRCP_BATTERYSTATUS_NORMAL	Battery operation is in normal state
BTSDK_AVRCP_BATTERYSTATUS_WARNING	Unable to operate soon. Specified when battery going down
BTSDK_AVRCP_BATTERYSTATUS_CRITICAL	Can not operate any more. Specified when battery going down.
BTSDK_AVRCP_BATTERYSTATUS_EXTERNAL	Connecting to external power supply
BTSDK_AVRCP_BATTERYSTATUS_FULL_CHARGE	When the device is completely charged.

Notification Type	Description
BTSDK_AVRCP_RSP_INTERIM	Interim Rsponse. The initial response to RegisterNotification command shall this notification type.
BTSDK_AVRCP_RSP_CHANGED	Changed Notificion. When the status of the events specified has been changed, should notify the CT with this notification type.

6.2.3.28 BtSdkRegisterNotifiReqStru

Definition	typedef struct _ BtSdkRegisterNotifiReqStru { BTUINT32 size; BTUINT8 event_id; BTUINT32 playback_interval; } BtSdkRegisterNotifiReqStru, *PBtSdkRegisterNotifiReqStru;	
Description	This structure is used in the input parameter of function 错误！未找到引用源。RegNotifReq.	
Members	*size*	Size of the stucture, in bytes. Should be *size* >= sizeof(BtSdkRegisterNotifiReqStru).
	event_id	Event for which the CT requires notifications.
	playback_interval	Specifies the time interval (in seconds) at which the change in playback position will be notified. If the song is being forwarded/rewound, a notification will be received whenever the playback position will change by this value. (Applicable only for EventID BTSDK_AVRCP_EVENT_PLAYBACK_POS_CHANGED. For other events, value of this parameter is ignored.)

event_id	Description
BTSDK_AVRCP_EVENT_PLAYBACK_STATUS_CHANGED	Change in playback status of the current track.
BTSDK_AVRCP_EVENT_TRACK_CHANGED	Change of current track
BTSDK_AVRCP_EVENT_TRACK_REACHED_END	Reached end of a track
BTSDK_AVRCP_EVENT_TRACK_REACHED_START	Reached start of a track
BTSDK_AVRCP_EVENT_PLAYBACK_POS_CHANGED	Change in playback position. Returned after the specified playback notification change notification interval
BTSDK_AVRCP_EVENT_BATT_STATUS_CHANGED	Change in battery status
BTSDK_AVRCP_EVENT_SYSTEM_STATUS_CHANGED	Change in system status
BTSDK_AVRCP_EVENT_PLAYER_APPLICATION_SETTING_CHANGED	Change in player application setting

BTSDK_AVRCP_EVENT_NOW_PLAYING_CONTENT_CHANGED	The content of the Now Playing list has changed
BTSDK_AVRCP_EVENT_AVAILABLE_PLAYERS_CHANGED	The available players have changed
BTSDK_AVRCP_EVENT_ADDRESSED_PLAYER_CHANGED	The Addressed Player has been changed
BTSDK_AVRCP_EVENT_UIDS_CHANGED	The UIDs have changed
BTSDK_AVRCP_EVENT_VOLUME_CHANGED	The volume has been changed locally on the TG

6.2.3.29 BtSdkSetBrowsedPlayerRspStru

Definition	typedef struct _BtsdkSetBrowsedPlayerRsp {
	BTUINT32 size;
	BTUINT32 subpacket_type;
	union {
	BtsdkSetBrowsedPlayerRspHeadStru packet_head;
	/*BTSDK_AVRCP_PACKET_HEAD*/
	BtsdkSetBrowsedPlayerRspItemStru folder_item;
	/*BTSDK_AVRCP_SUBPACKET*/};
	} BtSdkSetBrowsedPlayerRspStru, *PBtSdkSetBrowsedPlayerRspStru;	

Description	This structure is used in the input parameter of function **错误！未找到引用源。**.	
Members	*size*	Size of the stucture, in bytes. Should be *size* >= sizeof(BtsdkSetBrowsedPlayerRspHeadStru).
	subpacket_type	Subpacket type. When *subpacket_type* == BTSDK_AVRCP_PACKET_HEAD, use the *packet_head* element to specify the Attributes of the Browsed Player which is specified by the SetBrowsedPlayer Command and the folder depth of the current folder. The member *size* = 2 * sizeof(BTUINT32) + sizeof(BtSdkSetBrowsedPlayerRspHeadStru). When *subpacket_type* == BTSDK_AVRCP_SUBPACKET, use the *folder_item* element to specify current browsed path. The member *size* = 2 * sizeof(BTUINT32) + sizeof(BtsdkSetBrowsedPlayerRspItemStru) + (folder_name_len - 1) * sizeof(BTUINT8), folder_name_len is the folder name length.
	packet_head	The **错误！未找到引用源。** struct to specify the Attributes of the Browsed Player which is specified by the SetBrowsedPlayer Command and the folder depth of the current folder.
	folder_item	The **错误！未找到引用源。** struct to specify current browsed path.

6.2.3.30 BtsdkSetBrowsedPlayerRspHeadStru

Definition	typedef struct _ BtsdkSetBrowsedPlayerRspHead{ BTUINT32 items_num; BTUINT16 uid_counter; BTUINT16 characterset_id; BTUINT8 folder_depth; BTUINT8 status; }BtsdkSetBrowsedPlayerRspHeadStru, *PBtsdkSetBrowsedPlayerRspHeadStru;	
Description	This structure is used specify the Attributes of the Browsed Player and the depth of the current folder.	
Members	*items_num*	The number of the items in current folder.
	uid_counter	UID counter.
	characterset_id	The character set ID to be displayed on CT. BTSDK_AVRCP_CHARACTERSETID_UTF8
	folder_depth	The number of Folder Name Length/Folder Name pairs which follow.
	status	The result of the SetBrowsedPlayer operation. If an error has occurred then this is the only field present in the response.

6.2.3.31 BtsdkSetBrowsedPlayerRspItemStru

Definition	typedef struct_BtsdkSetBrowsedPlayerRspItem{ BTUINT16 folder_name_len; BTUINT8 folder_name[1]; }BtsdkSetBrowsedPlayerRspItemStru, *PBtsdkSetBrowsedPlayerRspItemStru;	
Description	This structure is used to specify the Folder Name Length/Folder Name pair.	
Members	*folder_name_len*	The length of the folder name.
	folder_name	The buffer contains Folder name.

6.2.3.32 BtSdkGetFolderItemRspStru

Definition	typedef struct _BtSdkGetFolderItemsRsp {
	BTUINT32 size;
	BTUINT32 subpacket_type;
	union {
	BtSdkBrowsableItemStru item;
	BtSdk4IDStringStru element_attr;
	};
	} BtSdkGetFolderItemRspStru, *PBtSdkGetFolderItemRspStru;	

Description	This structure is used in the input parameter of function 错误！未找到引用源。.	
Members	*size*	Size of the stucture, in bytes.
	subpacket_type	Subpacket Type. When *subpacket_type* == BTSDK_AVRCP_PACKET_BROWSEABLE_ITEM, use the *packet_head* member to specify the Browseable Items. The member *size* = 2 * sizeof(BTUINT32) + *item.item_len*. When *subpacket_type* == BTSDK_AVRCP_PACKET_MEDIA_ATTR, use the *element_attr* member only for specifing the Media Element Attributes, when *item.item_type* == BTSDK_AVRCP_ITEMTYPE_MEDIAELEMENT_ITEM. The member *size* = 2 * sizeof(BTUINT32) + sizeof(BtSdk4IDStringStru) + (*element_attr.len* - 1) * sizeof(BTUINT8).
	item	The 错误！未找到引用源。 struct to specify the Valid Browsed Items.
	element_attr	The 错误！未找到引用源。 struct to specify the Media Element Attributes.

6.2.3.33 BtSdkBrowsableItemStru

Definition	typedef struct _ BtSdkBrowsableItemStru {
	BTUINT16 items_num;
	BTUINT16 uid_counter;
	BTUINT16 item_len;
	BTUINT8 item_type;
	BTUINT8 status;
	union {
	BtSdkMediaPlayerItemStru player_item;
	BtSdkFolderItemStru folder_item;
	BtSdkMediaElementItemStru element_item;
	}
	} BtSdkBrowsableItemStru, *PBtSdkBrowsableItemStru;	

Description	This structure is used to specify the Browseable Item.	
Members	*items_num*	The total number of items returned in this listing.
	uid_counter	The UID Counter.
	item_len	The length of this item, in bytes.
	item_type	See the following Table list of Browsable Items Type. When item_type == BTSDK_AVRCP_ITEMTYPE_MEDIAPLAYER_ITEM, use the player_item member to specify a player item. The member item_len = 2 * sizeof(BTUINT32) + sizeof(BtSdkMediaPlayerItemStru) + (player_item.name_len - 1) * sizeof(BTUINT8). When item_type == BTSDK_AVRCP_ITEMTYPE_FOLDER_ITEM, use the folder_item member to specify a folder item. The member item_len = 2 * sizeof(BTUINT32) + sizeof(BtSdkFolderItemStru) + (folder_item.name_len - 1) * sizeof(BTUINT8). When item_type == BTSDK_AVRCP_ITEMTYPE_MEDIAELEMENT_ITEM, use the element_item member to specify a media element item. The member item_len = 2 * sizeof(BTUINT32) + sizeof(BtSdkMediaElementItemStru) + (element_item.name_len - 1) * sizeof(BTUINT8).

	status	The result of the GetFolderItems operation. If an error has occurred then this is the only field present in the response.
	player_item	The 错误!未找到引用源。struct to specify a media player item.
	folder_item	The 错误!未找到引用源。struct to specify a folder item.
	element_item	The 错误!未找到引用源。struct ro specify a media element item.

Remarks

item_type	Decription
BTSDK_AVRCP_ITEMTYPE_MEDIAPLAYER_ITEM	Media Player Item
BTSDK_AVRCP_ITEMTYPE_FOLDER_ITEM	Folder Item
BTSDK_AVRCP_ITEMTYPE_MEDIAELEMENT_ITEM	Media Element Item

6.2.3.34 BtSdkMediaPlayerItemStru

Definition	typedef struct _BtSdkMediaPlayerItem { BTUINT16 player_id; BTUINT8 play_status; BTUINT8 major_player_type; BTUINT32 player_subtype; BTUINT8 feature_bitmask[16]; BTUINT16 characterset_id; BTUINT16 name_len; BTUINT8 name[1]; } BtSdkMediaPlayerItemStru, *PBtSdkMediaPlayerItemStru;	
Description	This structure is used to specify the Media Player Item.	
Members	*player_id*	A unique identifier for the media player.
	play_status	Player play status, allowed value: BTSDK_AVRCP_PLAYSTATUS_STOPED; BTSDK_AVRCP_PLAYSTATUS_PLAYING; BTSDK_AVRCP_PLAYSTATUS_PAUSED; BTSDK_AVRCP_PLAYSTATUS_FWD_SEEK; BTSDK_AVRCP_PLAYSTATUS_FEV_SEEK; BTSDK_AVRCP_PLAYSTATUS_ERROR.
	major_player_type	Major player type, this filed is a bitmask, allowed value: BTSDK_AVRCP_MAJORPLAYERTYPE_AUDIO; BTSDK_AVRCP_MAJORPLAYERTYPE_VIDEO; BTSDK_AVRCP_MAJORPLAYERTYPE_BROADCASTING_AUDIO; BTSDK_AVRCP_MAJORPLAYERTYPE_BROADCASTING_VIDEO.
	Player_subtype	Player sub type, this filed is a bitmask, allowed value: BTSDK_AVRCP_PLAYERSUBTYPE_AUDIOBOOK; BTSDK_AVRCP_PLAYERSUBTYPE_PODCAST.
	feature_bitmask	Feature bit mask. See the following table .
	characterset_id	The character set ID to be displayed on CT.

	name_len	The length of the player name
	name	The buffer contains player name.

Remarks

No.	Parameter Description	Octet	Bit
BTSDK_AVRCP_FBM_SELECT	Select. This PASSTHROUGH command is supported.	0	0
BTSDK_AVRCP_FBM_UP	Up. This PASSTHROUGH command is supported.	0	1
BTSDK_AVRCP_FBM_DOWN	Down. This PASSTHROUGH command is supported.	0	2
BTSDK_AVRCP_FBM_LEFT	Left. This PASSTHROUGH command is supported.	0	3
BTSDK_AVRCP_FBM_RIGHT	Right. This PASSTHROUGH command is supported.	0	4
BTSDK_AVRCP_FBM_RIGHTUP	right-up. This PASSTHROUGH command is supported.	0	5
BTSDK_AVRCP_FBM_RIGHTDOWN	right-down. This PASSTHROUGH command is supported.	0	6
BTSDK_AVRCP_FBM_LEFTUP	left-up. This PASSTHROUGH command is supported.	0	7
BTSDK_AVRCP_FBM_LEFTDOWN	left-down. This PASSTHROUGH command is supported.	1	0
BTSDK_AVRCP_FBM_ROOTMENU	root menu. This PASSTHROUGH command is supported.	1	1
BTSDK_AVRCP_FBM_SETUPMENU	setup menu. This PASSTHROUGH command is supported.	1	2
BTSDK_AVRCP_FBM_CONENTMENU	contents menu. This PASSTHROUGH command is supported.	1	3
BTSDK_AVRCP_FBM_FAVORITEMNUMU	favorite menu. This PASSTHROUGH command is supported.	1	4
BTSDK_AVRCP_FBM_EXIT	Exit. This PASSTHROUGH command is supported.	1	5
BTSDK_AVRCP_FBM_0	0. This PASSTHROUGH command is supported.	1	6
BTSDK_AVRCP_FBM_1	1. This PASSTHROUGH command is supported.	1	7
BTSDK_AVRCP_FBM_2	2. This PASSTHROUGH command is supported.	2	0
BTSDK_AVRCP_FBM_3	3. This PASSTHROUGH command is supported.	2	1
BTSDK_AVRCP_FBM_4	4. This PASSTHROUGH command is supported.	2	2
BTSDK_AVRCP_FBM_5	5. This PASSTHROUGH command is supported.	2	3
BTSDK_AVRCP_FBM_6	6. This PASSTHROUGH command is supported.	2	4
BTSDK_AVRCP_FBM_7	7. This PASSTHROUGH command is supported.	2	5
BTSDK_AVRCP_FBM_8	8. This PASSTHROUGH command is supported.	2	6
BTSDK_AVRCP_FBM_9	9. This PASSTHROUGH command is supported.	2	7
BTSDK_AVRCP_FBM_DO	Dot. This PASSTHROUGH command is supported.	3	0

T			
BTSDK_AVRCP_FBM_ENTER	Enter. This PASSTHROUGH command is supported.	3	1
BTSDK_AVRCP_FBM_CLEAR	Clear. This PASSTHROUGH command is supported.	3	2
BTSDK_AVRCP_FBM_CHANNLEUP	channel up. This PASSTHROUGH command is supported.	3	3
BTSDK_AVRCP_FBM_CHANNLEDOWN	channel down. This PASSTHROUGH command is supported.	3	4
BTSDK_AVRCP_FBM_PREVIOUSCHANNEL	previous channel. This PASSTHROUGH command is supported.	3	5
BTSDK_AVRCP_FBM_SOUNDSELECT	sound select. This PASSTHROUGH command is supported.	3	6
BTSDK_AVRCP_FBM_INPUTSELCET	input select. This PASSTHROUGH command is supported.	3	7
BTSDK_AVRCP_FBM_DISPLAY_INFORMATION	Display information. This PASSTHROUGH command is supported.	4	0
BTSDK_AVRCP_FBM_HELP	Help. This PASSTHROUGH command is supported.	4	1
BTSDK_AVRCP_FBM_PAGEUP	page up. This PASSTHROUGH command is supported.	4	2
BTSDK_AVRCP_FBM_PAGEDOWN	page down. This PASSTHROUGH command is supported.	4	3
BTSDK_AVRCP_FBM_POWER	Power. This PASSTHROUGH command is supported.	4	4
BTSDK_AVRCP_FBM_VOLUMEUP	volume up. This PASSTHROUGH command is supported.	4	5
BTSDK_AVRCP_FBM_VOLUMEDOWN	volume down. This PASSTHROUGH command is supported.	4	6
BTSDK_AVRCP_FBM_MUTE	Mute. This PASSTHROUGH command is supported.	4	7
BTSDK_AVRCP_FBM_PLAY	Play. This PASSTHROUGH command is supported.	5	0
BTSDK_AVRCP_FBM_STOP	Stop. This PASSTHROUGH command is supported.	5	1
BTSDK_AVRCP_FBM_PAUSE	Pause. This PASSTHROUGH command is supported.	5	2
BTSDK_AVRCP_FBM_RECORD	Record. This PASSTHROUGH command is supported.	5	3
BTSDK_AVRCP_FBM_REWIND	Rewind. This PASSTHROUGH command is supported.	5	4
BTSDK_AVRCP_FBM_FASTFORWARD	fast forward. This PASSTHROUGH command is supported.	5	5

BTSDK_AVRCP_FBM_EJECT	Eject. This PASSTHROUGH command is supported.	5	6
BTSDK_AVRCP_FBM_FORWARD	Forward. This PASSTHROUGH command is supported.	5	7
BTSDK_AVRCP_FBM_BACKWARD	Backward. This PASSTHROUGH command is supported.	6	0
BTSDK_AVRCP_FBM_ANGLE	Angle. This PASSTHROUGH command is supported.	6	1
BTSDK_AVRCP_FBM_SUBPICTURE	Subpicture. This PASSTHROUGH command is supported.	6	2
BTSDK_AVRCP_FBM_F1	F1. This PASSTHROUGH command is supported.	6	3
BTSDK_AVRCP_FBM_F2	F2. This PASSTHROUGH command is supported.	6	4
BTSDK_AVRCP_FBM_F3	F3. This PASSTHROUGH command is supported.	6	5
BTSDK_AVRCP_FBM_F4	F4. This PASSTHROUGH command is supported.	6	6
BTSDK_AVRCP_FBM_F5	F5. This PASSTHROUGH command is supported.	6	7
BTSDK_AVRCP_FBM_VENDOR_UNIQUE	Vendor unique. This PASSTHROUGH command is supported.	7	0
BTSDK_AVRCP_FBM_BASIC_GROUP_NAVIGATION	Basic Group Navigation. This overrules the SDP entry as it is set per player.	7	1
BTSDK_AVRCP_FBM_ADVANCED_CONTROL_PLAYER	Advanced Control Player. This bit is set if the player supports at least AVRCP 1.4.	7	2
BTSDK_AVRCP_FBM_BROWSING	Browsing. This bit is set if the player supports browsing.	7	3
BTSDK_AVRCP_FBM_SEARCHING	Searching. This bit is set if the player supports searching.	7	4
BTSDK_AVRCP_FBM_ADDTO_NOWPLAYING	AddToNowPlaying. This bit is set if the player supports the AddToNowPlaying command.	7	5
BTSDK_AVRCP_FBM_UIDS_UNIQUE_INPLAYERVBROWSE_TREE	UIDs unique in player browse tree. This bit is set if the player is able to maintain unique UIDs across the player browse tree.	7	6
BTSDK_AVRCP_FBM_ONLY_BROWSABLE_WHEN_ADDRESSED	OnlyBrowsableWhenAddressed. This bit is set if the player is only able to be browsed when it is set as the Addressed Player.	7	7
BTSDK_AVRCP_FBM_ONLY_SERCHABLE_WHEN_ADDRESSED	OnlySearchableWhenAddressed. This bit is set if the player is only able to be searched when it is set as the Addressed player.	8	0
BTSDK_AVRCP_FBM_NOWPLAYING	NowPlaying. This bit is set if the player supports the NowPlaying folder. Note that for all players that support browsing this bit shall be set.	8	1
BTSDK_AVRCP_FBM_UIDPERSISTENCY	UIDPersistency. This bit is set if the Player is able to persist UID values between AVRCP Browse Reconnects	8	2

6.2.3.35 BtSdkFolderItemStru

Definition	typedef struct _BtSdkFolderItem { BTUINT8 folder_uid[8]; BTUINT8 folder_type; BTUINT8 is_playable; BTUINT16 characterset_id; BTUINT16 name_len; BTUINT8 name[1]; } BtSdkFolderItemStru, *PBtSdkFolderItemStru;	
Description	This structure is used to specify the Folder Item.	
Members	*folder_uid*	UID of the Folder.
	floder_type	Folder type to indicate what it contains. Allowed value: BTSDK_AVRCP_FOLDERTYPE_MIXED; BTSDK_AVRCP_FOLDERTYPE_TITLES; BTSDK_AVRCP_FOLDERTYPE_ALBUMS; BTSDK_AVRCP_FOLDERTYPE_ARTISTIS; BTSDK_AVRCP_FOLDERTYPE_GENRES; BTSDK_AVRCP_FOLDERTYPE_PLAYLISTS; BTSDK_AVRCP_FOLDERTYPE_YEARS.
	is_playable	If the folder is playable, allowed value: BTSDK_AVRCP_ISPLAYABLE_CANNOT: The folder cannot be played. This means that thefolder UID shall not be passed to either the PlayItem or AddToNowPlaying commands. BTSDK_AVRCP_ISPLAYABLE_CAN: The folder can be played. The folder UID may be passed to the PlayItem and AddtoNowPlaying (if supported) commands. The media player behavior on playing a folder should be same as on the local user interface.
	characterset_id	The character set ID to be displayed on CT.
	name_len	The length of the folder name
	name	The buffer contains folder name.

Remarks

Folders of type Titles shall contain media elements only. The Mixed folder type is used for any folder whose type is not known, or is a combination, for example a folder which contains media elements and subfolders. All other folder types contain only folders. It is recommended that the

TG assign any folder containing media elements the type Titles if it contains only media elements, or Mixed if it contains both media elements and subfolders.
The Folder Type can be used by the CT to improve user experience, e.g. to display fixed icons for each type to the user.

6.2.3.36 BtSdkMediaElementItemStru

Definition	typedef struct _MediaElementItem { BTUINT8 element_uid[8]; BTUINT8 media_type; BTUINT8 attr_num; BTUINT16 characterset_id; BTUINT16 name_len; BTUINT8 name[1]; } BtSdkMediaElementItemStru, *PBtSdkMediaElementItemStru;	
Description	This structure is used to specify the Media Element Item.	
Members	*element_uid*	UID of the Media Element.
	media_type	Folder type, allowed value: BTSDK_AVRCP_MEDIATYPE_AUDIO; BTSDK_AVRCP_MEDIATYPE_VIDEO.
	attr_num	The total number of the media element attributes.
	characterset_id	The character set ID to be displayed on CT.
	name_len	The length of the media element name
	name	The buffer contains media element name.

6.2.3.37 BtSdk4IDStringStru

Definition	typedef struct _BtSdk4IDString { BTUINT32 attr_id; BTUINT16 characterset_id; BTUINT16 len; BTUINT8 value[1]; } BtSdk4IDStringStru, *PBtSdk4IDStringStru;	
Description	This structure is used to specifie the value of the Media Element attributes	
Members	*attr_id*	Attributes ID.
	characterset_id	Character set ID.
	len	Length of the value of the attribue.
	value	The buffer contains the value of the attribute.

6.2.3.38 BtSdkGetCapabilitiesRspStru

Definition	typedef struct _ BtSdkGetCapabilitiesRspStru { BTUINT32 size; BTUINT8 capability_id; BTUINT8 count; BTUINT8 capability[1]; } BtSdkGetCapabilitiesRspStru, *PBtSdkGetCapabilitiesRspStru;	
Description	This structure is used in the input parameter of function 错误！未找到引用源。.	
Members	*size*	Size of the stucture, in bytes. Should be *size* >= sizeof(BtSdkGetCapabilitiesRspStru).
	capability_id	Specifies capability requested.
	count	Specifies the number of CompanyID or EventID returned.
	capability	The buffer contains the list of CompanyID or EventID.

Remarks

The *capability* element of this structure is a variable length array of octets. The application must ensure the correctness and integrity of the parameters.

When *capability_id* = BTSDK_AVRCP_CAPABILITYID_COMPANY_ID, *capability* contains the list of CompanyID supported. Each CompanyID is 3 octets long. All TG devices are expected to send the BT SIG CompanyID(0x001958) as the first supported CompanyID.

When *capability_id* = BTSDK_AVRCP_CAPABILITYID_EVENTS_SUPPORTE, *capability* contains the list of events supported. EventIDs are 1 octet each. TG is expected to respond with all the events supported including the mandatory events defined in AVRCP specification.

EventID	Description
BTSDK_AVRCP_EVENT_PLAYBACK_STATUS_CHANGED	Change in playback status of the current track.
BTSDK_AVRCP_EVENT_TRACK_CHANGED	Change of current track
BTSDK_AVRCP_EVENT_TRACK_REACHED_END	Reached end of a track
BTSDK_AVRCP_EVENT_TRACK_REACHED_START	Reached start of a track
BTSDK_AVRCP_EVENT_PLAYBACK_POS_CHANGED	Change in playback position. Returned after the specified playback notification change notification interval
BTSDK_AVRCP_EVENT_BATT_STATUS_CHANGED	Change in battery status
BTSDK_AVRCP_EVENT_SYSTEM_STATUS_CHANGED	Change in system status

BTSDK_AVRCP_EVENT_PLAYER_APPLICATION_SETTING_CHANGED	Change in player application setting
BTSDK_AVRCP_EVENT_NOW_PLAYING_CONTENT_CHANGED	The content of the Now Playing list has changed
BTSDK_AVRCP_EVENT_AVAILABLE_PLAYERS_CHANGED	The available players have changed,
BTSDK_AVRCP_EVENT_ADDRESSED_PLAYER_CHANGED	The Addressed Player has been changed
BTSDK_AVRCP_EVENT_UIDS_CHANGED	The UIDs have changed
BTSDK_AVRCP_EVENT_VOLUME_CHANGED	The volume has been changed locally on the TG

6.2.3.39 BtSdkGetItemAttrRspStru

Definition	typedef struct _BtSdkGetItemAttrRsp {
	BTUINT32 size;
	BTUINT32 subpacket_type;
	union {
	BtSdkGetItemAttrRspHeadStru packet_head;
	BtSdk4IDStringStru entry;
	};
	} BtSdkGetItemAttrRspStru, *PBtSdkGetItemAttrRspStru;	

Description	This structure is used in the input parameter of function 错误！未找到引用源。.	
Members	*size*	Size of the stucture, in bytes.
	subpacket_type	Subpacket Type. When *subpacket_type* == BTSDK_AVRCP_PACKET_HEAD, use the *packet_head* member to specify the item status and the number of attributes. The member *size* = 2 * sizeof(BTUINT32) + sizeof(BtSdkGetItemAttrRspHeadStru). When *subpacket_type* == BTSDK_AVRCP_SUBPACKET, use the *entry* member to specify an item attribute. The member *size* = 2 * sizeof(BTUINT32) + sizeof(BtSdk4IDStringStru) + (*entry.len* - 1) * sizeof(BTUINT8).
	packet_head	The 错误!未找到引用源。struct to specify the item status and the number of attributes.
	entry	The 错误！未找到引用源。 struct to specify the Media Element Attributes.

6.2.3.40 BtSdkGroupNaviReqStru

Definition	typedef struct _ BtSdkGroupNaviReqStru{ BTDEVHDL dev_hdl; BTUINT8 state_flag; BTUINT16 vendor_unique_id; } BtSdkGroupNaviReqStru, * PBtSdkGroupNaviReqStru;	
Description	This structure is used in the input parameter of function Btsdk_AVRCP_Group_NavigateReq	
Members	dev_hdl	Handle to the peer device
	state_flag	Button state(0: pressed 1: released)
	vendor_unique_id	Vendor Unique Operation IDs

6.2.3.41 BtSdkGetItemAttrRspHeadStru

Definition	typedef struct _BtSdkGetItemAttrRspHead{ BTUINT8 status; BTUINT8 attr_num; } BtSdkGetItemAttrRspHeadStru, *PBtSdkGetItemAttrRspHeadStru;	
Description	This structure is used in the struct 错误！未找到引用源。.	
Members	*status*	The result of the GetItemAttributes operation. If an error has occurred then this is the only field present in the response.
	attr_num	The number of attribute of the item.

6.2.3.42 BtSdkGetItemAttrRspStru

Definition	typedef struct _BtSdkGetItemAttrRsp {
	BTUINT32 size;
	BTUINT32 subpacket_type;
	union {
	BtSdkGetItemAttrRspHeadStru packet_head;
	BtSdk4IDStringStru entry;
	};
	} BtSdkGetItemAttrRspStru, *PBtSdkGetItemAttrRspStru;	

Description	This structure is used in the input parameter of function 错误！未找到引用源。.	
Members	*size*	Size of the stucture, in bytes.
	subpacket_type	Subpacket Type. When *subpacket_type* == BTSDK_AVRCP_PACKET_HEAD, use the *packet_head* member to specify the item status and the number of attributes. The member *size* = 2 * sizeof(BTUINT32) + sizeof(BtSdkGetItemAttrRspHeadStru). When *subpacket_type* == BTSDK_AVRCP_SUBPACKET, use the *entry* member to specify an item attribute. The member *size* = 2 * sizeof(BTUINT32) + sizeof(BtSdk4IDStringStru) + (*entry.len* - 1) * sizeof(BTUINT8).
	packet_head	The 错误!未找到引用源。struct to specify the item status and the number of attributes.
	entry	The 错误！未找到引用源。 struct to specify the Media Element Attributes.

6.2.3.43 BtSdkPlayStatusChangedStru

Definition	typedef struct _ BtSdkPlayStatusChangedStru { BTUINT32 size; BTUINT8 rsp_code; BTUINT8 id; } BtSdkPlayStatusChangedStru, *PBtSdkPlayStatusChangedStru,	
Description	This structure is used in the input parameter of function 错误！未找到引用源。.	
Members	*size*	Size of the stucture, in bytes. Should be *size* >= sizeof(BtSdkPlayStatusChangedRspStru).
	rsp_code	Notifaction Type
	id	The current status of playback. Allowed value: BTSDK_AVRCP_PLAYSTATUS_STOPPED BTSDK_AVRCP_PLAYSTATUS_PLAYING BTSDK_AVRCP_PLAYSTATUS_PAUSED BTSDK_AVRCP_PLAYSTATUS_FWD_SEEK BTSDK_AVRCP_PLAYSTATUS_REV_SEEK BTSDK_AVRCP_PLAYSTATUS_ERROR

Remarks

Response Code	Description
BTSDK_AVRCP_RSP_INTERIM	Interim Rsponse. The initial response to RegisterNotification command shall be this notification type.
BTSDK_AVRCP_RSP_CHANGED	Changed Notificion. When the status of the events specified has been changed, should notify the CT with this notification type.

6.2.3.44 BtSdkSysStatusChangedStru

Definition	typedef struct _ BtSdkSysStatusChangedStru { BTUINT32 size; BTUINT8 rsp_code; BTUINT8 id; } BtSdkSysStatusChangedStru, *PBtSdkSysStatusChangedStru,	
Description	This structure is used in the input parameter of function 错误！未找到引用源。.	
Members	*size*	Size of the stucture, in bytes. Should be *size* >= sizeof(BtSdkSysStatusChangedStru).
	rsp_code	Notifaction Type.
	id	The current System status, Allowed values: BTSDK_AVRCP_SYSTEM_POWER_ON BTSDK_AVRCP_SYSTEM_POWER_OFF BTSDK_AVRCP_SYSTEM_UNPLUGGED

Notification Type	Description
BTSDK_AVRCP_RSP_INTERIM	Interim Rsponse. The initial response to RegisterNotification command shall be this notification type.
BTSDK_AVRCP_RSP_CHANGED	Changed Notificion. When the status of the events specified has been changed, should notify the CT with this notification type.

6.2.3.45 BtSdkVolChangedStru

Definition	typedef struct _ BtSdkVolChangedStru { BTUINT32 size; BTUINT8 rsp_code; BTUINT8 id; } BtSdkVolChangedStru, *PBtSdkVolChangedStru;	
Description	This structure is used in the input parameter of function 错误！未找到引用源。.	
Members	*size*	Size of the stucture, in bytes. Should be *size* >= sizeof(BtSdkVolChangedStru).
	notifi_type	Notifaction Type.
	id	The current Absolute Volume. An Absolute Volume is represented in one octet. The top bit (bit 7) is reserved for future addition (RFA). The volume is specified as a percentage of the maximum. The value 0x0 corresponds to 0%. The value 0x7F corresponds to 100%. Scaling should be applied to achieve values between these two. The existence of this scale does not impose any restriction on the granularity of the volume control scale on the TG.

Notification Type	Description
BTSDK_AVRCP_RSP_INTERIM	Interim Rsponse. The initial response to RegisterNotification command shall be this notification type.
BTSDK_AVRCP_RSP_CHANGED	Changed Notificion. When the status of the events specified has been changed, should notify the CT with this notification type.

6.2.3.46 BtSdkAddrPlayerChangedStru

Definition	typedef struct _ BtSdkAddrPlayerChangedStru{ BTUINT32 size; BTUINT8 rsp_code; BTUINT16 player_id; BTUINT16 uid_counter; } BtSdkAddrPlayerChangedStru, *PBtSdkAddrPlayerChangedStru;	
Description	This structure is used in the input parameter of function 错误！未找到引用源。.	
Members	*size*	Size of the stucture, in bytes. Should be *size* >= sizeof(BtSdkAddrPlayerChangedRspStru).
	rsp_code	Notification type.
	player_id	Unique Media Player Id.
	uid_counter	UID Counter.

Notification Type	Description
BTSDK_AVRCP_RSP_INTERIM	Interim Rsponse. The initial response to RegisterNotification command shall be this notification type.
BTSDK_AVRCP_RSP_CHANGED	Changed Notificion. When the status of the events specified has been changed, should notify the CT with this notification type.

6.2.3.47 BtSdkUIDSChangedStru

Definition	typedef struct _BtSdkUIDSChangedStru { BTUINT32 size; BTUINT8 rsp_code; BTUINT16 uid_counter; } BtSdkUIDSChangedStru, *PBtSdkUIDSChangedStru;	
Description	This structure is used in the input parameter of function 错误！未找到引用源。.	
Members	*size*	Size of the stucture, in bytes. Should be *size* >= sizeof(BtSdkUIDSChangedStru).
	rsp_code	Notifaction Type, See the following table.
	id	The UID Counter of the currently browsed player

Notification Type	Description
BTSDK_AVRCP_RSP_INTERIM	Interim Rsponse. The initial response to RegisterNotification command shall be this notification type.
BTSDK_AVRCP_RSP_CHANGED	Changed Notificion. When the status of the events specified has been changed, should notify the CT with this notification type.

6.2.3.48 BtSdkGetPlayerAppSetAttrTxtReqStru

Definition	typedef struct _ BtSdkGetPlayerAppSetAttrTxtReqStru { BTUINT32 size; BTUINT8 num; BTUINT8 id[1]; }BtSdkGetPlayerAppSetAttrTxtReqStru, *PBtSdkGetPlayerAppSetAttrTxtReqStru;	
Description	This structure is used in the input parameter of function 错误！未找到引用源。.	
Members	*size*	Size of the stucture, in bytes. Should be *size* >= sizeof(BtSdkGetPlayerAppSetAttrTxtReqStru).
	num	Number of player application setting value IDs for which corresponding string is needed.
	id	Pointer to the buffer contains the player application setting attribute ID for which the corresponding attribute diaplayable text is needed.

Remarks

The *id* element of this structure is a variable length array of octet. Each *id* is 1 octet long. The application must ensure the correctness and integrity of the parameters. See following Table provide a List of player application setting attributes ID.

id	Description
BTSDK_AVRCP_PASA_EQUALIZER_ONOFF_STATUS	Equalizer ON/OFF status
BTSDK_AVRCP_PASA_REPEAT_MODE_STATUS	Repeat Mode status
BTSDK_AVRCP_PASA_SHUFFLE_ONOFF_STATUS	Shuffle ON/OFF status
BTSDK_AVRCP_PASA_SCAN_ONOFF_STATUS	Scan ON/OFF status

6.2.3.49 BtSdkGetPlayerAppSetValTxtReqStru

Definition	typedef struct _ BtSdkGetPlayerAppSetValTxtReqStru { BTUINT32 size; BTUINT8 attr_id; BTUINT8 num; BTUINT8 value_id[1]; }BtSdkGetPlayerAppSetValTxtReqStru, *PBtSdkGetPlayerAppSetValTxtReqStru;	
Description	This structure is used in the input parameter of function **错误！未找到引用源。**.	
Members	*size*	Size of the stucture, in bytes. Should be *size* >= sizeof(BtSdkGetPlayerAppSetValTxtReqStru).
	attr_id	Attribute ID
	num	Number of player application setting value for which corresponding string is needed.
	id	Pointer to the buffer contains the player application setting value ID for which the corresponding value string is needed.

Remarks

The *id* element of this structure is a variable length array of octet. Each *id* is 1 octet long. The application must ensure the correctness and integrity of the parameters. The following **错误！未找到引用源。**.

AttributeID	ValueID	Description
BTSDK_AVRCP_PASA_EQUALIZER_ONOFF_STATUS	BTSDK_AVRCP_EQUALIZER_OFF	Equalizer ON
	BTSDK_AVRCP_EQUALIZER_ON	Equalizer OFF
BTSDK_AVRCP_PASA_REPEAT_MODE_STATUS	BTSDK_AVRCP_REPEAT_MODE_OFF	Repeat Mode OFF
	BTSDK_AVRCP_REPEAT_MODE_SINGLE_TRACK_REPEAT	Single track repeat
	BTSDK_AVRCP_REPEAT_MODE_ALL_TRACK_REPEAT	All track repeat
	BTSDK_AVRCP_REPEAT_MODE_GROUP_REPEAT	Group repeat
BTSDK_AVRCP_PASA_SHUFFLE_ONOFF_STATUS	BTSDK_AVRCP_SHUFFLE_OFF	Shuffle OFF
	BTSDK_AVRCP_SHUFFLE_ALL_TRACKS_SHUFFLE	All tracks shuffle
	BTSDK_AVRCP_SHUFFLE_GROUP_SHUFFLE	Group shuffle
BTSDK_AVRCP_PASA_SCA	BTSDK_AVRCP_SCAN_OFF	Scan OFF

N_ONOFF_STATUS	BTSDK_AVRCP_SCAN_ALL_TRACKS_SCAN	All tracks scan
	BTSDK_AVRCP_SCAN_GROUP_SCAN	Group scan

6.2.3.50 BtSdkInformCharSetReqStru

Definition	typedef struct _ BtSdkInformCharSetReqStru { BTUINT32 size; BTUINT8 num; BTUINT16 characterset_id[1]; } BtSdkInformCharSetReqStru, *PBtSdkInformCharSetReqStru;	
Description	This structure is used in the input parameter of function 错误！未找到引用源。.	
Members	*size*	Size of the stucture, in bytes. Should be *size* >= sizeof(BtSdkInformCharSetReqStru).
	num	Number of displayable character sets.
	characterset_id	Pointer to the buffer contains the character set ID to be displayed on CT.

Remarks

The *characterset_id* element of this structure is a variable length array of octet. Each *characterset_id* is 2 octet long. The application must ensure the correctness and integrity of the parameters.

6.2.3.51 BtSdkSetAddresedPlayerReqStru

Definition	typedef struct _ BtSdkSetAddresedPlayerReqStru { BTUINT32 size; BTUINT16 id; } BtSdkSetAddresedPlayerReqStru, *PBtSdkSetAddresedPlayerReqStru	
Description	This structure is used in the input parameter of function **错误！未找到引用源。**.	
Members	*size*	Size of the stucture, in bytes. Should be *size* >= sizeof(BtSdkSetAddresedPlayerReqStru).
	id	Unique Media Player Id.

6.2.3.52 BtSdkSetBrowsedPlayerReqStru

Definition	typedef struct _ BtSdkSetBrowsedPlayerReqStru { BTUINT32 size; BTUINT16 id; } BtSdkSetBrowsedPlayerReqStru, *PBtSdkSetBrowsedPlayerReqStru	
Description	This structure is used in the input parameter of function 错误！未找到引用源。.	
Members	*size*	Size of the stucture, in bytes. Should be *size* >= sizeof(BtSdkSetBrowsedPlayerReqStru).
	id	Unique Media Player Id.

6.2.3.53 BtSdkChangePathReqStru

Definition	typedef struct _ BtSdkChangePathReqStru { BTUINT32 size; BTUINT16 uid_counter; BTUINT8 direction; BTUINT8 folder_uid[8]; } BtSdkChangePathReqStru, *PBtSdkChangePathReqStru;	
Description	This structure is used in the input parameter of function **错误！未找到引用源。**.	
Members	*size*	Size of the stucture, in bytes. Should be *size* >= sizeof(BtSdkChangePathReqStru).
	uid_counter	The UID Counter.
	direction	The value maybe: BTSDK_AVRCP_DIRECTION_FOLDER_UP BTSDK_AVRCP_DIRECTION_FOLDER_DOWN
	folder_uid	The UID of the folder to naviagete to. This may be retrieved via a GetFolderItems command. If the navigation command is Folder Up this field is reserved.

6.2.3.54 BtSdkGetFolderItemReqStru

Definition	typedef struct _ BtSdkGetFolderItemReqStru { BTUINT32 size; BTUINT8 scope; BTUINT32 start_item; BTUINT32 end_item; BTUINT8 attr_count; BTUINT32 attr_id[1]; } BtSdkGetFolderItemReqStru, *PBtSdkGetFolderItemReqStru;	
Description	This structure is used in the input parameter of function 错误！未找到引用源。.	
Members	*size*	Size of the stucture, in bytes. Should be *size* >= sizeof(BtSdkGetFolderItemReqStru).
	scope	The scope in which media contect navigation may take place. BTSDK_AVRCP_SCOPE_MEDIAPLAYER_LIST BTSDK_AVRCP_SCOPE_MEDIAPLAYER_VIRTUAL_FILESYSTEM BTSDK_AVRCP_SCOPE_MEDIAPLAYER_SEARCH BTSDK_AVRCP_SCOPE_MEDIAPLAYER_NOWPLAYING
	start_item	The offset within the listing of the item which should be the first returned item. The first media element in the listing is at offset 0.
	end_item	The offset within the listing of the item which should be the final returned item.
	attr_count	Allowed values: 0x00 All attributes are requied. There is no following Attribute list. 0x01-0xFE The following Attribute List contains this number of attributes. 0xFF No attributes are requied. There is no following Attribute List.
	attr_id	Attributes which are requested to be returned for each item returned. See the following Table provide a List of Media Attributes

Scope	Valid Browseable Items	Description	Applicable Player

BTSDK_AVRCP_SCOPE_MEDIAPLAYER_LIST	BTSDK_AVRCP_ITEMTYPE_MEDIAPLAYER_ITEM	Contains all available media players.	None
BTSDK_AVRCP_SCOPE_MEDIAPLAYER_VIRTUAL_FILESYSTEM	BTSDK_AVRCP_ITEMTYPE_FOLDER_ITEM BTSDK_AVRCP_ITEMTYPE_MEDIAELEMENT_ITEM	The virtual filesystem containing the media content of the browsed player.	Browsed
BTSDK_AVRCP_SCOPE_MEDIAPLAYER_SEARCH	BTSDK_AVRCP_ITEMTYPE_MEDIAELEMENT_ITEM	The results of a search operation on the browsed player.	Browsed
BTSDK_AVRCP_SCOPE_MEDIAPLAYER_NOWPLAYING	BTSDK_AVRCP_ITEMTYPE_MEDIAELEMENT_ITEM	The Now Playing list (or queue) of the addressed player	Addressed

Browseable item type

item_type	Decription
BTSDK_AVRCP_ITEMTYPE_MEDIAPLAYER_ITEM	Media Player Item
BTSDK_AVRCP_ITEMTYPE_FOLDER_ITEM	Folder Item
BTSDK_AVRCP_ITEMTYPE_MEDIAELEMENT_ITEM	Media Element Item

Media attribute

attr_id	Description
BTSDK_AVRCP_MA_TITLEOF_MEDIA	Title of the media. Any text encoded in specified character set.
BTSDK_AVRCP_MA_NAMEOF_ARTIST	Name of the artist. Any text encoded in specified character set.
BTSDK_AVRCP_MA_NAMEOF_ALBUM	Name of the album. Any text encoded in specified character set.
BTSDK_AVRCP_MA_NUMBEROF_MEDIA	Number of the media (ex. Track number of the CD). Numeric ASCII text with zero suppresses.
BTSDK_AVRCP_MA_TOTALNUMBEROF_MEDIA	Total number of the media (ex. Total track number of the CD). Numeric ASCII text with zero suppresses.
BTSDK_AVRCP_MA_GENRE	Genre. Any text encoded in specified character set.
BTSDK_AVRCP_MA_PLAYING_TIME	Playing time in millisecond. Numeric ASCII text with zero suppresses. (Ex. 2min30sec -> 150000)

6.2.3.55 BtSdkGetItemAttrReqStru

Definition	typedef struct _ BtSdkGetItemAttrReqStru { BTUINT32 size; BTUINT8 scope; BTUINT8 uid[8]; BTUINT16 uid_counter; BTUINT8 attr_num; BTUINT32 attr_id[1]; } BtSdkGetItemAttrReqStru, *PBtSdkGetItemAttrReqStru;	
Description	This structure is used in the input parameter of function 错误！未找到引用源。.	
Members	*size*	Size of the stucture, in bytes. Should be *size* >= sizeof(BtSdkGetItemAttrReqStru).
	scope	The scope in which the UID of the media element item or folder item is valid. Allowed values: BTSDK_AVRCP_SCOPE_MEDIAPLAYER_LIST BTSDK_AVRCP_SCOPE_MEDIAPLAYER_VIRTUAL_FILESYSTEM BTSDK_AVRCP_SCOPE_MEDIAPLAYER_SEARCH BTSDK_AVRCP_SCOPE_MEDIAPLAYER_NOWPLAYING
	uid	The UID of the media element item or folder item to return the attributes.
	uid_counter	The UID Counter
	attr_num	The number of attribute IDs in the following Attribute ID list. If this value is zero then all attributes are requested.
	attr_id	Attribute ID list. See List of Media Attributes of Media Attributes

attr_id	Description
BTSDK_AVRCP_MA_TITLEOF_MEDIA	Title of the media. Any text encoded in specified character set.
BTSDK_AVRCP_MA_NAMEOF_ARTIST	Name of the artist. Any text encoded in specified character set.
BTSDK_AVRCP_MA_NAMEOF_ALBUM	Name of the album. Any text encoded in specified character set.
BTSDK_AVRCP_MA_NUMBEROF_MEDIA	Number of the media (ex. Track number of the CD). Numeric ASCII

	text with zero suppresses.
BTSDK_AVRCP_MA_TOTALNUMBEROF_MEDIA	Total number of the media (ex. Total track number of the CD). Numeric ASCII text with zero suppresses.
BTSDK_AVRCP_MA_GENRE	Genre. Any text encoded in specified character set.
BTSDK_AVRCP_MA_PLAYING_TIME	Playing time in millisecond. Numeric ASCII text with zero suppresses. (Ex. 2min30sec -> 150000)

6.2.3.56 BtSdkSearchReqStru

Definition	typedef struct _BtSdkSearchReq { BTUINT32 size; BTUINT16 characterset_id; BTUINT16 len; BTUINT8 string[1]; } BtSdkSearchReqStru, *PBtSdkSearchReqStru;	
Description	This structure is used in the input parameter of function 错误！未找到引用源。.	
Members	*size*	Size of the stucture, in bytes. Should be *size* >= sizeof(BtSdkSearchReqStru).
	characterset_id	Specifies the Character Set only support UTF-8. BTSDK_AVRCP_CHARACTERSETID_UTF8
	len	The length of the search string in octets.
	string	The string to search on in the specified character set.

6.2.3.57 BtSdkPlayItemReqStru

Definition	typedef struct _ BtSdkPlayItemReqStru { BTUINT32 size; BTUINT8 scope; BTUINT8 uid[8]; BTUINT16 uid_counter; } BtSdkPlayItemReqStru, *PBtSdkPlayItemReqStru,	
Description	This structure is used in the input parameter of function 错误！未找到引用源。.	
Members	*size*	Size of the stucture, in bytes. Should be *size* >= sizeof(BtSdkPlayItemReqStru).
	scope	The scope in which the UID of the media element item or folder item is valid. Refter to Bluetooth AVRCP 1.4 specification 6.10.1. BTSDK_AVRCP_SCOPE_MEDIAPLAYER_LIST BTSDK_AVRCP_SCOPE_MEDIAPLAYER_VIRTUAL_FILESYSTEM BTSDK_AVRCP_SCOPE_MEDIAPLAYER_SEARCH BTSDK_AVRCP_SCOPE_MEDIAPLAYER_NOWPLAYING
	uid	The UID of the media element item or folder item, if supported, to be played. Refter to Bluetooth AVRCP 1.4 specification 6.10.3
	uid_counter	The UID Counter.

6.2.3.58 BtSdkAddToNowPlayingReqStru

Definition	typedef struct _ BtSdkAddToNowPlayingReqStru { BTUINT32 size; BTUINT8 scope; BTUINT8 uid[8]; BTUINT16 uid_counter; } BtSdkAddToNowPlayingReqStru, *PBtSdkAddToNowPlayingReqStru;	
Description	This structure is used in the input parameter of function 错误！未找到引用源。.	
Members	**size**	Size of the stucture, in bytes. Should be *size* >= sizeof(BtSdkAddToNowPlayingReqStru).
	scope	The scope in which the UID of the media element item or folder item, if supported, is valid. Please refter to Bluetooth specification “AVRCP_SPEC_V14r00.pdf” 6.10.1. BTSDK_AVRCP_SCOPE_MEDIAPLAYER_LIST BTSDK_AVRCP_SCOPE_MEDIAPLAYER_VIRTUAL_FILESYSTEM BTSDK_AVRCP_SCOPE_MEDIAPLAYER_SEARCH BTSDK_AVRCP_SCOPE_MEDIAPLAYER_NOWPLAYING
	uid	The UID of the media element item or folder item, if supported, to be added to the now playing folder. Please refter to Bluetooth specification “AVRCP_SPEC_V14r00.pdf” 6.10.3.
	uid_counter	The UID Counter

6.2.3.59 BtSdkSetAbsoluteVolReqStru

Definition	typedef struct _ BtSdkSetAbsoluteVolReqStru { BTUINT32 size; BTUINT8 id; } BtSdkSetAbsoluteVolReqStru, *PbtSdkSetAbsoluteVolReqStru;	
Description	This structure is used in the input parameter of function **错误！未找到引用源。**.	
Members	**size**	Size of the stucture, in bytes. Should be *size* >= sizeof(BtSdkSetAbsoluteVolReqStru).
	id	The volume which is requested. An absolute Volume is represented in one octet. The top bit(bit 7) is reserved for future addition(RFA). The volume is specified as a percentage of the maximum. The value 0x0 corresponds to 0%. The value 0x7F corresponds to 100%. Scaling should be applied to achieve values between these two. The existence of this scale does not impose any restriction on the granularity of the volume control scale on the TG.

6.2.3.60 BtsdkIDPairStru

Definition	typedef struct _ BtSdkIDPairStru { BTUINT8 attr_id; BTUINT8 value_id; } BtSdkIDPairStru, *PBtSdkIDPairStru;	
Description	This structure contains information about the current set values on the TG for the provided player application setting attributes list. .	
Members	*attr_id*	Player application setting's attribute ID for which the values is returned.
	value_id	Currently set player application.

See the following table provide a List of player application setting values ID.

AttributeID	ValueID	Description
BTSDK_AVRCP_PASA_EQUALIZER_ONOFF_STATUS	BTSDK_AVRCP_EQUALIZER_OFF	Equalizer ON
	BTSDK_AVRCP_EQUALIZER_ON	Equalizer OFF
BTSDK_AVRCP_PASA_REPEAT_MODE_STATUS	BTSDK_AVRCP_REPEAT_MODE_OFF	Repeat Mode OFF
	BTSDK_AVRCP_REPEAT_MODE_SINGLE_TRACK_REPEAT	Single track repeat
	BTSDK_AVRCP_REPEAT_MODE_ALL_TRACK_REPEAT	All track repeat
	BTSDK_AVRCP_REPEAT_MODE_GROUP_REPEAT	Group repeat
BTSDK_AVRCP_PASA_SHUFFLE_ONOFF_STATUS	BTSDK_AVRCP_SHUFFLE_OFF	Shuffle OFF
	BTSDK_AVRCP_SHUFFLE_ALL_TRACKS_SHUFFLE	All tracks shuffle
	BTSDK_AVRCP_SHUFFLE_GROUP_SHUFFLE	Group shuffle
BTSDK_AVRCP_PASA_SCAN_ONOFF_STATUS	BTSDK_AVRCP_SCAN_OFF	Scan OFF
	BTSDK_AVRCP_SCAN_ALL_TRACKS_SCAN	All tracks scan
	BTSDK_AVRCP_SCAN_GROUP_SCAN	Group scan

6.2.3.61 BtsdkSetCurPlayerAppSetValReqStru

Definition	typedef struct _ BtSdkSetCurPlayerAppSetValReqStru { BTUINT32 size; BTUINT8 num; BtSdkIDPairStru id[1]; }BtSdkSetCurPlayerAppSetValReqStru, *PBtSdkSetCurPlayerAppSetValReqStru;	
Description	This structure is used in the input parameter of function Btsdk_AVRCP_SetCurPlayerAppSetValReq.	
Members	*size*	Size of the stucture, in bytes. Should be *size* >= sizeof(BtSdkSetCurPlayerAppSetValReqStru).
	num	Number of player application settings value provided.
	id	The **错误!未找到引用源。** structure that specifies the setting values for the provided player application setting attributes list.

Remarks

The *id* element of this structure is a variable length array of BtSdkIDPairStru. Each *id* is 2 octets long. The application must ensure the correctness and integrity of the parameters.

6.2.3.62 BtSdkGetCurPlayerAppSetValRspStru

Definition	typedef struct _ BtSdkGetCurPlayerAppSetValRspStru { BTUINT32 size; BTUINT8 num; BtSdkIDPairStru id[1]; }BtSdkGetCurPlayerAppSetValRspStru, *PBtSdkGetCurPlayerAppSetValRspStru,;	
Description	This structure is used in the input parameter of function **错误！未找到引用源。**.	
Members	*size*	Size of the stucture, in bytes. Should be *size* >= sizeof(BtSdkGetCurPlayerAppSetValRspStru).
	num	Number of player application settings value provided.
	id	The **错误！未找到引用源。** structure that specifies the setting values for the provided player application setting attributes list.

Remarks

The *id* element of this structure is a variable length array of BtSdkIDPairStru. Each *id* is 2 octets long. The application must ensure the correctness and integrity of the parameters.

6.2.3.63 BtSdkGetElementAttrRspStru

Definition	typedef struct _ BtSdkGetElementAttrRspStru {
	BTUINT32 size;
	BTUINT32 subpacket_type;
	union {
	BTUINT8 id_num;
	BtSdk4IDStringStru id_value; };
	} BtSdkGetElementAttrRspStru, *PBtSdkGetElementAttrRspStru;	

Description	The structure is used in the input parameter of function Btsdk_AVRCP_GetItemAttrRsp.	
Members	*size*	Size of the structure, in bytes.
	subpacket_type	Subpacket type. When subpacket_type == BTSDK_AVRCP_PACKET_HEAD, use the id_num element to specify the attributes num. The member size=2*sizeof(BTUINT32)+sizeof(BTUINT8). When subpacket_type == BTSDK_AVRCP_SUBPACKET, use the id_value element to specify an attribute. The member size = sizeof(BtSdkGetElementAttrRspStru)+(len-1)*sizeof(BTUINT8), len is the length of the value of the value of the attribute.
	id_num	Number of attributes.
	id_ value	The Btsdk4IDStringStru struct that specifies the value of the attributes.

List of Media Attributes

attr_id	Description
BTSDK_AVRCP_MA_TITLEOF_MEDIA	Title of the media. Any text encoded in specified character set.
BTSDK_AVRCP_MA_NAMEOF_ARTIST	Name of the artist. Any text encoded in specified character set.
BTSDK_AVRCP_MA_NAMEOF_ALBUM	Name of the album. Any text encoded in specified character set.

BTSDK_AVRCP_MA_NUMBEROF_MEDIA	Number of the media (ex. Track number of the CD). Numeric ASCII text with zero suppresses.
BTSDK_AVRCP_MA_TOTALNUMBEROF_MEDIA	Total number of the media (ex. Total track number of the CD). Numeric ASCII text with zero suppresses.
BTSDK_AVRCP_MA_GENRE	Genre. Any text encoded in specified character set.
BTSDK_AVRCP_MA_PLAYING_TIME	Playing time in millisecond. Numeric ASCII text with zero suppresses. (Ex. 2min30sec -> 150000)

6.2.3.64 BtSdkListPlayerAppSetAttrRspStru

Definition	typedef struct _ BtSdkListPlayerAppSetAttrRspStru { BTUINT32 size; BTUINT8 num; BTUINT8 id[1]; }BtSdkListPlayerAppSetAttrRspStru, *PbtSdkListPlayerAppSetAttrRspStru;	
Description	This structure is used in the input parameter of function 错误！未找到引用源。.	
Members	*size*	Size of the stucture, in bytes. Should be *size* >= sizeof(BtSdkListPlayerAppSetAttrRspStru).
	num	Number of attributes provided.
	id	Pointer to the buffer contains the player application setting attribute ID.

Remarks

The *id* element of this structure is a variable length array of octets. Each *id* is 1 octet long. The application must ensure the correctness and integrity of the parameters. See the following Table provide a List of player application setting attributes ID.

id	Description
BTSDK_AVRCP_PASA_EQUALIZER_ONOFF_STATUS	Equalizer ON/OFF status
BTSDK_AVRCP_PASA_REPEAT_MODE_STATUS	Repeat Mode status
BTSDK_AVRCP_PASA_SHUFFLE_ONOFF_STATUS	Shuffle ON/OFF status
BTSDK_AVRCP_PASA_SCAN_ONOFF_STATUS	Scan ON/OFF status

6.2.3.65 BtSdkPlayStatusRspStru

Definition	typedef struct _ BtSdkPlayStatusRspStru { BTUINT32 size; BTUINT32 song_length; BTUINT32 song_position; BTUINT8 play_status; } BtSdkPlayStatusRspStru, *PBtSdkPlayStatusRspStru;	
Description	This structure is used in the input parameter of function **错误！未找到引用源。**.	
Members	*size*	Size of the stucture, in bytes. Should be *size* >= sizeof(BtSdkPlayStatusRspStru).
	song_length	The Total length of the playing song in milliseconds. The allowed values are 0-(2^{32}-1). (Ex. 02:30 = 150000)
	song_position	The current position of the playing in milliseconds elapsed. The allowed values are 0-(2^{32}-1). (Ex. 02:30 = 150000)
	play_status	Current status of playing. Allowed Values: BTSDK_AVRCP_PLAYSTATUS_STOPPED BTSDK_AVRCP_PLAYSTATUS_PLAYING BTSDK_AVRCP_PLAYSTATUS_PAUSED BTSDK_AVRCP_PLAYSTATUS_FWD_SEEK BTSDK_AVRCP_PLAYSTATUS_REV_SEEK BTSDK_AVRCP_PLAYSTATUS_ERROR

6.2.3.66 BtSdkSetAddresedPlayerRspStru

Definition	typedef struct _ BtSdkSetAddresedPlayerRspStru { BTUINT32 size; BTUINT8 id; } BtSdkSetAddresedPlayerRspStru, *PBtSdkSetAddresedPlayerRspStru;	
Description	This structure is used in the input parameter of function **错误！未找到引用源。**.	
Members	*size*	Size of the stucture, in bytes. Should be *size* >= sizeof(BtSdkSetAddresedPlayerRspStru).
	id	The result of the SetAddresedPlayer operation, the value is BTSDK_AVRCP_ERROR_SUCCESSFUL。

6.2.3.67 BtSdkChangePathRspStru

Definition	typedef struct _ BtSdkChangePathRspStru { BTUINT32 size; BTUINT8 status; BTUINT32 items_num; } BtSdkChangePathRspStru, *PBtSdkChangePathRspStru;	
Description	This structure is used in the input parameter of function 错误！未找到引用源。.	
Members	*size*	Size of the stucture, in bytes. Should be *size* >= sizeof(BtSdkChangePathRspStru).
	status	The result of the ChangePath operation.
	items_num	If the ChangePath succeeded the number of items in the folder which has been changed to, ie the new current folder.

6.2.3.68 BtSdkSearchRspStru

Definition	typedef struct _ BtSdkSearchRspStru { BTUINT32 size; BTUINT8 status; BTUINT16 uid_counter; BTUINT32 items_num; } BtSdkSearchRspStru, *PBtSdkSearchRspStru;	
Description	This structure is used in the input parameter of function 错误！未找到引用源。.	
Members	*size*	Size of the stucture, in bytes. Should be *size* >= sizeof(BtSdkSearchRspStru).
	status	The result of the Search operation. If an error has occured then this is the only field present in the response.
	uid_counter	The UID Counter. Please to Bluetooth Spec “AVRCP_SPEC_V14r00.pdf” 6.10.3.
	items_num	The number of media element items found in the search.

6.2.3.69 BtSdkPlayItemRspStru

Definition	typedef struct _ BtSdkPlayItemRspStru { BTUINT32 size; BTUINT8 id; } BtSdkPlayItemRspStru, *PBtSdkPlayItemRspStru;	
Description	This structure is used in the input parameter of function **错误！未找到引用源。**.	
Members	**size**	Size of the stucture, in bytes. Should be *size* >= sizeof(BtSdkPlayItemRspStru).
	id	The result of the PlayItem operation. Please refter to Table11

6.2.3.70 BtSdkAddToNowPlayingRspStru

Definition	typedef struct _ BtSdkAddToNowPlayingRspStru { BTUINT32 size; BTUINT8 id; }BtSdkAddToNowPlayingRspStru, *PbtSdkAddToNowPlayingRspStru;	
Description	This structure is used in the input parameter of function **错误！未找到引用源。**.	
Members	**size**	Size of the stucture, in bytes. Should be *size* >= sizeof(BtSdkAddToNowPlayingRspStru).
	id	The result of the AddToNowPlaying operation. Please refer to Table11.

6.2.3.71 BtSdkSetAbsoluteVolRspStru

Definition	typedef struct _ BtSdkSetAbsoluteVolRspStru { BTUINT32 size; BTUINT8 id; } BtSdkSetAbsoluteVolRspStru, *PBtSdkSetAbsoluteVolRspStru;	
Description	This structure is used in the input parameter of function **错误！未找到引用源。错误！未找到引用源。**.	
Members	**size**	Size of the stucture, in bytes. Should be *size* >= sizeof(BtSdkSetAbsoluteVolRspStru).
	id	The volume which has actually been set.

6.2.3.72 BtSdkGeneralRejectRspStru

Definition	typedef struct _ BtSdkGeneralRejectRspStru { BTUINT32 size; BTUINT32 cmd_type; BTUINT8 error_code; } BtSdkGeneralRejectRspStru, *PBtSdkGeneralRejectRspStru;	
Description	This structure is used in the input parameter of function 错误！未找到引用源。.	
Members	*size*	Size of the stucture, in bytes. Should be *size* >= sizeof(BtSdkGeneralRejectRspStru).
	cmd_type	Type of the command being rejected.
	error_code	Error code. See the following Table provide a List of Error Status Code

cmd_type	*Description*
BTSDK_APP_EV_AVRCP_GET_CAPABILITIES_RSP	
BTSDK_APP_EV_AVRCP_LIST_PLAYER_SETTING_ATTR_RSP	
BTSDK_APP_EV_AVRCP_LIST_PLAYER_SETTING_VALUES_RSP	
BTSDK_APP_EV_AVRCP_GET_CURRENTPLAYER_SETTING_VALUE_RSP	
BTSDK_APP_EV_AVRCP_SET_CURRENTPLAYER_SETTING_VALUE_RSP	
BTSDK_APP_EV_AVRCP_GET_PLAYER_SETTING_ATTR_TEXT_RSP	
BTSDK_APP_EV_AVRCP_GET_PLAYER_SETTING_VALUE_TEXT_RSP	
BTSDK_APP_EV_AVRCP_INFORM_CHARACTERSET_RSP	
BTSDK_APP_EV_AVRCP_INFORM_BATTERYSTATUS_OF_CT_RSP	
BTSDK_APP_EV_AVRCP_GET_ELEMENT_ATTR_RSP	
BTSDK_APP_EV_AVRCP_GET_PLAY_STATUS_RSP	
BTSDK_APP_EV_AVRCP_SET_ABSOLUTE_VOLUME_RSP	

BTSDK_APP_EV_AVRCP_SET_ADDRESSED_PLAYER_RSP	
BTSDK_APP_EV_AVRCP_SET_BROWSED_PLAYER_RSP	
BTSDK_APP_EV_AVRCP_GET_FOLDER_ITEMS_RSP	
BTSDK_APP_EV_AVRCP_CHANGE_PATH_RSP	
BTSDK_APP_EV_AVRCP_GET_ITEM_ATTRIBUTES_RSP	
BTSDK_APP_EV_AVRCP_PLAY_ITEM_RSP	
BTSDK_APP_EV_AVRCP_SEARCH_RSP	
BTSDK_APP_EV_AVRCP_ADDTO_NOWPLAYING_RSP	
BTSDK_APP_EV_AVRCP_GENERAL_REJECT_RSP	
BTSDK_APP_EV_AVRCP_PLAYBACK_STATUS_CHANGED_NOTIF	
BTSDK_APP_EV_AVRCP_TRACK_CHANGED_NOTIF	
BTSDK_APP_EV_AVRCP_TRACK_REACHED_END_NOTIF	
BTSDK_APP_EV_AVRCP_TRACK_REACHED_START_NOTIF	
BTSDK_APP_EV_AVRCP_PLAYBACK_POS_CHANGED_NOTIF	
BTSDK_APP_EV_AVRCP_BATT_STATUS_CHANGED_NOTIF	
BTSDK_APP_EV_AVRCP_SYSTEM_STATUS_CHANGED_NOTIF	
BTSDK_APP_EV_AVRCP_PLAYER_APPLICATION_SETTING_CHANGED_NOTIF	
BTSDK_APP_EV_AVRCP_NOW_PLAYING_CONTENT_CHANGED_NOTIF	
BTSDK_APP_EV_AVRCP_AVAILABLE_PLAYERS_CHANGED_NOTIF	
BTSDK_APP_EV_AVRCP_ADDRESSED_PLAYER_CHANGED_NOTIF	
BTSDK_APP_EV_AVRCP_UIDS_CHANGED_NOTIF	
BTSDK_APP_EV_AVRCP_VOLUME_CHANGED_NOTIF	

error_code	*Description*
BTSDK_AVRCP_ERROR_INVALID_COMMAND	Invalid command, sent if TG received a PDU that it did not understand.
BTSDK_AVRCP_ERROR_INVALID_PARAMETER	Invalid parameter, sent if the TG received a PDU with a parameter ID that it did not understand. Sent if there is only one parameter ID in the PDU.
BTSDK_AVRCP_ERROR_SPECIFIED_PARAMETER_NOTFOUND	Specified parameter not found., sent if the parameter ID is understood, but content is wrong or corrupted.
BTSDK_AVRCP_ERROR_INTERNAL_ERROR	Internal Error, sent if there are error conditions not covered by a more specific error code.
BTSDK_AVRCP_ERROR_UID_CHANGED	UID Changed – The UIDs on the device have changed
BTSDK_AVRCP_ERROR_RESERVED	Reserved
BTSDK_AVRCP_ERROR_INVALID_DIRECTION	Invalid Direction – The Direction parameter is invalid
BTSDK_AVRCP_ERROR_NOTA_DIRECTORY	Not a Directory – The UID provided does not refer to a folder item
BTSDK_AVRCP_ERROR_UID_DOESNOT_EXIST	Does Not Exist – The UID provided does not refer to any currently valid item
BTSDK_AVRCP_ERROR_INVALID_SCOPE	Invalid Scope – The scope parameter is invalid
BTSDK_AVRCP_ERROR_RANGE_OUTOF_BOUNDS	Range Out of Bounds – The start of range provided is not valid
BTSDK_AVRCP_ERROR_UID_ISA_DIRECTORY	UID is a Directory – The UID provided refers to a directory, which cannot be handled by this media player
BTSDK_AVRCP_ERROR_MEDIA_INUSE	Media in Use – The media is not able to be used for this operation at this time
BTSDK_AVRCP_ERROR_NOWPLAYING_LISTFULL	Now Playing List Full – No more items can be added to the Now Playing List
BTSDK_AVRCP_ERROR_SEARCH_NOTSUPPORTED	Search Not Supported – The Browsed Media Player does not support search
BTSDK_AVRCP_ERROR_SEARCH_INPROGRESS	Search in Progress – A search operation is already in progress
BTSDK_AVRCP_ERROR_INVALID_PLAYERID	Invalid Player Id – The specified Player Id does not refer to a valid player
BTSDK_AVRCP_ERROR_PLAYER_NOT_BROWSABLE	Player Not Browsable – The Player Id supplied refers to a Media Player which does not support browsing.
BTSDK_AVRCP_ERROR_PLAYER_NOT_ADDRESSED	Player Not Addressed. The Player Id supplied refers to a player which is not currently addressed, and the command is not able to be

	performed if the player is not set as addressed.
BTSDK_AVRCP_ERROR_NO_VALID_SEARCH_RESULTS	No valid Search Results – The Search result list does not contain valid entries, e.g. after being invalidated due to change of browsed player
BTSDK_AVRCP_ERROR_NO_AVAILABLE_PLAYERS	No available players
BTSDK_AVRCP_ERROR_ADDRESSED_PLAYER_CHANGED	Addressed Player Changed

6.2.3.73 BtSdkListPlayerAppSetValRspStru

Definition	typedef struct _ BtSdkListPlayerAppSetValRspStru { BTUINT32 size; BTUINT8 num; BTUINT8 id[1]; }BtSdkListPlayerAppSetValRspStru, *PBtSdkListPlayerAppSetValRspStru;	
Description	This structure is used in the input parameter of function **错误！未找到引用源。**	
Members	*size*	Size of the stucture, in bytes. Should be *size* >= sizeof(BtSdkListPlayerAppSetValRspStru).
	num	Number of player application setting values.
	id	Pointer to the buffer contains the player application setting value ID.

Remarks

The *id* element of this structure is a variable length array of octets. Each *id* is 1 octet long. The application must ensure the correctness and integrity of the parameters. See the following Table provide a List of player application setting values ID.

AttributeID	ValueID	Description
BTSDK_AVRCP_PASA_EQUALIZER_ONOFF_STATUS	BTSDK_AVRCP_EQUALIZER_OFF	Equalizer ON
	BTSDK_AVRCP_EQUALIZER_ON	Equalizer OFF
BTSDK_AVRCP_PASA_REPEAT_MODE_STATUS	BTSDK_AVRCP_REPEAT_MODE_OFF	Repeat Mode OFF
	BTSDK_AVRCP_REPEAT_MODE_SINGLE_TRACK_REPEAT	Single track repeat
	BTSDK_AVRCP_REPEAT_MODE_ALL_TRACK_REPEAT	All track repeat
	BTSDK_AVRCP_REPEAT_MODE_GROUP_REPEAT	Group repeat
BTSDK_AVRCP_PASA_SHUFFLE_ONOFF_STATUS	BTSDK_AVRCP_SHUFFLE_OFF	Shuffle OFF
	BTSDK_AVRCP_SHUFFLE_ALL_TRACKS_SHUFFLE	All tracks shuffle
	BTSDK_AVRCP_SHUFFLE_GROUP_SHUFFLE	Group shuffle
BTSDK_AVRCP_PASA_SCAN_ONOFF_STATUS	BTSDK_AVRCP_SCAN_OFF	Scan OFF
	BTSDK_AVRCP_SCAN_ALL_TRACKS_SCAN	All tracks scan
	BTSDK_AVRCP_SCAN_GROUP_SCAN	Group scan

6.2.3.74 BtSdkPBAPParamStru

Definition	typedef struct _BtSdkPBAPParamStru { BTUINT16 mask; BTUINT8 filter[8]; BTUINT16 max_count; BTUINT16 list_offset; BTUINT8 order; BTUINT8 format; BTUINT8 *search_val; BTUINT8 search_attrib; BTUINT8 missed_calls; BTUINT16 pb_size; } BtSdkPBAPParamStru, *PBtSdkPBAPParamStru;	
Description	The structure BtSdkPBAPParamStru contains parameters used to request to and reponse from PullPhoneBook, PullvCardListing and PullvCardEntry.	
Members	*mask*	A set of flags which specify the validity of the member.
	filter	The filter of vCard's attributes, used to PullPhoneBook and PullvCardEntry request. If BTSDK_PBAP_PM_FILTER is not set in *mask*, or the filter contains all-0, then all the attribute values are returned by the server.
	max_count	Specifies the maximum number of vCard handled by PCE. Valid range: 0x0000 to 0xFFFF. 0x0000: the PCE would like to know the number of the vCard object in the phone book. In this case, the other parameters of this structure will be ignored and the server returns the number of the vCard object by *pb_size*. 0xFFFF: the PCE of the vCard does not limit the number of objects. This parameter is only applicable to the request of PullPhoneBook and PullvCardListing. If BTSDK_PBAP_PM_MAXCOUNT is not set in mask, max count is processed as 0xFFFF as default.

	list_offset	Specifies the offset between the returned vCard object and the first vCard object of server. This parameter is only applicable to the request of PullPhoneBook and PullvCardListing. If BTSDK_PBAP_PM_LISTOFFSET is not set in mask, *list_offset* is processed as 0 as default.
	order	Specifies the Sort by vCard object. This parameter is only applicable to the request of PullvCardListing. If BTSDK_PBAP_PM_ORDER is not set in mask, vCard objects are arranged according to the storing number.
	format	Specifies the format of vCard returned from server. It can be one of these values, BTSDK_PBAP_FMT_VCARD21 – vCard2.1, BTSDK_PBAP_FMT_VCARD30 – vCard3.0. This parameter is only applicable to the request of PullPhoneBook and PullvCardListing. If BTSDK_PBAP_PM_FORMAT is not set in mask, the default format is processed as vCard2.1.
	search_val	Specifies value match the vCard attribute. This string must be coded in UTF-8 format. This parameter is only applicable to the request of PullvCardListing. If BTSDK_PBAP_PM_SCHVALUE is not set in mask, all vCard objects are returned as default.
	search_attrib	Specifies flag of value match the vCard attribute. This parameter is only applicable to the request of PullvCardListing. If BTSDK_PBAP_PM_SCHATTR is not set in mask, the default matching is 'N'.
	missed_calls	Specifies the number of un-check missed call. This parameter makes sense only when BTSDK_PBAP_PM_MISSEDCALLS is set in mask
	pb_size	Specifies the number of vCard objects store in PSE. This parameter makes sense only when BTSDK_PBAP_PM_PBSIZE is set in mask

The ***mask*** member can be one or more of these values.

Value	Description
BTSDK_PBAP_PM_FILTER	The value of the *filter* is available.
BTSDK_PBAP_PM_MAXCOUNT	The value of the *max_count* is available.
BTSDK_PBAP_PM_LISTOFFSET	The value of the *list_offset* is available.
BTSDK_PBAP_PM_ORDER	The value of the *order* is available.

BTSDK_PBAP_PM_FORMAT	The value of the *format* is available.
BTSDK_PBAP_PM_SCHVALUE	The value of the *search_val* is available.
BTSDK_PBAP_PM_SCHATTR	The value of the *search_attrib* is available.
BTSDK_PBAP_PM_MISSEDCALLS	The value of the *missed_calls* is available.
BTSDK_PBAP_PM_PBSIZE	The value of the *pb_size* is available.

The ***order*** member can be one of these values.

Value	Description
BTSDK_PBAP_ORDER_INDEXED	Accordance with the storage number vCard sort.
BTSDK_PBAP_ORDER_NAME	Accordance with the vCard object attribute value 'N' to sort.
BTSDK_PBAP_ORDER_PHONETIC	Accordance with the vCard object attribute value 'SOUND' to sort. Note that: only for the plain text of the attribute value 'SOUND'.

The ***search_attrib*** member can be one of these values.

Value	Description
BTSDK_PBAP_SCHATTR_NAME	Used to attribute value 'N'.
BTSDK_PBAP_SCHATTR_NUMBER	Used to attribute value 'TEL'.
BTSDK_PBAP_SCHATTR_SOUND	Used to attribute value 'SOUND'.

6.2.3.75 BtSdkPBAPParserRoutinesStru

Definition	typedef struct _BtSdkPBAPParserRoutinesStru{ Btsdk_vCardParser_Open_Func parse_open; Btsdk_vCardParser_GetProperty_Func get_prop; Btsdk_vCardParser_Close_Func parse_close; Btsdk_vCardParser_FreeProperty_Func parse_free; Btsdk_vCardParser_FindFirstProperty_Func parse_findfirst; Btsdk_vCardParser_FindNextProperty_Func parse_findnext; Btsdk_vCardParser_FindPropertyClose_Func parse_findclose; } BtSdkPBAPParserRoutinesStru, *PBtSdkPBAPParserRoutinesStru;	
Description	The structure BtSdkPBAPParserRoutinesStru contains seven function pointers which are responsible for parsing vCard objects. The server must implement all these functions. The client does not need to implement them.	
Members	*parse_open*	The pointer to a function initializing operation.
	get_prop	The pointer to a function which gets the attribute value. If the vCard object contains the same attribute, the application decides which value will be returned.
	parse_close	The pointer to a function which ends the parse operation.
	parse_free	The pointer to a function which releases the attribute value got by *get_prop*.
	parse_findfirst	The pointer to a function which finds the first attribute.
	parse_findnext	The pointer to a function whicn finds the next attribute.
	parse_findclose	The pointer to a function which ends the current searching process.

6.2.3.76 BtSdkPBAPFindFileRoutinesStru

Definition	typedef struct _BtSdkPBAPFindFileRoutinesStru { Btsdk_FindFirstFile_Func find_first; Btsdk_FindNextFile_Func find_next; Btsdk_FindFileClose_Func find_close; } BtSdkPBAPFindFileRoutinesStru, *PBtSdkPBAPFindFileRoutinesStru;	
Description	The structure BtSdkPBAPFindFileRoutinesStru contains vCard finding functions. The server must implement all these functions. The client does not need to implement them.	
Members	*find_first*	The pointer to a function which finds the first vCard object.
	find_next	The pointer to a function which finds the next vCard object.
	find_close	The pointer to a function which ends the current searching process.

6.2.3.77 BtSdkPBAPFileIORoutinesStru

Definition	typedef struct _BtSdkPBAPFileIORoutinesStru{ Btsdk_OpenFile_Func open_file; Btsdk_CreateFile_Func create_file; Btsdk_WriteFile_Func write_file; Btsdk_ReadFile_Func read_file; Btsdk_GetFileSize_Func get_file_size; Btsdk_RewindFile_Func rewind_file; Btsdk_CloseFile_Func close_file; } BtSdkPBAPFileIORoutinesStru, *PBtSdkPBAPFileIORoutinesStru;	
Description	The structure BtSdkHoldModeStru contains file operation functions. The file here can be a practical file or can be a vCard object. The server must implement all these functions. The client must implement *write_file*.	
Members	*open_file*	The pointer to function which opens a file.
	create_file	The pointer to function which creates a file.
	write_file	The pointer to function which writes a file.
	read_file	The pointer to function which reads a file.
	get_file_size	The pointer to function which gets the size of a file.
	rewind_file	The pointer to function which moves the file pointer to a specified location.
	close_file	The pointer to function which closes a file.

6.2.3.78 BtSdkPBAPDirCtrlRoutinesStru

Definition	typedef struct _BtSdkPBAPDirCtrlRoutinesStru{ Btsdk_ChangDir_Func change_dir; Btsdk_CreateDir_Func create_dir; } BtSdkPBAPDirCtrlRoutinesStru, *PBtSdkPBAPDirCtrlRoutinesStru;	
Description	The structure BtSdkPBAPDirCtrlRoutinesStru contains directory access functions. The directory can be the actual file directory or the virtual directory in line with PBAP standards. The server must implement all these functions. The client does not need to implement them.	
Members	*change_dir*	Change current work directory.
	create_dir	Create a new directory.

6.2.3.79 BtSdkPBAPSvrCBStru

Definition	typedef struct _BtSdkPBAPSvrCBStru{ BtSdkPBAPParserRoutinesStru cardparser_rtns; BtSdkPBAPFindFileRoutinesStru findfile_rtns; BtSdkPBAPFileIORoutinesStru fileio_rtns; BtSdkPBAPDirCtrlRoutinesStru dirctrl_rtns; Btsdk_PBAP_GetMissedCalls_Func get_new_missedcalls; } BtSdkPBAPSvrCBStru, *PBtSdkPBAPSvrCBStru;	
Description	The structure BtSdkPBAPSvrCBStru contains five function sets to finish PSE operation.	
Members	*cardparser_rtns*	Specifies structure which contains vCard parse funcions.
	findfile_rtns	Specifies structure which contains vCard finding funcions.
	fileio_rtns	Specifies structure which contains file operation funcions.
	dirctrl_rtns	Specifies structure which contains directory operation funcions.
	get_new_missedcalls	The pointer to function which gets the number of missed calls that have not yet been checked before this function call. If the service application does not support missed calls, then this function could be ignored.

6.2.3.80 BtSdk_SDAP_PNPINFO

Definition	struct BtSdk_SDAP_PNPINFO{
	BTUINT16 size;
	BTUINT16 mask;
	BTUINT32 svc_hdl;
	BTUINT16 spec_id;
	BTUINT16 vendor_id;
	BTUINT16 product_id;
	BTUINT16 version_value;
	BTUINT16 vendor_id_src;
	};	

Description	The structure BtSdk_SDAP_PNPINFO contains the information of Plug and Play.	
Members	*size*	The size of the structure BtSdk_SDAP_PNPINFO.
	mask	Specify the optional or mandatory bool type attribute mask.
	svc_hdl	The service handle.
	spec_id	Specify the specification ID.
	vendor_id	Specify the vendor ID.
	product_id	Specify the product ID.
	version_value	Specify the version.
	vendor_id_src	Specify the vendor ID source.

Remarks

6.2.3.81 BtSdkRmtDISvcExtAttrStru

Definition	typedef struct BtSdkRmtDISvcExtAttrStru{
	BTUINT32 size;
	BTUINT16 mask;
	BTUINT16 spec_id;
	BTUINT16 vendor_id;
	BTUINT16 product_id;
	BTUINT16 version;
	BTBOOL primary_record;
	BTUINT16 vendor_id_source;
	BTUINT16 list_size;
	BTUINT8 str_url_list[1];
	};	

Description	The structure BtSdkRmtDISvcExtAttrStru contains the information of device ID.	
Members	*size*	The size of the structure BtSdkRmtDISvcExtAttrStru.
	mask	Specify whether an optional attribute value is available.
	spec_id	Specify the specification ID.
	vendor_id	Specify the vendor ID.
	product_id	Specify the product ID.
	version	Specify the version.
	primary_record	Specify the primary record.
	vendor_id_source	Specify the vendor ID source.
	list_size	The size of the text string list.
	str_url_list[1]	Specify the List of ClientExecutableURL, DocumentationURL and ServiceDescription attributes.

Remarks

6.2.3.82 BtSdkRmtMASSvcAttrStru

Definition	typedef struct _BtSdkRmtMASSvcAttrStru { BTUINT32 size; BTUINT16 mask; BTUINT8 mas_inst_id; BTUINT8 sup_msg_types; } BtSdkRmtMASSvcAttrStru, *PBtSdkRmtMASSvcAttrStru;	
Description	The structure BtSdkRmtMASSvcAttrStru contains the attribute of PSE, to specify ext_attributes of BtSdkRemoteServiceAttrStru	
Members	*Size*	The length of the structure
	mask	A flag which specifies parameter read or set. Currently, it is reserved.
	mas_inst_id	The handle of a MAS instance. Each MSE device can support many MAS service instances. Each MAS service instance has unique name and service handle.
	sup_msg_types	The message type that is supported by the MAS service handle. It can be set to values below: BTSDK_MAP_SUP_MSG_EMAIL – support email, BTSDK_MAP_SUP_MSG_SMSGSM– support GSM, SMS, BTSDK_MAP_SUP_MSG_SMSCDMA – support CDMA, SMS, BTSDK_MAP_SUP_MSG_MMS – support 3GPP, MMS

6.2.3.83 BtSdkMAPEvReportObjStru

Definition	typedef struct _BtSdkMAPEvReportObjStru { BTUINT8 ev_type; BTUINT8 msg_type[BTSDK_MAP_MSGTYPE_LEN]; BTUINT8 msg_handle[BTSDK_MAP_MSGHDL_LEN]; BTUINT8 folder[BTSDK_MAP_PATH_LEN]; BTUINT8 old_folder[BTSDK_MAP_PATH_LEN]; BTUINT8 mas_inst_id; } BtSdkMAPEvReportObjStru, *PBtSdkMAPEvReportObjStru;	
Description	The structure BtSdkMAPEvReportObjStru contains the value of service attributes of Printer Administractive User Interface.	
Members	*ev_type*	Event type. It could be one of the values below: BTSDK_MAP_EVT_NEWMSG – NewMessage, BTSDK_MAP_EVT_DELIVERY_OK – DeliverySuccess, BTSDK_MAP_EVT_SEND_OK – SendingSuccess, BTSDK_MAP_EVT_DELIVERY_FAIL – DeliveryFailure, BTSDK_MAP_EVT_SEND_FAIL – SendingFailure, BTSDK_MAP_EVT_MEM_FULL – MemoryFull, BTSDK_MAP_EVT_MEM_READY – MemoryAvailable, the event only occurs when BTSDK_MAP_EVT_MEM_FULL event has occurred. BTSDK_MAP_EVT_MSG_DELETED – MessageDeleted, BTSDK_MAP_EVT_MSG_SHIFT – MessageShift. When the event type is BTSDK_MAP_EVT_MEM_FULL or BTSDK_MAP_EVT_MEM_AVAILABLE, other parameters of this structure can be omitted.
	msg_type	Message type that is been operated. It could be one of the character arrays below: “EMAIL” – e-mail, “SMS_GSM” – short messages for GSM networks “SMS_CDMA” – short messages for CDMA networks “MMS” – 3GPP MMS messages Other character arrays – the message type that current MAP version doesn’t support.
	msg_handle	A null-terminated UTF-8 string which specifies message handle with 16 hexadecimal digits.

	folder	A null-terminated ANSI string which specifies the directory (include path) used by MSE device to store operated message.
	old_folder	A null-terminated ANSI string which specifies the directory (include path) which is used to store message before the message has been transferred. The parameter is valid only when the event type is BTSDK_MAP_MSGT_MSG_SHIFT
	mas_inst_id	MASInstanceID of the MAS service instance delivering this event.

6.2.3.84 BtSdkMAPFindFolderRoutinesStru

Definition	typedef struct _BtSdkMAPFindFolderRoutinesStru { Btsdk_MAP_FindFirstFolder_Func find_first_folder; Btsdk_MAP_FindNextFolder_Func find_next_folder; Btsdk_MAP_FindFolderClose_Func find_folder_close; } BtSdkMAPFindFolderRoutinesStru, *PBtSdkMAPFindFolderRoutinesStru;	
Description	The structure BtSdkMAPFindFolderRoutinesStru contains folder finding functions. The server must implement all these functions. These functions are used by MAS Server to search a directory for all folder objects in case of receiving GetFolderListing request.	
Members	*find_first_folder*	The pointer to a function which finds the first folder object.
	find_next_folder	The pointer to a function which finds the next folder object.
	find_folder_close	The pointer to a function which ends the current searching process.

6.2.3.85 BtSdkMAPFindMsgRoutinesStru

Definition	typedef struct _BtSdkMAPFindMsgRoutinesStru { Btsdk_MAP_FindFirstMsg_Func find_first_msg; Btsdk_MAP_FindNextMsg_Func find_next_msg; Btsdk_MAP_FindMsgClose_Func find_msg_close; } BtSdkMAPFindMsgRoutinesStru, *PBtSdkMAPFindMsgRoutinesStru;	
Description	The structure BtSdkMAPFindMsgRoutinesStru contains message finding functions. The server must implement all these functions. These functions are used by MAS Server to search a directory for all message objects in case of receiving GetMessageListing request.	
Members	*find_first_msg*	The pointer to a function which finds the first message object.
	find_next_msg	The pointer to a function which finds the next message object.
	find_msg_close	The pointer to a function which ends the current searching process.

6.2.3.86 BtSdkMAPFileIORoutinesStru

Definition	typedef struct _BtSdkMAPFileIORoutinesStru { Btsdk_OpenFile_Func open_file; Btsdk_CreateFile_Func create_file; Btsdk_WriteFile_Func write_file; Btsdk_ReadFile_Func read_file; Btsdk_GetFileSize_Func get_file_size; Btsdk_RewindFile_Func rewind_file; Btsdk_CloseFile_Func close_file; } BtSdkMAPFileIORoutinesStru, *PBtSdkMAPFileIORoutinesStru;	
Description	The structure BtSdkMAPFileIORoutinesStru contains file operation functions. The file here can mean a real file or a message object. Each member of the structure corresponds to a pointer of a function. The server should realize all these functions, and the client must realize write_file and read_file.	
Members	*open_file*	The pointer to function which opens a file.
	create_file	The pointer to function which creates a file.
	write_file	The pointer to function which writes a file.
	read_file	The pointer to function which reads a file.
	get_file_size	The pointer to function which gets the size of a file.
	rewind_file	The pointer to function which moves the file pointer to a specified location.
	close_file	The pointer to function which closes a file.

6.2.3.87 BtSdkMAPMsgIORoutinesStru

Definition	typedef struct _BtSdkMAPMsgIORoutinesStru { Btsdk_MAP_ModifyMsgStatus_Func modify_msg_status; Btsdk_MAP_CreateBMsgFile_Func create_bmsg_file; Btsdk_MAP_OpenBMsgFile_Func open_bmsg_file; Btsdk_MAP_PushMsg_Func push_msg; } BtSdkMAPMsgIORoutinesStru, *PBtSdkMAPMsgIORoutinesStru;	
Description	The structure BtSdkMAPMsgIORoutinesStru contains message operation functions. The server must implement all these functions.	
Members	*modify_msg_status*	The pointer to a function which modifys the status of a message.
	create_bmsg_file	The pointer to a function which creates a new empty message object.
	open_bmsg_file	The pointer to a function which opens an existing message object.
	push_msg	The pointer to a function which deals with a message pushed by the MSE client recently.

6.2.3.88 BtSdkMAPMSEStatusRoutinesStru

Definition	typedef struct _BtSdkMAPMSEStatusRoutinesStru { Btsdk_MAP_RegisterNotification_Func register_notification; Btsdk_MAP_UnpdateInbox_Func update_inbox; Btsdk_MAP_GetMSETime_Func get_mse_time; } BtSdkMAPMSEStatusRoutinesStru, *PBtSdkMAPMSEStatusRoutinesStru;	
Description	The structure BtSdkMAPMSEStatusRoutinesStru contains functions that get or change MSE service status. The server must implement at least get_mse_time.	
Members	*register_notification*	The pointer to a function which changes the MSE's notification status.
	update_inbox	The pointer to a function which initiates an update of the MSE's inbox.
	get_mse_time	The pointer to a function which returns the local Time basis of the MSE and its UTC offset.

6.2.3.89 BtSdkMASSvrCBStru

Definition	typedef struct _BtSdkMASSvrCBStru { BtSdkMAPFindFolderRoutinesStru find_folder_rtns; BtSdkMAPFindMsgRoutinesStru find_msg_rtns; BtSdkMAPFileIORoutinesStru file_io_rtns; BtSdkMAPMsgIORoutinesStru msg_io_rtns; BtSdkMAPMSEStatusRoutinesStru mse_status_rtns; } BtSdkMASSvrCBStru, *PBtSdkMASSvrCBStru;	
Description	The structure BtSdkMASSvrCBStru contains function sets to finish MSE server operation.	
Members	*find_folder_rtns*	Specifies structure which contains folder finding funcions.
	find_msg_rtns	Specifies structure which contains message finding funcions.
	file_io_rtns	Specifies structure which contains file operation funcions.
	msg_io_rtns	Specifies structure which contains message operation funcions.
	mse_status_rtns	The pointer to function which contains functions that get or change MSE service status.

6.2.3.90 BtSdkMAPGetFolderListParamStru

Definition	typedef struct _BtSdkMAPGetFolderListParamStru { BTUINT16 mask; BTUINT16 max_count; BTUINT16 start_off; BTUINT16 list_size; } BtSdkMAPGetFolderListParamStru, *PBtSdkMAPGetFolderListParamStru;	
Description	The structure BtSdkMAPGetFolderListParamStru contains parameters for GetFolderListing request and its response.	
Members	*mask*	A set of flags which specify the validity of the member. It could be values or operations below: BTSDK_MAP_GFLP_MAXCOUNT – set the value of max_count BTSDK_MAP_GFLP_STARTOFF – set the value of start_off BTSDK_MAP_GFLP_LISTSIZE – set the value of list_size If the specific flag is not set in mask, the corresponding parameter will not show in OBEX data package
	max_count	The max number of directories that can be returned. The value range is 0 – 0xFFFF. 0 means MCE wants to know the total number of directories. Under this circumstance, other parameters of this structure will be omitted. The object number that server returns will be transferred to application through the "list_size" parameter. If BTSDK_MAP_GFLP_MAXCOUNT is not set in mask, it will be treated as 1024 as default.
	start_off	Number of bytes from *origin*. For example: if the total number of directories is 5, and the "start_off" is set to 2, so just return the last 3 directories. If BTSDK_MAP_GFLP_STARTOFF is not set in mask, it will be treated as 0. And the ListStartOffset parameter will not be set in OBEX request.
	list_size	The number of subdirectory under current directory in server. Only if BTSDK_MAP_GFLP_LISTSIZE is set in mask, the value will be of sense.

6.2.3.91 BtSdkMAPGetMsgListParamStru

<table>
<tr><td>Definition</td><td colspan="2">typedef struct _BtSdkMAPGetMsgListParamStru {
BTUINT32 mask;
BTUINT8 folder[BTSDK_MAP_FOLDER_LEN];
BTUINT16 max_count;
BTUINT16 start_off;
BTUINT32 param_mask;
BTUINT8 filter_period_begin[BTSDK_MAP_TIME_LEN];
BTUINT8 filter_period_end[BTSDK_MAP_TIME_LEN];
BTUINT8 filter_originator[BTSDK_MAP_USERNAME_LEN];
BTUINT8 filter_recipient[BTSDK_MAP_USERNAME_LEN];
BTUINT8 filter_msg_type;
BTUINT8 filter_read_status;
BTUINT8 filter_priority;
BTUINT8 subject_length;
BTUINT16 list_size;
BTBOOL new_msg;
BTUINT8 mse_time[BTSDK_MAP_MSE_TIME_LEN];
} BtSdkMAPGetMsgListParamStru, *PBtSdkMAPGetMsgListParamStru;</td></tr>
<tr><td>Description</td><td colspan="2">The structure BtSdkMAPGetMsgListParamStru contains parameters for GetMessageListing request and their response</td></tr>
<tr><td>Members</td><td>mask</td><td>A set of flags which specify the validity of the member. It could be values below:
BTSDK_MAP_GMLP_MAXCOUNT – set the value of max_count
BTSDK_MAP_GMLP_STARTOFF – set the value of start_offset
BTSDK_MAP_GMLP_PARAMMASK – set the value of param_mask
BTSDK_MAP_GMLP_PERIODBEGIN – set the value of filter_period_begin
BTSDK_MAP_GMLP_PERIODEND – set the value of filter_period_end
BTSDK_MAP_GMLP_ORIGINATOR – set the value of filter_originator
BTSDK_MAP_GMLP_RECIPIENT – set the value of filter_recipient
BTSDK_MAP_GMLP_MSGTYPE – set the value of filter_msg_type</td></tr>
</table>

		BTSDK_MAP_GMLP_READSTATUS – set the value of filter_read_status BTSDK_MAP_GMLP_PRIORITY – set the value of filter_priority BTSDK_MAP_GMLP_SUBJECTLENTH – set the value of subject_length BTSDK_MAP_GMLP_LISTSIZE – set the value of list_size BTSDK_MAP_GMLP_NEWMSG – set the value of new_msg BTSDK_MAP_GMLP_MSETIME – set the value of mse_time If the specific flag is not set in mask, the corresponding parameter will not show in OBEX data package
	folder	A null-terminated UTF-8 string which specifies the directory (not include path) which contains list of message that should be got. Empty character array means current directory under MSE is set as default.
	max_count	The max number of directories that can be returned. The value range is 0 – 0xFFFF. 0 means MCE wants to know the total number of directories. The object number that server returns will be transferred to application through the "list_size" parameter. If BTSDK_MAP_GMLP_MAXCOUNT is not set in mask, it will be treated as 1024 as default.
	start_offset	Number of bytes from *origin*. For example: if the total number of directories is 5, and the "start_off" is set to 2, so just return the last 3 directories. If BTSDK_MAP_GMLP_STARTOFF is not set in mask, it will be treated as 0. And the ListStartOffset parameter will not be set in OBEX request.
	param_mask	A set of flags which specify the parameters contained in the requested Messages-Listing objects. It could be one or more of the values below: BTSDK_MAP_MP_SUBJECT – subject parameter BTSDK_MAP_MP_DATATIME – datatime parameter BTSDK_MAP_MP_SENDERNAME – sender_name parameter BTSDK_MAP_MP_SENDERADDR – sender_addressing parameter BTSDK_MAP_MP_RECIPIENTNAME – recipient_name parameter BTSDK_MAP_MP_RECIPIENTADDR – recipient_addressing parameter BTSDK_MAP_MP_TYPE– type parameter BTSDK_MAP_MP_SIZE – size parameter BTSDK_MAP_MP_RECPSTATUS – reception_status parameter

		BTSDK_MAP_MP_TEXT– text parameter BTSDK_MAP_MP_ATTACHSIZE – attachment_size parameter BTSDK_MAP_MP_PRIORITY – priority parameter BTSDK_MAP_MP_READ – read parameter BTSDK_MAP_MP_SENT – sent parameter BTSDK_MAP_MP_PROTECTED – protected parameter BTSDK_MAP_MP_REPLY2ADDR– replyto_addressing parameter If BTSDK_MAP_MP_PARAMMASK is not set or param_mask is set to 0 in mask, all parameters will be returned as default
	filter_period_begin	A null-terminated string which specifies the begin time of the time interval when the specific message returns. The type of time is "YYYYMMDDTHHMMSS" If BTSDK_MAP_GMLP_PERIODBEGIN is not set in mask or filter_period_begin equals to empty character array, the default return is the message that has been received before the time that filter_period_end points (not include the time point). If neither filter_period_begin nor filter_period_end is set, then return the message that meets all the other conditions. If the time that filter_period_end points is earlier than the time that filter_period_begin points, don't return anything.
	filter_period_end	A null-terminated string which specifies the end time of the time interval when the specific message returns. The type of time is "YYYYMMDDTHHMMSS" If BTSDK_MAP_GMLP_PERIODEND is not set in mask or filter_period_end equals to empty character array, the default return is the message that has been received after the time that filter_period_begin points (include the time point). If neither filter_period_begin nor filter_period_end is set, then return the message that meets all the other conditions. If the time that filter_period_end points is earlier than the time that filter_period_begin points, don't return anything.
	filter_originator	A null-terminated UTF-8 string which specifies the conditions of filter originator that the returned message should meet. MSE uses the condition to match the attribute "N", "TEL", "EMAIL" of vCard, if they contain filter_originator, the match is successful. If BTSDK_MAP_GMLP_ORIGINATOR is not set or the value of filter_originator equals to empty character array, return messages of random filter originator.

	filter_receipient	A null-terminated UTF-8 string which specifies the conditions of filter recipient that the returned message should meet. MSE uses the condition to match the attribute “N”, “TEL”, “EMAIL” of vCard, if they contain filter_recipient, the match is successful. If BTSDK_MAP_GMLP_RECIPIENT is not set or the value of filter_recipient equals to empty character array, return random message of random filter recipient.
	filter_msg_type	The message type that should not be returned. It can be values below:. BTSDK_MAP_FILTEROUT_SMSGSM – omit GSM message BTSDK_MAP_FILTEROUT_SMSCDMA – omit CDMA message BTSDK_MAP_FILTEROUT_EMAIL – omit email BTSDK_MAP_FILTEROUT_MMS – omit 3GPP MMS If BTSDK_MAP_GMLP_MSGTYPE is not set in mask, return all type of messages as default.
	filter_read_status	The reading status that the returned message should meet. It could be one of the values below: BTSDK_MAP_MSG_FILTER_ST_READ – the message has been read BTSDK_MAP_MSG_FILTER_ST_UNREAD – the message hasn’t been read BTSDK_MAP_MSG_FILTER_ST_ALL – do not distinguish the reading status If BTSDK_MAP_GMLP_READSTATUS is not set in mask, BTSDK_MAP_MSG_FILTER_ST_ALL is set as default.
	filter_priority	The condition of priority level that returned message should meet. It could be one of the values below: BTSDK_MAP_MSG_FILTER_PRI_HIGH – high priority level message BTSDK_MAP_MSG_FILTER_PRI_NOHIGH – not high priority level message BTSDK_MAP_MSG_FILTER_PRI_ALL – do not distinguish priority level If BTSDK_MAP_GMLP_PRIORITY is not set in mask, BTSDK_MAP_MSG_FILTER_PRI_ALL is set as default.
	subject_length	The max length of subject of returned message. If BTSDK_MAP_GMLP_SUBJECTLENGTH is not set in mask, do not limit the length of subject of returned message.
	list_size	The number of messages which meet the conditions under current directory. Only if BTSDK_MAP_GMLP_LISTSIZE is set in mask, the value is of sense.

	new_msg	A flag which declares the returned listing of message contains unread messages or not. It could be one of the values below: BTSDK_TRUE – contains BTSDK_FALSE – do not contain Only if BTSDK_MAP_GMLP_NEWMSG is set in mask, the value is of sense.
	mse_time	A null-terminated string which returns current base value of time in server and the offset of UTC. The type is "YYYYMMDDTHHMMSS±hhmm". If server could not get current UTC time, it won't return the offset. So the type is "YYYYMMDDTHHMM". Only if BTSDK_MAP_GMLP_NEWMSG is set in mask, the value is of sense.

6.2.3.92 BtSdkMAPGetMsgParamStru

Definition	typedef struct _BtSdkMAPGetMsgParamStru { BTUINT8 msg_handle[BTSDK_MAP_MSGHDL_LEN]; BTUINT8 charset; BTBOOL attachment; BTUINT8 fraction_req; BTUINT8 fraction_deliver; } BtSdkMAPGetMsgParamStru, *PBtSdkMAPGetMsgParamStru;	
Description	The structure BtSdkMAPGetMsgParamStru contains parameters for GetMessage request and its response	
Members	*msg_handle*	A null-terminated UTF-8 string which specifies message handle which points the message that will be got with 16 hexadecimal digits.
	charset	The encoder mode of message that is returned by server. It could be one of the values below: BTSDK_MAP_CHARSET_NATIVE – return integrated SMS PDU in intrinsic encoder mode. The MSE does not do any change to the type. BTSDK_MAP_CHARSET_UTF8 – the MSE only transfer the textual content of message, and change the type to UTF-8 before sending. The message that the type is email or MMS can only adopt BTSDK_MAP_CHARSET_UTF8. If you set the two type of message to BTSDK_MAP_CHARSET_NATIVE, the MSE will reject the request immediately. The message that the type is SMS can adopt one of the two values. If message doesn' t contain any textual content, the MSE will reject the request that charset equals BTSDK_MAP_CHARSET_UTF8 immediately.
	attachment	A flag which specifies attachment should be contained or not in a returned message. It could be one of the values below: BTSDK_TRUE – need to contain attachment(if there is any) BTSDK_FALSE – do not need to contain attachment

	fraction_req	A flag which points whether to use multi requests to get email which is received in separate part. It could be one of the values below: BTSDK_MAP_FRACT_NONE – get integrated email one time. It means MSE will be responsibility packaging all separated part. BTSDK_MAP_FRACT_REQFIRST – get the first separated part of message BTSDK_MAP_FRACT_REQNEXT – get the next separated part of message Fraction_deliver is used to decide whether it is the last separated part of message or not
	fraction_deliver	A flag which decides whether the received part of message is the last separated part or not. It could be one of the values below: BTSDK_MAP_FRACT_RSPMORE – there is still more separated part of message should be received BTSDK_MAP_FRACT_RSPLAST – this is the last separated part of message. And it is also suitable for the situation that getting integrated email one time.

6.2.3.93 BtSdkMAPPushMsgParamStru

Definition	typedef struct _BtSdkMAPPushMsgParamStru { BTUINT8 folder[BTSDK_MAP_FOLDER_LEN]; BTBOOL save_copy; BTBOOL retry; BTUINT8 charset; BTUINT8 msg_handle[BTSDK_MAP_MSGHDL_LEN]; } BtSdkMAPPushMsgParamStru, *PBtSdkMAPPushMsgParamStru;	
Description	The structure BtSdkMAPPushMsgParamStru contains parameters for PushMessage request and its response	
Members	*folder*	A null-terminated UTF-8 string which specifies the directory (not include path) of this message which saved in MSE. If folder equals to empty character array or NULL, current directory will be saved as default.
	save_copy	A flag which decides whether to save copy under directory of "have sent" in MSE after message has been send to network successfully. It could be one of the values below: BTSDK_TRUE – save BTSDK_FALSE – not save
	retry	A flag which decides whether to retry to send message in MSE after message hasn't been sent because network could not be accessed. It could be one of the values below: BTSDK_TRUE – retry to send BTSDK_FALSE – do not retry to send
	charset	A flag which decides whether to encode the message again before the message has been sent. It could be one of the values below: BTSDK_MAP_CHARSET_NATIVE – send the message under the original encoding format. It is only used under the situation when message contains SMS PDU that can be sent directly. BTSDK_MAP_CHARSET_UTF8 – the data which is under the UTF-8 encoding format will be transformed to the prescribed encoding format before sending to the network, for example, email and MMS
	msg_handle	A null-terminated UTF-8 string which specifies message handle that MSE appoints for this message with 16 hexadecimal digits.

6.2.3.94 BtSdkMAPFolderObjStru

Definition	typedef struct _BtSdkMAPFolderObjStru { BTUINT32 size; BTUINT8 name[BTSDK_MAP_FOLDER_LEN]; BTUINT8 create_time[BTSDK_MAP_TIME_LEN]; BTUINT8 access_time[BTSDK_MAP_TIME_LEN]; BTUINT8 modify_time[BTSDK_MAP_TIME_LEN]; } BtSdkMAPFolderObjStru, *PBtSdkMAPFolderObjStru;	
Description	The structure BtSdkMAPFolderObjStru contains the attributes of directory.	
Members	*size*	The size of the directory. It is only an estimated value.
	name	A null-terminated UTF-8 string which specifies the name of the directory(not include path)
	create_time	A null-terminated UTF-8 string which specifies the time that the directory is established. The type of time is "YYYYMMDDTHHMMSS", or "YYYYMMDDTHHMMSSZ" under UTC type of time.
	access_time	A null-terminated UTF-8 string which specifies the time that the directory has been accessed the last time. The type of time is "YYYYMMDDTHHMMSS", or "YYYYMMDDTHHMMSSZ" under UTC type of time.
	modify_time	A null-terminated UTF-8 string which specifies the time that the directory has been modified the last time. The type of time is "YYYYMMDDTHHMMSS", or "YYYYMMDDTHHMMSSZ" under UTC type of time.

6.2.3.95 BtSdkMAPMsgObjStru

Definition	typedef struct _BtSdkMAPMsgObjStru { BTUINT8 msg_handle[BTSDK_MAP_MSGHDL_LEN]; BTUINT32 mask; BTUINT32 msg_size; BTUINT32 attachment_size; BTUINT8 subject[BTSDK_MAP_SUBJECT_LEN]; BTUINT8 sender_name[BTSDK_MAP_USERNAME_LEN]; BTUINT8 sender_addr[BTSDK_MAP_ADDR_LEN]; BTUINT8 replyto_addr[BTSDK_MAP_ADDR_LEN]; BTUINT8 recipient_name[BTSDK_MAP_USERNAME_LEN]; BTUINT8 recipient_addr[BTSDK_MAP_ADDR_LEN]; BTUINT8 msg_type[BTSDK_MAP_MSGTYPE_LEN]; BTUINT8 date_time[BTSDK_MAP_TIME_LEN]; BTUINT8 reception_status; BTBOOL text; BTBOOL read; BTBOOL sent; BTBOOL protect; BTBOOL priority; } BtSdkMAPMsgObjStru, *PBtSdkMAPMsgObjStru;	
Description	The structure BtSdkMAPMsgObjStru contains the attributes of messages.	
Members	*msg_handle*	A null-terminated UTF-8 string which specifies message handle that MES appoints for this message with 16 hexadecimal digits.
	mask	A set of flags which specify the validity of the member. It could be values or operations below: BTSDK_MAP_MP_SIZE – set the value of msg_size BTSDK_MAP_MP_TEXT – set the value of text BTSDK_MAP_MP_ATTACHSIZE – set the value of attachment_size BTSDK_MAP_MP_SUBJECT – set the value of subject BTSDK_MAP_MP_SENDERNAME – set the value of sender_name BTSDK_MAP_MP_SENDERADDR – set the value of sender_addr BTSDK_MAP_MP_REPLY2ADDR – set the value of replyto_addr

		BTSDK_MAP_MP_RECIPIENTNAME – set the value of recipient_name BTSDK_MAP_MP_RECIPIENTADDR – set the value of recipient_addr BTSDK_MAP_MP_TYPE – set the value of msg_type BTSDK_MAP_MP_DATATIME – set the value of date_time BTSDK_MAP_MP_RECPSTATUS – set the value of reception_status BTSDK_MAP_MP_READ – set the value of read BTSDK_MAP_MP_SENT – set the value of sent BTSDK_MAP_MP_PROTECTED – set the value of protected BTSDK_MAP_MP_PRIORITY – set the value of priority If the specific flag is not set in mask, the corresponding parameter will not show in OBEX data package
	msg_size	The original size of message that has been received from network in bytes.
	attachment_size	The total length of attachment in bytes. 0 means there's no attachment.
	subject	The subject of the message
	sender_name	The name of sender. The length should not be over 257 bytes including 0 to indicate the end.
	sender_addr	The address of sender. It could be email address or telephone number. The length should not be over 257 bytes including 0 to indicate the end.
	replyto_addr	The email address that the sender has written in reply-to field. It is only suitable to the message which the type of email. The length should not be over 257 bytes including 0 to indicate the end.
	recipient_name	The name of recipient. The length should not be over 257 bytes including 0 to indicate the end.
	recipient_addr	The address of recipient. It could be a email address or multi email addresses (with ", " to separate them), or telephone number. The length should not be over 257 bytes including 0 to indicate the end.
	msg_type	The type of message. It could be one of the character arrays below: "EMAIL" –e-mail，RFC 2822 or MIME "SMS_GSM" – GSM "SMS_CDMA" – CDMA "MMS" – 3GPP MMS Other character arrays – the message type that current MAP version doesn't support

	date_time	A null-terminated UTF-8 string which specifies the time that message has been sent or the time message has been received. It depends on MSE. The type of time is "YYYYMMDDTHHMMSS"
	reception_status	The status of receiving message. It could be one of the values below: BTSDK_MAP_MSG_RCVST_COMPLETE – MSE has received complete message BTSDK_MAP_MSG_RCVST_FRACTION – MSE only received part of the message BTSDK_MAP_MSG_NOTIFICATION – MSE gets the information that there is a new message got. It means the content of the message has not been received yet.
	text	A flag to present whether there is any textual content in the message or not BTSDK_TRUE – contain textual content BTSDK_FALSE – do not contain textual content
	read	A flag to present whether the message has been read or not BTSDK_TRUE – has been read BTSDK_FALSE – has not been read yet
	sent	A flag to present whether the message has been sent to the recipient yet or not BTSDK_TRUE – has been sent BTSDK_FALSE – has not been sent yet
	protect	A flag to present whether the message has been protected under the method of DRM or not BTSDK_TRUE – yes BTSDK_FALSE – no
	priority	A flag to present whether the message is of high priority level or not BTSDK_TRUE – yes BTSDK_FALSE – no

6.2.3.96 BtSdkMAPMsgFilterStru

Definition	typedef struct _BtSdkMAPMsgFilterStru { BTUINT32 mask; BTUINT32 param_mask; BTUINT8 filter_period_begin[BTSDK_MAP_TIME_LEN]; BTUINT8 filter_period_end[BTSDK_MAP_TIME_LEN]; BTUINT8 filter_originator[BTSDK_MAP_USERNAME_LEN]; BTUINT8 filter_recipient[BTSDK_MAP_USERNAME_LEN]; BTUINT8 filter_msg_type; BTUINT8 filter_read_status; BTUINT8 filter_priority; BTUINT8 subject_length; } BtSdkMAPMsgFilterStru, *PBtSdkMAPMsgFilterStru;	
Description	The structure BtSdkMAPMsgFilterStru contains the parameters from the GetMessageListing request.	
Members	*mask*	A set of flags which specify the validity of the member. It could be values below: BTSDK_MAP_GMLP_PARAMMASK – set the value of param_mask BTSDK_MAP_GMLP_PERIODBEGIN – set the value of filter_period_begin BTSDK_MAP_GMLP_PERIODEND – set the value of filter_period_end BTSDK_MAP_GMLP_ORIGINATOR – set the value of filter_originator BTSDK_MAP_GMLP_RECIPIENT – set the value of filter_recipient BTSDK_MAP_GMLP_MSGTYPE – set the value of filter_msg_type BTSDK_MAP_GMLP_READSTATUS – set the value of filter_read_status BTSDK_MAP_GMLP_PRIORITY – set the value of filter_priority BTSDK_MAP_GMLP_SUBJECTLENTH – set the value of subject_length If the specific flag is not set in mask, the corresponding parameter value shall be ignored.
	param_mask	A set of flags which specify the parameter values to be retrieved for the specified message object. It could be values below:

		BTSDK_MAP_MP_SUBJECT – subject parameter BTSDK_MAP_MP_DATATIME – datatime parameter BTSDK_MAP_MP_SENDERNAME – sender_name parameter BTSDK_MAP_MP_SENDERADDR – sender_addressing parameter BTSDK_MAP_MP_RECIPIENTNAME – recipient_name parameter BTSDK_MAP_MP_RECIPIENTADDR – recipient_addressing parameter BTSDK_MAP_MP_TYPE– type parameter BTSDK_MAP_MP_SIZE – size parameter BTSDK_MAP_MP_RECPSTATUS – reception_status parameter BTSDK_MAP_MP_TEXT– text parameter BTSDK_MAP_MP_ATTACHSIZE – attachment_size parameter BTSDK_MAP_MP_PRIORITY – priority parameter BTSDK_MAP_MP_READ – read parameter BTSDK_MAP_MP_SENT – sent parameter BTSDK_MAP_MP_PROTECTED – protected parameter BTSDK_MAP_MP_REPLY2ADDR– replyto_addressing parameter
	`filter_period_begin`	A null-terminated string which specifies the begin time of the time interval when the specific message returns. The type of time is "YYYYMMDDTHHMMSS" If BTSDK_MAP_GMLP_PERIODBEGIN is not set in mask or filter_period_begin equals to empty character array, the default return is the message that has been received before the time that filter_period_end points (not include the time point). If neither filter_period_begin nor filter_period_end is set, then return the message that meets all the other conditions. If the time that filter_period_end points is earlier than the time that filter_period_begin points, don't return anything.
	filter_period_end	A null-terminated string which specifies the end time of the time interval when the specific message returns. The type of time is "YYYYMMDDTHHMMSS" If BTSDK_MAP_GMLP_PERIODEND is not set in mask or filter_period_end equals to empty character array, the default return is the message that has been received after the time that filter_period_begin points (include the time point). If neither filter_period_begin nor filter_period_end is set, then return the message that meets all the other conditions. If the time that filter_period_end points is earlier than the time that filter_period_begin points, don't return anything.

	filter_originator	A null-terminated UTF-8 string which specifies the conditions of filter originator that the returned message should meet. MSE uses the condition to match the attribute “N”, “TEL”, “EMAIL” of vCard, if they contain filter_originator, the match is successful. If BTSDK_MAP_GMLP_ORIGINATOR is not set or the value of filter_originator equals to empty character array, return messages of random filter originator.
	filter_recipient	A null-terminated UTF-8 string which specifies the conditions of filter recipient that the returned message should meet. MSE uses the condition to match the attribute “N”, “TEL”, “EMAIL” of vCard, if they contain filter_recipient, the match is successful. If BTSDK_MAP_GMLP_RECIPIENT is not set or the value of filter_recipient equals to empty character array, return random message of random filter recipient.
	filter_msg_type	The message type that should not be returned. It can be values below:. BTSDK_MAP_FILTEROUT_SMSGSM – omit GSM message BTSDK_MAP_FILTEROUT_SMSCDMA – omit CDMA message BTSDK_MAP_FILTEROUT_EMAIL – omit email BTSDK_MAP_FILTEROUT_MMS – omit 3GPP MMS If BTSDK_MAP_GMLP_MSGTYPE is not set in mask, return all type of messages as default.
	filter_read_status	The reading status that the returned message should meet. It could be one of the values below: BTSDK_MAP_MSG_FILTER_ST_READ – the message has been read BTSDK_MAP_MSG_FILTER_ST_UNREAD – the message hasn’t been read BTSDK_MAP_MSG_FILTER_ST_ALL – do not distinguish the reading status If BTSDK_MAP_GMLP_READSTATUS is not set in mask, BTSDK_MAP_MSG_FILTER_ST_ALL is set as default.
	filter_priority	The condition of priority level that returned message should meet. It could be one of the values below: BTSDK_MAP_MSG_FILTER_PRI_HIGH – high priority level message BTSDK_MAP_MSG_FILTER_PRI_NOHIGH – not high priority level message BTSDK_MAP_MSG_FILTER_PRI_ALL – do not distinguish priority level If BTSDK_MAP_GMLP_PRIORITY is not set in mask, BTSDK_MAP_MSG_FILTER_PRI_ALL is set as default.

	subject_length	The max length of subject of returned message. If BTSDK_MAP_GMLP_SUBJECTLENGTH is not set in mask, do not limit the length of subject of returned message.

6.2.3.97 BtSdkMAPMsgStatusStru

Definition	typedef struct _BtSdkMAPMsgStatusStru { BTUINT8 msg_handle[BTSDK_MAP_MSGHDL_LEN]; BTUINT8 status_indicator; BTUINT8 status_value; } BtSdkMAPMsgStatusStru, *PBtSdkMAPMsgStatusStru;	
Description	This structure contains parameters from SetMessageStatus request.	
Members	*msg_handle*	Handle of the message the status of which shall be modified. It is a null-terminated UTF-8 string with 16 hexadecimal digits.
	status_indicator	Indicates which status information is to be modified. It can be one of following values: BTSDK_MAP_MSG_READ_STATUS – Read status; BTSDK_MAP_MSG_DELETE_STATUS – Deleted status.
	status_value	Indicate the new value of the status indicator to be modified. It can be one of following values: BTSDK_MAP_MSG_STATUS_NO – Unread for the read status or Undeleted for the deleted status; BTSDK_MAP_MSG_STATUS_YES – Read for the read status or Deleted for the deleted status.

6.2.3.98 BtSdkMAPMSETimeStru

Definition	typedef struct _BtSdkMAPMSETimeStru { BTUINT8 mse_time[BTSDK_MAP_MSE_TIME_LEN]; } BtSdkMAPMSETimeStru, *PBtSdkMAPMSETimeStru;	
Description	This structure contains MSE time value.	
Members	*mse_time*	A null-terminated string which returns current base value of time in server and the offset of UTC. The type is "YYYYMMDDTHHMMSS ± hhmm". If server could not get current UTC time, it won't return the offset. So the type is "YYYYMMDDTHHMM".

6.2.3.99 BtSdkMAPMsgHandleStru

Definition	typedef struct _BtSdkMAPMsgHandleStru { BTUINT8 msg_handle[BTSDK_MAP_MSGHDL_LEN]; } BtSdkMAPMsgHandleStru, *PBtSdkMAPMsgHandleStru;	
Description	This structure contains message handle value.	
Members	*msg_handle*	Message handle. It is a null-terminated UTF-8 string with 16 hexadecimal digits.

6.3 API Functions

6.3.1 File Transfer Profile

The format of a path string depends on the target platform running the application. For example, the path string can be “C:\\Bluetooth” in the Windows PC OS, or “/usr/Bluetooth” in the Linux OS.

Currently, if not specified additionally in the release note, the path string and the file name parameters use the default code page of the target platform.

6.3.1.1 General

6.3.1.1.1 Btsdk_FTPRegisterStatusCallback4ThirdParty

Prototype	void Btsdk_FTPRegisterStatusCallback4ThirdParty (BTCONNHDL conn_hdl, Btsdk_FTP_STATUS_INFO_CB* func);	
Description	The **Btsdk_FTPRegisterStatusCallback4ThirdParty** function registers an application-defined callback function used to deal with FTP tranfer file status information.	
Parameters	*conn_hdl*	[in] Handle to the FTP connection. For a FTP client connection, this handle value is returned by a previous successful call to functions *Btsdk_Connect* or *Btsdk_ConnectEx*. For a FTP server connection, this handle value is returned by the BTSDK_CONNECTION_EVENT_IND callback function.
	func	[in] Pointer to the callback function of Btsdk_FTP_STATUS_INFO_CB type.
Return:		

Remarks

This function registers callback function of FTP transfer file status information for the specified FTP connection. Only one callback function of Btsdk_FTP_STATUS_INFO_CB type is allowed for the same *conn_hdl* value. That is, if the application calls *Btsdk_FTPRegisterStatusCallback* twice to register different callback functions for the same connection handle, the second callback function will replace the first one.

If *func* is NULL, the call to *Btsdk_FTPRegisterStatusCallback* will remove the callback for the specified connection handle.

6.3.1.1.2 Btsdk_FTP_STATUS_INFO_CB

Prototype	typedef void (Btsdk_FTP_STATUS_INFO_CB)(BTUINT8 first, BTUINT8 last, BTUINT8* filename, BTUINT32 filesize, BTUINT32 cursize);	
Description	The **Btsdk_FTP_STATUS_INFO_CB** function prototype is the prototype of application defined callback function used to deal with file transfer status.	
Parameters	*first*	[in] Flag specifies whether it is the first call to this function. Any none zero (TRUE) value means it is the fist call. Otherwise, it is a continuous call.
	last	[in] Flag specifies whether it is the last call to this function. Any none zero (TRUE) value means it is the last call. Otherwise, it is not a last call.
	filename	[in] Pointer to the buffer contains the file name. It is valid only when first flag is not zero.
	filesize	[in] Specifies full size of the file to be transferred in bytes, only valid when first flag is not zero.
	cursize	[in] Specifies current transferred size in bytes.
Return:		

Remarks

This callback function needs to be registered using *Btsdk_FTPRegisterStatusCallback4ThirdParty* function. It is always called when the device sends/receives an OBEX package over the specified FTP connection

6.3.1.2 FTP Server

6.3.1.2.1 Btsdk_FTPRegisterDealReceiveFileCB4ThirdParty

Prototype	Void Btsdk_FTPRegisterDealReceiveFileCB4ThirdParty (BTSDK_FTP_UIDealReceiveFile* func);	
Description	The **Btsdk_FTPRegisterDealReceiveFileCB4ThirdParty** function registers an application-defined callback function used to process file transferring mode selection requests from the remote FTP client.	
Parameters	*func*	[in] Pointer to the callback function of BTSDK_FTP_UIDealReceiveFile type.
Return:		

Remarks

If the application wants to intervene in the file transfer procedure, e.g. to allow the user to determine whether to accept the file uploading request, it shall register a callback function after the local FTP service is enabled.

6.3.1.2.2 BTSDK_FTP_UIDealReceiveFile

Prototype	typedef BTBOOL (BTSDK_FTP_UIDealReceiveFile)(PBtSdkFileTransferReqStru pFileInfo);	
Description	The **BTSDK_FTP_UIDealReceiveFile** function prototype is the prototype of application defined callback function used to deal with file transfering requests from the remote FTP client.	
Parameters	*pFileInfo*	[in/out] Pointer to a BtSdkFileTransferReqStru structure specifies the information of the file transfer request.
Return:	If the function succeeds, the return value is TRUE. If the function fails, the return value is an error code listed in FALSE.	

Remarks

On input, if *pFileInfo->flag* is set to BTSDK_ER_CONTINUE, following operation is allowed:

(1) If the application wants to save the file using a different name, copy the new file name to *pFileInfo->file_name*.

(2) If the application wants to reject the file upload or delete request, change the *pFileInfo->flag* to one of OBEX error code except for BTSDK_ER_CONTINUE and BTSDK_ER_SUCCESS.

(3) If the application allows saving the file, just keep *pFileInfo->flag* unchanged.

6.3.1.3 FTP Client

6.3.1.3.1 Btsdk_FTPBrowseFolder

Prototype	BTINT32 Btsdk_FTPBrowseFolder (BTCONNHDL conn_hdl, BTUINT8 * szPath, BTSDK_FTP_UIShowBrowseFile* pShowFunc, BTUINT8 op_type);	
Description	The **Btsdk_FTPBrowseFolder** function browses the remote device folder.	
Parameters	*conn_hdl*	[in] Handle to the FTP connection.
	szPath	[in] Specifies the remote path to be browsed. A NULL pointer is used to specify the root directory.
	pShowFunc	[in] Pointer to the callback function of BTSDK_FTP_UIShowBrowseFile type.
	op_type	[in] Specifies the operation type.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code.	

The ***op_type*** member can be one of these values.

Value	Description
FTP_OP_REFRESH	Refresh the current directory. The *szPath* shall contain the name of the current directory.
FTP_OP_UPDIR	Up one level directory. The *szPath* is ignored.
FTP_OP_NEXT	Change the current directory to *szPath* and show the content of the directory. The *szPath* shall be the name of a sub-folder of the current directory.

Remarks

Before calling *Btsdk_FTPBrowseFolder*, a FTP connection between local device and the target device must be created first.

The *Btsdk_FTPBrowseFolder* function will go through the specified folder and report information of each file or sub-folder to the application through the callback function *pShowFunc*.

6.3.1.3.2 BTSDK_FTP_UIShowBrowseFile

Prototype	typedef void (BTSDK_FTP_UIShowBrowseFile) (BTUINT8* SYS_FIND_DATA);	
Description	The **BTSDK_FTP_UIShowBrowseFile** function prototype is the prototype of application defined callback function used to show file or folder information on the remote device.	
Parameters	*SYS_FIND_DATA*	[in] Pointer to an OS dependent structure describes the file found. The application should use the *Btsdk_FreeMemory* function to free the buffer pointed to by the *SYS_FIND_DATA* when it is no longer needed
Return:		

Remarks

Refers to the porting guide for detail information of the structure type of SYS_FIND_DATA

Currently, the SYS_FIND_DATA shall be converted to a pointer of *WIN32_FIND_DATA* type if the application runs in the Windows OS (98/2000/XP/CE).

6.3.1.3.3 Btsdk_FTPSetRmtDir

Prototype	BTINT32 Btsdk_FTPSetRmtDir (BTCONNHDL conn_hdl, BTUINT8 * szDir);	
Description	The **Btsdk_FTPSetRmtDir** function sets the current directory of the remote device.	
Parameters	*conn_hdl*	[in] Handle to the FTP connection.
	szDir	[in] Pointer to a buffer that contains the current directory to be set. It must be a relative path start with '\', which means the root directory, e.g. "\dir1\dir2". If szDir is NULL, root directory will be set. The path size must be smaller than BTSDK_PATH_MAXLENGTH.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code.	

Remarks

Before calling *Btsdk_FTPSetRmtDir*, a FTP connection between local device and the specified remote device must be created first.

After calling this function successfully, the application can call *Btsdk_FTPGetRmtDir* to get the current directory, call *Btsdk_FTPBrowseFolder* to browse the contents or call *Btsdk_FTPBackDir* to go up one level directory.

6.3.1.3.4 Btsdk_FTPGetRmtDir

Prototype	BTINT32 Btsdk_FTPGetRmtDir (BTCONNHDL conn_hdl, BTUINT8 * szDir);	
Description	The **Btsdk_FTPGetRmtDir** function gets the current directory of the remote device.	
Parameters	*conn_hdl*	[in] Handle to the FTP connection.
	szDir	[out] Pointer to a buffer used to receive the current directory. The size of this buffer shall be larger than BTSDK_PATH_MAXLENGTH in bytes.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code.	

Remarks

Before calling *Btsdk_FTPGetRmtDir*, a FTP connection between local device and the specified remote device must be created first.

The application can call *Btsdk_FTPSetRmtDir* to set the current directory of the remote device first. If the application does not call *Btsdk_FTPSetRmtDir* before, calling *Btsdk_FTPGetRmtDir* may get the root directory of the remote device.

After calling this function, the application can call *Btsdk_FTPBrowseFolder* to browse the contents of the current directory on the remote device.

6.3.1.3.5 Btsdk_FTPCreateDir

Prototype	BTINT32 Btsdk_FTPCreateDir (BTCONNHDL conn_hdl, BTUINT8 * szDir);	
Description	The **Btsdk_FTPCreateDir** function creates a new folder on the remote FTP server.	
Parameters	*conn_hdl*	[in] Handle to the FTP connection.
	szDir	[in] Pointer to a buffer contains the name of the new folder to be created.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code.	

Remarks

Before calling *Btsdk_FTPCreateDir*, a FTP connection between local device and the specified remote device must be created first.

After calling this function successfully, the application can call *Btsdk_FTPDeleteDir* to delete the directory or call *Btsdk_FTPSetRmtDir* to set it as the current directory.

6.3.1.3.6 Btsdk_FTPDeleteDir

Prototype	BTINT32 Btsdk_FTPDeleteDir (BTCONNHDL conn_hdl, BTUINT8 * szDir);	
Description	The **Btsdk_FTPDeleteDir** function deletes a folder on the remote FTP server.	
Parameters	*conn_hdl*	[in] Handle to the FTP connection.
	szDir	[in] Pointer to a buffer contains the name of the folder to be deleted.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code.	

Remarks

Before calling *Btsdk_FTPDeleteDir*, a FTP connection between local device and the specified remote device must be created first.

6.3.1.3.7 Btsdk_FTPDeleteFile

Prototype	BTINT32 Btsdk_FTPDeleteFile (BTCONNHDL conn_hdl, BTUINT8 * szFile);	
Description	The **Btsdk_FTPDeleteFile** function deletes a file on the remote FTP server.	
Parameters	*conn_hdl*	[in] Handle to the FTP connection.
	szFile	[in] Pointer to a buffer contains the name of the file to be deleted.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code.	

Remarks

Before calling *Btsdk_FTPDeleteFile*, a FTP connection between local device and the specified remote device must be created first.

6.3.1.3.8 Btsdk_FTPCancelTransfer

Prototype	BTINT32 Btsdk_FTPCancelTransfer (BTCONNHDL conn_hdl,);	
Description	The **Btsdk_FTPCancelTransfer** function terminates the file transferring procedure.	
Parameters	*conn_hdl*	[in] Handle to the FTP connection.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code.	

Remarks

This function only terminates the ongoing file transfer procedures over the specified connection. It DOES NOT release the specified connection.

6.3.1.3.9 Btsdk_FTPPutDir

Prototype	BTINT32 Btsdk_FTPPutDir (BTCONNHDL conn_hdl, BTUINT8 * loc_dir, BTUINT8* new_dir);	
Description	The **Btsdk_FTPPutDir** function uploads all contents under the specified directory to the remote FTP server.	
Parameters	*conn_hdl*	[in] Handle to the FTP connection.
	loc_dir	[in] Pointer to a buffer contains the full path of the local directory to be uploaded. The path size must be smaller than BTSDK_PATH_MAXLENGTH.
	new_dir	[in] Pointer to a buffer contains the name of the destination folder on the remote FTP server.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code.	

Remarks

Before calling *Btsdk_FTPPutDir*, a FTP connection between local device and the specified remote device must be created first.

The application can call *Btsdk_FTPCancelTransfer* function to terminate the transfer procedure.

6.3.1.3.10 Btsdk_FTPPutFile

Prototype	BTINT32 Btsdk_FTPPutFile (BTCONNHDL conn_hdl, BTUINT8 * loc_file, BTUINT8* new_file);	
Description	The **Btsdk_FTPPutFile** function uploads all contents under the specified directory to the remote FTP server.	
Parameters	*conn_hdl*	[in] Handle to the FTP connection.
	loc_file	[in] Pointer to a buffer contains the full path of the local file to be uploaded. The path size must be smaller than BTSDK_PATH_MAXLENGTH.
	new_file	[in] Pointer to a buffer contains the name of the destination file on the remote FTP server.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code.	

Remarks

Before calling *Btsdk_FTPPutFile*, a FTP connection between local device and the specified remote device must be created first.

The application can call *Btsdk_FTPCancelTransfer* function to terminate the transfer procedure.

6.3.1.3.11 Btsdk_FTPGetDir

Prototype	BTINT32 Btsdk_FTPGetDir (BTCONNHDL conn_hdl, BTUINT8 * rmt_dir, BTUINT8* new_dir);	
Description	The **Btsdk_FTPGetDir** function downloads all contents under the specified directory from the remote FTP server.	
Parameters	*conn_hdl*	[in] Handle to the FTP connection.
	rmt_dir	[in] Pointer to a buffer contains the name of the source folder on the remote FTP server.
	new_dir	[in] Pointer to a buffer contains the full path of the local directory to receive the downloaded contents. The path size must be smaller than BTSDK_PATH_MAXLENGTH.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code.	

Remarks

Before calling *Btsdk_FTPGetDir*, a FTP connection between local device and the specified remote device must be created first.

The application can call *Btsdk_FTPCancelTransfer* function to terminate the transfer procedure.

6.3.1.3.12 Btsdk_FTPGetFile

Prototype	BTINT32 Btsdk_FTPGetFile (BTCONNHDL conn_hdl, BTUINT8 * rmt_file, BTUINT8* new_file);	
Description	The **Btsdk_FTPGetFile** function downloads a file from the remote FTP server.	
Parameters	*conn_hdl*	[in] Handle to the FTP connection.
	rmt_file	[in] Pointer to a buffer contains the name of the source file on the remote FTP server.
	new_file	[in] Pointer to a buffer contains the full path of the local file to store the downloaded content. The path size must be smaller than BTSDK_PATH_MAXLENGTH.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code.	

Remarks

Before calling *Btsdk_FTPGetFile*, a FTP connection between local device and the specified remote device must be created first.

The application can call *Btsdk_FTPCancelTransfer* function to terminate the transfer procedure.

6.3.1.3.13 Btsdk_FTPBackDir

Prototype	BTINT32 Btsdk_FTPBackDir (BTCONNHDL conn_hdl,);	
Description	The **Btsdk_FTPBackDir** function changes the current directory on the remote FTP server to its parent directory.	
Parameters	*conn_hdl*	[in] Handle to the FTP connection.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code.	

Remarks

Before calling *Btsdk_FTPBackDir*, a FTP connection between local device and the specified remote device must be created first.

The application can call this function to go up one step of the remote directory after calling *Btsdk_FTPSetRmtDir* successfully.

6.3.2 Object Push Profile

The format of a path string depends on the target platform running the application. For example, the path string can be “C:\\Bluetooth” in the Windows PC OS, or “/usr/Bluetooth” in the Linux OS.

Currently, if not specified additionally in the release note, the path string and the file name parameters use the default code page of the target platform.

6.3.2.1 General

6.3.2.1.1 Btsdk_OPPRegisterStatusCallback4ThirdParty

Prototype	void Btsdk_OPPRegisterStatusCallback4ThirdParty (BTCONNHDL conn_hdl, Btsdk_OPP_STATUS_INFO_CB* func);	
Description	The **Btsdk_OPPRegisterStatusCallback4ThirdParty** function registers an application-defined callback function used to deal with FTP tranferring file status information.	
Parameters	*conn_hdl*	[in] Handle to the OPP connection. For an OPP client connection, this handle value is returned by a previous successful call to functions *Btsdk_Connect* or *Btsdk_ConnectEx*. For an OPP server connection, this handle value is returned by the BTSDK_CONNECTION_EVENT_IND callback function.
	func	[in] Pointer to the callback function of Btsdk_OPP_STATUS_INFO_CB type.
Return:		

Remarks

This function registers callback function of OPP transfer file status information for the specified OPP connection. Only one callback function of Btsdk_OPP_STATUS_INFO_CB type is allowed for the same *conn_hdl* value. That is, if the application calls *Btsdk_OPPRegisterStatusCallback4ThirdParty* twice to register different callback functions for the same connection handle, the second callback function will replace the first one.

If *func* is NULL, the call to *Btsdk_OPPRegisterStatusCallback4ThirdParty* will remove the callback for the specified connection handle.

6.3.2.1.2 Btsdk_OPP_STATUS_INFO_CB

Prototype	typedef void (Btsdk_OPP_STATUS_INFO_CB)(BTUINT8 first, BTUINT8 last, BTUINT8* filename, BTUINT32 filesize, BTUINT32 cursize);	
Description	The **Btsdk_FTP_STATUS_INFO_CB** function prototype is the prototype of application defined callback function used to deal with file transfer status.	
Parameters	*first*	[in] Flag specifies whether it is the first call to this function. Any none zero (TRUE) value means it is the fist call. Otherwise, it is a continuous call.
	last	[in] Flag specifies whether it is the last call to this function. Any none zero (TRUE) value means it is the last call. Otherwise, it is not a last call.
	filename	[in] Pointer to the buffer contains the file name. It is valid only when first flag is not zero.
	filesize	[in] Specifies full size of the file to be transferred in bytes, only valid when first flag is not zero.
	cursize	[in] Specifies current transferred size in bytes.
Return:		

Remarks

This callback function needs to be registered using *Btsdk_OPPRegisterStatusCallback* function. It is always called when the device sends/receives an OBEX package over the specified OPP connection.

6.3.2.2 OPP Server

6.3.2.2.1 Btsdk_OPPRegisterDealReceiveFileCB4ThirdParty

Prototype	Void Btsdk_OPPRegisterDealReceiveFileCB4ThirdParty (BTSDK_OPP_UIDealReceiveFile* func);	
Description	The **Btsdk_OPPRegisterDealReceiveFileCB4ThirdParty** function registers an application-defined callback function used to process file transfer mode selection requests from the remote OPP client.	
Parameters	*func*	[in] Pointer to the callback function of BTSDK_OPP_UIDealReceiveFile type.
Return:		

Remarks

If the application wants to intervene in the file transfer procedure, e.g. to allow the user to determine whether to accept the file uploading request, it shall register a callback function after the local OPP service is enabled.

6.3.2.2.2 BTSDK_OPP_UIDealReceiveFile

Prototype	typedef BTBOOL (BTSDK_OPP_UIDealReceiveFile) (PBtSdkFileTransferReqStru pFileInfo);	
Description	The **BTSDK_OPP_UIDealReceiveFile** function prototype is the prototype of application defined callback function used to deal with file transfer requests from the remote OPP client.	
Parameters	*pFileInfo*	[in/out] Pointer to a BtSdkFileTransferReqStru structure specifies the information of the file transfer request.
Return:	If the function succeeds, the return value is TRUE. If the function fails, the return value is an error code listed in FALSE.	

Remarks

On input, if *pFileInfo->flag* is set to BTSDK_ER_CONTINUE, following operation is allowed:

(4) If the application wants to save the file using a different name, copy the new file name to *pFileInfo->file_name*.

(5) If the application wants to reject the file upload request, change the *pFileInfo->flag* to one of OBEX error code except for BTSDK_ER_CONTINUE and BTSDK_ER_SUCCESS.

(6) If the application allows saving the file, just keep *pFileInfo->flag* unchanged.

6.3.2.3 OPP Client

6.3.2.3.1 Btsdk_OPPCancelTransfer

Prototype	BTINT32 Btsdk_OPPCancelTransfer (BTCONNHDL conn_hdl,);	
Description	The **Btsdk_OPPCancelTransfer** function terminates the file transfer procedure.	
Parameters	*conn_hdl*	[in] Handle to the OPP connection.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code.	

Remarks

This function only terminates the ongoing file transfer procedures over the specified connection. It DOES NOT release the specified connection.

6.3.2.3.2 Btsdk_OPPPushObj

Prototype	BTINT32 Btsdk_OPPPushObj (BTCONNHDL conn_hdl, BTUINT8 * szPushFilePath);	
Description	The **Btsdk_OPPPushObj** function pushes an object to the remote OPP server. Currently, the object contents must be stored in a file.	
Parameters	*conn_hdl*	[in] Handle to the OPP connection.
	szPushFilePath	[in] Pointer to a buffer contains the full path of the local file containing the object contents to be pushed. The path size must be smaller than BTSDK_PATH_MAXLENGTH.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code.	

Remarks

Before calling *Btsdk_OPPPushObj*, an OPP connection between local device and the specified remote device must be created first.

The application can call *Btsdk_OPPCancelTransfer* function to terminate the transfer procedure.

6.3.2.3.3 Btsdk_OPPPullObj

Prototype	BTINT32 Btsdk_OPPPullObj (BTCONNHDL conn_hdl, BTUINT8 * szPushFilePath);	
Description	The **Btsdk_OPPPullObj** function pulls the owner's business card form the remote OPP server.	
Parameters	*conn_hdl*	[in] Handle to the OPP connection.
	szPushFilePath	[in] Pointer to a buffer contains the local path to store the business card file. The path size must be smaller than BTSDK_PATH_MAXLENGTH.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code.	

Remarks

Before calling *Btsdk_OPPPullObj*, a FTP connection between local device and the specified remote device must be created first.

Currently, the received business card file is always named as "remote.vcf".

The application can call *Btsdk_OPPCancelTransfer* function to terminate the transfer procedure.

6.3.2.3.4 Btsdk_OPPExchangeObj

Prototype	BTINT32 Btsdk_OPPExchangeObj (BTCONNHDL conn_hdl, BTUINT8 * szPushFilePath, BTUINT8 * szPullFilePath, BTINT32 * npushError, BTINT32 * npullError);	
Description	The **Btsdk_OPPExchangeObj** function exchanges business card with the remote OPP server.	
Parameters	*conn_hdl*	[in] Handle to the OPP connection.
	szPushFilePath	[in] Pointer to a buffer contains the full path of the local file containing the object contents to be pushed. The path size must be smaller than BTSDK_PATH_MAXLENGTH.
	szPullFilePath	[in] Pointer to a buffer contains the local path to store the business card file. The path size must be smaller than BTSDK_PATH_MAXLENGTH.
	nPushError	[out] Pointer to a buffer to receive the push operation result.
	nPullError	[out] Pointer to a buffer to receive the pull operation result.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code. Check *npushError and npullError result of push and pull operation separately.	

Remarks

Before calling *Btsdk_OPPExchangeObj*, an OPP connection between local device and the specified remote device must be created first.

Currently, the received business card file is always named as “remote.vcf”.

The application can call *Btsdk_OPPCancelTransfer* function to terminate the transfer procedure.

6.3.3 Personal Area Networking Profile

6.3.3.1 General

6.3.3.1.1 Btsdk_PAN_RegIndCbk4ThirdParty

Prototype	void Btsdk_PAN_RegIndCbk4ThirdParty (Btsdk_PAN_Event_Ind_Func *pfunc);	
Description	The **Btsdk_PAN_RegIndCbk4ThirdParty** function registers an application-defined callback function used to deal with PAN callback messages.	
Parameters	*pfunc*	[in] Pointer to the callback function of Btsdk_PAN_Event_Ind_Func type.
Return:		

Remarks

Only one callback function of Btsdk_PAN_Event_Ind_Func type is allowed at a time. That is, if the application calls *Btsdk_PAN_RegIndCbk* twice to register different callback functions, the second callback function will replace the first one.

If *func* is NULL, the call to *Btsdk_PAN_RegIndCbk* will remove the callback function information

6.3.3.1.2 Btsdk_PAN_Event_Ind_Func

Prototype	typedef void (Btsdk_PAN_Event_Ind_Func)(BTUINT16 event, BTUINT16 len, BTUINT8* param);	
Description	The **Btsdk_PAN_Event_Ind_Func** function prototype is the prototype of application defined callback function used to deal with PAN messages.	
Parameters	*event*	[in] Event identifier.
	len	[in] If *param* is not set to NULL, *len* specifies the size of the buffer pointed to by the *param* parameter in bytes. Otherwise, it is set to 0.
	param	[in] Event specific parameter.
Return:		

The ***event*** parameter can be one of these values,

Value	Description
BTSDK_PAN_EV_IP_CHANGE	The IP address of the Bluetooth network adapter is changed. The *param* parameter is a pointer to a 32bit integer contains the new IP address value.

6.3.4 Audio/Video Remote Control Profile

6.3.4.1 AVRCP Target (TG)

6.3.4.1.1 Btsdk_AVRCP_RegPassThrCmdCbk4ThirdParty

Prototype	void Btsdk_AVRCP_RegPassThrCmdCbk4ThirdParty (Btsdk_AVRCP_PassThr_Cmd_Func *pfunc);	
Description	The **Btsdk_AVRCP_RegPassThrCmdCbk4ThirdParty** function registers an application-defined callback function used to deal with PASS THROUTH command from the Controller.	
Parameters	*pfunc*	[in] Pointer to the callback function of Btsdk_AVRCP_PassThr_Cmd_Func type. If p*func* is NULL, BlueSoleil will remove the callback information registered before.
Return:		

Remarks

Only one callback function of Btsdk_AVRCP_PassThr_Cmd_Func type is allowed at a time. That is, if the application calls *Btsdk_AVRCP_RegPassThrCmdCbk* twice to register different callback functions, the second callback function will replace the first one.

6.3.4.1.2 Btsdk_AVRCP_PassThr_Cmd_Func

Prototype	typedef void (Btsdk_AVRCP_PassThr_Cmd_Func) (BTUINT8 op_id, BTUINT8 state_flag,);	
Description	The **Btsdk_AVRCP_PassThr_Cmd_Func** function prototype is the prototype of application defined callback function used to deal with PASS THROUTH command from the Controller.	
Parameters	*op_id*	[in] Operation identifier specifies the command.
	statte_flag	[in] Button status.
Return:		

The ***op_id*** parameter can be one of these values,

Value	Description
BTSDK_AVRCP_OPID_AVC_PANEL_POWER	Power operation.
BTSDK_AVRCP_OPID_AVC_PANEL_VOLUME_UP	Volume Up operation.
BTSDK_AVRCP_OPID_AVC_PANEL_VOLUME_DOWN	Volume Down operation.
BTSDK_AVRCP_OPID_AVC_PANEL_MUTE	Mute operation.
BTSDK_AVRCP_OPID_AVC_PANEL_PLAY	Play operation.
BTSDK_AVRCP_OPID_AVC_PANEL_STOP	Stop operation.
BTSDK_AVRCP_OPID_AVC_PANEL_PAUSE	Pause operation.
BTSDK_AVRCP_OPID_AVC_PANEL_RECORD	Record operation.
BTSDK_AVRCP_OPID_AVC_PANEL_REWIND	Rewind operation.
BTSDK_AVRCP_OPID_AVC_PANEL_FAST_FORWARD	Fast Forward operation.
BTSDK_AVRCP_OPID_AVC_PANEL_EJECT	Reject operation.
BTSDK_AVRCP_OPID_AVC_PANEL_FORWARD	Forward operation.
BTSDK_AVRCP_OPID_AVC_PANEL_BACKWARD	Backward operation.

The ***state_flag*** parameter can be one of these values,

Value	Description
BTSDK_AVRCP_BUTTON_STATE_PRESSED	Button is pressed down.
BTSDK_AVRCP_BUTTON_STATE_RELEASED	Button is released.

Remarks

All operation requests from the remote Controller are transferred to the application using this callback function.

6.3.4.1.3 Btsdk_AVRCP_RegIndCbk4ThirdParty

Prototype	void Btsdk_AVRCP_RegIndCbk4ThirdParty (Btsdk_AVRCP_Event_Ind_Func *pfunc);	
Description	The **Btsdk_AVRCP_RegIndCbk4ThirdParty** function is register client another callback function to Bssdk.dll. If one client call Btsdk_AVRCP_RegIndCbk and this function to register callback the same time, bssdk will call these two callback functions.	
Parameters	*pfunc*	[in] pointer to Btsdk_AVRCP_Event_Ind_Func.
Return:		

Remarks

Only one callback function of Btsdk_AVRCP_Event_Ind_Func type is allowed at a time. That is, if the application calls *Btsdk_AVRCP_RegIndCbk4ThirdParty* twice to register different callback functions, the second callback function will replace the first one.

Two events:
BTSDK_APP_EV_AVRCP_IND_CONN and BTSDK_APP_EV_AVRCP_IND_DISCONN need to be processed.For example, the application needs implement something to control player.

The application should free the param.

Example

```
Btsdk_AVRCP_RegIndCbk4ThirdParty (AVRCP_Event_CbkFunc);

void AVRCP_Event_CbkFunc(BTUINT8 event, BTUINT8 *param)
{
    switch (event)
    {
        case BTSDK_APP_EV_AVRCP_IND_CONN:
            /*prepare to control player */
            break;
        case BTSDK_APP_EV_AVRCP_IND_DISCONN:
            /*exit to control player*/
            break;
```

```
                default:
                        break;
            }
        if (NULL != param)
        {
                Btsdk_FreeMemory(param);
        }
}
```

6.3.4.1.4 Btsdk_AVRCP_TGRegCommandCbk

Prototype	void Btsdk_AVRCP_TGRegCommandCbk (Btsdk_AVRCP_TG_Command_Cbk_Func *pfunc);	
Description	The **Btsdk_AVRCP_TGRegCommandCbk** function registers an TG callback function used to deal with request of CT.	
Parameters	*pfunc*	[in] Pointer to the callback function of Btsdk_AVRCP_TG_Command_Cbk_Func. If p*func* is NULL, BlueSoleil will remove the callback information registered before.
Return:		

Remarks

Only one callback function of Btsdk_AVRCP_TG_Command_Cbk_Func type is allowed at a time. That is, if the application calls Btsdk_AVRCP_TGRegCommandCbk twice to register different callback functions, the second callback function will replace the first one.

6.3.4.1.5 Btsdk_AVRCP_TG_Command_Cbk_Func

Prototype	Typedef BTBOOL (Btsdk_AVRCP_TG_Command_Cbk_Func) (BTDEVHDL dev_hdl, BTUINT8 tl, BTUINT16 cmd_type, BTUINT8 *param);	
Description	The **Btsdk_AVRCP_TG_Command_Cbk_Func** function used to deal with events from CT resquested.	
Parameters	*dev_hdl*	[in] Handle to the remote device
	tl	[in] Transaction labeling.
	cmd_type	[in] Event specific type.
	param	[in] Event specific parameter.
Return:	If the function succeeds, the return value is BTSDK_TRUE. If the function fails, the return value is BTSDK_FALSE.	

The *cmd_type* parameter can be one of these values,

cmd_type	*Description*
BTSDK_APP_EV_AVRCP_GET_CAPABILITIES_IND	TG received GetCapabilities request from CT. TG should call Btsdk_AVRCP_GetCapabilitiesRsp function to response.
BTSDK_APP_EV_AVRCP_LIST_PLAYER_SETTING_ATTR_IND	TG received ListPlayerApplicationSettingAttributes request from CT. TG should call Btsdk_AVRCP_ListPlayerAppSetAttrRsp function to response.
BTSDK_APP_EV_AVRCP_LIST_PLAYER_SETTING_VALUES_IND	TG received ListPlayerApplicationSettingValues request from CT. TG should call Btsdk_AVRCP_ListPlayerAppSetValRsp function to response.
BTSDK_APP_EV_AVRCP_GET_CURRENTPLAYER_SETTING_VALUE_IND	TG received GetCurrentPlayerApplicationSettingValue request from CT. TG should call Btsdk_AVRCP_GetCurPlayerAppSetValRsp function to response.

BTSDK_APP_EV_AVRCP_SET_CURRENTPLAYER_SETTING_VALUE_IND	TG received SetCurrentPlayerApplicationSettingValue request from CT. TG should call Btsdk_AVRCP_SetCurPlayerAppSetValRsp function to response.
BTSDK_APP_EV_AVRCP_GET_PLAYER_SETTING_ATTR_TEXT_IND	TG received GetPlayerApplicationSettingAttributeText request from CT. TG should call Btsdk_AVRCP_GetPlayerAppSetAttrTxtRsp function to response.
BTSDK_APP_EV_AVRCP_GET_PLAYER_SETTING_VALUE_TEXT_IND	TG received GetPlayerApplicationSettingValueText request from CT. TG should call Btsdk_AVRCP_GetPlayerAppSetValTxtRsp function to response.
BTSDK_APP_EV_AVRCP_INFORM_CHARACTERSET_IND	TG received InformDisplayableCharacterSet request from CT. TG should call Btsdk_AVRCP_InformCharSetRsp function to response.
BTSDK_APP_EV_AVRCP_INFORM_BATTERYSTATUS_OF_CT_IND	TG received InformBatteryStatutsOfCT request from CT. TG should call Btsdk_AVRCP_InformBattStatusRsp function to response.
BTSDK_APP_EV_AVRCP_GET_ELEMENT_ATTR_IND	TG received GetElementAttributes request from CT. TG should call Btsdk_AVRCP_GetElementAttrRsp function to response.
BTSDK_APP_EV_AVRCP_GET_PLAY_STATUS_IND	TG received GetPlayStauts request from CT. TG should call Btsdk_AVRCP_GetPlayStatusRsp function to response.
BTSDK_APP_EV_AVRCP_SET_ABSOLUTE_VOLUME_IND	TG received SetAbsoluteVolume request from CT. TG should call Btsdk_AVRCP_SetAbsoluteVolRsp function to response.
BTSDK_APP_EV_AVRCP_SET_ADDRESSED_PLAYER_IND	TG received SetAddressedPlayer request from CT. TG should call Btsdk_AVRCP_SetAddressedPlayerRsp function to response.
BTSDK_APP_EV_AVRCP_SET_BROWSED_PLAYER_IND	TG received SetBrowsedPlayer request from CT. TG should call Btsdk_AVRCP_SetBrowsedPlayerRsp function to response.

BTSDK_APP_EV_AVRCP_GET_FOLDER_ITEMS_IND	TG received GetFolderItem request from CT. TG should call Btsdk_AVRCP_GetFolderItemsRsp function to response.
BTSDK_APP_EV_AVRCP_CHANGE_PATH_IND	TG received ChangePath request from CT. TG should call Btsdk_AVRCP_ChangePathRsp function to response.
BTSDK_APP_EV_AVRCP_GET_ITEM_ATTRIBUTES_IND	TG received GetItemAttributes request from CT. TG should call Btsdk_AVRCP_GetItemAttrRsp function to response.
BTSDK_APP_EV_AVRCP_PLAY_ITEM_IND	TG received PlayItem request from CT. TG should call Btsdk_AVRCP_PlayItemRsp function to response.
BTSDK_APP_EV_AVRCP_SEARCH_IND	TG received Search request from CT. TG should call Btsdk_AVRCP_SearchRsp function to response.
BTSDK_APP_EV_AVRCP_ADDTO_NOWPLAYING_IND	TG received AddToNowPlaying request from CT. TG should call Btsdk_AVRCP_AddToNowPlayingRsp function to response.
BTSDK_AVRCP_EVENT_PLAYBACK_STATUS_CHANGED	TG received the registration for Play back status changed notification request from CT. TG should call Btsdk_AVRCP_EventPlayStatusChanged function to response or when the play back status changed.
BTSDK_AVRCP_EVENT_TRACK_CHANGED	TG received the registration for track changed notification request from CT. TG should call Btsdk_AVRCP_EventTrackChanged function to response or when the track changed.
BTSDK_AVRCP_EVENT_TRACK_REACHED_END	TG received the registration for track reached the end notification request from CT. TG should call Btsdk_AVRCP_EventTrackReachEnd function to response or when the track reached end.
BTSDK_AVRCP_EVENT_TRACK_REACHED_START	TG received the registration for track reached the start notification from CT. TG should call Btsdk_AVRCP_EventTrackReachStart function to response or when the track reached the start.
BTSDK_AVRCP_EVENT_PLAYBACK_POS_CHANGED	TG received the registration for play back position changed notification request from CT.

	TG should call Btsdk_AVRCP_EventPlayPosChanged function to response or when the player's play back position changed.
BTSDK_AVRCP_EVENT_BATT_STATUS_CHANGED	TG received the registration for battery status changed notification request from CT. TG should call Btsdk_AVRCP_EventBattStatusChanged function to response or when the TG's battery status changed.
BTSDK_AVRCP_EVENT_SYSTEM_STATUS_CHANGED	TG received the registration for system status changed notification request from. TG should call Btsdk_AVRCP_EventSysStatusChanged function to response or when the TG's system status changed.
BTSDK_AVRCP_EVENT_PLAYER_APPLICATION_SETTING_CHANGED	TG received the registration for player application's setting changed notification request from CT. TG shoul call Btsdk_AVRCP_EventPlayerAppSetChanged function to response or when the TG's player application's setting changed.
BTSDK_AVRCP_EVENT_NOW_PLAYING_CONTENT_CHANGED	TG received the registration for "content of the NowPlaying folder for the Addressed player is changed" notification request from CT. TG should call Btsdk_AVRCP_EventNowPlayingContentChanged to response or when the content of the NowPlaying folder for the Addressed player is changed.
BTSDK_AVRCP_EVENT_AVAILABLE_PLAYERS_CHANGED	TG received the registration for "a new player becomes available to be addressed or a player ceases to be avaliable" notification request from CT. TG should call Btsdk_AVRCP_EventAvailablePlayerChanged function to response or when the event "a new player becomes available to be addressed or a player ceases to be available" occur.
BTSDK_AVRCP_EVENT_ADDRESSED_PLAYER_CHANGED	TG received the registration for addressed player changed notification request from CT. TG should call Btsdk_AVRCP_EventAddrPlayerChanged function to response or when the addressed player changed.

BTSDK_AVRCP_EVENT_UIDS_CHANGED	TG received the registration for UIDs changed notification request from CT. TG should call Btsdk_AVRCP_EventUIDSChanged function to respose or when the UIDs changed.
BTSDK_AVRCP_EVENT_VOLUME_CHANGED	TG received the registration for “volume changed locally on TG” notification request from CT. TG should call Btsdk_AVRCP_EventVolChanged function to response or when the volume changed on TG.

Remarks

6.3.4.1.6 Btsdk_AVRCP_PassThroughRspEx

Prototype	BTINT32 Btsdk_AVRCP_PassThroughRspEx (BTDEVHDL hdl, BTUINT8 tl, PBtSdkPassThroughStru param);	
Description	The **Btsdk_AVRCP_PassThroughRspEx** responds to the PassThrough command from CT	
Parameters	*hdl*	[in] Handle to the peer device.
	tl	[in] Transaction labeling. Get from the Command Callback.
	param	[in] Pointer to the BtSdkPassThroughStru structure just get from CT
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

if received BTSDK_APP_EV_AVRCP_PASSTHROUGH_IND, TG should call Btsdk_AVRCP_PassThroughRspEx function to response.

6.3.4.1.7 Btsdk_AVRCP_GetCapabilitiesRsp

Prototype	BTINT32 Btsdk_AVRCP_GetCapabilitiesRsp (BTDEVHDL hdl, BTUINT8 tl, PBtSdkGetCapabilitiesRspStru param);	
Description	The **Btsdk_AVRCP_GetCapabilitiesRsp** responds to the GetCapabilities command from CT.	
Parameters	*hdl*	[in] Handle to the peer device.
	tl	[in] Transaction labeling. Get from the Command Callback.
	param	[in] Pointer to the 错误！未找到引用源。 structure specifies the capabilities supported by TG.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

6.3.4.1.8 Btsdk_AVRCP_ListPlayerAppSetAttrRsp

Prototype	BTINT32 Btsdk_AVRCP_ListPlayerAppSetAttrRsp (BTDEVHDL hdl, BTUINT8 tl, PBtSdkListPlayerAppSetAttrRspStru param);	
Description	The **Btsdk_AVRCP_ListPlayerAppSetAttrRsp** responds to the ListPlayerApplicationSettingAttributes command from CT.	
Parameters	*hdl*	[in] Handle to the peer device.
	tl	[in] Transaction labeling. Get from the Command Callback.
	param	[in] Pointer to the 错误！未找到引用源。 structure specifies the supported player application setting attributes.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

6.3.4.1.9 Btsdk_AVRCP_ListPlayerAppSetValRsp

Prototype	BTINT32 Btsdk_AVRCP_ListPlayerAppSetValRsp (BTDEVHDL hdl, BTUINT8 tl, PBtSdkListPlayerAppSetValRspStru param);	
Description	The **Btsdk_AVRCP_ListPlayerAppSetValRsp** responds to the ListPlayerApplicationSettingValues command from CT.	
Parameters	*hdl*	[in] Handle to the peer device.
	tl	[in] Transaction labeling. Get from the Command Callback.
	param	[in] Pointer to the 错误！未找到引用源。 structure list the values of the requested player application setting attribute.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

6.3.4.1.10 Btsdk_AVRCP_GetCurPlayerAppSetValRsp

Prototype	BTINT32 Btsdk_AVRCP_GetCurPlayerAppSetValRsp (BTDEVHDL hdl, BTUINT8 tl, PBtSdkGetCurPlayerAppSetValRspStru param);	
Description	The **Btsdk_AVRCP_GetCurPlayerAppSetValRsp** responds to the GetCurrentPlayerApplicationSettingValue command from CT.	
Parameters	*hdl*	[in] Handle to the peer device.
	tl	[in] Transaction labeling. Get from the Command Callback.
	param	[in] Pointer to the **错误！未找到引用源。** structure provides the player application setting list of player application current setting values.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

6.3.4.1.11 Btsdk_AVRCP_SetCurPlayerAppSetValRsp

Prototype	BTINT32 Btsdk_AVRCP_SetCurPlayerAppSetValRsp (BTDEVHDL hdl, BTUINT8 tl,);	
Description	The **Btsdk_AVRCP_SetCurPlayerAppSetValRsp** responds to the SetCurrentPlayerAppSettingValue command from CT.	
Parameters	*hdl*	[in] Handle to the peer device.
	tl	[in] Transaction labeling. Get from the Command Callback.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

Setting of a value by CT does not implicitly mean that the setting will take effect on TG. The setting shall take effect after a play command from CT. If currently playing, it is up to the TG to decide when the setting shall take effect. There shall be an 错误！未找到引用源。 sent back if there are errors in attributes and/or value.

6.3.4.1.12 Btsdk_AVRCP_GetPlayerAppSetAttrTxtRsp

Prototype	BTINT32 Btsdk_AVRCP_GetPlayerAppSetAttrTxtRsp (BTDEVHDL hdl, BTUINT8 tl, PBtSdkGetPlayerAppSetAttrTxtRspStru param);	
Description	The **Btsdk_AVRCP_GetPlayerAppSetAttrTxtRsp** responds to the GetPlayerApplicationSettingAttributeText command from CT.	
Parameters	*hdl*	[in] Handle to the peer device.
	tl	[in] Transaction labeling. Get from the Command Callback.
	param	[in] Pointer to the 错误！未找到引用源。 structure provides player application setting attributes displayable text for the provided AttributeIDs.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

When responds the GetPlayerApplicationSettingAttributeText command, call this function first with subpacket_type set to BTSDK_AVRCP_PACKET_HEAD and id_num set to the total number of attributes, and so on for the other attributes.

6.3.4.1.13 Btsdk_AVRCP_GetPlayerAppSetValTxtRsp

Prototype	BTINT32 Btsdk_AVRCP_GetPlayerAppSetValTxtRsp (BTDEVHDL hdl, BTUINT8 tl, PBtSdkGetPlayerAppSetValTxtRspStru param);	
Description	The **Btsdk_AVRCP_GetPlayerAppSetValTxtRsp** responds to the GetPlayerApplicationSettingValueText command from CT.	
Parameters	*hdl*	[in] Handle to the peer device.
	tl	[in] Transaction labeling. Get from the Command Callback.
	param	[in] Pointer to the 错误！未找到引用源。 structure provides player application setting values displayable text.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

When responds to the GetPlayerApplicationSettingValueText command, call this function first with *subpacket_type* set to BTSDK_AVRCP_PACKET_HEAD and *id_num* set to the total number of attributes. Then, call it with *subpacket_type* set to BTSDK_AVRCP_SUBPACKET and *id_string* specify an attribute, and so on for the other attributes.

6.3.4.1.14 Btsdk_AVRCP_InformCharSetRsp

Prototype	BTINT32 Btsdk_AVRCP_InformCharSetRsp (BTDEVHDL hdl BTUINT8 tl);	
Description	The **Btsdk_AVRCP_InformCharSetRsp** responds to the InformCharacterSet command from CT.	
Parameters	*hdl*	[in] Handle to the peer device.
	tl	[in] Transaction labeling. Get from the Command Callback.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

When TG receives InformDisplayableCharacterSet Command, the TG can send a string in the character set that is specified in this command. If there is no character set which CT has. TG will send a string in UTF-8. By default TG shall send strings in UTF-8 if this command has not been sent by CT to TG.

6.3.4.1.15 Btsdk_AVRCP_InformBattStatusRsp

Prototype	BTINT32 Btsdk_AVRCP_InformBattStatusRsp (BTDEVHDL hdl BTUINT8 tl);	
Description	This function responds to the InformBatteryStatusRsp command from CT.	
Parameters	*hdl*	[in] Handle to the peer device.
	tl	[in] Transaction labeling. Get from the Command Callback.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

6.3.4.1.16 Btsdk_AVRCP_GetElementAttrRsp

Prototype	BTINT32 Btsdk_AVRCP_GetElementAttrRsp (BTDEVHDL hdl, BTUINT8 tl, PBtSdkGetElementAttrRspStru *param*);	
Description	The **Btsdk_AVRCP_GetElementAttrRsp** responds to the GetElementAttributes command from CT.	
Parameters	*hdl*	[in] Handle to the peer device.
	tl	[in] Transaction labeling. Get from the Command Callback.
	param	[in] Pointer to the 错误！未找到引用源。 structure specifies the attribute values.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

When responds to the GetElementAttributes command, call this function first with *subpacket_type* set to BTSDK_AVRCP_PACKET_HEAD and *id_num* set to the total number of attributes. Then, call it with *subpacket_type* set to BTSDK_AVRCP_SUBPACKET and *id_value* specify an attribute, and so on for the other attributes.

6.3.4.1.17 Btsdk_AVRCP_GetPlayStatusRsp

Prototype	BTINT32 Btsdk_AVRCP_GetPlayStatusRsp (BTDEVHDL hdl, BTUINT8 tl, PBtSdkPlayStatusRspStru param);	
Description	The **Btsdk_AVRCP_GetPlayStatusRsp** responds to the GetPlayStatus command from CT.	
Parameters	*hdl*	[in] Handle to the peer device.
	tl	[in] Transaction labeling. Get from the Command Callback.
	param	[in] Pointer to the 错误！未找到引用源。 structure specifies the status.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

6.3.4.1.18 Btsdk_AVRCP_SetAddressedPlayerRsp

Prototype	BTINT32 Btsdk_AVRCP_SetAddressedPlayerRsp (BTDEVHDL hdl, BTUINT8 tl, PBtSdkSetAddresedPlayerRspStru param);	
Description	The **Btsdk_AVRCP_SetAddressedPlayerRsp** responds to the SetAddressedPlayer command from CT.	
Parameters	*hdl*	[in] Handle to the peer device.
	tl	[in] Transaction labeling. Get from the Command Callback.
	param	[in] Pointer to the 错误！未找到引用源。 structure specifies the status.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

6.3.4.1.19 Btsdk_AVRCP_SetBrowsedPlayerRsp

Prototype	BTINT32 Btsdk_AVRCP_SetBrowsedPlayerRsp(BTDEVHDL hdl, BTUINT8 tl, PBtSdkSetBrowsedPlayerRspStru param);	
Description	The **Btsdk_AVRCP_SetBrowsedPlayerRsp** responds to the SetBrowsedPlayer command from CT. It contains the current browsed path of the player.	
Parameters	*hdl*	[in] Handle to the peer device.
	tl	[in] Transaction labeling. Get from the Command Callback.
	param	[in] Pointer to the 错误！未找到引用源。 structure specifies the status.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

Some players may support browsing only when set as the Addressed Player. This is shown in the player feature bitmask. If a SetBrowsedPlayer command is received by the TG for a Player Id which does not support browsing while not addressed it shall return the PlayerNotAddressed error in the status field of the response.

The response contains the current browsed path of the player. This is built up through a sequence of name/value pairs as illustrated in the example.

When responds to the SetBrowsedPlayer command, call this function first with *subpacket_type* set to BTSDK_AVRCP_PACKET_HEAD and *packet_head* specify the Attributes of the player and the folder depth of the current folder. Then, call it with *subpacket_type* set to BTSDK_AVRCP_SUBPACKET and *folder_item* specify an folder in the browsed path, and so on for the other folders.

Example:

This example shows a successful switch to a browsed player with the current directory DEF, which is the child of the folder BC, which itself is the child of the root folder A.

```
void AVRCP_Exp_GetElementAttrInd(BTDEVHDL dev_hdl, BTUINT8 tl,
PBtSdkSetBrowsedPlayerReqStru in)
{
    PBtSdkSetBrowsedPlayerRspStru prsp = NULL;
    UINT16 player_id = in->id;
    UINT16 k = 0;
    BTUINT16 len = 0
    UINT8 folder_depth = 3;
```

```
    /* Specifies the Attributes of the Browsed Player which is specified by the
SetBrowsedPlayer Command and the folder depth of the current folder. */
    size = 2 * sizeof(BTUINT32) + sizeof(BtSdkSetBrowsedPlayerRspHeadStru);
    prsp = (PBtSdkSetBrowsedPlayerRspStru)malloc(size);
    prsp->size = size;
    prsp->subpacket_type = BTSDK_AVRCP_PACKET_HEAD;
    prsp->packet_head.status = BTSDK_AVRCP_ERROR_SUCCESSFUL;
    prsp->packet_head.uid_counter = 0x1357;
    prsp->packet_head.items_num = 0x0005;
    prsp->packet_head.characterset_id = folder_depth;
    prsp->packet_head.folder_depth = folder_depth;
    Btsdk_AVRCP_SetBrowsedPlayerRsp(hdl, tl, prsp);
    free(prsp);

    /* Specifies the root folder. */
    len = strlen("A");
    size = 2 * sizeof(BTUINT32) + sizeof(BtSdkSetBrowsedPlayerRspItemStru) + (len
- 1) * sizeof(BTUINT8);
    prsp = (PBtSdkSetBrowsedPlayerRspStru)malloc(size);
    prsp->size = size;
    prsp->subpacket_type = BTSDK_AVRCP_SUBPACKET;
    prsp->folder_item.folder_name_len = len;
    memcpy(prsp->folder_item.folder_name, "A");
    Btsdk_AVRCP_SetBrowsedPlayerRsp(hdl, tl, prsp);
    free(prsp);

    /* Specifies the second level folder. */
    len = strlen("BC");
    size = 2 * sizeof(BTUINT32) + sizeof(BtSdkSetBrowsedPlayerRspItemStru) + (len
- 1) * sizeof(BTUINT8);
    prsp = (PBtSdkSetBrowsedPlayerRspStru)malloc(size);
    prsp->size = size;
    prsp->subpacket_type = BTSDK_AVRCP_SUBPACKET;
    prsp->folder_item.folder_name_len = len;
    memcpy(prsp->folder_item.folder_name, "BC");
    Btsdk_AVRCP_SetBrowsedPlayerRsp(hdl, tl, prsp);
    free(prsp);

    /* Specifies the current folder. */
    len = strlen("DEF");
    size = 2 * sizeof(BTUINT32) + sizeof(BtSdkSetBrowsedPlayerRspItemStru) + (len
- 1) * sizeof(BTUINT8);
    prsp = (PBtSdkSetBrowsedPlayerRspStru)malloc(size);
    prsp->size = size;
```

```
    prsp->subpacket_type = BTSDK_AVRCP_SUBPACKET;
    prsp->folder_item.folder_name_len = len;
    memcpy(prsp->folder_item.folder_name, "DEF");
    Btsdk_AVRCP_SetBrowsedPlayerRsp(hdl, tl, prsp);
    free(prsp);
}
```

6.3.4.1.20 Btsdk_AVRCP_ChangePathRsp

Prototype	BTINT32 Btsdk_AVRCP_ChangePathRsp (BTDEVHDL hdl, BTUINT8 tl, PBtSdkChangePathRspStru param);	
Description	The **Btsdk_AVRCP_ChangePathRsp** responds to the ChangePath command from CT.	
Parameters	*hdl*	[in] Handle to the peer device.
	tl	[in] Transaction labeling. Get from the Command Callback.
	param	[in] Pointer to the 错误！未找到引用源。 structure.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

6.3.4.1.21 Btsdk_AVRCP_GetFolderItemsRsp

Prototype	BTINT32 Btsdk_AVRCP_GetFolderItemsRsp (BTDEVHDL hdl, BTUINT8 tl, PBtSdkGetFolderItemRspStru param);	
Description	The **Btsdk_AVRCP_GetFolderItemsRsp** responds to the GetFolderItems command from CT.	
Parameters	*hdl*	[in] Handle to the peer device.
	tl	[in] Transaction labeling. Get from the Command Callback.
	param	[in] Pointer to the 错误！未找到引用源。 structure.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

The CT may specify a range of entries to be returned. This means that a CT which can only display a limited number of items can obtain a listing one part at a time as the user scrolls the display. If possible, the returned list should resemble the order used on the local display on the TG, but should list all folder items before media element items to facilitate browsing on the CT.

To allow the CT to request specific Metadata Attributes be returned along with each media element in the folder listing the command shall include a filter specifying which metadata attributes are requested to be returned by the TG. The TG should provide the available attribute values in the response. The TG is not required to provide a value for all requested attributes.

If the TG receives a GetFolderItems command for an empty folder then the TG shall return the error (= Range Out of Bounds) in the status field of the GetFolderItems response.

When responds to the GetFolderItems command, call this function with *subpacket_type* set to BTSDK_AVRCP_PACKETTYPE_BROWSABLE_ITME and *item* specify a Browsable Item, and so on for the other items. And the *item.item_num* should be always set to the total number of the items. If the *item.item_type* is BTSDK_AVRCP_ITEMTYPE_MEDIAELEMENT_ITEM, call this function with *subpacket_type* set to BTSDK_AVRCP_PACKETTYPE_MEDIA_ATTR and *element_attr* specify an attribute of this Media Element Attribute, and so on for the other attributes.

6.3.4.1.22 Btsdk_AVRCP_GetItemAttrRsp

Prototype	BTINT32 Btsdk_AVRCP_GetItemAttrRsp (BTDEVHDL hdl, BTUINT8 tl, PBtSdkGetItemAttrRspStru param);	
Description	The **Btsdk_AVRCP_GetItemAttrRsp** responds to the GetItemAttribute command from CT.	
Parameters	*hdl*	[in] Handle to the peer device.
	tl	[in] Transaction labeling. Get from the Command Callback.
	param	[in] Pointer to the 错误！未找到引用源。 structure.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

6.3.4.1.23 Btsdk_AVRCP_SearchRsp

Prototype	BTINT32 Btsdk_AVRCP_SearchRsp (BTDEVHDL hdl, BTUINT8 tl, PBtSdkSearchRspStru param);	
Description	The **Btsdk_AVRCP_SearchRsp** responds to the Search command from CT.	
Parameters	*hdl*	[in] Handle to the peer device.
	tl	[in] Transaction labeling. Get from the Command Callback.
	param	[in] Pointer to the 错误！未找到引用源。 structure.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

6.3.4.1.24 Btsdk_AVRCP_PlayItemRsp

Prototype	BTINT32 Btsdk_AVRCP_PlayItemRsp (BTDEVHDL hdl, BTUINT8 tl, PBtSdkPlayItemRspStru param);	
Description	The **Btsdk_AVRCP_PlayItemRsp** responds to the PlayItem command from CT.	
Parameters	*hdl*	[in] Handle to the peer device.
	tl	[in] Transaction labeling. Get from the Command Callback.
	param	[in] Pointer to the 错误！未找到引用源。 structure.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

6.3.4.1.25 Btsdk_AVRCP_AddToNowPlayingRsp

Prototype	BTINT32 Btsdk_AVRCP_AddToNowPlayingRsp (BTDEVHDL hdl, BTUINT8 tl, PBtSdkAddToNowPlayingRspStru param);	
Description	The **Btsdk_AVRCP_AddToNowPlayingRsp** responds to the AddToNowPlaying command from CT, if the operation succeed.	
Parameters	*hdl*	[in] Handle to the peer device.
	tl	[in] Transaction labeling. Get from the Command Callback.
	param	[in] Pointer to the 错误！未找到引用源。 structure.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

6.3.4.1.26 Btsdk_AVRCP_SetAbsoluteVolRsp

Prototype	BTINT32 Btsdk_AVRCP_SetAbsoluteVolRsp (BTDEVHDL hdl, BTUINT8 tl, PBtSdkSetAbsoluteVolRspStru param);	
Description	The **Btsdk_AVRCP_SetAbsoluteVolRsp** responds to the SetAbsoluteVolume command from CT.	
Parameters	*hdl*	[in] Handle to the peer device.
	tl	[in] Transaction labeling. Get from the Command Callback.
	param	[in] Pointer to the 错误！未找到引用源。 structure.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

6.3.4.1.27 Btsdk_AVRCP_GeneralRejectRsp

Prototype	BTINT32 Btsdk_AVRCP_GeneralRejectRsp (BTDEVHDL hdl, BTUINT8 tl, PBtSdkGeneralRejectRspStru param);	
Description	The **Btsdk_AVRCP_GeneralRejectRsp** is used as error handling for the recieved command.	
Parameters	*hdl*	[in] Handle to the peer device.
	tl	[in] Transaction labeling. Get from the Command Callback.
	param	[in] Pointer to the 错误！未找到引用源。 structure.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

6.3.4.1.28 Btsdk_UnregisterAVRCPTGService

Prototype	BTUINT32 Btsdk_UnregisterAVRCPTGService (void);	
Description	The **Btsdk_UnregisterAVRCPTGService** function is unregister TG service.	
Parameters	*void*	
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

6.3.4.1.29 Btsdk_RegisterAVRCPTGService

Prototype	BTSVCHDL Btsdk_RegisterAVRCPTGService (void);	
Description	The **Btsdk_RegisterAVRCPTGService** function is used to Registration of TG service. BlueSoleil registers the TG service by default.	
Parameters	*void*	
Return:	If the function succeeds, the return value is the handle If the function fails, the return value is BTSDK_INVALID_HANDLE	

Remarks

6.3.4.2 AVRCP Control(CT)

6.3.4.2.1 Btsdk_AVRCP_CTRegResponseCbk

Prototype	void Btsdk_AVRCP_CTRegResponseCbk (Btsdk_AVRCP_CT_Response_Cbk_Func *pfunc);	
Description	The **Btsdk_AVRCP_CTRegResponseCbk** function registers a CT callback function used to deal with the responses from the TG.	
Parameters	*pfunc*	[in] Pointer to the callback function of Btsdk_AVRCP_CT_Response_Cbk_Func. If p*func* is NULL, BlueSoleil will remove the callback information registered before.
Return:		

Remarks

6.3.4.2.2 Btsdk_AVRCP_CT_Response_Cbk_Func

Prototype	typedef BTBOOL (Btsdk_AVRCP_CT_Response_Cbk_Func) (BTDEVHDL dev_hdl, BTUINT16 rsp_type, BTUINT8 *param);	
Description	The **Btsdk_AVRCP_CT_Response_Cbk_Func** function is used to deal with response from TG. This function can not be blocked. So if application do something cost time, please do it in a new thread.	
Parameters	*dev_hdl*	[in] Handle to the remote device
	rsp_type	[in] Event specific type. Please refer to the following table, different rsp_type corresponds to different param.
	param	[in] Event specific parameter.
Return:	If the function succeeds, the return value is BTSDK_TRUE. If the function fails, the return value is BTSDK_FALSE.	

The *cmd_type* parameter can be one of these values,

rsp_type	*Description*
BTSDK_APP_EV_AVRCP_GET_CAPABILITIES_RSP	CT received the GetCapabilities respond from the TG.
BTSDK_APP_EV_AVRCP_LIST_PLAYER_SETTING_ATTR_RSP	CT received the ListPlayerApplicationSettingAttributes respond from TG.
BTSDK_APP_EV_AVRCP_LIST_PLAYER_SETTING_VALUES_RSP	CT received the ListPlayerApplicationSettingValues respond from TG.
BTSDK_APP_EV_AVRCP_GET_CURRENTPLAYER_SETTING_VALUE_RSP	CT received the GetCurrentPlayerApplicationSettingValue respond from TG.
BTSDK_APP_EV_AVRCP_SET_CURRENTPLAYER_SETTING_VALUE_RSP	CT received the SetPlayerApplicationSettingValues respond from TG.
BTSDK_APP_EV_AVRCP_GET_PLAYER_SETTING_ATTR_TEXT_RSP	CT received the GetPlayerApplicationSettingAttributeText respond from TG.
BTSDK_APP_EV_AVRCP_GET_PLAYER_SETTING_VALUE_TEXT_RSP	CT received the GetPlayerApplicationSettingValueText respond from TG.

BTSDK_APP_EV_AVRCP_INFORM_CHARACTERSET_RSP	CT received the InformDisplayableCharacterSet respond from TG.
BTSDK_APP_EV_AVRCP_INFORM_BATTERYSTATUS_OF_CT_RSP	CT received the InformBatteryStatusOfCT respond from TG.
BTSDK_APP_EV_AVRCP_GET_ELEMENT_ATTR_RSP	CT received the GetElementAttributes respond from TG.
BTSDK_APP_EV_AVRCP_GET_PLAY_STATUS_RSP	CT received the GetPlayStatus.respond from TG.
BTSDK_APP_EV_AVRCP_SET_ABSOLUTE_VOLUME_RSP	CT received the SetAbsoluteVolume respond from TG.
BTSDK_APP_EV_AVRCP_SET_ADDRESSED_PLAYER_RSP	CT received the SetAddressedPlayer respond from TG.
BTSDK_APP_EV_AVRCP_SET_BROWSED_PLAYER_RSP	CT received the SetBrowsedPlayer respond from TG.
BTSDK_APP_EV_AVRCP_GET_FOLDER_ITEMS_RSP	CT received the GetFolderItems respond from TG.
BTSDK_APP_EV_AVRCP_CHANGE_PATH_RSP	CT received the ChangePath respond from TG.
BTSDK_APP_EV_AVRCP_GET_ITEM_ATTRIBUTES_RSP	CT received the GetItemAttributes respond from TG.
BTSDK_APP_EV_AVRCP_PLAY_ITEM_RSP	CT received from TG for a respond to a request of PlayItem.
BTSDK_APP_EV_AVRCP_SEARCH_RSP	CT received the Search respond from TG.
BTSDK_APP_EV_AVRCP_ADDTO_NOWPLAYING_RSP	CT received the AddToNowPlaying respond from TG.
BTSDK_APP_EV_AVRCP_GENERAL_REJECT_RSP	CT received the General Reject respond from TG.
BTSDK_APP_EV_AVRCP_PLAYBACK_STATUS_CHANGED_NOTIF	CT received an EVENT_PLAYBACK_STATUC_CHANGED notification from TG.
BTSDK_APP_EV_AVRCP_TRACK_CHANGED_NOTIF	CT received an EVENT_TRACK_CHANGED notification from TG.
BTSDK_APP_EV_AVRCP_TRACK_REACHED_END_NOTIF	CT received an EVENT_TRACK_REACHED_END notification from TG.
BTSDK_APP_EV_AVRCP_TRACK_REACHED_START_NOTIF	CT received an EVENT_TRACK_REACHED_START notification from TG.
BTSDK_APP_EV_AVRCP_PLAYBACK_POS_CHANGED_NOTIF	CT received an EVENT_PLAYBACK_POS_CHANGE

	D notification from TG.
BTSDK_APP_EV_AVRCP_BATT_STATUS_CHANGED_NOTIF	CT received an EVENT_BATT_STATUS_CHANEGED notification from TG.
BTSDK_APP_EV_AVRCP_SYSTEM_STATUS_CHANGED_NOTIF	CT received an EVENT_SYSTEM_STATUS_CHANGED notification from TG.
BTSDK_APP_EV_AVRCP_PLAYER_APPLICATION_SETTING_CHANGED_NOTIF	CT received an EVENT_PLAYER_APPLICATION_SETTING_CHANGED _CHANGED notification from TG.
BTSDK_APP_EV_AVRCP_NOW_PLAYING_CONTENT_CHANGED_NOTIF	CT received an EVENT_NOW_PLAYING_CONTENT_CH notification from TG.
BTSDK_APP_EV_AVRCP_AVAILABLE_PLAYERS_CHANGED_NOTIF	CT received an EVENT_AVAILABLE_PLAYERS_CHANGE notification from TG.
BTSDK_APP_EV_AVRCP_ADDRESSED_PLAYER_CHANGED_NOTIF	CT received an EVENT_ADDRESSED_PLAYER_CHANGED notification from TG.
BTSDK_APP_EV_AVRCP_UIDS_CHANGED_NOTIF	CT received an EVENT_UID_CHANGED notification from TG.
BTSDK_APP_EV_AVRCP_VOLUME_CHANGED_NOTIF	CT received an EVENT_VOLUME_CHANGED notification from TG.

rsp_type	*param*
BTSDK_APP_EV_AVRCP_GET_CAPABILITIES_RSP	**错误！未找到引用源。**
BTSDK_APP_EV_AVRCP_LIST_PLAYER_SETTING_ATTR_RSP	**错误！未找到引用源。**
BTSDK_APP_EV_AVRCP_LIST_PLAYER_SETTING_VALUES_RSP	**错误！未找到引用源。**
BTSDK_APP_EV_AVRCP_GET_CURRENTPLAYER_SETTING_VALUE_RSP	**错误！未找到引用源。**
BTSDK_APP_EV_AVRCP_SET_CURRENTPLAYER_SETTING_VALUE_RSP	NULL
BTSDK_APP_EV_AVRCP_GET_PLAYER_SETTING_ATTR_TEXT_RSP	**错误！未找到引用源。**
BTSDK_APP_EV_AVRCP_GET_PLAYER_SETTING_VALUE_TEXT_RSP	**错误！未找到引用源。**

BTSDK_APP_EV_AVRCP_INFORM_CHARACTERSET_RSP	NULL
BTSDK_APP_EV_AVRCP_INFORM_BATTERYSTATUS_OF_CT_RSP	NULL
BTSDK_APP_EV_AVRCP_GET_ELEMENT_ATTR_RSP	错误！未找到引用源。
BTSDK_APP_EV_AVRCP_GET_PLAY_STATUS_RSP	错误！未找到引用源。
BTSDK_APP_EV_AVRCP_SET_ABSOLUTE_VOLUME_RSP	错误！未找到引用源。
BTSDK_APP_EV_AVRCP_SET_ADDRESSED_PLAYER_RSP	错误！未找到引用源。
BTSDK_APP_EV_AVRCP_SET_BROWSED_PLAYER_RSP	错误！未找到引用源。
BTSDK_APP_EV_AVRCP_GET_FOLDER_ITEMS_RSP	错误！未找到引用源。
BTSDK_APP_EV_AVRCP_CHANGE_PATH_RSP	错误！未找到引用源。
BTSDK_APP_EV_AVRCP_GET_ITEM_ATTRIBUTES_RSP	错误！未找到引用源。
BTSDK_APP_EV_AVRCP_PLAY_ITEM_RSP	错误！未找到引用源。
BTSDK_APP_EV_AVRCP_SEARCH_RSP	错误！未找到引用源。
BTSDK_APP_EV_AVRCP_ADDTO_NOWPLAYING_RSP	错误！未找到引用源。
BTSDK_APP_EV_AVRCP_GENERAL_REJECT_RSP	错误！未找到引用源。
BTSDK_APP_EV_AVRCP_PLAYBACK_STATUS_CHANGED_NOTIF	BtSdkPlayStatusChangedStru
BTSDK_APP_EV_AVRCP_TRACK_CHANGED_NOTIF	错误！未找到引用源。
BTSDK_APP_EV_AVRCP_TRACK_REACHED_END_NOTIF	错误！未找到引用源。
BTSDK_APP_EV_AVRCP_TRACK_REACHED_START_NOTIF	错误！未找到引用源。
BTSDK_APP_EV_AVRCP_PLAYBACK_POS_CHANGED_NOTIF	错误！未找到引用源。
BTSDK_APP_EV_AVRCP_BATT_STATUS_CHANGED_NOTIF	错误！未找到引用源。
BTSDK_APP_EV_AVRCP_SYSTEM_STATUS_CHANGED_NOTIF	错误！未找到引用源。
BTSDK_APP_EV_AVRCP_PLAYER_APPLICATION_SETTING_CHANGED_NOTIF	错误！未找到引用源。

BTSDK_APP_EV_AVRCP_NOW_PLAYING_CONTENT_CHANGED_NOTIF	**错误！未找到引用源。**
BTSDK_APP_EV_AVRCP_AVAILABLE_PLAYERS_CHANGED_NOTIF	**错误！未找到引用源。**
BTSDK_APP_EV_AVRCP_ADDRESSED_PLAYER_CHANGED_NOTIF	NULL
BTSDK_APP_EV_AVRCP_UIDS_CHANGED_NOTIF	**错误！未找到引用源。**
BTSDK_APP_EV_AVRCP_VOLUME_CHANGED_NOTIF	**错误！未找到引用源。**

Remarks

This callback function is called to deal with events of TG response.

6.3.4.2.3 Btsdk_AVRCP_GetElementAttrReq

Prototype	BTINT32 Btsdk_AVRCP_GetElementAttrReq (BTDEVHDL *hdl*, PBtSdkGetElementAttrReqStru *param*);	
Description	The **Btsdk_AVRCP_GetElementAttrReq** function sends the GetElementAttributes command to request TG to provide the attributes of the element specified in the parameter.	
Parameters	*hdl*	[in] Handle to the peer device.
	param	[in] Pointer to the 错误！未找到引用源。 structure specifies the attributes.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

This shall only be used to retrieve Metadata for the currently playing track from the Addressed Player on the Control channel when GetItemAttributes is not supported. Use 错误！未找到引用源。 function to retriving Metadata for other items.

6.3.4.2.4 Btsdk_AVRCP_ GetPlayStatusReq

Prototype	BTINT32 Btsdk_AVRCP_GetPlayStatusReq (BTDEVHDL hdl);	
Description	The **Btsdk_AVRCP_GetPlayStatusReq** function sends the GetPlayStatus command to get the status of the currently playing media at TG.	
Parameters	*hdl*	[in] Handle to the peer device.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

6.3.4.2.5 Btsdk_AVRCP_RegNotifReq

Prototype	BTINT32 Btsdk_AVRCP_RegNotifReq (BTDEVHDL dev_hdl, PBtSdkRegisterNotifiReqStru pRegNotif);	
Description	The **Btsdk_AVRCP_CTRegResponseCbk** function registers with the TG to receive notification asynchronously based on specific events occurring. The initial response to this Notify command shall be an INTERIM response with current status, or a REJECTED/NOT IMPLEMENTED response. The following response shall be a CHANGED response. A registed notification gets changed on receiving CHANGED event notification. For a new notification additional NOTIFY command is expected to be sent.	
Parameters	*dev_hdl*	[in] Handle to the remote device
	pRegNotif	[in] Pointer to the 错误！未找到引用源。 structure specifies the event for which the CT requires notifications.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

When this function is called in CT, the event type of callback function in TG is BTSDK_APP_EV_AVRCP_REGISTER_NOTIFICATION_IND.

6.3.4.2.6 Btsdk_AVRCP_PassThroughReq

Prototype	BTINT32 Btsdk_AVRCP_PassThroughReq(PBtSdkPassThrReqStru preq);	
Description	The **Btsdk_AVRCP_PassThroughReq** function is to send a pass through command to the TG.	
Parameters	*preq*	[in] point to a BtSdkPassThrReqStru structure which contains PASS THROUGH command.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

6.3.4.2.7 Btsdk_AVRCP_PassThroughReqEx

Prototype	BTINT32 Btsdk_AVRCP_PassThroughReqEx(BTDEVHDL hdl, PBtSdkPassThroughStru param);	
Description	The **Btsdk_AVRCP_PassThroughReqEx** function is to send a pass through command to the TG.	
Parameters	*hdl*	[in] Handle to the peer device.
	preq	[in] point to a BtSdkPassThroughStru structure which contains PASS THROUGH command.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

BlueSoleil recommend use Btsdk_AVRCP_PassThroughReqEx instead of Btsdk_AVRCP_PasThroughReq.

6.3.4.2.8 Btsdk_AVRCP_SetAbsoluteVolReq

Prototype	BTINT32 Btsdk_AVRCP_SetAbsoluteVolReq (BTDEVHDL hdl, PBtSdkSetAbsoluteVolReqStru param);	
Description	The **Btsdk_AVRCP_SetAbsoluteVolReq** function is to set an absolutevolume to be used by the rendering device. This is in additional to the relative volume PASS THROUGH commands. It is expected that the audio sink will perform as the TG for this command.	
Parameters	*hdl*	[in] Handle to the peer device.
	param	[in] Pointer to the 错误！未找到引用源。 structure specifies the volume which is requested.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

6.3.4.2.9 Btsdk_AVRCP_AddToNowPlayingReq

Prototype	BTINT32 Btsdk_AVRCP_AddToNowPlayingReq (BTDEVHDL hdl, PBtSdkAddToNowPlayingReqStru param);	
Description	The **Btsdk_AVRCP_AddToNowPlayingReq** function sends the AddToNowPlaying command to add an item indicated by the UID to the Now Playing queue.	
Parameters	*hdl*	[in] Handle to the peer device.
	param	[in] Pointer to the structure **错误！未找到引用源。**.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

6.3.4.2.10 Btsdk_AVRCP_PlayItemReq

Prototype	BTINT32 Btsdk_AVRCP_PlayItemReq (BTDEVHDL hdl, PBtSdkPlayItemReqStru param);	
Description	The **Btsdk_AVRCP_PlayItemReq** function starts playing an item indicated by the UID. It is routed to the Addressed Player.	
Parameters	*hdl*	[in] Handle to the peer device.
	param	[in] Pointer to the 错误！未找到引用源。 structure specific the item requested.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

6.3.4.2.11 Btsdk_AVRCP_SearchReq

Prototype	BTINT32 Btsdk_AVRCP_SearchReq (BTDEVHDL hdl, PBtSdkSearchReqStru param);	
Description	The **Btsdk_AVRCP_SearchReq** function provides basic search functionality. Regular expression shall not be supported. Seach string interpretation by the TG should be consistent between the local user interface and AVRCP search.	
Parameters	*hdl*	[in] Handle to the peer device.
	param	[in] Pointer to the 错误！未找到引用源。 structure specific the search string.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

6.3.4.2.12 Btsdk_AVRCP_GetItemAttrReq

Prototype	BTINT32 Btsdk_AVRCP_GetItemAttrReq (BTDEVHDL hdl, PBtSdkGetItemAttrReqStru param);	
Description	The **Btsdk_AVRCP_GetItemAttrReq** function sends the GetItemAttributes command to retrieve the metadata attributes for a particular media element item or folder item.	
Parameters	*hdl*	[in] Handle to the peer device.
	param	[in] Pointer to the 错误！未找到引用源。 structure specific the item and item's attributes requested.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

6.3.4.2.13 Btsdk_AVRCP_GetFolderItemsReq

Prototype	BTINT32 Btsdk_AVRCP_GetFolderItemsReq (BTDEVHDL hdl, PBtSdkGetFolderItemReqStru param);	
Description	The **Btsdk_AVRCP_GetFolderItemsReq** function is used to retrieve a listing of the contents of a folder.	
Parameters	*hdl*	[in] Handle to the peer device.
	param	[in] Pointer to the 错误！未找到引用源。 structure specific a range of entries to be returned.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

6.3.4.2.14 Btsdk_AVRCP_ChangePathReq

Prototype	BTINT32 Btsdk_AVRCP_ChangePathReq (BTDEVHDL hdl, PBtSdkChangePathReqStru param);	
Description	The **Btsdk_AVRCP_ChangePathReq** function sends the ChangePath command to navigate the virtual filesystem. This command allows the CT to navigate one level up or down in the virtual filesystem.	
Parameters	*hdl*	[in] Handle to the peer device.
	param	[in] Pointer to the 错误！未找到引用源。 structure specific the path.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

6.3.4.2.15 Btsdk_AVRCP_SetBrowsedPlayerReq

Prototype	BTINT32 Btsdk_AVRCP_SetBrowsedPlayerReq (BTDEVHDL hdl, PBtSdkSetBrowsedPlayerReqStru param);	
Description	The **Btsdk_AVRCP_SetBrowsedPlayerReq** function sends the SetBrowsedPlayer command to inform the TG of which media player browsing commands should be routed. It shall be sent successfully before any other commands are sent on the browsing chanel except GetFolderItems in the media Player List scope. If the browsed player has become unavailable the SetBrowsedPlayer command shall be sent successfully again before further commands are sent on the browsing channel.	
Parameters	*hdl*	[in] Handle to the peer device.
	param	[in] Pointer to the BtSdkSetBrowsedPlayerReqStru structure specific the player ID.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

6.3.4.2.16 Btsdk_AVRCP_ SetAddressedPlayerReq

Prototype	BTINT32 Btsdk_AVRCP_SetAddressedPlayerReq (BTDEVHDL hdl, PBtSdkSetAddresedPlayerReqStru param);	
Description	The **Btsdk_AVRCP_SetAddressedPlayerReq** function used to inform the TG of which media player the CT wishes to control.	
Parameters	*hdl*	[in] Handle to the peer device.
	param	[in] Pointer to the 错误！未找到引用源。 structure specific the player ID.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

6.3.4.2.17 Btsdk_AVRCP_InformBattStatusReq

Prototype	BTINT32 Btsdk_AVRCP_InformBattStatusReq (BTDEVHDL hdl, PBtSdkInformBattStatusReqStru param);	
Description	The **Btsdk_AVRCP_InformBattStatusReq** function sends the InformBatteryStatusOfCT command to TG to inform the CT's battery status to the TG, whenever the CT's battery status has been changed.	
Parameters	*hdl*	[in] Handle to the peer device.
	param	[in] Pointer to the 错误！未找到引用源。 structure specific the CT's battery status.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

6.3.4.2.18 Btsdk_AVRCP_InformCharSetReq

Prototype	BTINT32 Btsdk_AVRCP_InformCharSetReq (BTDEVHDL hdl, PBtSdkInformCharSetReqStru param);	
Description	The **Btsdk_AVRCP_InformCharSetReq** function sends the InformDisplayableCharacterSet command to TG to inform the list of character set supported by CT.	
Parameters	*hdl*	[in] Handle to the peer device.
	param	[in] Pointer to the 错误！未找到引用源。 structure specifies the CharacterSetIDs.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

If this command is not issued, UTF-8 shall be used for any strings as default character set. It is mandatory for CT to send UTF-8 as one of the supported character set in the PDU parameters.

The CT should send this command before it sends any commands that support multiple character sets as follows:

错误！未找到引用源。

错误！未找到引用源。

GetElementAttributes

SetBrowsedPlayer

GetFolderItems

Search

6.3.4.2.19 Btsdk_AVRCP_GetPlayerAppSetValTxtReq

Prototype	BTINT32 Btsdk_AVRCP_GetPlayerAppSetValTxtReq (BTDEVHDL hdl, PBtSdkGetPlayerAppSetValTxtReqStru param);	
Description	The **Btsdk_AVRCP_GetPlayerAppSetValTxtReq** function sends the GetPlayerApplicationSettingValueText command to request TG to provide supported player application setting value displayable text for the provided player application setting attribute values.	
Parameters	*hdl*	[in] Handle to the peer device.
	param	[in] Pointer to the 错误！未找到引用源。 structure specifies the provided AttributeID and attribute values.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

This command is expected to be used only for extended attributes for menu navigation. It is assumed that all <attributes, value> pairs used for menu extensions are statically defined by TG.

6.3.4.2.20 Btsdk_AVRCP_GetPlayerAppSetAttrTxtReq

Prototype	BTINT32 Btsdk_AVRCP_GetPlayerAppSetAttrTxtReq (BTDEVHDL hdl, PBtSdkGetPlayerAppSetAttrTxtReqStru param);	
Description	The **Btsdk_AVRCP_GetPlayerAppSetAttrTxtReq** function sends the GetPlayerApplicationAttributeText command to request TG to provide supported player application setting attribute displayable text for the provided PlayerApplicationSettingAttributeIDs.	
Parameters	*hdl*	[in] Handle to the peer device.
	param	[in] Pointer to the 错误！未找到引用源。 structure specifies the provided AttributesIDs list.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

This command is expected to be used only for extended attributes for menu navigation. It is assumed that all <attributes, value> pairs used for menu extensions are statically defined by TG.

6.3.4.2.21 Btsdk_AVRCP_GetCurPlayerAppSetValReq

Prototype	BTINT32 Btsdk_AVRCP_GetCurPlayerAppSetValReq (BTDEVHDL hdl, PBtSdkGetCurPlayerAppSetValReqStru param);	
Description	The **Btsdk_AVRCP_GetCurPlayerAppSetValReq** function sends the GetCurrentPlayerApplicationSettingValue command to request TG to provide the current set values for the provided player application setting attributes list.	
Parameters	*hdl*	[in] Handle to the peer device.
	param	[in] Pointer to the 错误！未找到引用源。 structure specifies the player application setting attributes list.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

6.3.4.2.22 Btsdk_AVRCP_SetCurPlayerAppSetValReq

Prototype	BTINT32 Btsdk_AVRCP_SetCurPlayerAppSetValReq (BTDEVHDL hdl, PBtSdkSetCurPlayerAppSetValReqStru param);	
Description	The Btsdk_AVRCP_SetCurPlayerAppSetValReq function requests the TG to set the player application setting list of player application setting values.	
Parameters	*hdl*	[in] Handle to the peer device.
	param	[in] Pointer to the BtSdkSetCurPlayerAppSetValReqStru structure Specifies the player application setting attributes list.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

6.3.4.2.23 Btsdk_AVRCP_Group_NavigateReq

Prototype	BTINT32 Btsdk_AVRCP_Group_NavigateReq (PBtSdkGroupNaviReqStru param);	
Description	The **Btsdk_AVRCP_Group_NavigateReq** function is to send Group Navigate Command request to TG.	
Parameters	*param*	[in] Group Navigate Command request parameters
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

6.3.4.2.24 Btsdk_AVRCP_ListPlayerAppSetValReq

Prototype	BTINT32 Btsdk_AVRCP_ListPlayerAppSetValReq (BTDEVHDL hdl, PBtSdkListPlayerAppSetValReqStru param);	
Description	The **Btsdk_AVRCP_ListPlayerAppSetValReq** function sends the ListPlayerApplicationSettingValues command to request TG to list the set of possible values for the requested player application setting attributes.	
Parameters	*hdl*	[in] Handle to the peer device.
	param	[in] Pointer to the 错误！未找到引用源。 structure specifies the player application setting attribute ID.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

6.3.4.2.25 Btsdk_AVRCP_ListPlayerAppSetAttrReq

Prototype	BTINT32 Btsdk_AVRCP_ListPlayerAppSetAttrReq (BTDEVHDL hdl);	
Description	The **Btsdk_AVRCP_ListPlayerAppSetAttrReq** function sends the ListPlayerAppliactionSettingAttributes command to request TG to provide supported player application setting attributes.	
Parameters	*hdl*	[in] Handle to the peer device.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

6.3.4.2.26 Btsdk_AVRCP_GetCapabilitiesReq

Prototype	BTINT32 Btsdk_AVRCP_GetCapabilitiesReq (BTDEVHDL hdl, PBtSdkGetCapabilitiesReqStru param);	
Description	This function sends the GetCapabilities command to get the capabilities supported by TG.	
Parameters	*hdl*	[in] Handle to the peer device.
	param	[in] Pointer to the 错误！未找到引用源。 structure specifies the capabilities requested.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

The CT should be aware that the capabilities supported by the TG may be subject to change. This may occur if the application on the TG changes, or the application changes mode, for instance different fuctionality may be available when the TG is playing locally stored audio tracks to when if is acting as a radio. How this is handled by the CT is implementaion dependent. If the TG application changes to support less functionality the CT may receive error responses indicating that the function requesed is not implemented. The CT may then decide to reissue the GetCapabilities to get the most current capabilities if the TG appplication changes to support more features the CT may be happy to continue using the original set of features supported. If not it may choose to occasionally poll the TG with a GetCapabilities to determine when further capabilitise are available.

6.3.4.3 AVRCP Control(Event)

6.3.4.3.1 Btsdk_AVRCP_Event_Ind_Func

Prototype	typedef void (Btsdk_AVRCP_Event_Ind_Func) (BTUINT16 event, BTUINT8* param,);	
Description	The **Btsdk_AVRCP_Event_Ind_Func** function prototype is the prototype of application defined callback function used to deal with AVRCP connection events. This function can not be blocked.	
Parameters	*event*	[in] Event identifier.
	param	[in] Event specific parameter.
Return:		

The ***event*** parameter can be one of these values,

Value	Description
BTSDK_APP_EV_AVTG_ATTACHPLAYER_IND	A remote Controller connects to the local TG service. The application can now select a media player program to be controlled by the remote Controller. The *param* parameter is ignored.
BTSDK_APP_EV_AVRCP_DETACHPLAYER_IND	The connection from the remote Controller is released. The application can now release the control to the selected media player program. The *param* parameter is ignored.

Remarks

This callback function is called local avrcp target connect with or disconnect from remote avrcp controller.

6.3.4.3.2 Btsdk_AVRCP_EventTrackChanged

Prototype	BTINT32 Btsdk_AVRCP_EventTrackChanged (PBtSdkTrackChangedStru param,);	
Description	The **Btsdk_AVRCP_EventTrackChanged** function is called to respond to the RegisterNotification command from CT with the event_id set to BTSDK_AVRCP_EVENT_TRACK_CHANGED, and notify CT when the track has been changed.	
Parameters	*param*	[in] Pointer to the BtsdkTrackChangedStru structure.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

This function is called when the player's track information changed .It can only be effect when CT register BTSDK_AVRCP_EVENT_TRACK_CHANGED notification request and TG give a response.

6.3.4.3.3 Btsdk_AVRCP_EventPlayStatusChanged

Prototype	BTINT32 Btsdk_AVRCP_EventPlayStatusChanged (PBtSdkPlayStatusChangedStru param);	
Description	The **Btsdk_AVRCP_EventPlayStatusChanged** function is used to respond to the RegisterNotification command from CT with the event_id set to BTSDK_AVRCP_EVENT_PLAYBACK_STATUS_CHANGED, and notify CT when the Playback status has been changed.	
Parameters	param	[in] Pointer to the BtsdkPlayStatusChangedStru structure.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

6.3.4.3.4 Btsdk_AVRCP_EventTrackReachEnd

Prototype	BTINT32 Btsdk_AVRCP_EventTrackReachEnd (PBtSdkTrackReachEndStru param);	
Description	The **Btsdk_AVRCP_EventTrackReachEnd** function sends a CHANGED event notification to CT, when the track on the TG reaches the end. If any action (e.g. GetElementAttributes) is undertaken on the CT as reaction to the EVENT_TRACK_REACHED_END, the CT should register the EVENT_TRACK_REACHED_END again before initiation this action in order to get informed about intermediate changes regarding the track status.	
Parameters	param	[in] Pointer to the BtsdkTrackReachEndStru structure.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

6.3.4.3.5 Btsdk_AVRCP_EventTrackReachStart

Prototype	BTINT32 Btsdk_AVRCP_EventTrackReachStart (PBtSdkTrackReachStartStru param);	
Description	The **Btsdk_AVRCP_EventTrackReachStart** function sends a CHANGED event notification to CT, when the track on the TG reaches the start. If any action (e.g. GetElementAttributes) is undertaken on the CT as reaction to the EVENT_TRACK_REACHED_START, the CT should register the EVENT_TRAC_REACHED_START again before initiation this action in order to get informed about intermediate changes regarding the track status.	
Parameters	param	[in] Pointer to the BtsdkTrackReachStartStru structure.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

6.3.4.3.6 Btsdk_AVRCP_EventBattStatusChanged

Prototype	BTINT32 Btsdk_AVRCP_EventBattStatusChanged (PBtSdkBattStatusChangedStru param);	
Description	The **Btsdk_AVRCP_EventBattStatusChanged** function sends a CHANGED event notification to CT, when the TG's battery status is changed.	
Parameters	param	[in] Pointer the BtsdkBattStatusChangedStru structure.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

6.3.4.3.7 Btsdk_AVRCP_EventSysStatusChanged

Prototype	BTINT32 Btsdk_AVRCP_EventSysStatusChanged (PBtSdkSysStatusChangedStru param);	
Description	The **Btsdk_AVRCP_EventSysStatusChanged** function sendsa CHANGED event notification to CT, when the system status is changed. POWER_OFF and UNPLUGGED are used for Bluetooth Accessories which attach to Media Players. In this case, it will happen that Audio Player's power state is "POWER OFF" or Audio Player is detached from Bluetooth Adapter (UNPLUGGED).	
Parameters	param	[in] Pointer the the BtsdkSysStatusChangedStru structure.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

6.3.4.3.8 Btsdk_AVRCP_EventPlayerAppSetChanged

Prototype	BTINT32 Btsdk_AVRCP_EventPlayerAppSetChanged (PBtSdkPlayerAppSetChangedStru param);	
Description	The **Btsdk_AVRCP_EventPlayerAppSetChanged** function sends a CHANGED event notification to CT, when the Player Application Setting is changed. Note that as settings may be addes or removed all player application settings are returned to enable the CT to determine which settings have changed.	
Parameters	param	[in] Pointer to the BtsdkPlayerAppSetChangedStru structure specifies current setting.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

6.3.4.3.9 Btsdk_AVRCP_EventVolChanged

Prototype	BTINT32 Btsdk_AVRCP_EventVolChanged (PBtSdkVolChangedStru param);	
Description	The **Btsdk_AVRCP_EventVolChanged** function sends a CHANGED event notification to CT, when the volume has been changed locally on the TG, or what the actual volume level is following use of relative volume commands. Note thar if this is being used to detect the result of a relative volume command then if the relative volume command results in no volume change (for example the TG is already at maximum volume and receives a volume up command) then there will be no volume change and hence the TG shall not completes an outstanding Volume Change Notification.	
Parameters	param	[in] Pointer to the BtsdkVolChangedStru.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

6.3.4.3.10 Btsdk_AVRCP_EventAddrPlayerChanged

Prototype	BTINT32 Btsdk_AVRCP_EventAddrPlayerChanged (PBtSdkAddrPlayerChangedStru param);	
Description	The **Btsdk_AVRCP_EventAddrPlayerChanged** function sends a CHANGED event notification to CT, when the addressed player on the TG is changed. The interim response to the Notify shall contain the Player ID of the current addressed player. If the CT registers this Notify before sending a SetAddressedPlayer command the interim response contains the Player ID of the default player on the TG. On completion of the Addressed Player Changed notification the TG shall complete all player specific notifications with AV/C C-Type REJECTED with error code Addressed Player Changed. Which notification are defined as player specific in the following: EVENT_PLAYBACK_STATUS_CHANGED EVENT_TRACK_CHANGED EVENT_TRACK_REACHED_END EVENT_TRACK_REACHED_START EVENT_PLAYBACK_POS_CHANGED EVENT_PLAYER_APPLICATION_SETTING_CHANGED EVENT_NOW_PLAYING_CONTENT_CHANGED	
Parameters	param	[in] Pointer to the BtsdkAddrPlayerChangedStru structure.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

6.3.4.3.11 Btsdk_AVRCP_EventAvailablePlayerChanged

Prototype	BTINT32 Btsdk_AVRCP_EventAvailablePlayerChanged (PBtSdkAvailablePlayerChangedStru param);	
Description	The **Btsdk_AVRCP_EventAvailablePlayerChanged** function sends a CHANGED event notification to CT, when a new player becomes available to be addressed (for instance started, or installed) or a player ceases to be available. Note that to view information about available player, such as their status, the Media Player List may be browsed, If the Media Player List is browsed as reaction to the EVENT_AVAILABLE_PLAYER_CHANGED, the CT should register the EVENT_AVALIABLE_PLAYERS_CHANGED again before broesing the Media Player List in order to get informed about intermediate changes of the available players.	
Parameters	param	[in] Pointer to the BtSdkAvailablePlayerChangedStru structure.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

6.3.4.3.12 Btsdk_AVRCP_EventUIDSChanged

Prototype	BTINT32 Btsdk_AVRCP_EventUIDSChanged (PBtSdkUIDSChangedStru param);	
Description	The **Btsdk_AVRCP_EventUIDSChanged** function sends a CHANGED event notification to CT, when the UIDs have changed on the TG. A database unaware player may accept and complete UIDs changed notifications as it may be able to detect some changes to the available media. However it should be noted that the UID counter value shall always be 0. Note that to refresh UID information after having received an EVENT_UIDS_CHANGED, the Media Player Virtual FileSystem may be browsed. If the Media Player Virtual FileSystem is browsed as reaction to the EVENT_UIDS_CHANGED again before browsing the Media Player Virtual FileSystem in order to informed about intermediate changes within the filesystem.	
Parameters	*param*	[in] Pointer to the BtSdkUIDSChangedStru structure specifies the UID Counter.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

6.3.4.3.13 Btsdk_AVRCP_ EventNowPlayingContentChanged

Prototype	BTINT32 Btsdk_AVRCP_EventNowPlayingContentChanged (PBtSdkNowPlayingContentChangedStru param);	
Description	The **Btsdk_AVRCP_EventNowPlayingContentChanged** function sends a CHANGED event notification to CT, when the content of the NowPlaying folder for the Addressed Player is changed. The Notification should not be completd if only the track has changed or the order of the tracks on the now playing list has changed. Note that to retrieve the content of the NowPlaying folder, the NowPlaying folder can be browsed. If the NowPlaying folder is browsed as reaction to the EVENT_NOW_PLAYING_CONTECT_CHANGED, the CT should register the EVENT_NOW_PLAYING_CONTENT_CHANGED again before browsing the NowPlaying folder in order to get informed about intermediate changes in that folder.	
Parameters	param	[in] Pointer to the BtSdkNowPlayingContentChangedStru structure.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

6.3.4.3.14 Btsdk_AVRCP_EventPlayPosChanged

Prototype	BTINT32 Btsdk_AVRCP_ EventPlayPosChanged (PBtSdkPlayPosChangedStru param);	
Description	The **Btsdk_AVRCP_EventPlayPosChanged** function sends a CHANGED event notification to CT. EVENT_PLAYBACK_POS_CHANGED shall be notified in the following conditions: TG has reached the registered playback Interval time. Changed PLAY STATUS. Changed Current Track.. Reached end or beginning of track.	
Parameters	*param*	[in] Pointer to the BtSdkPlayPosChangedStru structure.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

This function is called when TG send the position-value of player to CT. The callback event in CT get the postion- value to update the progress bar. The type of event is BTSDK_APP_EV_AVRCP_PLAYBACK_POS_CHANGED_NOTIF.

6.3.5 Serial Port Profile

6.3.5.1 Btsdk_InitCommObj

Prototype	BTINT32 Btsdk_InitCommObj (BTUINT8 com_idx, BTUINT16 svc_class);	
Description	The **Btsdk_InitCommObj** function initializes the COM port object.	
Parameters	*com_idx*	Integer that specifies the COM port to be initialized.
	svc_class	Type of the service record. It can be one of the values listed in the Table 2.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code of either BTSDK_ER_COM_INUSED or BTSDK_ER_INVALID_PARAMETER.	

Remarks

6.3.5.2 Btsdk_DeinitCommObj

Prototype	BTINT32 Btsdk_DeinitCommObj (BTUINT8 com_idx);	
Description	The **Btsdk_DeinitCommObj** function deletes the COM port designated by com_idx.	
Parameters	*com_idx*	Integer that specifies the COM port to be deleted.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code of BTSDK_ER_COM_INUSED.	

Remarks

6.3.5.3 Btsdk_GetClientPort

Prototype	BTINT16 Btsdk_GetClientPort(BTCONNHDL conn_hdl);	
Description	The **Btsdk_GetClientPort** function gets the client COM port of the SPP, DUN, and LAP connection.	
Parameters	*conn_hdl*	[in] Handle of the connection.
Return:	If the function succeeds, the return value is the ID number of COM port. If an error, the return value is 0.	

Remarks

Before calling Btsdk_GetClientPort, the local device must be enabled by a previous successful call to *Btsdk_StartBluetooth*.

6.3.5.4 Btsdk_GetAvailableExtSPPCOMPort

Prototype	BTUINT8 Btsdk_GetAvailableExtSPPCOMPort (BTBOOL bIsForLocalSPPService);	
Description	The **Btsdk_GetAvailableExtSPPCOMPort** function gets available COM port used for 128 bit spp.	
Parameters	*bIsForLocalSPPService*	[in] Notify BlueSoleil the usage of this COM port. BTSDK_TRUE: This COM port is used to register a local defined SPP-based service record. BTSDK_FALSE: This COM port is used to connect to an application defined SPP-based service record.
Return:	If there is a COM port available, the return value is the ID number of the serial port. If there is no COM port available, the return value is 0.	

Remarks

6.3.5.5 Btsdk_SearchAppExtSPPService

Prototype	BTUINT32 Btsdk_SearchAppExtSPPService (BTDEVHDL dev_hdl, PBtSdkAppExtSPPAttrStru psvc,);	
Description	The **Btsdk_SearchAppExtSPPService** function searches a remote device for the application-defined service.	
Parameters	*dev_hdl*	[in] Handle to the remote device to search for the specified service.
	psvc	[in/out] Pointer to a BtSdkAppExtSPPAttrStru structure. On input, it must specify the value of service_class_128. On output, rf_svr_chnl, svc_name and sdp_record_handle are set to the values retrieved during SDP transaction. com_index is ignored by this function.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code.	

Remarks

Before calling *Btsdk_SearchAppExtSPPService*, the local device must be enabled by a previous successful call to *Btsdk_StartBluetooth*.

6.3.5.6 Btsdk_ConnectAppExtSPPService

Prototype	BTUINT32 Btsdk_ConnectAppExtSPPService (BTDEVHDL dev_hdl, PBtSdkAppExtSPPAttrStru psvc, BTCONNHDL *conn_hdl);	
Description	The **Btsdk_ConnectAppExtSPPService** function connects to an application defined SPP-based service record.	
Parameters	*dev_hdl*	[in] Handle to the remote device to connect.
	psvc	[in/out] Pointer to a BtSdkAppExtSPPAttrStru structure. On input, it must specify the value of service_class_128, and may specify the value of com_index. If com_index is set to 0, you can use Btsdk_GetAvailableExtSPPCOMPort(BTSDK_FALSE) to get one com port. On output, rf_svr_chnl, svc_name and sdp_record_handle are set to the values retrieved during SDP transaction. If com_index provided by the application is 0, SDK will set it to the value assigned internally.
	conn_hdl	[out] Pointer to a BTCONNHDL variable. If connection created successfully, it will be set to the handle to the connection. Otherwise, it will be set to BTSDK_INVALID_HANDLE.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code.	

Remarks

Before calling *Btsdk_ConnectAppExtSPPService*, the local device must be enabled by a previous successful call to *Btsdk_StartBluetooth*.

Currently, both SPP client and server connections are combined with Bluetooth virtual serial ports

pre-installed in the OS. After SPP connection is created, the application can use the standard OS serial port I/O functions to transfer data over the SPP connection.

6.3.5.7 Btsdk_GetASerialNum

Prototype	BTUINT32 Btsdk_GetASerialNum();	
Description	The **Btsdk_GetASerialNum** function gets a currently available serial number of COM port.	
Parameters	*None*	
Return:	The return value is the currently available serial number of COM port.	

Remarks

6.3.5.8 Btsdk_PlugInVComm

Prototype	BOOL Btsdk_PlugInVComm (UINT serialNum, ULONG *comportNumber, UINT usageType, ULONG flag, DWORD dwTimeout);	
Description	The **Btsdk_PlugInVComm** function plugs in a currently available COM port.	
Parameters	*serialNum*	[in] Serial number of COM port which is return value of the function Btsdk_GetASerialNum
	comportNumber	[in/out] Pointer to buffer containing Com port number specified by OS
	usageType	[in] This parameter must be 1
	flag	[in] This parameter must be COMM_SET_RECORD|COMM_SET_USAGETYPE
	dwTimeout	[in] The timeouts of plugging in the serial port.
Return:	If the function succeeds, the return value is TRUE. If the function fails, the return value is FALSE	

Remarks

Before calling Btsdk_PlugInVComm, the Btsdk_GetASerialNum must be called to get the parameter of *serialNum*

After calling Btsdk_PlugInVComm, the Btsdk_InitCommObj should be called to initialize the COM port.

Flag description:

Value	Description
COMM_SET_RECORD	This macro indicates if the COM port is recorded by BlueSoleil.

COMM_SET_USAGETYPE	This macro is an identity of BlueSoleil designated by OS.

6.3.5.9 Btsdk_CommNumToSerialNum

Prototype	BTUINT32 Btsdk_CommNumToSerialNum (int comportNum);	
Description	The **Btsdk_CommNumToSerialNum** gets the serial number of COM port from COM port number.	
Parameters	*comportNum*	[in] Com port number specified by OS
Return:	The return value is a serial number of COM port.	

Remarks

6.3.5.10 Btsdk_PlugOutVComm

Prototype	void Btsdk_PlugOutVComm (UINT serialNum, ULONG flag);	
Description	The **Btsdk_PlugInVComm** function plugs out a COM port.	
Parameters	*serialNum*	[in] Serial number of COM port which is return value of the function Btsdk_GetASerialNum
	flag	[in] This parameter must be COMM_SET_RECORD
Return:		

Remarks

After calling Btsdk_PlugOutVComm, the Btsdk_DeinitCommObj should be called to delete the COM port.

6.3.6 Hands-free and Headset Profile

BlueSoleil SDK provides the same APIs for these two profiles.

6.3.6.1 General

6.3.6.1.1 Btsdk_RegisterHFPService

Prototype	BTSVCHDL Btsdk_RegisterHFPService(BTUINT8 *svc_name, BTUINT16 svc_class, BTUINT16 features);	
Description	The **Btsdk_RegisterHFPService** function registers a HFP or HEP service.	
Parameters	svc_name	[in] User friendly name of the new service. It shall be a null-terminated UTF-8 string. It can't be NULL. Its length shall be limited within BTSDK_SERVICENAME_MAXLENGTH, including the terminated '\0'.
	svc_class	[in] 16bit UUID specifies the service type. It can be one of: BTSDK_CLS_HANDSFREE, BTSDK_CLS_HANDSFREE_AG, BTSDK_CLS_HEADSET, BTSDK_CLS_HEADSET_AG.
	features	[in] A set of flags specifies the BRSF features supported by the new Hands-free HF or AG service. Its value is ignored If the service is of Headset HS or AG type.
Return:	The handle of the service.	

The ***features*** parameter can be binary combination of the following values:

Value	Description
BTSDK_AG_BRSF_3WAYCALL	Three-way calling
BTSDK_AG_BRSF_NREC	EC and NR function
BTSDK_AG_BRSF_BVRA	Voice recognition function

BTSDK_AG_BRSF_INBANDRING	In-band ring tone capability
BTSDK_AG_BRSF_BINP	Attach a number to a voice tag
BTSDK_AG_BRSF_REJECT_CALL	Ability to reject a call
BTSDK_AG_BRSF_ENHANCED_CALLSTATUS	Enhanced call status
BTSDK_AG_BRSF_ENHANCED_CALLCONTROL	Enhanced call control
BTSDK_AG_BRSF_EXTENDED_ERRORRESULT	Extended Error Result Codes
BTSDK_AG_BRSF_ALL	Support all the upper features
BTSDK_HF_BRSF_NREC	EC and/or NR function
BTSDK_HF_BRSF_3WAYCALL	Call waiting and 3-way calling
BTSDK_HF_BRSF_CLIP	CLI presentation capability
BTSDK_HF_BRSF_BVRA	Voice recognition activation
BTSDK_HF_BRSF_RMTVOLCTRL	Remote volume control
BTSDK_HF_BRSF_ENHANCED_CALLSTATUS	Enhanced call status
BTSDK_HF_BRSF_ENHANCED_CALLCONTROL	Enhanced call control
BTSDK_HF_BRSF_ALL	Support all the upper features

Remarks

This function MUST be called and the return value MUST be BTSDK_OK before any other HFP functions is called.

This function will enable both Hands-free and Headset services at the same time. But only one connection is allowed every time, no matter which side (local or remote application) initiates the connection. For example, if a connection between the local Hands-free AG and a remote Hands-free Unit is created, no more connections with other Hands-free Units or Headsets can be created until the previous connection is released.

6.3.6.1.2 Btsdk_UnregisterHFPService

Prototype	BTUINT32 Btsdk_UnregisterHFPService(BTSVCHDL svc_hdl);	
Description	The **Btsdk_UnregisterHFPService** function unregisters HFP service.	
Parameters	*svc_hdl*	The handle of the service.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code.	

Remarks

6.3.6.1.3 Btsdk_HFP_Callback

Prototype	typedef void (Btsdk_HFP_Callback)(BTCONNHDL hdl, BTUINT16 event, BTUINT8 *param, BTUINT16 len);	
Description	The **Btsdk_HFP_Callback** function prototype is the prototype of application defined callback function used to process Hands-free/Headset events.	
Parameters	hdl	[in] Handle to the HFP connection with a remote HF that is to send the call answered indication.
	event	[in] Event identifier.
	param	[in] First event parameters. It is usually a pointer to an evevt specific variable.
	len	[in] Specify the length, in bytes, of the string pointed to by the param, not including the terminated NULL.
Return:		

The ***event*** and ***param***:

event	param
BTSDK_HFP_EV_SPP_ESTABLISHED_IND	Btsdk_HFP_ConnInfoStru
BTSDK_HFP_EV_SLC_ESTABLISHED_IND	Btsdk_HFP_ConnInfoStru
BTSDK_HFP_EV_SLC_RELEASED_IND	Btsdk_HFP_ConnInfoStru
BTSDK_HFP_EV_STANDBY_IND	NULL
BTSDK_HFP_EV_ONGOINGCALL_IND	NULL
BTSDK_HFP_EV_RINGING_IND	BTUINT8: Specify the type of ring tone. 0 – Local ring tone; 1 – In-band ring tone
BTSDK_HFP_EV_OUTGOINGCALL_IND	NULL
BTSDK_HFP_EV_CALLHELD_IND	NULL
BTSDK_ HFP_EV_CALL_WAITING_IND	Btsdk_HFP_PhoneInfoStru
BTSDK_HFP_EV_TBUSY_IND	NULL

BTSDK_HFP_EV_GENERATE_INBAND_RINGTONE_IND	NULL
BTSDK_HFP_EV_TERMINATE_LOCAL_RINGTONE_IND	NULL
BTSDK_HFP_EV_VOICE_RECOGN_ACTIVATED_IND	NULL
BTSDK_HFP_EV_VOICE_RECOGN_DEACTIVATED_IND	NULL
BTSDK_HFP_EV_NETWORK_AVAILABLE_IND	NULL
BTSDK_HFP_EV_NETWORK_UNAVAILABLE_IND	NULL
BTSDK_HFP_EV_ROAMING_RESET_IND	NULL
BTSDK_HFP_EV_ROAMING_ACTIVE_IND	NULL
BTSDK_HFP_EV_SIGNAL_STRENGTH_IND	BTUINT8: The signal strength value.
BTSDK_HFP_EV_BATTERY_CHARGE_IND	BTUINT8: Battery charge indicator value.
BTSDK_HFP_EV_CHLDHELD_ACTIVATED_IND	NULL
BTSDK_HFP_EV_CHLDHELD_RELEASED_IND	NULL
BTSDK_HFP_EV_MICVOL_CHANGED_IND	BTUINT8: The gain value of microphone.
BTSDK_HFP_EV_SPKVOL_CHANGED_IND	BTUINT8: The speaker gain value.
BTSDK_HFP_EV_CURRENT_CALLS_REQ	NULL
BTSDK_HFP_EV_NETWORK_OPERATOR_FORMAT_REQ	NULL
BTSDK_HFP_EV_NETWORK_OPERATOR_REQ	NULL
BTSDK_HFP_EV_SUBSCRIBER_NUMBER_REQ	NULL
BTSDK_HFP_EV_VOICETAG_PHONE_NUM_REQ	NULL
BTSDK_HFP_EV_CUR_INDICATOR_VAL_REQ	NULL
BTSDK_HFP_EV_HF_DIAL_REQ	For a HFP-AG connection, it points to a buffer containing the phone number to dial; For a HSP-AG connection, it is set to NULL.
BTSDK_HFP_EV_HF_MEM_DIAL_REQ	BTUINT8*: Points to a buffer containing the memory location index.
BTSDK_HFP_EV_HF_LASTNUM_REDIAL_REQ	NULL
BTSDK_HFP_EV_MANUFACTURER_REQ	NULL

BTSDK_HFP_EV_MODEL_REQ	NULL
BTSDK_HFP_EV_NREC_DISABLE_REQ	NULL
BTSDK_HFP_EV_DTMF_REQ	BTUINT8: The DTMF code.
BTSDK_HFP_EV_ANSWER_CALL_REQ	BTUINT8: Specifies the type of the call to answer. It can be one of BTSDK_HFP_TYPE_INCOMING_CALL, BTSDK_HFP_TYPE_HELDINCOMING_CALL.
BTSDK_HFP_EV_CANCEL_CALL_REQ	BTUINT8: Specifies the type of the call to release. It can be one of BTSDK_HFP_TYPE_ALL_CALLS, BTSDK_HFP_TYPE_INCOMING_CALL, BTSDK_HFP_TYPE_HELDINCOMING_CALL, BTSDK_HFP_TYPE_OUTGOING_CALL, BTSDK_HFP_TYPE_ONGOING_CALL.
BTSDK_HFP_EV_HOLD_CALL_REQ	NULL
BTSDK_HFP_EV_REJECTWAITINGCALL_REQ	NULL
BTSDK_HFP_EV_ACPTWAIT_RELEASEACTIVE_REQ	BTUINT8:The value of idx is specified by AT+CHLD=1<idx>
BTSDK_HFP_EV_HOLDACTIVECALL_REQ	BTUINT8: The value of idx is specified by AT+CHLD=2<idx>
BTSDK_HFP_EV_ADD_ONEHELDCALL_2ACTIVE_REQ	NULL
BTSDK_HFP_EV_LEAVE3WAYCALLING_REQ	NULL
BTSDK_HFP_EV_AUDIO_CONN_ESTABLISHED_IND	BTUINT16: the SCO connection handle.
BTSDK_HFP_EV_AUDIO_CONN_RELEASED_IND	BTUINT16: the SCO connection handle.
BTSDK_HFP_EV_EXTEND_CMD_IND	BTUINT8*: Points to the buffer contains the full extended AT command including the ending <cr>, or extended result code, including the starting and ending <cr><lf>.
BTSDK_HFP_EV_PRE_SCO_CONNECTION_IND	Btsdk_AGAP_PreSCOConnIndStru
BTSDK_HFP_EV_SIGNAL_STRENGTH_IND	BTUINT8: The signal strength value.
BTSDK_HFP_EV_BATTERY_CHARGE_IND	BTUINT8: Battery charge indicator value.
BTSDK_HFP_EV_CHLDHELD_ACTIVATED_IND	
BTSDK_HFP_EV_CHLDHELD_RELEASED_IND	

BTSDK_HFP_EV_MICVOL_CHANGED_IND	BTUINT8: The gain value of microphone.
BTSDK_HFP_EV_SPKVOL_CHANGED_IND	BTUINT8: The gain value of speaker.
BTSDK_HFP_EV_ATCMD_RESULT	Btsdk_HFP_ATCmdResultStru
BTSDK_HFP_EV_CLIP_IND	Btsdk_HFP_PhoneInfoStru
BTSDK_HFP_EV_CURRENT_CALLS_IND	Btsdk_HFP_CLCCInfoStru
BTSDK_HFP_EV_NETWORK_OPERATOR_IND	Btsdk_HFP_COPSInfoStru
BTSDK_HFP_EV_SUBSCRIBER_NUMBER_IND	Btsdk_HFP_PhoneInfoStru
BTSDK_HFP_EV_VOICETAG_PHONE_NUM_IND	Btsdk_HFP_PhoneInfoStru
BTSDK_HFP_EV_SIGNAL_STRENGTH_IND	BTUINT8: The signal strength value.
BTSDK_HFP_EV_BATTERY_CHARGE_IND	BTUINT8: Battery charge indicator value.
BTSDK_HFP_EV_HF_MANUFACTURERID_IND	BTUINT8* - Manufacturer ID of the AG device, a null-terminated ASCII string.
BTSDK_HFP_EV_HF_MODELID_IND	BTUINT8* - Model ID of the AG device, a null-terminated ASCII string.

Remarks

If not specified in the upper table, the event parameters shall be ignored.

6.3.6.1.4 Btsdk_HFP_ExtendCmd

Prototype	BTUINT32 Btsdk_HFP_ExtendCmd(BTCONNHDL hdl, void *cmd, BTUINT16 len, BTUINT32 timeout);	
Description	The **Btsdk_HFP_ExtendCmd** function is called to transmit the extended command to AG/HF device. If it is an AT command, the application will receive BTSDK_HFP_EV_ATCMD_RESULT event after the remote AG responds to the command or the specified time expires. If it is a result code, the application will receive no confirms.	
Parameters	*hdl*	[in] Handle to the HF connection to send the command.
	cmd	[in] Pointer to the AT command or result code to be transmitted. It shall be an AT command if local device acts as HF/HS in the specified connection, including the ending <cr>. E.g., "AT+CGMM\r".It can be any bytes stream if local device acts as AG in the specified connection.
	len	[in] Size of the content stored in the cmd buffer. If local device acts as HF/HS in the specified connection, the length of the AT command shall exclude the terminated null. E.g. strlen("AT+CGMM\r").
	timeout	[in] Specifies the maximum time, in seconds, the lower HF entity will wait for the response to this command. If the time expires before the remote AG response to the command, the command execution will be considered to have failed. If timeout is 0, a default time value will be adopted.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code.	

Remarks

The hands-free profile uses a subset of AT commands and result codes from existing standards. The application may require transferring more AT commands and result codes. This function

provides the application this kind of ability.

6.3.6.2 Hands-free/Headset Audio Gateway (AG)

6.3.6.2.1 Btsdk_AGAP_APPRegCbk4ThirdParty

Prototype	BTUINT32 Btsdk_AGAP_APPRegCbk4ThirdParty(Btsdk_HFP_Callback *pfunc);	
Description	The **Btsdk_AGAP_APPRegCbk4ThirdParty** function registers an application-defined callback function used to process Hands-free/Headset AG messages created by the BlueSoleil.	
Parameters	*pfunc*	[in] Pointer to the callback function of Btsdk_HFP_Callback type. If p*func* is NULL, BlueSoleil will remove the callback information registered before.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code.	

Remarks

All messages of both Hands-free AG and Headset AG from BlueSolei are transferred to the applications using the same callback function. That is, if the application calls *Btsdk_AGAP_APPRegCbk4ThirdParty* twice to register different callback functions, the second callback function will replace the first one.

6.3.6.2.2 Btsdk_AGAP_AnswerCall

Prototype	BTUINT32 Btsdk_AGAP_AnswerCall(BTCONNHDL hdl, BTUINT8 mode);	
Description	The **Btsdk_AGAP_AnswerCall** function informs the HF that the AG has answered the incoming call.	
Parameters	*hdl*	[in] Handle to the HFP connection with a remote HF that is to send the call answered indication.
	mode	[in] Specify whether to setup SCO connection.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code.	

Remarks

The ***mode*** parameter can be one of these values

Value	Description
BTSDK_HFP_AG_PRIVATE_MODE	Do not setup SCO connection.
BTSDK_HFP_AG_HANDSFREE_MODE	Setup SCO connection.

6.3.6.2.3 Btsdk_AGAP_OriginateCall

Prototype	BTUINT32 Btsdk_AGAP_OriginateCall(BTCONNHDL hdl, BTUINT8 mode);	
Description	The **Btsdk_AGAP_OriginateCall** function informs the HF that the AG has originated a call. (This function can be called when the AG application starts to call the remote party after a successful voice recognition procedure.)	
Parameters	*hdl*	[in] Handle to the HFP connection with a remote HF that is to send the call answered indication.
	mode	[in] Specify whether to setup SCO connection.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code.	

Remarks

The ***mode*** parameter can be one of these values

Value	Description
BTSDK_HFP_AG_PRIVATE_MODE	Do not setup SCO connection.
BTSDK_HFP_AG_HANDSFREE_MODE	Setup SCO connection.

6.3.6.2.4 Btsdk_AGAP_CancelCall

Prototype	BTUINT32 Btsdk_AGAP_CancelCall(BTCONNHDL hdl, BTUINT8 type);	
Description	The **Btsdk_AGAP_CancelCall** function informs the HF that the AG has cancelled a call. (AG may reject an incoming call or terminate an outgoing call or release an ongoing call.)	
Parameters	*hdl*	[in] Handle to the HFP connection with a remote HF that is to send the call canceled indication.
	type	[in] Specifies the type of the call released.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code.	

Remarks

The ***type*** parameter can be one of these values

Value	Description
BTSDK_HFP_CANCELED_ALLCALL	AG has released all the existing calls (active, outgoing, waiting, holding).
BTSDK_HFP_CANCELED_CALLSETUP	AG has rejected a waiting call or terminated an outgoing call.
BTSDK_HFP_CANCELED_LASTCALL	AG has released the last active call.

6.3.6.2.5 Btsdk_AGAP_ChangeInbandRingSetting

Prototype	BTUINT32 Btsdk_AGAP_ChangeInbandRingSetting(BTCONNHDL hdl, BTUINT8 inband_ring);	
Description	The **Btsdk_AGAP_ChangeInbandRingSetting** function informs the HF device the new in-band ring tone setting.	
Parameters	*hdl*	[in] Handle to the HFP connection with a remote HF that is to send the call cancelled indication.
	inband_ring	[in] Specify whether the AG will provide the in-band ring tones or not. 0 -- The AG won't provide the in-band ring tones. 1 -- The AG will provide the in-band ring tones.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code.	

Remarks

6.3.6.2.6 Btsdk_AGAP_NetworkEvent

Prototype	BTUINT32 Btsdk_AGAP_NetworkEvent(BTCONNHDL hdl, BTUINT8 ev, void *param);	
Description	The **Btsdk_AGAP_NetworkEvent** function informs BlueSoleil that the AG application receives an event from the external network, e.g. a result code from the cellular network.	
Parameters	*hdl*	[in] Handle to the HFP connection with a remote HF that is to send the network indication.
	ev	[in] Event identifier
	param	[in] event parameter.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code.	

The ***event*** parameter can be one of these values,

Value	Description
BTSDK_AGAP_NETWORK_RMT_IS_BUSY	The remote called party is already in communication. For example, the answer to the ATD command is BUSY.
BTSDK_AGAP_NETWORK_ALERTING_RMT	The remote called party is reached and being alerted. For example, the answer to the ATD command is “0” (OK).

BTSDK_AGAP_NETWORK_INCOMING_CALL	The AG application receives an incoming call from the network. For example, “RING” (may be followed by a “+CLIP<number>”) is received. *param* is a pointer to a buffer that contains a NULL terminated ASCII string that specifies the phone number if it is available. *param* shall be set to NULL if the phone number is unavailable.
BTSDK_AGAP_NETWORK_RMT_ANSWER_CALL	The AG application detects that the remote called party has answered the call. For example, the answer to the AT+CLCC command is “+CLCC: 0, 0, …”.
BTSDK_AGAP_NETWORK_LINK_NOT_ESTABLISHED	The remote called party can’t be reached or the remote called party hang-up the ongoing call. For example, the AG application receives NO ANSWER, NO CARRIER, or NO DIALTONE.
BTSDK_AGAP_NETWORK_SVC_UNAVAILABLE	The AG application detects that the network service is unavailable.
BTSDK_AGAP_NETWORK_SVC_AVAILABLE	The AG application detects that the network service is available.
BTSDK_AGAP_NETWORK_SIGNAL_STRENGTH	
BTSDK_AGAP_NETWORK_ROAMING_RESET	
BTSDK_AGAP_NETWORK_ROAMING_ACTIVE	

1. BTSDK_AGAP_NETWORK_RMT_IS_BUSY: Pointer to a BTUINT8 variable specifies which call is canceld by this busy event. Its value can be one of BTSDK_HFP_CANCELED_LASTCALL, BTSDK_HFP_CANCELED_CALLSETUP and BTSDK_HFP_CANCELED_CALLHELD).
The default value is BTSDK_HFP_CANCELED_LASTCALL if param is set to NULL.

2. BTSDK_AGAP_NETWORK_INCOMING_CALL: Pointer to a Btsdk_HFP_PhoneInfoStru structure contains the phone number.

3. BTSDK_AGAP_NETWORK_SIGNAL_STRENGTH: Pointer to a BTUINT8 variable specifies the signal strength. Its range is from 0 to 5.

4. For all the other events, the param is ignored and should be NULL.

6.3.6.2.7 Btsdk_AGAP_VoiceRecognitionReq

Prototype	BTUINT32 Btsdk_AGAP_VoiceRecognitionReq(BTCONNHDL hdl, BTUINT8 param);	
Description	The **Btsdk_AGAP_VoiceRecognitionReq** function informs the HF device that AG has activated or deactivated the voice recognition.	
Parameters	hdl	[in] Handle to the HFP connection with a remote AG that is to attach the voice tag.
	param	[in] 1=enable, 0=disable.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code.	

Remarks

6.3.6.2.8 Btsdk_AGAP_VoiceTagPhoneNumRsp

Prototype	BTUINT32 Btsdk_AGAP_VoiceTagPhoneNumRsp(BTCONNHDL hdl, void *phone_num, BTUINT8 len);	
Description	The **Btsdk_AGAP_VoiceTagPhoneNumRsp** function specifies a phone number to be attached to a voice tag in the HF side.	
Parameters	hdl	[in] Handle to the HFP connection with a remote HF that is to send the indication.
	phone_num	[in] Pointer to a buffer contains the phone number string .
	len	[in] Length of the string, not including the terminated null character.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code.	

Remarks

6.3.6.2.9 Btsdk_AGAP_DialRsp

Prototype	BTUINT32 Btsdk_AGAP_DialRsp(BTCONNHDL hdl, BTUINT8 err_code);	
Description	The **Btsdk_AGAP_DialRsp** function responds the AG dialing status, which HF device requested in ATD, ATD> and AT+BLDN.	
Parameters	hdl	[in] Handle to the HFP connection with a remote HF that is to send the indication.
	status	[in] Specifies the result of dialing operation. It shall be one of BTSDK_HFP_OK, CME error codes and standard error result codes. BTSDK_HFP_OK - The operation is successful. Otherwise, CME error codes or standard error result codes.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code.	

Remarks

6.3.6.2.10 Btsdk_AGAP_HoldIncomingCall

Prototype	BTUINT32 Btsdk_AGAP_HoldIncomingCall(BTCONNHDL hdl);	
Description	The **Btsdk_AGAP_HoldIncomingCall** function informs HF device that AG has put the incoming call on hold.	
Parameters	*hdl*	[in] Handle to the HFP connection with a remote HF that is to send the indication.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code.	

Remarks

6.3.6.2.11 Btsdk_AGAP_AcceptHeldIncomingCall

Prototype	BTUINT32 Btsdk_AGAP_AcceptHeldIncomingCall(BTCONNHDL hdl, BTUINT8 mode);	
Description	The **Btsdk_AGAP_AcceptHeldIncomingCall** function informs HF device that AG has accepted the held incoming call.	
Parameters	*hdl*	[in] Handle to the HFP connection with a remote HF that is to send the indication.
	mode	[in] Specify whether to setup SCO connection.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code.	

Remarks

The ***mode*** parameter can be one of these values

Value	Description
BTSDK_HFP_AG_PRIVATE_MODE	Do not setup SCO connection.
BTSDK_HFP_AG_HANDSFREE_MODE	Setup SCO connection.

6.3.6.2.12 Btsdk_AGAP_RejectHeldIncomingCall

Prototype	BTUINT32 Btsdk_AGAP_RejectHeldIncomingCall(BTCONNHDL hdl);	
Description	The **Btsdk_AGAP_RejectHeldIncomingCall** function informs HF device that AG has rejected the held incoming call.	
Parameters	*hdl*	[in] Handle to the HFP connection with a remote HF that is to send the indication.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code.	

Remarks

6.3.6.2.13 Btsdk_AGAP_NetworkOperatorRsp

Prototype	BTUINT32 Btsdk_AGAP_NetworkOperatorRsp(BTCONNHDL hdl, PBtsdk_HFP_COPSInfoStru op_info);	
Description	The **Btsdk_AGAP_NetworkOperatorRsp** function is called to respond to the AT+COPS? command..	
Parameters	*hdl*	[in] Handle to the HFP connection with a remote HF that is to send the indication.
	op_info	[in] Pointer to the Btsdk_HFP_COPSInfoStru structure contains the operator information. If the operator information is unavailable, op_info shall be NULL.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code.	

Remarks

6.3.6.2.14 Btsdk_AGAP_SubscriberNumberRsp

Prototype	BTUINT32 Btsdk_AGAP_SubscriberNumberRsp(BTCONNHDL hdl, PBtsdk_HFP_PhoneInfoStru usr_info, BTUINT8 complete);	
Description	The **Btsdk_AGAP_SubscriberNumberRsp** function is called to respond to the AT+CNUM command. If there are multiple subscriber numbers available, this function shall only be called once for a number.	
Parameters	*hdl*	[in] Handle to the HFP connection with a remote HF that is to send the indication.
	usr_info	[in] Pointer to the Btsdk_HFP_PhoneInfoStru structure containing one subscriber number's information. The type, service, num_len and number member of this structure shall be set to the proper value. All the other members are ignored.
	complete	[in] Specify whether it is the last subscriber number. 0 - There are still more numbers to be sent. 1 - This is the last number. "\r\nOK\r\n" won't be sent to the HF device until complete is set to 1.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code.	

Remarks

If no subscriber number information is available, usr_info shall be set to NULL and complete shall be set to 1.

6.3.6.2.15 Btsdk_AGAP_CurrentCallRsp

Prototype	BTUINT32 Btsdk_AGAP_CurrentCallRsp(BTCONNHDL hdl, PBtsdk_HFP_CLCCInfoStru call_info, BTUINT8 complete);	
Description	The **Btsdk_AGAP_CurrentCallRsp** function is called to respond to the AT+CLCC command.If there are multiple concurrent calls, this function shall only be called once for a call.	
Parameters	*hdl*	[in] Handle to the HFP connection with a remote HF that is to send the indication.
	call_info	[in] Pointer to the Btsdk_HFP_CLCCInfoStru structure containing information of one of the current call.
	complete	[in] Specify whether it is the last available call. 0 - There are still more is called to be sent. 1 - This is the last call. "\r\nOK\r\n" won't be sent to the HF device until is_last is set to 1.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code.	

Remarks

If no calls are available, call_info shall be set to NULL and complete shall be set to 1.

6.3.6.2.16 Btsdk_AGAP_ManufacturerIDRsp

Prototype	BTUINT32 Btsdk_AGAP_ManufacturerIDRsp(BTCONNHDL hdl, BTINT8 *mid, BTUINT16 len);	
Description	The **Btsdk_AGAP_ManufacturerIDRsp** function is called to transmit response to AT+CGMI command.	
Parameters	*hdl*	[in] Handle to the HFP connection with a remote HF that is to send the indication.
	mid	[in] Pointer to the buffer containing the manufacturer identification. It shall be an ASCII text string.
	len	[in] Specify the length, in bytes, of the string pointed to by the mid, not including the terminated NULL.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code.	

Remarks

6.3.6.2.17 Btsdk_AGAP_ ModelIDRsp

Prototype	BTUINT32 Btsdk_AGAP_ ModelIDRsp (BTCONNHDL hdl, BTINT8 *mid, BTUINT16 len);	
Description	The **Btsdk_AGAP_ ModelIDRsp** function is called to transmit response to AT+CGMM command.	
Parameters	*hdl*	[in] Handle to the HFP connection with a remote HF that is to send the indication.
	mid	[in] Pointer to the buffer contains the model identification.It shall be an ASCII text string.
	len	[in] Specify the length, in bytes, of the string pointed to by the mid, not included the terminated NULL.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code.	

Remarks

6.3.6.2.18 Btsdk_AGAP_SendBatteryChargeIndicator

Prototype	BTUINT32 Btsdk_AGAP_SendBatteryChargeIndicator(BTCONNHDL hdl, BTUINT8 indicator);	
Description	The **Btsdk_AGAP_SendBatteryChargeIndicator** function is called to transmit current battery charge indicator.	
Parameters	*hdl*	[in] Handle to the HFP connection with a remote HF that is to send the indication.
	indicator	[in] Specify the current battery charge indicator value. Range: 0 - 5.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code.	

Remarks

6.3.6.2.19 Btsdk_AGAP_ SendErrorMessage

Prototype	BTUINT32 Btsdk_AGAP_SendErrorMessage(BTCONNHDL hdl, BTUINT8 err_code);	
Description	The **Btsdk_AGAP_SendErrorMessage** function is called to transmit "+CME ERROR" result code to the HF.	
Parameters	*hdl*	[in] Handle to the HFP connection with a remote HF that is to send the indication.
	err_code	[in] Specify the error code. It shall be one of CME error codes.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code.	

Remarks

6.3.6.2.20 Btsdk_AGAP_SetSpkVol

Prototype	BTUINT32 Btsdk_AGAP_SetSpkVol(BTCONNHDL hdl, BTUINT8 spk_vol);	
Description	The **Btsdk_AGAP_SetSpkVol** function is called to set the speaker volume of HF device..	
Parameters	*hdl*	[in] Handle to the HFP connection with a remote HF that is to send the indication.
	spk_vol	[in] The speaker volume level. Range: 0 - 15
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code.	

Remarks

6.3.6.2.21 Btsdk_AGAP_SetMicVol

Prototype	BTUINT32 BTUINT32 Btsdk_AGAP_SetMicVol(BTCONNHDL hdl, BTUINT8 mic_vol);	
Description	The **Btsdk_AGAP_SetMicVol** function is called to set the microphone volume of HF device.	
Parameters	*hdl*	[in] Handle to the HFP connection with a remote HF that is to send the indication.
	mic_vol	[in] The microphone volume level. Range: 0 - 15
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code.	

Remarks

6.3.6.2.22 Btsdk_AGAP_SetCurIndicatorVal

Prototype	BTUINT32 Btsdk_AGAP_SetCurIndicatorVal(BTCONNHDL hdl, PBtsdk_HFP_CINDInfoStru indicators);	
Description	The **Btsdk_AGAP_SetCurIndicatorVal** function sets the current call/service indicator value in order to synchronize the state with the HF during the service level connection establishing procedure with the HF.	
Parameters	*hdl*	[in] Handle to the HFP connection with a remote HF that is to send the indication.
	indicators	[in] Pointer to the Btsdk_HFP_CINDInfoStru containing the current value of the HFP defined indicators.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code.	

Remarks

6.3.6.2.23 Btsdk_AGAP_AudioConnTrans

Prototype	BTUINT32 Btsdk_AGAP_AudioConnTrans(BTCONNHDL hdl);	
Description	The **Btsdk_AGAP_AudioConnTrans** function transfers the audio path of the ongoing call from or towards the HF.	
Parameters	*hdl*	[in] Handle to the HFP connection with a remote HF that is to transfer the audio connection.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code.	

Remarks

If there is no audio connection established between the AG and the HF, this function transfers the audio path of the ongoing call from the AG towards the HF. If the audio connection already exists, this function transfers the audio path of the ongoing call from the HF towards the AG.

6.3.6.2.24 Btsdk_AGAP_GetAGState

Prototype	BTUINT32 Btsdk_AGAP_GetAGState(BTUINT16* agstate);	
Description	The **Btsdk_AGAP_GetAGState** function gets current AG's state.	
Parameters	*agstate*	[out] Pointer to the variable which indicates current AG's state.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code.	

The ***agstate*** member can be one of these values.

Value	Description
BTSDK_AGAP_ST_IDLE	Before service level connection is established.
BTSDK_AGAP_ST_STANDBY	Service level connection is established.
BTSDK_AGAP_ST_RINGING	Ringing state.
BTSDK_AGAP_ST_OUTGOINGCALL	Outgoing call state.
BTSDK_AGAP_ST_ONGOINGCALL	Ongoing call state.
BTSDK_AGAP_ST_BVRA	Voice recognition is ongoing.
BTSDK_AGAP_ST_VOVG	SCO link doesn't exist between AG and HF while a call is ongoing.
BTSDK_AGAP_ST_HELDINCOMINGCALL	the incoming call is held
BTSDK_AGAP_ST_THREEWAYCALLING	three way calling

Remarks

6.3.6.2.25 Btsdk_AGAP_CurrentCallSync

Prototype	BTUINT32 Btsdk_AGAP_CurrentCallSync(BTCONNHDL hdl, PBtsdk_HFP_CLCCInfoStru call_info, BTUINT8 complete);	
Description	The **Btsdk_AGAP_CurrentCallSync** function is used to tell the lower HFP AG module current existing phone calls. It is different from the Btsdk_AGAP_CurrentCallRsp that it would not send any result code to the remote HF device. If there are multiple concurrent calls, this function shall only be called once for a call. If no calls are available, *call_info* shall be set to NULL and the *complete* shall be set to 1.	
Parameters	*hdl*	[in] Handle to the local HFP AG entity.
	call_info	[in] Pointer to the Btsdk_HFP_CLCCInfoStru structure contains information of one of the current call.
	complete	[in] Specify whether it is the last available call. 0 - There are still more calls to be synchronize. 1 - This is the last call.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code.	

Remarks

6.3.6.2.26 Btsdk_AGAP_3WayCallingHandler

Prototype	BTUINT32 Btsdk_AGAP_3WayCallingHandler(BTCONNHDL hdl, BTUINT16 op_code, BTUINT8 idx);	
Description	The **Btsdk_AGAP_3WayCallingHandler** function calls by the AG application to notify the HF current 3way-calling status. It is called after the AG application sends AT+CHLD=<n> command to the mobile network. The application should first call Btsdk_AGAP_CurrentCallSync to synchronize with the lower HFP AG module with the current available call list. Then it sends AT+CHLD=<n> command to the mobile network. After it got "OK" from the network, it calls Btsdk_HFAP_3WayCallingHandler.	
Parameters	*hdl*	[in] Handle to the HFP connection with a remote HF that is to handle the 3way-calling.
	op_code	[in] Operation code for the 3way-calling. It can be one of the: BTSDK_HFP_CMD_CHLD_0, BTSDK_HFP_CMD_CHLD_1, BTSDK_HFP_CMD_CHLD_2, BTSDK_HFP_CMD_CHLD_3, BTSDK_HFP_CMD_CHLD_4.
	idx	[in] Specify the call to be handled separately. If op_code is one of BTSDK_HFP_CMD_CHLD_0, BTSDK_HFP_CMD_CHLD_3 and BTSDK_HFP_CMD_CHLD_4, idx is ignored and shall be set to 0. If op_code is BTSDK_HFP_CMD_CHLD_1, a none-zero idx specify the call to released; a zero idx force to release all active calls if any exist. If op_code is BTSDK_HFP_CMD_CHLD_2, a none-zero idx specify the call not to be placed on hold; a zero idx force to hold all active calls if any exist.
Return:	BTSDK_TRUE if the specified status is set. BTSDK_FALSE if the specified status is not set.	

Remarks

The op_code along with idx determines the AT command the application sends to the mobile network.

The ***op_code*** can be one of these values.

Value	idx	AT Command
BTSDK_HFP_CMD_CHLD_0	0	AT+CHLD=0
BTSDK_HFP_CMD_CHLD_1		AT+CHLD=1
BTSDK_HFP_CMD_CHLD_1	>0	AT+CHLD=1<idx>
BTSDK_HFP_CMD_CHLD_2	0	AT+CHLD=2
BTSDK_HFP_CMD_CHLD_2	>0	AT+CHLD=2<idx>
BTSDK_HFP_CMD_CHLD_3	0	AT+CHLD=3
BTSDK_HFP_CMD_CHLD_4	0	AT+CHLD=4

6.3.6.2.27 Btsdk_AGAP_IsAudioConnExisted

Prototype	BTUINT32 Btsdk_AGAP_IsAudioConnExisted(BTBOOL* audioconn);	
Description	The **Btsdk_AGAP_IsAudioConnExisted** function judges whether SCO connection is established.	
Parameters	*audioconn*	[out] status of SCO connection BTSDK_TRUE: SCO is established BTSDK_FALSE: SCO is released
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code.	

Remarks

6.3.6.2.28 Btsdk_AGAP_SetDialHandlerFlag

Prototype	BTBOOL Btsdk_AGAP_SetDialHandlerFlag (BTBOOL bFlag);	
Description	The **Btsdk_AGAP_SetDialHandlerFlag** function sets whether the HFAG dial indication is handled by user application or not.	
Parameters	*bFlag*	[in] status of handling HFAG dial indication BTSDK_TRUE: Applications need to handle HFAG dial indication. BTSDK_FALSE: Applications don't care about HFAG dial indication anymore.
Return:	If the function succeeds, the return value is BTSDK_TRUE. If the function fails, the return value is BTSDK_FALSE.	

Remarks

This function must be called immediately after the callback event BTSDK_APP_EV_AGAP_HF_AVAILABLE_IND is received by application, and it must be called every time after this callback event is received.

After calling this function, when the BTSDK_APP_EV_AGAP_HF_LASTNUM_REDIAL_IND, BTSDK_APP_EV_AGAP_HF_MEM_DIAL_IND or BTSDK_APP_EV_AGAP_HF_DIAL_IND is received, the function **Btsdk_DialRsp** has to be called to imform the dialing result to BlueSoleil.

6.3.6.3 Hands-free Unit/Headset (HF/HS)

6.3.6.3.1 Btsdk_HFAP_APPRegCbk4ThirdParty

Prototype	void Btsdk_HFAP_APPRegCbk4ThirdParty (Btsdk_HFP_Callback *pfunc);	
Description	The **Btsdk_HFAP_APPRegCbk4ThirdParty** function registers an application-defined callback function used to process HF/HS messages created by the BlueSoleil.	
Parameters	*pfunc*	[in] Pointer to the callback function of Btsdk_HFP_Callback type. If p*func* is NULL, BlueSoleil will remove the callback information registered before.
Return:		

Remarks

All messages of both HF and HS from BlueSoleil are transferred to the application using the same callback function. That is, if the application calls *Btsdk_HFAP_APPRegCbk4ThirdParty* twice to register different callback functions, the second callback function will replace the first one.

6.3.6.3.2 Btsdk_HFAP_AnswerCall

Prototype	BTUINT32 Btsdk_HFAP_AnswerCall(BTCONNHDL hdl);	
Description	The **Btsdk_HFAP_AnswerCall** function informs the AG that the HF has been answered the incoming call.	
Parameters	hdl	[in] Handle to the HFP connection with a remote AG that is to answer the call.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code.	

Remarks

6.3.6.3.3 Btsdk_HFAP_CancelCall

Prototype	BTUINT32 Btsdk_HFAP_CancelCall(BTCONNHDL hdl);	
Description	The **Btsdk_HFAP_CancelCall** function informs the AG that the HF has cancelled a call. (HF may reject an incoming call or terminate an outgoing call or release an ongoing call.)	
Parameters	hdl	[in] Handle to the HFP connection with a remote AG that is to cancel the call.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code.	

Remarks

6.3.6.3.4 Btsdk_HFAP_LastNumRedial

Prototype	BTUINT32 Btsdk_HFAP_LastNumRedial(BTCONNHDL hdl);	
Description	The **Btsdk_HFAP_LastNumRedial** function instructs the AG to redial the last dialed number.	
Parameters	hdl	[in] Handle to the HFP connection with a remote AG that is to dial the number.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code.	

Remarks

6.3.6.3.5 Btsdk_HFAP_MemNumDial

Prototype	BTUINT32 Btsdk_HFAP_MemNumDial(BTCONNHDL hdl, void *mem_location, BTUINT16 len);	
Description	The **Btsdk_HFAP_MemNumDial** function instructs the AG to dial the phone number stored in the AG memory location given by a specific index.	
Parameters	hdl	[in] Handle to the HFP connection with a remote AG that is to dial the number.
	mem_location	[in] Pointer to a buffer contains the index string that specifies the AG memory location.
	len	[in] Length of the string, not including the terminated null character.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code.	

Remarks

6.3.6.3.6 Btsdk_HFAP_Dial

Prototype	BTUINT32 Btsdk_HFAP_Dial (void* phone_num, BTUINT16 len);	
Description	The **Btsdk_HFAP_Dial** function instructs the AG to dial the provided phone number.	
Parameters	*hdl*	[in] Handle to the HFP connection with a remote AG that is to dial the number.
	phone_num	[in] Pointer to a buffer contains the phone number string.
	len	[in] Length of the string, not including the terminated null character.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code.	

Remarks

6.3.6.3.7 Btsdk_HFAP_VoiceRecognitionReq

Prototype	BTUINT32 Btsdk_HFAP_VoiceRecognitionReq(BTCONNHDL hdl, BTUINT8 param);	
Description	The **Btsdk_HFAP_VoiceRecognitionReq** function requests the AG to activate or deactivate the voice recognition procedure.	
Parameters	*hdl*	[in] Handle to the HFP connection with a remote AG that is to activate or deactivate voice recognition.
	param	[in] 1=enable, 0=disable.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code.	

6.3.6.3.8 Btsdk_HFAP_3WayCallingHandler

Prototype	BTUINT32 Btsdk_HFAP_3WayCallingHandler(BTCONNHDL hdl, BTUINT16 op_code, BTUINT8 idx);	
Description	The **Btsdk_HFAP_3WayCallingHandler** function is called to handle the 3way-calling. It sends AT+CHLD=<n> command to the remote AG. <n> is determined by the values of op_code and idx.	
Parameters	*hdl*	[in] Handle to the HFP connection with a remote AG that is to handle the 3way-calling.
	op_code	[in] Specify the call to be handled separately. If op_code is one of BTSDK_HFP_CMD_CHLD_0, BTSDK_HFP_CMD_CHLD_3 and BTSDK_HFP_CMD_CHLD_4, idx is ignored and shall be set to 0. If op_code is BTSDK_HFP_CMD_CHLD_1, a none-zero idx specify the call to released; a zero idx force to release all active calls if any exist. If op_code is BTSDK_HFP_CMD_CHLD_2, a none-zero idx specify the call not to be placed on hold; a zero idx force to hold all active calls if any exist.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code.	

Remarks

The ***op_code*** of this function is the similar as the one of Btsdk_HFAP_3WayCallingHandler.

6.3.6.3.9 Btsdk_HFAP_DisableNREC

Prototype	BTUINT32 Btsdk_HFAP_DisableNREC(BTCONNHDL hdl);	
Description	The **Btsdk_HFAP_DisableNREC** function to request AG to disable NREC function.	
Parameters	*hdl*	[in] Handle to the HFP connection with a remote AG that is to disable NREC function.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code.	

Remarks

6.3.6.3.10 Btsdk_HFAP_TxDTMF

Prototype	BTUINT32 Btsdk_HFAP_TxDTMF(BTCONNHDL hdl, BTUINT8 chr);	
Description	The **Btsdk_HFAP_TxDTMF** function is called to instruct AG to transmit the specific DTMF code.	
Parameters	*hdl*	[in] Handle to the HFP connection with a remote AG that is to transmit the DTMF code.
	chr	[in] The DTMF character.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code.	

6.3.6.3.11 Btsdk_HFAP_SetSpkVol

Prototype	BTUINT32 Btsdk_HFAP_SetSpkVol(BTCONNHDL hdl, BTUINT8 spk_vol);	
Description	The **Btsdk_HFAP_SetSpkVol** function informs the remote AG that the speaker volume of the HF has been changed.	
Parameters	hdl	[in] Handle to the HFP connection with a remote AG that is to set the speaker volume.
	spk_vol	[in] The current speaker volume level. Range from 0 to 15. 0 = minimum gain; 15 = maximum gain.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code.	

6.3.6.3.12 Btsdk_HFAP_SetMicVol

Prototype	BTUINT32 Btsdk_HFAP_SetMicVol(BTCONNHDL hdl, BTUINT8 mic_vol);	
Description	The **Btsdk_HFAP_SetMicVol** function is called to inform AG that the microphone volume of HF device has been changed.	
Parameters	hdl	[in] Handle to the HFP connection with a remote AG that is to set the microphone volume.
	mic_vol	[in] The current micphone volume level. Range from 0 to 15. 0 = minimum gain; 15 = maximum gain.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code.	

6.3.6.3.13 Btsdk_HFAP_VoiceTagPhoneNumReq

Prototype	BTUINT32 Btsdk_HFAP_VoiceTagPhoneNumReq(BTCONNHDL hdl);	
Description	The **Btsdk_HFAP_VoiceTagPhoneNumReq** function is called to request AG to enter a phone number to be attached to the HF device's voice-tag, which is used for voice recognition. The phone number will be returned by the BTSDK_HFP_EV_VOICETAG_PHONE_NUM_IND event.	
Parameters	*hdl*	[in] Handle to the HFP connection with a remote AG that is to attach the voice tag.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code.	

Remarks

The phone number provided by the remote AG will be sent to the HF application through the BTSDK_APP_EV_HFAP_VOICETAG_PHONE_NUM_RSP message.

6.3.6.3.14 Btsdk_ HFAP_GetManufacturerID

Prototype	BTUINT32 Btsdk_HFAP_GetManufacturerID(BTCONNHDL hdl, BTUINT8 *manufacturer_id, BTUINT16 *id_len);	
Description	The **Btsdk_ HFAP_ManufacturerIDRsp** function is called to get the manufacturer ID information.	
Parameters	*hdl*	[in] Handle to the HFP connection with a remote AG that is to get the manufacturer ID.
	manufacturer_id	[out]: The buffer to store manufacture ID, If it is NULL, the *id_len should be set as 0; Otherwise, the *id_len specifies the manufacturer_id buffer size
	id_len	[in]: The size of the manufacturer_id buffer [out]: The real length of the manufacturer ID, including the terminated null character is returned.
Return:	If manufacturer ID is got, the return value is BTSDK_OK. If the return value is BTSDK_ER_FUNCTION_NOTSUPPORT, it indicates that the function failed to get the manufacturer ID of the AG.	

Remarks

6.3.6.3.15 Btsdk_ HFAP_GetModelID

Prototype	BTUINT32 Btsdk_HFAP_GetModelID(BTCONNHDL hdl, BTUINT8 *model_id, BTUINT16 *id_len);	
Description	The **Btsdk_HFAP_GetModelID** function is called to get the model nformation.	
Parameters	*hdl*	[in] Handle to the HFP connection with a remote HF that is to send the indication.
	model_id	[out] The buffer used to store model D, If it is NULL, the *id_len should be set as 0; Otherwise, the *id_len specifies the model_id buffer size.
	id_len	[in/out] The size of the model_id buffer (Input). On output, returns the real length of the model ID, including the terminated null charater.
Return:	If model ID is got, the return value is BTSDK_OK. If the return value is BTSDK_ER_FUNCTION_NOTSUPPORT, it indicates that the function failed to get the model ID of the AG.	

Remarks

6.3.6.3.16 Btsdk_HFAP_AudioConnTrans

Prototype	BTUINT32 Btsdk_HFAP_AudioConnTrans(BTCONNHDL hdl)	
Description	The **Btsdk_HFAP_AudioConnTrans** function is called to transfer the audio connection.	
Parameters	*hdl*	[in] Handle to the HFP connection with a remote AG that is to transfer the audio connection.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code.	

Remarks

6.3.6.3.17 Btsdk_HFAP_NetworkOperatorReq

Prototype	BTUINT32 Btsdk_HFAP_NetworkOperatorReq(BTCONNHDL hdl);	
Description	The **Btsdk_HFAP_NetworkOperatorReq** function is called to request for the network operator name of the AG device.The operator name will be returned by the BTSDK_HFP_EV_NETWORK_OPERATOR_IND event.	
Parameters	hdl	[in] Handle to the HFP connection with a remote AG that is to get the operator name.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code.	

Remarks

6.3.6.3.18 Btsdk_HFAP_SetExtendedErrors

Prototype	BTUINT32 Btsdk_HFAP_SetExtendedErrors(BTCONNHDL hdl, BTUINT8 enable);	
Description	The **Btsdk_HFAP_SetExtendedError**s function is called to enable the Extended Audio Gateway Error Result Code in the AG.	
Parameters	hdl	[in] Handle to the HFP connection with a remote AG that is to enable the extended error result code.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code.	

Remarks

6.3.6.3.19 Btsdk_HFAP_GetResponseHoldStatus

Prototype	BTUINT32 Btsdk_HFAP_GetResponseHoldStatus(BTCONNHDL hdl)	
Description	The **Btsdk_HFAP_GetResponseHoldStatus** function is called to query the current Response and Hold status of the AG. If the AG responds with "+BTRH:0", the application will be informed by the BTSDK_HFP_EV_CALLHELD_IND event.	
Parameters	hdl	[in] Handle to the HFP connection with a remote AG that is to query the Response and Hold status.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code.	

Remarks

6.3.6.3.20 Btsdk_HFAP_HoldIncomingCall

Prototype	BTUINT32 Btsdk_HFAP_HoldIncomingCall(BTCONNHDL hdl)	
Description	The **Btsdk_HFAP_HoldIncomingCall** function is called to inform the AG to hold the incoming call.If the AG responds with "+BTRH:0", the application will be informed by the BTSDK_HFP_EV_CALLHELD_IND event.	
Parameters	hdl	[in] Handle to the HFP connection with a remote AG that is to hold the incoming call.
Return:	BTSDK_OK if the request is sent to the AG. Other for error code. No events are generated in this case.	

Remarks

6.3.6.3.21 Btsdk_HFAP_AcceptHeldIncomingCall

Prototype	BTUINT32 Btsdk_HFAP_AcceptHeldIncomingCall(BTCONNHDL hdl)	
Description	The **Btsdk_HFAP_AcceptHeldIncomingCall** function is called to inform the AG to accept the held incoming call. If the AG responds with "+BTRH:1", the application will be informed by the BTSDK_HFP_EV_ONGOINGCALL_IND event.	
Parameters	hdl	[in] Handle to the HFP connection with a remote AG that is to accept the held call.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code.	

Remarks

6.3.6.3.22 Btsdk_HFAP_RejectHeldIncomingCall

Prototype	BTUINT32 Btsdk_HFAP_RejectHeldIncomingCall(BTCONNHDL hdl);	
Description	The **Btsdk_HFAP_RejectHeldIncomingCall** function is called to inform the AG to reject the held incoming call. If the AG responds with "+BTRH:2", the application will be informed by the BTSDK_HFP_EV_STANDBY_IND event.	
Parameters	hdl	[in] Handle to the HFP connection with a remote AG that is to reject the held call.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code.	

Remarks

6.3.6.3.23 Btsdk_HFAP_GetSubscriberNumber

Prototype	BTUINT32 Btsdk_HFAP_GetSubscriberNumber(BTCONNHDL hdl)	
Description	The **Btsdk_HFAP_GetSubscriberNumber** function is called to get the subscriber number information of AG. The subscriber number will be returned by the BTSDK_HFP_EV_SUBSCRIBER_NUMBER_IND event. If the AG responds with one of "OK", "ERROR" and "+CMER:" or the local timer expired before receiving the upper result code from the AG, the event BTSDK_HFP_EV_ATCMD_RESULT is reported to the application.	
Parameters	hdl	[in] Handle to the HFP connection with a remote AG that is to get the number.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code.	

Remarks

6.3.6.3.24 Btsdk_HFAP_GetCurrentCalls

Prototype	BTUINT32 Btsdk_HFAP_GetCurrentCalls(BTCONNHDL hdl)	
Description	The **Btsdk_HFAP_GetCurrentCalls** function is called to query the list of current calls. Information of each existing call will be returned by a BTSDK_HFP_EV_CURRENT_CALLS_IND event.	
Parameters	hdl	[in] Handle to the HFP connection with a remote AG that is to get the call list.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code.	

Remarks

6.3.6.3.25 Btsdk_HFAP_GetAGFeatures

Prototype	BTUINT32 Btsdk_HFAP_GetAGFeatures(BTCONNHDL hdl)	
Description	The **Btsdk_HFAP_GetAGFeatures** function is called to query the features of the remote AG.	
Parameters	hdl	[in] Handle to the HFP connection with a remote AG that is to get the features.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code.	

Remarks

6.3.6.3.26 Btsdk_HFAP_GetCurrHFState

Prototype	BTUINT32 Btsdk_HFAP_GetCurrHFState (BTUINT16 *agstate);	
Description	The **Btsdk_HFAP_GetCurrHFState** function gets current state of Hands-free device.	
Parameters	*agstate*	[out] A pointer to the variable which indicates current Hands-free device's state.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code.	

The ***agstate*** can be one of these values.

Value	Description
BTSDK_HFAP_ST_IDLE	Before service level connection is established.
BTSDK_HFAP_ST_STANDBY	Service level connection is established.
BTSDK_HFAP_ST_RINGING	Ringing state.
BTSDK_HFAP_ST_OUTGOINGCALL	Outgoing call state.
BTSDK_HFAP_ST_ONGOINGCALL	Ongoing call state.
BTSDK_HFAP_ST_BVRA	voice recognition is ongoing
BTSDK_HFAP_ST_VOVG	SCO link doesn't exist between AG and HF while a call is ongoing.
BTSDK_HFAP_ST_HELDINCOMINGCALL	the incoming call is held

Remarks

6.3.6.3.27 Btsdk_HFAP_SetWaveInDevice

Prototype	BTBOOL Btsdk_HFAP_SetWaveInDevice(BTUINT8* pWaveInDevice, BTUINT32 devNamelen);	
Description	The **Btsdk_HFAP_SetWaveInDevice** function sets wavein audio device for Handsfree application.	
Parameters	*pWaveInDevice*	[in] A pointer to the buffer that contains the wavein audio device name. If this parameter is NULL, the default audio device will be opened
	devNamelen	[in] Specifies the size in bytes of the string pointed to by the *pWaveInDevice* parameter.
Return:	If the function succeeds, the return value is BTSDK_TRUE. If the function fails, the return value is BTSDK_FALSE.	

Remarks

This function can be called to set wavein audio device in advance before establishing HF connection. It is also can be called to dynamically switch wavein audio device after establishing SCO link.

This funciton does not set the wavein audio device specified by pWaveInDevice as the default audio device.

6.3.6.3.28 Btsdk_HFAP_SetWaveOutDevice

Prototype	BTBOOL Btsdk_HFAP_SetWaveOutDevice(BTUINT8* pWaveOutDevice, BTUINT32 devNamelen);	
Description	The **Btsdk_HFAP_SetWaveInDevice** function sets waveout audio device for Handsfree application.	
Parameters	*pWaveOutDevice*	[in] A pointer to the buffer that contains the waveout audio device name. If this parameter is NULL, the default audio device will be opened.
	devNamelen	[in] Specifies the size in bytes of the string pointed to by the *pWaveOutDevice* parameter.
Return:	If the function succeeds, the return value is BTSDK_TRUE. If the function fails, the return value is BTSDK_FALSE.	

Remarks

This function can be called to set waveout audio device in advance before establishing HF connection. It is also can be called to dynamically switch waveout audio device after establishing SCO link.

This funciton does not set the wavein audio device specified by *pWaveOutDevice* as the default audio device.

6.3.7 Advancecd Audio Distribute Profile

6.3.7.1 A2DP Source

6.3.7.1.1 Btsdk_RegisterA2DPSRCService

Prototype	BTSVCHDL Btsdk_RegisterA2DPSRCService (void);	
Description	The **Btsdk_RegisterA2DPSRCService** function adds an A2DP SRC service record to SDK service database and then activates it.	
Parameters		
Return:	If the function succeeds, the return value is the handle to the new service record. If the function fails, the return value is BTSDK_INVALID_HANDLE.	

Remarks

Before calling *Btsdk_RegisterA2DPSRCService*, the service database must be initialized by a previous successful call to *Btsdk_Init*.

Currently, only one A2DP SRC service record is allowed at a time. That is, if the application calls the *Btsdk_RegisterA2DPSRCService* function twice, the second call will first remove the first A2DP SRC service record and then add a new A2DP SRC service record.

6.3.7.1.2 Btsdk_UnregisterA2DPSRCService

Prototype	BTUINT32 Btsdk_UnregisterA2DPSRCService (void);	
Description	The **Btsdk_UnregisterA2DPSRCService** function removes the current A2DP SRC service record from the SDK service database. If an A2DP SNK connects the SRC service, this function will release the connection first.	
Parameters		
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code.	

Remarks

This A2DP SRC service record is added to the service database by a previous call to the function *Btsdk_RegisterA2DPSRCService function*.

6.3.7.2 A2DP Sink

6.3.7.2.1 Btsdk_RegisterA2DPSNKService

Prototype	BTSVCHDL Btsdk_RegisterA2DPSNKService(BTUINT16 len, const BTUINT8* audio_card);	
Description	The **Btsdk_RegisterA2DPSNKService** function adds an A2DP SNK service record to SDK service database and then activates it.	
Parameters	*len*	[in] Specifies the size, in bytes, of the buffer pointed to by the *audio_card* parameter. It shall be smaller than BTSDK_A2DP_AUDIOCARD_NAME_LEN.
	audio_card	[in] A null-terminated string that specifies the playback device used to play the audio stream received over the Bluetooth A2DP connection.
Return:	If the function succeeds, the return value is the handle to the new service record. If the function fails, the return value is BTSDK_INVALID_HANDLE.	

Remarks

Before calling *Btsdk_RegisterA2DPSNKService*, the service database must be initialized by a previous successful call to *Btsdk_Init*.

Currently, only one A2DP SNK service record is allowed at a time. That is, if the application calls the *Btsdk_RegisterA2DPSNKService* function twice, the second call will first remove the first A2DP SNK service record and then add a new A2DP SNK service record.

6.3.7.2.2 Btsdk_UnregisterA2DPSNKService

Prototype	BTUINT32 Btsdk_UnregisterA2DPSNKService (void);	
Description	The **Btsdk_UnregisterA2DPSNKService** function removes the current A2DP SNK service record from the SDK service database. If an A2DP SRC connects the SNK service, this function will release the connection first.	
Parameters		
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code.	

Remarks

This A2DP SNK service record is added to the service database by a previous call to the function *Btsdk_RegisterA2DPSNKService function*.

6.3.8 Human Interface Device Profile

6.3.8.1 Btsdk_Hid_ClntUnPluggedDev

Prototype	BTUINT32 Btsdk_Hid_ClntUnPluggedDev(BTUINT8 * bdaddr);	
Description	The **Btsdk_Hid_ClntUnPluggedDev** function unplugs hid device.	
Parameters	*bdaddr*	[in] Pointer to the remote Bluetooth device address.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code.	

Remarks

6.3.8.2 Btsdk_Hid_LEHostConnect

Prototype	BTCONNHDL Btsdk_Hid_LEHostConnect(BTDEVHDL dev_hdl, struct BTP_HID_HOSTINFO *host_info);	
Description	The **Btsdk_Hid_LEHostConnect** function connect LE hid host.	
Parameters	*Dev_hdl*	[in] The handle to the remote device.
	host_info	[out] Pointer to buffer into which to return hid host information.
Return:	If the function succeeds, he return value is the connected handle of Bluetooth device. If the function fails, the return value is BTSDK_INVALID_HANDLE.	

Remarks

6.3.9 Phone Book Access Profile

6.3.9.1 General

6.3.9.1.1 Btsdk_RegisterPBAPService

Prototype	BTSVCHDL Btsdk_RegisterPBAPService(BTUINT8* svc_name, PBtSdkLocalPSEServerAttrStru svr_attr, PBtSdkPBAPSvrCBStru cb_funcs);	
Description	The **Btsdk_RegisterPBAPService** function registers Phone Book Server service record to SDK service database and then activates it.	
Parameters	*svc_name*	[in] User friendly name of the new service. It shall be a null-terminated UTF-8 string. It can't be NULL. Its length shall be limited within BTSDK_SERVICENAME_MAXLENGTH, including the terminated '\0'.
	svr_attr	[in] Specifies attribute of the new service. All members of this structure must be set. It may be a NULL pointer, and the default path delimiter is OS dependent, e.g. "\" in Windows PC. The default root directory is represented by a path delimiter.
	cb_funcs	[in] Pointer to the callback function which sets of this service.
Return:	If the function succeeds, the return value is the handle of the new service record. If the function fails, the return value is BTSDK_INVALID_HANDLE.	

Remarks

Before calling *Btsdk_RegisterPBAPService*, the service database must be initialized by a previous successful call to *Btsdk_Init*.

Currently, only one Phone Book Server service record is allowed at a time. That is, if the

application calls the *Btsdk_RegisterPBAPService* function twice, the second call will first remove the first Phone Book Server service record and then add a new Phone Book Server service record.

6.3.9.1.2 Btsdk_vCardParser_Open_Func

Prototype	typedef BTUINT8*(*Btsdk_vCardParser_Open_Func)(BTSDKHANDLE file_hdl);	
Description	The **Btsdk_vCardParser_Open_Func** function prototype is the prototype of application defined callback function used to initialize the parse operation of the vCard object contained in a specified file.	
Parameters	*file_hdl*	[in] The handle of the vCard objects are contained in a specified file. It is returned by *Btsdk_OpenFile_Func* or *Btsdk_OpenFile_Func.*
Return:	If the specified file contains correct vCard objects, a pointer is returned to identify this operation. It will be used for other vCard object analytic functions. If the spedified file is not able to parse by the application, the NULL is returned.	

6.3.9.1.3 Btsdk_vCardParser_GetProperty_Func

Prototype	typedef BTUINT8* (*Btsdk_vCardParser_GetProperty_Func)(BTUINT8* v_obj, BTUINT8* prop, BTINT32* len);	
Description	The **Btsdk_vCardParser_GetProperty_Func** function prototype is the prototype of application defined callback function used to read the attribute value of the specified vCard object. If the vCard object contains a number of specified attributes, the application selects one of them to be returned.	
Parameters	*v_obj*	[in] A pointer returned by *Btsdk_vCardParser_Open_Func*.
	prop	[in] A null-terminated ANSI string which specifies the name of the attribute to be returned, e.g. 'N'.
	len	[out] The length of the returned attribute.
Return:	If the vCard object contains the specified attribute, then the whole attribute is returned. This string must be coded in UTF-8 format, e.g. "N: David \ r \ n". Finally, the memory will be released by calling *Btsdk_vCardParser_FreeProperty_Func*. If the application is not able to find the specified attribute or find an error during parse processing, then NULL is returned.	

6.3.9.1.4 Btsdk_vCardParser_FindFirstProperty_Func

Prototype	typedef BTUINT8* (*Btsdk_vCardParser_FindFirstProperty_Func)(BTUINT8* v_obj, BTUINT8* prop, BTSDKHANDLE* find_hdl, BTINT32* len);	
Description	The **Btsdk_vCardParser_FindFirstProperty_Func** function prototype is the prototype of application defined callback function used to find the specified attribute from a vCard object, and then return the value of the first instance.	
Parameters	*v_obj*	[in] A pointer returned by *Btsdk_vCardParser_Open_Func*.
	prop	[in] A null-terminated ANSI string which specifies the name of the attribute to be returned, e.g. 'N'.
	find_hdl	[in/out] Specifies the returned file handle is used for *Btsdk_vCardParser_FindNextProperty_Func* and *Btsdk_vCardParser_FindPropertyClose_Func.*
	len	[out] The length of returned attribute.
Return:	If the vCard object contains the specified attribute, then the whole attribute is returned. This string must be coded in UTF-8 format, e.g. "N: David \ r \ n". Finally, the memory will be released by calling *Btsdk_vCardParser_FreeProperty_Func*. If the application is not able to find the specified attribute or find an error during parse processing, the NULL is returned.	

6.3.9.1.5 Btsdk_vCardParser_FindNextProperty_Func

Prototype	typedef BTUINT8*(*Btsdk_vCardParser_FindNextProperty_Func)(BTSDKHANDLE find_hdl, BTINT32* len);	
Description	The **Btsdk_vCardParser_FindNextProperty_Func** function prototype is the prototype of application defined callback function used to find the next specified attribute from vCard object, and return the value of the next instance of the specified attribute.	
Parameters	*find_hdl*	[in] A handle returned by *Btsdk_vCardParser_FindFirstProperty_Func*.
	len	[out] The length of returned attribute.
Return:	If the vCard object contains the specified attribute, then the whole attribute is returned. This string must be coded in UTF-8 format, e.g. "N: David \ r \ n". Finally, the memory will be released by calling *Btsdk_vCardParser_FreeProperty_Func*. If the application is not able to find the specified attribute or find an error during parse processing, the NULL is returned.	

6.3.9.1.6 Btsdk_vCardParser_FindPropertyClose_Func

Prototype	typedef void (*Btsdk_vCardParser_FindPropertyClose_Func)(BTSDKHANDLE find_hdl);	
Description	The **Btsdk_vCardParser_FindPropertyClose_Func** function prototype is the prototype of application defined callback function used to close searching process.	
Parameters	*find_hdl*	[in] A handle returned by *Btsdk_vCardParser_FindFirstProperty_Func*.
Return:		

6.3.9.1.7 Btsdk_vCardParser_FreeProperty_Func

Prototype	typedef void (*Btsdk_vCardParser_FreeProperty_Func)(BTUINT8* buf);	
Description	The **Btsdk_vCardParser_FreeProperty_Func** function prototype is the prototype of application defined callback function used to release memory.	
Parameters	*buf*	[in] Pointer to the buffer which is allocated by Btsdk_vCardParser_GetProperty_Func, Btsdk_vCardParser_FindFirstProperty_Func or Btsdk_vCardParser_FindNextProperty_Func.
Return:		

6.3.9.1.8 Btsdk_vCardParser_Close_Func

Prototype	typedef void (*Btsdk_vCardParser_Close_Func)(BTUINT8* v_obj);	
Description	The **Btsdk_vCardParser_Close_Func** function prototype is the prototype of application defined callback function used to close parse processing started by *Btsdk_vCardParser_Open_Func*.	
Parameters	*v_obj*	[in]A pointer returned by *Btsdk_vCardParser_Open_Func*.
Return:		

6.3.9.1.9 Btsdk_FindFirstFile_Func

Prototype	typedef BTSDKHANDLE (*Btsdk_FindFirstFile_Func)(const BTUINT8* path, BTUINT8* file_name);	
Description	The **Btsdk_FindFirstFile_Func** function prototype is the prototype of application defined callback function used to find vCard objects from specified path, and then the name of the first file is returned.	
Parameters	*path*	[in] A null-terminated ANSI string specifies the path of searching. It should be less than BTSDK_PATH_MAXLENGTH
	file_name	[out] Receives the name of the file found. Any path information could not be included in this parameter.
Return:	If the function succeeds, the return value is a handle used for the following searching process. If the function fails, the return value is BTSDK_INVALID_HANDLE.	

6.3.9.1.10 Btsdk_FindNextFile_Func

Prototype	typedef BTINT32 (*Btsdk_FindNextFile_Func)(BTSDKHANDLE find_hdl, BTUINT8* file_name);	
Description	The **Btsdk_FindNextFile_Func** function prototype is the prototype of application defined callback function used to find the name of the next file.	
Parameters	*find_hdl*	[in]A handle returned by *Btsdk_FindFirstFile_Func*.
	file_name	[out] Receives the name of the file found. Any path information could not be included in this parameter.
Return:	If the function succeeds, the return value is 0. If the function fails, such as all the vCard files in the directory have been enumerated, then the return value is -1.	

6.3.9.1.11 Btsdk_FindFileClose_Func

Prototype	typedef void (*Btsdk_FindFileClose_Func)(BTSDKHANDLE find_hdl);	
Description	The **Btsdk_FindFileClose_Func** function prototype is the prototype of application defined callback function used to end the searching process.	
Parameters	*find_hdl*	[in]A handle returned by *Btsdk_FindFirstFile_Func*.
Return:		

6.3.9.1.12 Btsdk_OpenFile_Func

Prototype	typedef BTSDKHANDLE (*Btsdk_OpenFile_Func)(const BTUINT8* file_name);	
Description	The **Btsdk_OpenFile_Func** function prototype is the prototype of application defined callback function used to open file.	
Parameters	*file_name*	[in] A null-terminated ANSI string which specifies the name of the file. It contains path information.
Return:	If the function succeeds, the return value is the file handle. If the function fails, the return value is BTSDK_INVALID_HANDLE.	

6.3.9.1.13 Btsdk_CreateFile_Func

Prototype	typedef BTSDKHANDLE (*Btsdk_CreateFile_Func)(const BTUINT8* file_name);	
Description	The **Btsdk_CreateFile_Func** function prototype is the prototype of application defined callback function used to create or open a file. If the file does not exist, a new file will be created. If the file exists, the contents of the original file must be cleared.	
Parameters	*file_name*	[in] A null-terminated ANSI string which specifies the name of the file. It contains path information.
Return:	If the function succeeds, the return value is the handle for the following searching process. If the function fails, the return value is BTSDK_INVALID_HANDLE.	

6.3.9.1.14 Btsdk_WriteFile_Func

Prototype	typedef BTUINT32 (*Btsdk_WriteFile_Func)(BTSDKHANDLE file_hdl, BTUINT8* buf, BTUINT32 bytes_to_write);	
Description	The **Btsdk_WriteFile_Func** function prototype is the prototype of application defined callback function used to write data to a file.	
Parameters	*find_hdl*	[in]A handle returned by *Btsdk_OpenFile_Func* or *Btsdk_CreateFile_Func.*
	buf	[in] Pointer to the buffer containing the data to be written to the file.
	bytes_to_write	[in] Number of bytes to be written to the file.
Return:	The return value is the actual length of data written into the file.	

6.3.9.1.15 Btsdk_ReadFile_Func

Prototype	typedef BTUINT32 (*Btsdk_ReadFile_Func)(BTSDKHANDLE file_hdl, BTUINT8* buf, BTUINT32 len);	
Description	The **Btsdk_ReadFile_Func** function prototype is the prototype of application defined callback function used to read data from a file.	
Parameters	*find_hdl*	[in]A handle returned by *Btsdk_OpenFile_Func* or *Btsdk_CreateFile_Func.*
	buf	[in] A pointer to the buffer that receives the data read from a file.
	len	[in] Number of bytes to be read from the file.
Return:	The return value is the actual length of data read from the file. Its length no more than *len*.	

6.3.9.1.16 Btsdk_GetFileSize_Func

Prototype	typedef BTUINT32 (*Btsdk_GetFileSize_Func)(BTSDKHANDLE file_hdl);	
Description	The **Btsdk_GetFileSize_Func** function prototype is the prototype of application defined callback function used to retrieve the size of the file.	
Parameters	*find_hdl*	[in]A handle returned by *Btsdk_OpenFile_Func* or *Btsdk_CreateFile_Func.*
Return:	If the function succeeds, the return value is size of the file. If the function fails, the return value is 0.	

6.3.9.1.17 Btsdk_RewindFile_Func

Prototype	typedef BTINT32 (*Btsdk_RewindFile_Func)(BTSDKHANDLE file_hdl, BTUINT32 offset);	
Description	The **Btsdk_RewindFile_Func** function prototype is the prototype of application defined callback function used to move the file pointer to a specified location.	
Parameters	*find_hdl*	[in]A handle returned by *Btsdk_OpenFile_Func* or *Btsdk_CreateFile_Func.*
	offset	[in] Number of bytes from *origin*.
Return:	If the function succeeds, the return value is 0 If the function fails, the return value is nonzero.	

6.3.9.1.18 Btsdk_CloseFile_Func

Prototype	typedef void (*Btsdk_CloseFile_Func)(BTSDKHANDLE file_hdl);	
Description	The **Btsdk_CloseFile_Func** function prototype is the prototype of application defined callback function used to close a file.	
Parameters	*find_hdl*	[in]A handle returned by *Btsdk_OpenFile_Func* or *Btsdk_CreateFile_Func.*
Return:		

6.3.9.1.19 Btsdk_ChangDir_Func

Prototype	typedef BTINT32 (*Btsdk_ChangDir_Func)(const BTUINT8* path);	
Description	The **Btsdk_ChangDir_Func** function prototype is the prototype of application defined callback function used to change the current directory to the new specified directory.	
Parameters	*path*	[in] A null-terminated ANSI string which specifies the new directory.
Return:	If the function succeeds, the return value is 0 If the function fails, the return value is nonzero.	

6.3.9.1.20 Btsdk_CreateDir_Func

Prototype	typedef BTINT32 (*Btsdk_CreateDir_Func)(const BTUINT8* path);	
Description	The **Btsdk_CreateDir_Func** function prototype is the prototype of application defined callback function used to create a new directory.	
Parameters	*path*	[in] A null-terminated ANSI string which specifies the new directory.
Return:	If the function succeeds, the return value is 0 If the function fails, the return value is nonzero.	

6.3.9.1.21 Btsdk_GetMissedCalls_Func

Prototype	Typedef BTINT32 (*Btsdk_PBAP_GetMissedCalls_Func)(BTUINT8* path);	
Description	The **Btsdk_PBAP_GetMissedCalls_Func** function prototype is the prototype of application defined callback function used to retrieve the number of the missed calls.	
Parameters	*path*	[in] A null-terminated ANSI string which specifies the path of stored missed calls information. It can be ignored by applications.
Return:	The return value is the number of missed calls which have not yet been checked before this function call.	

6.3.9.1.22 Btsdk_PBAPRegisterSvrCallback

Prototype	BTINT32 Btsdk_PBAPRegisterSvrCallback(BTSVCHDL svc_hdl, PBtSdkPBAPSvrCBStru cb_funcs);	
Description	The **Btsdk_PBAPRegisterSvrCallback** function registers an application-defined callback function used to deal with the operation of Phone Book Server service.	
Parameters	*svc_hdl*	[in] The handle of the Phone Book Server service.
	cb_funcs	[in] Pointer to the callback functions of service.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code.	

Remarks

If the application calls *Btsdk_PBAPRegisterSvrCallback* twice to register different callback functions for the same service handle, the second callback function will replace the first one.

6.3.9.1.23 Btsdk_PBAPRegisterFileIORoutines

Prototype	BTINT32 Btsdk_PBAPRegisterFileIORoutines(BTCONNHDL conn_hdl, PBtSdkPBAPFileIORoutinesStru cb_funcs);	
Description	The **Btsdk_PBAPRegisterFileIORoutines** function registers the file I/O routines required by an outgoing PBAP connection.	
Parameters	*conn_hdl*	[in] Handle to the PBAP connection.
	cb_funcs	[in] Pointer to the set of callback functions.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code.	

Remarks

This function shall be called before calling any of Btsdk_PBAPGetPhonebook, Btsdk_PBAPGetCardListing and Btsdk_PBAPGetCardEntry.

6.3.9.1.24 Btsdk_UnregisterPBAPService

Prototype	BTINT32 Btsdk_UnregisterPBAPService (BTSVCHDL svc_hdl);	
Description	The **Btsdk_UnregisterPBAPService** function removes the current Phone Book Server service record from the SDK service database.	
Parameters	*svc_hdl*	[in] The handle of the Phone Book Server service.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code.	

Remarks

6.3.9.1.25 Btsdk_PBAPRegisterStatusCallback

Prototype	BTINT32 Btsdk_PBAPRegisterStatusCallback(BTCONNHDL conn_hdl, Btsdk_PBAP_STATUS_INFO_CB* func);	
Description	The **Btsdk_PAN_RegIndCbk4ThirdPart** function registers an application-defined callback function used to deal with PAN callback messages.	
Parameters	*conn_hdl*	[in] Handle to the PBAP connection.
	func	[in] Pointers to the callback function of Btsdk_PBAP_STATUS_INFO_CB type.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code.	

Remarks

If *func* is NULL, the calling of *Btsdk_PBAPRegisterStatusCallback* will remove the callback for the specified connection handle.

6.3.9.1.26 Btsdk_PBAP_STATUS_INFO_CB

Prototype	typedef void (Btsdk_PBAP_STATUS_INFO_CB)(BTUINT8 first, BTUINT8 last, BTUINT8* filename, BTUINT32 filesize, BTUINT32 cursize);	
Description	The **Btsdk_PBAP_STATUS_INFO_CB** function prototype is the prototype of application defined callback function used to deal with file transfer status.	
Parameters	*first*	[in] Flag specifies whether it is the first call this function. Any none zero (TRUE) value means it is the fist call. Otherwise, it is a continuous call.
	last	[in] Flag specifies whether it is the last call to this function. Any none zero (TRUE) value means it is the last call. Otherwise, it is not a last call.
	filename	[in] Pointer to the buffer contains the file name. It is valid only when first flag is not zero.
	filesize	[in] Specifies full size of the file to be transferred in bytes, only valid when first flag is not zero.
	cursize	[in] Specifies current transferred size in bytes.
Return:		

Remarks

This callback function needs to be registered using *Btsdk_PBAP_STATUS_INFO_CB* function. It is always called when the device sends/receives an OBEX package over the specified PBAP connection

6.3.9.1.27 Btsdk_PBAPPullPhoneBook

Prototype	BTINT32 Btsdk_PBAPPullPhoneBook(BTCONNHDL conn_hdl, BTUINT8* path, PBtSdkPBAPParamStru param, BTSDKHANDLE file_hdl);	
Description	The **Btsdk_PBAPPullPhoneBook** function retrieves an entire phone book object from the object exchange server.	
Parameters	*conn_hdl*	[in] The handle to the connection with the remote PBAP server.
	path	[in] A null-terminated ANSI string which specifies the name of the phone book objects to be retrieved. It shall contain the absolute path in the virtual folders architecture of the PSE, appended with the name of the file representation of the phone book object, e.g. telecom/pb.vcf.
	param	[in/out] For input type, it specifies the parameters required by the Pull Phone Book request. For output type, it specifies the parameters got from the Pull Phone Book response.
	file_hdl	[in] Handle to the file to store the retrieved phone book object.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code.	

Remarks

6.3.9.1.28 Btsdk_PBAPFilterComposer

Prototype	void Btsdk_PBAPFilterComposer(BTUINT8* filter, BTUINT8 flag);	
Description	The **Btsdk_PBAPFilterComposer** function sets the attribute masks table.	
Parameters	*filter*	[out] 8 bytes attribute masks in big Endian order. Any bit which is not needed to be set must be 0.
	flag	[in] Specifies the bits of the attribute masks which are needed to be set to 1.
Return:		

Remarks

The possible values of *nAddressType* are listed as follows:

Type	Defination
BTSDK_PBAP_FILTER_VERSION	No.0 bit:The attribute of VERSION
BTSDK_PBAP_FILTER_FN	No.1 bit:The attribute of FN
BTSDK_PBAP_FILTER_N	No.2 bit:The attribute of N.
BTSDK_PBAP_FILTER_PHOTO	No.3 bit:The attribute of PHOTO
BTSDK_PBAP_FILTER_BDAY	No.4 bit:The attribute of BDAY
BTSDK_PBAP_FILTER_ADR	No.5 bit:The attribute of ADR
BTSDK_PBAP_FILTER_LABEL	No.6 bit:The attribute of LABEL
BTSDK_PBAP_FILTER_TEL	No.7 bit:The attribute of TEL
BTSDK_PBAP_FILTER_EMAIL	No.8 bit:The attribute of EMAIL
BTSDK_PBAP_FILTER_MAILER	No.9 bit:The attribute of MAILER
BTSDK_PBAP_FILTER_TZ	No.10 bit:The attribute of TZ
BTSDK_PBAP_FILTER_GEO	No.11 bit:The attribute of GEO
BTSDK_PBAP_FILTER_TITLE	No.12 bit:No.0 bit:The attribute of TITLE
BTSDK_PBAP_FILTER_ROLE	No.13 bit:No.0 bit:The attribute of ROLE
BTSDK_PBAP_FILTER_LOGO	No.14 bit:The attribute of LOGO
BTSDK_PBAP_FILTER_AGENT	No.15 bit:The attribute of AGENT
BTSDK_PBAP_FILTER_ORG	No.16 bit:The attribute of ORG
BTSDK_PBAP_FILTER_NOTE	No.17 bit:The attribute of NOTE
BTSDK_PBAP_FILTER_REV	No.18 bit:The attribute of REV
BTSDK_PBAP_FILTER_SOUND	No.19 bit:The attribute of SOUND

BTSDK_PBAP_FILTER_URL	No.20 bit:The attribute of URL
BTSDK_PBAP_FILTER_UID	No.21 bit:The attribute of UID
BTSDK_PBAP_FILTER_KEY	No.22 bit:The attribute of KEY
BTSDK_PBAP_FILTER_NICKNAME	No.23 bit:The attribute of NICKNAME
BTSDK_PBAP_FILTER_CATEGORIES	No.24 bit:The attribute of CATEGORIES
BTSDK_PBAP_FILTER_PROID	No.25 bit:The attribute of PROID
BTSDK_PBAP_FILTER_CLASS	No.26 bit:The attribute of CLASS
BTSDK_PBAP_FILTER_SORT_STRING	No.27 bit:The attribute of SORT-STRING
BTSDK_PBAP_FILTER_X_IRMC_CALL_DATETIME	No.28 bit:The attribute of X-IRMC-CALL-DATETIME

6.3.9.1.29 Btsdk_PBAPSetPath

Prototype	BTINT32 Btsdk_PBAPSetPath(BTCONNHDL conn_hdl, BTUINT8* folder);	
Description	The **Btsdk_PBAPSetPath** function modifies the current folder path of the remote device.	
Parameters	*conn_hdl*	[in] The handle to the connection with the remote PBAP server.
	folder	[in] A null-terminated ANSI string which specifies the name of the folder. A NULL pointer or an empty string specifies the root folder. ".." Specifies the parent folder. Otherwise, it specifies a child folder, e.g. "telecom".
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code.	

Remarks

6.3.9.1.30 Btsdk_PBAPPullCardList

Prototype	BTINT32 Btsdk_PBAPPullCardList (BTCONNHDL conn_hdl, BTUINT8* folder, PBtSdkPBAPParamStru param, BTSDKHANDLE file_hdl);	
Description	The **Btsdk_PAN_RegIndCbk4ThirdPart** function retrieves the vCard-listing object from the object exchange server.	
Parameters	*conn_hdl*	[in] The handle to the connection with the remote PBAP server.
	folder	[in] A null-terminated ANSI string which specifies the name of the folder to be retrieved. It shall not include any path information.
	param	[in/out] For input type, it specifies the parameters required by the Pull vCard Listing request. For output type, it specifies the parameters got from the Pull vCard Listing response.
	file_hdl	[in] Handle to the file to store the retrieved vCard-listing object.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code.	

Remarks

6.3.9.1.31 Btsdk_PBAPPullCardEntry

Prototype	BTINT32 Btsdk_PBAPPullCardEntry (BTCONNHDL conn_hdl, BTUINT8* name, PBtSdkPBAPParamStru param, BTSDKHANDLE file_hdl);	
Description	The **Btsdk_PBAPPullCardEntry** function retrieves a specific vCard from the object exchange server.	
Parameters	*conn_hdl*	[in] The handle to the connection with the remote PBAP server.
	name	[in] A null-terminated ANSI string which specifies the name of the vCard to be retrieved. Any path information should not be included in this parameter.
	param	[in] Specifies the parameters required by the Pull vCard Entry request.
	file_hdl	[in] Handle to the file to store the retrieved vCard object.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code.	

Remarks

6.3.9.1.32 Btsdk_PBAPCancelTransfer

Prototype	BTINT32 Btsdk_PBAPCancelTransfer (BTCONNHDL conn_hdl);	
Description	The **Btsdk_PBAPCancelTransfer** function terminates the file transfer procedure.	
Parameters	*conn_hdl*	[in] The handle of the current PBAP connection handle.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code.	

Remarks

6.3.10 MAP Profile

6.3.10.1 General

6.3.10.1.1 Btsdk_MAP_STATUS_INFO_CB

Prototype	typedef void (Btsdk_MAP_STATUS_INFO_CB)(BTUINT8 first, BTUINT8 last, BTUINT8* filename, BTUINT32 filesize, BTUINT32 cursize);	
Description	The **Btsdk_MAP_STATUS_INFO_CB** function prototype is the prototype of application defined callback function used to deal with file transfer status. This call back must be realized.	
Parameters	*first*	[in] Flag specifies whether it is the first call to this function. Any none zero (TRUE) value means it is the fist call. Otherwise, it is a continuous call.
	last	[in] Flag specifies whether it is the last call to this function. Any none zero (TRUE) value means it is the last call. Otherwise, it is not a last call.
	filename	[in] Pointer to the buffer contains the file name. It is valid only when first flag is not zero.
	filesize	[in] Specifies full size of the file to be transferred in bytes, only valid when first flag is not zero.
	cursize	[in] Specifies current transferring size in bytes.
Return:		

Remarks

6.3.10.1.2 Btsdk_MNS_MessageNotification_Func

Prototype	typedef void (*Btsdk_MNS_MessageNotification_Func)(BTSVCHDL svc_hdl, PBtSdkMAPEvReportObjStru ev_obj);	
Description	This **Btsdk_MNS_MessageNotification_Func** prototype is the prototype of application defined callback function used to deal with evnt report from MNS	
Parameters	*svc_hdl*	[in] The handle of the MNS service.
	ev_obj	[in] Specific content of event notification
Return:		

Remarks

6.3.10.1.3 Btsdk_MAP_FindFirstFolder_Func

Prototype	typedef BTSDKHANDLE (*Btsdk_MAP_FindFirstFolder_Func)(const BTUINT8 *path, PBtSdkMAPFolderObjStru pfd);	
Description	The **Btsdk_MAP_FindFirstFolder_Func** function prototype is the prototype of application defined callback function used to find folder objects from specified path, and then the information of the first folder is returned.	
Parameters	*path*	[in] A null-terminated UTF-8 string specifies the path of searching. It should be less than BTSDK_PATH_MAXLENGTH
	pfd	[out] Receives the information of the folder found.
Return:	If the function succeeds, the return value is a handle used for the following searching process. If the function fails, the return value is BTSDK_INVALID_HANDLE.	

6.3.10.1.4 Btsdk_MAP_FindNextFolder_Func

Prototype	typedef BTBOOL (*Btsdk_MAP_FindNextFolder_Func)(BTSDKHANDLE find_hdl, PBtSdkMAPFolderObjStru pfd);	
Description	The **Btsdk_MAP_FindNextFolder_Func** function prototype is the prototype of application defined callback function used to find the information of the next folder.	
Parameters	*find_hdl*	[in]A handle returned by *Btsdk_MAP_FindNextFolder_Func*.
	pfd	[out] Receives the information of the folder found.
Return:	If the function succeeds, the return value is BTSDK_TRUE. If the function fails, such as all the folders in the directory have been enumerated, then the return value is BTSDK_FALSE.	

6.3.10.1.5 Btsdk_MAP_FindFolderClose_Func

Prototype	typedef BTBOOL (*Btsdk_MAP_FindFolderClose_Func)(BTSDKHANDLE find_hdl);	
Description	The **Btsdk_MAP_FindFolderClose_Func** function prototype is the prototype of application defined callback function used to end the searching process.	
Parameters	*find_hdl*	[in]A handle returned by *Btsdk_MAP_FindFirstFolder_Func*.
Return:	If the function succeeds, the return value is BTSDK_TRUE. If the function fails, the return value is BTSDK_FALSE.	

6.3.10.1.6 Btsdk_MAP_FindFirstMsg_Func

Prototype	typedef BTSDKHANDLE (*Btsdk_MAP_FindFirstMsg_Func)(const BTUINT8 *path, PBtSdkMAPMsgFilterStru pfilter, PBtSdkMAPMsgObjStru pmsg);	
Description	The **Btsdk_MAP_FindFirstMsg_Func** function prototype is the prototype of application defined callback function used to find message objects from specified path, and then the information of the first message is returned.	
Parameters	*path*	[in] A null-terminated UTF-8 string specifies the path of searching. It should be less than BTSDK_PATH_MAXLENGTH
	pfilter	[in] Contains the filter condition values specified by the GetMessageListing request.
	pmsg	[out] Receives the attribute values of the message found. If pmsg is NULL, means only to count matched message objects.
Return:	If the function succeeds, the return value is a handle used for the following searching process. If the function fails, the return value is BTSDK_INVALID_HANDLE.	

6.3.10.1.7 Btsdk_MAP_FindNextMsg_Func

Prototype	typedef BTBOOL (*Btsdk_MAP_FindNextMsg_Func)(BTSDKHANDLE find_hdl, PBtSdkMAPMsgFilterStru pfilter, PBtSdkMAPMsgObjStru pmsg);	
Description	The **Btsdk_MAP_FindNextMsg_Func** function prototype is the prototype of application defined callback function used to find the information of the next message.	
Parameters	*find_hdl*	[in]A handle returned by *Btsdk_MAP_FindNextFolder_Func*.
	pfilter	[in] Contains the filter condition values specified by the GetMessageListing request.
	pmsg	[out] Receives the attribute values of the message found. If pmsg is NULL, means only to count matched message objects.
Return:	If the function succeeds, the return value is BTSDK_TRUE. If the function fails, such as all the messages in the directory have been enumerated, then the return value is BTSDK_FALSE.	

6.3.10.1.8 Btsdk_MAP_FindMsgClose_Func

Prototype	typedef BTBOOL (*Btsdk_MAP_FindMsgClose_Func)(BTSDKHANDLE find_hdl);	
Description	The **Btsdk_MAP_FindMsgClose_Func** function prototype is the prototype of application defined callback function used to end the searching process.	
Parameters	*find_hdl*	[in]A handle returned by *Btsdk_MAP_FindFirstMsg_Func*.
Return:	If the function succeeds, the return value is BTSDK_TRUE. If the function fails, the return value is BTSDK_FALSE.	

6.3.10.1.9 Btsdk_MAP_ModifyMsgStatus_Func

Prototype	typedef BTBOOL (*Btsdk_MAP_ModifyMsgStatus_Func)(PBtSdkMAPMsgStatusStru msg_info);	
Description	The **Btsdk_MAP_ModifyMsgStatus_Func** function prototype is the prototype of application defined callback function used to change the status of the specified message. It is called when the MSE server receives the SetMessageStatus request.	
Parameters	*msg_info*	[in] Specifies the message to be modified and the new status value.
Return:	If the function succeeds, the return value is BTSDK_TRUE. If the function fails, the return value is BTSDK_FALSE.	

6.3.10.1.10 Btsdk_MAP_CreateBMsgFile_Func

Prototype	typedef BTSDKHANDLE (*Btsdk_MAP_CreateBMsgFile_Func)(const BTUINT8 *cur_path, PBtSdkMAPMsgHandleStru msg_hdl);	
Description	The **Btsdk_MAP_CreateBMsgFile_Func** function prototype is the prototype of application defined callback function used to create an empty message file. It is called when the MSE server receies the PushMessage request.	
Parameters	*cur_path*	[in] A null-terminated UTF-8 string specifies the path to save the message file. It should be less than BTSDK_PATH_MAXLENGTH.
	msg_hdl	[out] Returns the handle assigned to the new message created.
Return:	If the function succeeds, the return value is the file handle. If the function fails, the return value is BTSDK_INVALID_HANDLE.	

6.3.10.1.11 Btsdk_MAP_OpenBMsgFile_Func

Prototype	typedef BTSDKHANDLE (*Btsdk_MAP_OpenBMsgFile_Func)(PBtSdkMAPGetMsgParamStru msg_info);	
Description	The **Btsdk_MAP_OpenBMsgFile_Func** function prototype is the prototype of application defined callback function used to find and open a message object for reading. It is called when the MSE server receives the GetMessage request.	
Parameters	*msg_info*	[in] Specifies the message object to be opened and the format of message content.
Return:	If the function succeeds, the return value is the file handle. If the function fails, the return value is BTSDK_INVALID_HANDLE.	

6.3.10.1.12 Btsdk_MAP_PushMsg_Func

Prototype	typedef BTBOOL (*Btsdk_MAP_PushMsg_Func)(const BTUINT8 *cur_path, PBtSdkMAPPushMsgParamStru msg_info);	
Description	The **Btsdk_MAP_OpenBMsgFile_Func** function prototype is the prototype of application defined callback function used to deal with the message object received from the MCE device, e.g. send the message to the telecom network. It is called after the MSE server receiving the complete message object in a PushMessage operation.	
Parameters	*cur_path*	[in] A null-terminated UTF-8 string specifies the path of the message to be operated. It should be less than BTSDK_PATH_MAXLENGTH.
	msg_info	[in] Specifies the message object and the way to deal with the message.
Return:	If the function succeeds, the return value is BTSDK_TRUE. If the function fails, the return value is BTSDK_FALSE.	

6.3.10.1.13 Btsdk_OpenFile_Func

Prototype	typedef BTSDKHANDLE (*Btsdk_OpenFile_Func)(const BTUINT8* file_name);	
Description	The **Btsdk_OpenFile_Func** function prototype is the prototype of application defined callback function used to open file.	
Parameters	*file_name*	[in] A null-terminated UTF-8 string which specifies the name of the file. It contains path information.
Return:	If the function succeeds, the return value is the file handle. If the function fails, the return value is BTSDK_INVALID_HANDLE.	

Remarks

6.3.10.1.14 Btsdk_CreateFile_Func

Prototype	typedef BTSDKHANDLE (*Btsdk_CreateFile_Func)(const BTUINT8* file_name);	
Description	The **Btsdk_CreateFile_Func** function prototype is the prototype of application defined callback function used to create or open a file. If the file does not exist, a new file will be created. If the file exists, the contents of the original file must be cleared.	
Parameters	*file_name*	[in] A null-terminated UTF-8 string which specifies the name of the file. It contains path information.
Return:	If the function succeeds, the return value is the handle for the following searching process. If the function fails, the return value is BTSDK_INVALID_HANDLE.	

Remarks

6.3.10.1.15 Btsdk_WriteFile_Func

Prototype	typedef BTUINT32 (*Btsdk_WriteFile_Func)(BTSDKHANDLE file_hdl, BTUINT8* buf, BTUINT32 bytes_to_write);	
Description	The **Btsdk_WriteFile_Func** function prototype is the prototype of application defined callback function used to write data to a file.	
Parameters	*file_hdl*	[in] A handle returned by *Btsdk_OpenFile_Func, Btsdk_CreateFile_Func, Btsdk_MAP_OpenBMsgFile_Func or Btsdk_MAP_CreateBMsgFile_Func.*
	buf	[in] Pointer to the buffer containing the data to be written to the file.
	len	[in] Number of bytes to be written to the file.
Return:	The return value is the actual length of data written into the file.	

Remarks

6.3.10.1.16 Btsdk_ReadFile_Func

Prototype	typedef BTUINT32 (*Btsdk_ReadFile_Func)(BTSDKHANDLE file_hdl, BTUINT8* buf, BTUINT32 len, BTBOOL *is_end);	
Description	The **Btsdk_ReadFile_Func** function prototype is the prototype of application defined callback function used to read data from a file.	
Parameters	*file_hdl*	[in] A handle returned by *Btsdk_OpenFile_Func, Btsdk_CreateFile_Func, Btsdk_MAP_OpenBMsgFile_Func or Btsdk_MAP_CreateBMsgFile_Func.*
	buf	[out] A pointer to the buffer that receives the data read from a file.
	len	[in] Number of bytes to be read from the file.
	is_end	[out] Specify whether all data has been read from the file. It could be one of the values below: BTSDK_TRUE – All data has been read from the file BTSDK_FALSE – There is still data has not been read yet
Return:	The return value is the actual length of data read from the file. Its length no more than *len*.	

Remarks

6.3.10.1.17 Btsdk_GetFileSize_Func

Prototype	typedef BTUINT32 (*Btsdk_GetFileSize_Func)(BTSDKHANDLE file_hdl);	
Description	The **Btsdk_GetFileSize_Func** function prototype is the prototype of application defined callback function used to retrieve the size of the file.	
Parameters	*file_hdl*	[in] A handle returned by *Btsdk_OpenFile_Func*, *Btsdk_CreateFile_Func*, *Btsdk_MAP_OpenBMsgFile_Func or Btsdk_MAP_CreateBMsgFile_Func*.
Return:	If the function succeeds, the return value is size of the file. If the function fails, the return value is 0.	

Remarks

6.3.10.1.18 Btsdk_RewindFile_Func

Prototype	typedef BTINT32 (*Btsdk_RewindFile_Func)(BTSDKHANDLE file_hdl, BTUINT32 offset);	
Description	The **Btsdk_RewindFile_Func** function prototype is the prototype of application defined callback function used to move the file pointer to a specified location.	
Parameters	*file_hdl*	[in] A handle returned by *Btsdk_OpenFile_Func, Btsdk_CreateFile_Func, Btsdk_MAP_OpenBMsgFile_Func or Btsdk_MAP_CreateBMsgFile_Func.*
	offset	[in] Number of bytes from *origin*.
Return:	If the function succeeds, the return value is 0 If the function fails, the return value is nonzero.	

Remarks

6.3.10.1.19 Btsdk_CloseFile_Func

Prototype	typedef void (*Btsdk_CloseFile_Func)(BTSDKHANDLE file_hdl);	
Description	The **Btsdk_CloseFile_Func** function prototype is the prototype of application defined callback function used to close a file.	
Parameters	*file_hdl*	[in] A handle returned by *Btsdk_OpenFile_Func, Btsdk_CreateFile_Func, Btsdk_MAP_OpenBMsgFile_Func or Btsdk_MAP_CreateBMsgFile_Func.*
Return:		

Remarks

6.3.10.1.20 Btsdk_MAP_RegisterNotification_Func

Prototype	typedef BTBOOL (*Btsdk_MAP_RegisterNotification_Func)(BTCONNHDL mns_conn_hdl, BTSVCHDL mas_svc_hdl, BTBOOL turn_on);	
Description	The **Btsdk_MAP_RegisterNotification_Func** function prototype is the prototype of application defined callback function used to enable or disable MSE's notification function. It is called after the MSE server receiving the SetNotificationRegistration request. The lower Stack will connect with the remote MNS server automatically if the request is to enable notification. The lower Stack will disconnect the MNS connection if the request is to disable the notification.	
Parameters	*mns_conn_hdl*	[in] Handle to the MNS connection to send the notification.
	mas_svc_hdl	[in] Handle to the local MAS server receiving the SetNotificationRegistration request.
	turn_on	[in] It can be one of, BTSDK_TRUE – To enable the notification; BTSDK_FALSE – To disable the notification.
Return:	If the function succeeds, the return value is BTSDK_TRUE. If the function fails, the return value is BTSDK_FALSE.	

6.3.10.1.21 Btsdk_MAP_UnpdateInbox_Func

Prototype	typedef BTBOOL (*Btsdk_MAP_UnpdateInbox_Func)(void);	
Description	The **Btsdk_MAP_UnpdateInbox_Func** function prototype is the prototype of application defined callback function used to update MSE's inbox. It is called after the MSE server receiving the UpdateInbox request.	
Parameters		
Return:	If the function succeeds, the return value is BTSDK_TRUE. If the function fails, the return value is BTSDK_FALSE.	

6.3.10.1.22 Btsdk_MAP_GetMSETime_Func

Prototype	typedef BTBOOL (*Btsdk_MAP_GetMSETime_Func)(PBtSdkMAPMSETimeStru mse_time);	
Description	The **Btsdk_MAP_GetMSETime_Func** function prototype is the prototype of application defined callback function used to return the local Time basis of the MSE and its UTC offset. It is called after the MSE server receiving the GetMessageListing request.	
Parameters	*mse_time*	[out] Return the local Time basis of the MSE and its UTC offset. The format is "YYYYMMDDTHHMMSS±hhmm". If the server could not get current UTC time, the offset is not required. So the type is "YYYYMMDDTHHMM".
Return:	If the function succeeds, the return value is BTSDK_TRUE. If the function fails, the return value is BTSDK_FALSE.	

6.3.10.1.23 Btsdk_RegisterMASService

Prototype	BTSVCHDL Btsdk_RegisterMASService(BTUINT8 *svc_name, PBtSdkLocalMASServerAttrStru svr_attr, PBtSdkMASSvrCBStru cb_funcs);	
Description	The **Btsdk_RegisterMASService** function registers MAS service record to SDK service database and then activates it.	
Parameters	*svc_name*	[in] User friendly name of the new service. It shall be a null-terminated UTF-8 string. It can't be NULL. Its length shall be limited within BTSDK_SERVICENAME_MAXLENGTH, including the terminated '\0'.
	svr_attr	[in] Specifies attribute of the new service. All members of this structure must be set. It may be a NULL pointer, and the the default path delimiter is '/'; the default root directory is represented by a path delimiter; the default MASInstanceID is the smallest one within 0 - 255 that different from those of the existing MAS service instances.
	cb_funcs	[in] Pointer to the callback functions defined for this service.
Return:	If the function succeeds, the return value is the handle of the new service record. If the function fails, the return value is BTSDK_INVALID_HANDLE.	

Remarks

Before calling *Btsdk_RegisterMASService*, the service database must be initialized by a previous successful call to *Btsdk_Init*.

Currently, only one MAS service record is allowed at a time. That is, if the application calls the *Btsdk_RegisterMASService* function twice, the second call will first remove the first MAS service record and then add a new MAS service record.

6.3.10.1.24 Btsdk_MAPRegisterSvrCallback

Prototype	BTINT32 Btsdk_MAPRegisterSvrCallback (BTSVCHDL svc_hdl, PBtSdkMASSvrCBStru cb_funcs);	
Description	The **Btsdk_MAPRegisterSvrCallback** function is to register implementation dependent callback functions required by the MAS server.	
Parameters	*svc_hdl*	[in] The handle of the MAS service.
	cb_funcs	[in] Pointer to the callback functions of service.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code.	

Remarks

If the application calls *Btsdk_MPRegisterSvrCallback* twice to register different callback functions for the same service handle, the second callback function will replace the first one.

6.3.10.1.25 Btsdk_RegisterMNSService

Prototype	BTSVCHDL Btsdk_RegisterMNSService(BTUINT8* svc_name, Btsdk_MNS_MessageNotification_Func st_func, PBtSdkMAPFileIORoutinesStru file_ios);	
Description	The **Btsdk_RegisterMNSService** function registers MNS service record to SDKservice database and then activates it.	
Parameters	*svc_name*	[in] User friendly name of the new service. It shall be a null-terminated UTF-8 string. It can't be NULL. Its length shall be limited within BTSDK_SERVICENAME_MAXLENGTH, include the terminated '\0'.
	st_func	[in] The callback function that is used to deal with event notification reported from MNS
	file_ios	[in] Pointer to the set of callback functions. Create_file, read_file, write_file, rewind_file and close_file functions must be realized
Return:	If the function succeeds, the return value is the handle of the new service record. If the function fails, the return value is BTSDK_INVALID_HANDLE.	

Remarks

6.3.10.1.26 Btsdk_MAPRegisterFileIORoutines

Prototype	BTINT32 Btsdk_MAPRegisterFileIORoutines(BTCONNHDL conn_hdl, PBtSdkMAPFileIORoutinesStru cb_funcs);	
Description	The **Btsdk_MAPRegisterFileIORoutines** function is to register callback function which deal with file access in MAS client. The function must be called before Btsdk_GetFolderList, Btsdk_GetMessageList, Btsdk_GetMessage and Btsdk_PushMessage functions are called in client. If the function has been called several times, BlueSoleil will save the last input callback functions.	
Parameters	*conn_hdl*	[in] To client, conn_hdl is the connection handle with server. To server, conn_hdl is the service handle or connection handle that is passed to application by connection event.
	cb_func	[in] Collection of callback functions. read_file, write_file, get_file_size and rewind_file functions must be realized.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code.	

Remarks

6.3.10.1.27 Btsdk_UnregisterMAPService

Prototype	BTINT32 Btsdk_UnregisterMAPService (BTSVCHDL svc_hdl);	
Description	The **Btsdk_UnregisterMAPService** function removes the current MNS and MAS service record from SDK service database.	
Parameters	*svc_hdl*	[in] The handle of MNS and MAS service.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code.	

Remarks

6.3.10.1.28 Btsdk_MAPRegisterStatusCallback

Prototype	BTINT32 Btsdk_MAPRegisterStatusCallback(BTCONNHDL conn_hdl, Btsdk_MAP_STATUS_INFO_CB *func);	
Description	The **Btsdk_MAPRegisterStatusCallback** function is to register callback function to deal with MAP tranferring file status information.	
Parameters	*conn_hdl*	[in] For MAP client, conn_hdl is the connection handle to server. For server, conn_hdl is the service handle or connection handle which is reported to connection event callback. If the parameter is an invalid handle, the callback pointed by parameter “func” will be set as the default callback function. It will be used for all connections which do not specify Btsdk_MAP_STATUS_INFO_CB callback function.
	func	[in] Function which deals with status of file access. If the parameter is set to NULL, it equals to unregister callback function.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code.	

Remarks

6.3.10.1.29 Btsdk_MAPSetNotificationRegistration

Prototype	BTINT32 Btsdk_MAPSetNotificationRegistration (BTCONNHDL conn_hdl, BTBOOL turn_on);	
Description	The **Btsdk_MAPSetNotificationRegistration** function is to switch message notification in MSE, realize the function of SetNotificationRegistration.	
Parameters	*conn_hdl*	[in] Connection handle with MSE
	turn_on	[in] Switch for message notification on MSE. It could be one of the values below: BTSDK_TRUE – open the function of message notification BTSDK_FALSE – close the function of message notification Other values which do not equal to BTSDK_FALSE have the same meaning of BTSDK_TRUE
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code.	

Remarks

6.3.10.1.30 Btsdk_MAPSendEvent

Prototype	BTINT32 Btsdk_MAPSendEvent (BTCONNHDL conn_hdl, PBtSdkMAPEvReportObjStru ev_obj);	
Description	The **Btsdk_MAPSendEvent** function is to notify the MCE about any change of a Messages-Listing on the MSE side.	
Parameters	*conn_hdl*	[in] The handle of the MNS connection to send the notification.
	ev_obj	[in] Specifies the content of the event report object.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code.	

Remarks

6.3.10.1.31 Btsdk_MAPSetFolder

Prototype	BTINT32 Btsdk_MAPSetFolder(BTCONNHDL conn_hdl, BTUINT8* folder);	
Description	The **Btsdk_MAPSetFolder** function is to update current directory in MSE, and realize the function of SetFolder. Macro Btsdk_MAPSetRoot and Btsdk_MAPBackFolder can be used to return to root directory or back to parent directory separately.	
Parameters	*conn_hdl*	[in] Connection handle with MSE
	folder	[in] A null-terminated ANSI string which specifies the name of the file. It contains path information. NULL or empty character array means root directory. “..”is used to represent father directory. Other character array is the name of subdirectory.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code.	

Remarks

6.3.10.1.32 Btsdk_MAPGetFolderList

Prototype	BTINT32 Btsdk_MAPGetFolderList (BTCONNHDL conn_hdl, PBtSdkMAPGetFolderListParamStru param, BTSDKHANDLE file_hdl);	
Description	The **Btsdk_MAPGetFolderList** function is to read list of subdirectories under current directory from MSE. And realize the function of GetFolderListing	
Parameters	*conn_hdl*	[in] Connection handle with MSE
	param	[in] Parameters for GetFolderListing request, include MaxListCount, ListStartOffset. [out] Return FolderListingSize. And the validity of each parameter depends on the flags which are set by param->mask.
	file_hdl	[in] File handle to save received of object of directory listing, It is returned by the callback function of Btsdk_CreateFile_Func.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code.	

Remarks

6.3.10.1.33 Btsdk_MAPGetMessageList

Prototype	BTINT32 Btsdk_MAPGetMessageList (BTCONNHDL conn_hdl, PBtSdkMAPGetMsgListParamStru param, BTSDKHANDLE file_hdl);	
Description	The **Btsdk_MAPGetMessageList** function is to read list of messages specific directory from MSE. And realize the function of GetMessageListing	
Parameters	*conn_hdl*	[in] Connection handle with MSE
	path	[in] A null-terminated string that specifies the directory used to be read messages from.
	param	[in] Parameters for GetMessageListing request, include: Folder, MaxListCount, ListStartOffset, SubjectLength, ParameterMask, FilterMessageType, FilterPeriodBegin, FilterPeriodEnd, FilterReadStatus, FilterRecipient, FilterOriginator, FilterPriority [out] Return NewMessage, MSETime, MessagesListingSize And the validity of each parameter depends on the flags which are set by param->mask.
	file_hdl	[in] File handle to save received message listing. It is returned by the callback function of Btsdk_CreateFile_Func.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code.	

Remarks

6.3.10.1.34 Btsdk_MAPGetMessage

Prototype	BTINT32 Btsdk_MAPGetMessage (BTCONNHDL conn_hdl, PBtSdkMAPGetMsgParamStru param, BTSDKHANDLE file_hdl);	
Description	The **Btsdk_MAPGetMessage** function is to read message object from MSE, and realize the function of GetMessage	
Parameters	*conn_hdl*	[in] Connection handle with MSE
	param	[in] Parameters for GetMessage request, include: Handle, Charset, Attachment, FractionRequest [out] Return FractionDeliver.
	file_hdl	[in] File handle to save received message object. It is returned by the callback function of Btsdk_CreateFile_Func
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code.	

Remarks

6.3.10.1.35 Btsdk_MAPSetMessageStatus

Prototype	BTINT32 Btsdk_MAPSetMessageStatus (BTCONNHDL conn_hdl, BTUINT8 *msg_hdl, BTUINT8 status);	
Description	The **Btsdk_MAPSetMessageStatus** function is to notify MSE to update the status of specific message, and realize the function of SetMessageStatus.	
Parameters	*conn_hdl*	[in] Connection handle with MSE
	msg_hdl	[in] A null-terminated UTF-8 string which specifies message handle with 16 hexadecimal digits.
	status	[in] New status of message. It could be one of the values below: BTSDK_MAP_MSG_SETST_READ – set the reading status to "has been read" BTSDK_MAP_MSG_SETST_UNREAD – set the reading status to "not read yet" BTSDK_MAP_MSG_SETST_DELETED – set the status of message to "deleted", then transfer the message to directory of "deleted" by MSE device. BTSDK_MAP_MSG_SETST_UNDELETED – set the status of message to "not deleted", then transfer the message from the directory of "deleted" to inbox.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code.	

Remarks

6.3.10.1.36 Btsdk_MAPPushMessage

Prototype	BTINT32 Btsdk_MAPPushMessage (BTCONNHDL conn_hdl, PBtSdkMAPPushMsgParamStru param, BTSDKHANDLE file_hdl);	
Description	The **Btsdk_MAPPushMessage** function is to send message to MSE, and realize the function of PushMessage.	
Parameters	*conn_hdl*	[in] Connection handle with MSE
	param	[in] Parameters for PushtMessage request, include: Folder, Transparent, Retry, Charset [out] Return Handle.
	file_hdl	[in] File handle to save message object which is waiting for sending. It is return by the callback function of Btsdk_OpenFile_Func
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code.	

Remarks

6.3.10.1.37 Btsdk_MAPUpdateInbox

Prototype	BTINT32 Btsdk_MAPUpdateInbox (BTCONNHDL conn_hdl);	
Description	The **Btsdk_MAPUpdateInbox** function is to inform MSE to update inbox, and realize the function of UpdateInbox	
Parameters	*conn_hdl*	[in] Connection handle with MSE
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code.	

Remarks

6.3.10.1.38 Btsdk_MAPCancelTransfer

Prototype	BTINT32 Btsdk_MAPCancelTransfer (BTCONNHDL conn_hdl);	
Description	The **Btsdk_MAPCancelTransfer** function is to cancel the file transferring operation	
Parameters	*conn_hdl*	[in] Connection handle with MSE
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code.	

Remarks

6.3.10.1.39 Btsdk_MAPStartEnumFolderList

Prototype	BTSDKHANDLE Btsdk_MAPStartEnumFolderList (Btsdk_ReadFile_Func func_read, Btsdk_RewindFile_Func func_rewind, BTSDKHANDLE file_hdl);	
Description	The **Btsdk_MAPStartEnumFolderList** function is used to start the analyzing process of FolderListing object.	
Parameters	*func_read*	[in] Function that is defined by application which is to read data from the file which is pointed by file_hdl
	func_rewind	[in] Function that is defined by application which is to move the file pointer to the original location.
	file_hdl	[in] File handle to save the MessageListing object which is waiting for analyzing. It is returned by the callback function of Btsdk_CreateFile_Func or Btsdk_OpenFile_Func.
Return:	Return handle which is used to identify the analyzing process uniquely.	

Remarks

This function and Btsdk_MAPEnumFolderList, Btsdk_MAPEndEnumFolderList are used together to analyze FolderListing object.

6.3.10.1.40 Btsdk_MAPEnumFolderList

Prototype	PBtSdkMAPFolderObjStru Btsdk_MAPEnumFolderList (BTSDKHANDLE enum_hdl, PBtSdkMAPFolderObjStru item);	
Description	The **Btsdk_MAPEnumFolderList** function is to read attributes of current directory item from the list of directories, and move to the next directory item on the list.	
Parameters	*enum_hdl*	[in] The handle which is returned by Btsdk_MAPStartEnumFolderList
	item	[out] Returned value of attributes of directory.
Return:	If there are still directory item which has not been read yet, the return value is "item" pointer. If all directory items have been read, the return value is NULL.	

Remarks

The function and Btsdk_MAPStartEnumFolderList, Btsdk_MAPEndEnumFolderList are used together to analyze FolderListing object.

6.3.10.1.41 Btsdk_MAPEndEnumFolderList

Prototype	void Btsdk_MAPEndEnumFolderList (BTSDKHANDLE enum_hdl);	
Description	The **Btsdk_MAPEndEnumFolderList** function is to stop the analyzing process for list of directory items, and release associating resources.	
Parameters	*enum_hdl*	[in] The handle which is returned by Btsdk_MAPStartEnumFolderListing
Return:		

Remarks

The function and Btsdk_MAPStartEnumFolderList, Btsdk_MAPEndEnumFolderList are used together to analyze FolderListing object.

After analyzing, the function must be called to release resources.

6.3.10.1.42 Btsdk_MAPStartEnumMessageList

Prototype	BTSDKHANDLE Btsdk_MAPStartEnumMessageList (Btsdk_ReadFile_Func func_read, Btsdk_RewindFile_Func func_rewind, BTSDKHANDLE file_hdl);	
Description	The **Btsdk_MAPStartEnumMessageList** function is to start the analyzing process for MessageListing object.	
Parameters	*func_read*	[in] Function that is defined by application which is to read data from the file which is pointed by file_hdl
	func_rewind	[in] Function that is defined by application which is to move the file pointer to the original location.
	file_hdl	[in] File handle to save the MessageListing object which is waiting for analyzing. It is returned by the callback function of Btsdk_CreateFile_Func or Btsdk_OpenFile_Func.
Return:	The return value is the handle which is to identify the analyzing process.	

Remarks

The function and Btsdk_MAPEnumMessageList, Btsdk_MAPEndEnumMessageList are used together to analyze MessageListing object.

6.3.10.1.43 Btsdk_MAPEnumMessageList

Prototype	PBtSdkMAPMsgObjStru Btsdk_MAPEnumMessageList (BTSDKHANDLE enum_hdl, PBtSdkMAPMsgObjStru item);	
Description	The **Btsdk_MAPEnumMessageList** function is to read attributes of current message on the list of messages, and move to the next message on the list.	
Parameters	*enum_hdl*	[in] The handle which is returned by Btsdk_MAPStartEnumMessageList
	item	[out] Returned value of attributes of message.
Return:	If there are still message which has not been read yet, the return value is "item" pointer. If all messages have been read, the return value is NULL.	

Remarks

The function and Btsdk_MAPStartEnumMessageList, Btsdk_MAPEndEnumMessageList are used together to analyze MessageListing object.

6.3.10.1.44 Btsdk_MAPEndEnumMessageList

Prototype	void Btsdk_MAPEndEnumFolderListing (BTSDKHANDLE enum_hdl);	
Description	The **Btsdk_MAPEndEnumMessageList** function is to stop the analyzing process for the list of messages, and release associating resources.	
Parameters	*enum_hdl*	[in] The handle which is returned by Btsdk_MAPStartEnumMessageList.
Return:		

Remarks

The function and Btsdk_MAPStartEnumMessageList, MAPBtsdk_EnumMessageList are used together to analyze MessageListing object.

After anayzing, the function must be called to release resources.

6.3.11 BLE Profile

6.3.11.1 General

6.3.11.1.1 Btsdk_GATTGetServices

Prototype	BTINT32 Btsdk_GATTGetServices(BTDEVHDL hDevice, BTUINT16 ServicesBufferCount, PBtsdkGATTServiceStru ServicesBuffer, BTUINT16* ServicesBufferActual, BTUINT32 Flags);	
Description	The **Btsdk_GATTGetServices** function gets all the primary services available for a server.	
Parameters	*hDevice*	[in] Handle to the Bluetooth device from which to obtain the list of primary services.
	ServicesBufferCount	[in] The number of elements allocated for the ServicesBuffer parameter.
	ServicesBuffer	[out, optional] Pointer to an array of BtsdkGATTServiceStru structures that contains *ServicesBufferCount* elements.
	ServicesBufferActual	[out] The return value will be set to the actual number of services were returned in *ServicesBuffer* parameter.
	Flags	[in] Flags to modify the behavior of Btsdk_GATTGetServices. Refer to Table 11.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

Before calling *Btsdk_GATTGetService*, the client must be initialized by a previous successful call to *Btsdk_Init*.

When ServicesBufferCount is 0 and ServicesBuffer is NULL, *Btsdk_GATTGetServices* will get the actual number of services were returned in the ServicesBuffer parameter.

6.3.11.1.2 Btsdk_GATTGetIncludedServices

Prototype	BTINT32 Btsdk_GATTGetIncludedServices(BTDEVHDL hDevice, PBtsdkGATTServiceStru ParentService, BTUINT16 IncludedServicesBufferCount, PBtsdkGATTServiceStru IncludedServicesBuffer, BTUINT16* IncludedServicesBufferActual, BTUINT32 Flags);	
Description	The **BluetoothGATTGetIncludedServices** function gets all the included services available for a given service either from directly or from cache.	
Parameters	*hDevice*	[in] Handle to the Bluetooth device.
	ParentService	[in] Pointer to the parent service of the included services to be retrieved.
	IncludedServicesBufferCount	[in] The number of elements allocated for the IncludedServicesBuffer parameter.
	IncludedServicesBuffer	[out, optional] Pointer to buffer into which to return included services.
	IncludedServicesBufferActual	[out] Pointer to buffer into which the actual number of included services were returned in the IncludedServicesBuffer parameter.
	Flags	[in] Flags to modify the behavior of Btsdk_GATTGetIncludedServices. Refer to Table 11.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

Before calling *Btsdk_GATTGetIncludedServices*, the client must be initialized by a previous successful call to *Btsdk_Init*.

When IncludedServicesBufferCount is 0 and IncludedServicesBuffer is NULL, the actual number of services were returned in the IncludedServicesBuffer parameter.

One of the following conditions occurred will return BTSDK_LE_ER_INVALID_PARAMETER:

- hDevice is BTSDK_INVALID_HANDLE.
- ParentService is NULL.

6.3.11.1.3 Btsdk_GATTGetCharacteristics

Prototype	BTINT32 Btsdk_GATTGetCharacteristics(BTDEVHDL hDevice, PBtsdkGATTServiceStru Service, BTUINT16 CharacteristicsBufferCount, PBtsdkGATTCharacteristicStru CharacteristicsBuffer, BTUINT16* CharacteristicsBufferActual, BTUINT32 Flags);	
Description	The **BluetoothGATTGetCharacteristics** function gets all the characteristics available for the specified service.	
Parameters	*hDevice*	[in] Handle to the Bluetooth device.
	Service	[in] Pointer to the parent service of the characteristics to be retrieved.
	CharacteristicsBufferCount	[in] The number of elements allocated for the CharacteristicsBuffer parameter.
	CharacteristicsBuffer	[out, optional] Pointer to buffer into which to return characteristics.
	CharacteristicsBufferActual	[out] Pointer to buffer into which the actual number of characteristics were returned in the CharacteristicsBuffer parameter.
	Flags	[in] Flags to modify the behavior of Btsdk_GATTGetCharacteristics. Refer to Table 11.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

Before calling *Btsdk_GATTGetCharacteristics*, the client must be initialized by a previous successful call to *Btsdk_Init*.

The parent service must be present in the cache, otherwise this function will fail. The parent service must be got by a previous call of *Btsdk_GATTGetServices*, or *Btsdk_GATTGetIncludedServices.*

When CharacteristicsBufferCount is 0 and CharacteristicsBuffer is NULL, the actual number of characteristics were returned in CharacteristicsBuffer parameter.

One of the following conditions occurred will return BTSDK_LE_ER_INVALID_PARAMETER:

- hDevice is BTSDK_INVALID_HANDLE.
- Service is NULL.

6.3.11.1.4 Btsdk_GATTGetDescriptors

Prototype	BTINT32 Btsdk_GATTGetDescriptors(BTDEVHDL hDevice, PBtsdkGATTCharacteristicStru Characteristic, BTUINT16 DescriptorsBufferCount, PBtsdkGATTDescriptorStru DescriptorsBuffer, BTUINT16* DescriptorsBufferActual, BTUINT32 Flags);	
Description	The **BluetoothGATTGetDescriptors** function gets all the descriptors available for the specified characteristic.	
Parameters	*hDevice*	[in] Handle to the Bluetooth device.
	Characteristic	[in] Pointer to the parent characteristic of the descriptors to be retrieved.
	DescriptorsBufferCount	[in] The number of elements allocated for the DescriptorsBuffer parameter.
	DescriptorsBuffer	[out, optional] Pointer to buffer into which to return descriptors.
	DescriptorsBufferActual	[out] Pointer to buffer into which the actual number of descriptors were returned in the DescriptorsBuffer parameter.
	Flags	[in] Flags to modify the behavior of Btsdk_GATTGetDescriptors. Refer to Table 11.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

Before calling *Btsdk_GATTGetDescriptors*, the client must be initialized by a previous successful call to *Btsdk_Init*.

The parent characteristic must be present in the cache, otherwise this function will fail. The parent characteristic must be got by a previous call of *Btsdk_GATTGetCharacteristics*.

When DescriptorsBufferCount is 0 and DescriptorsBuffer is NULL, *Btsdk_GATTGetCharacteristics* will get the actual number of descriptors were returned in the DescriptorsBuffer parameter.

One of the following conditions occurred will return BTSDK_LE_ER_INVALID_PARAMETER:

- hDevice is BTSDK_INVALID_HANDLE.
- Characteristic is NULL.

6.3.11.1.5 Btsdk_GATTGetCharacteristicValue

Prototype	BTINT32 Btsdk_GATTGetCharacteristicValue(BTDEVHDL hDevice, PBtsdkGATTCharacteristicStru Characteristic, BTUINT16 CharacteristicValueDataSize, PBtsdkGATTCharacteristicValueStru CharacteristicValue, BTUINT16* CharacteristicValueSizeRequired, BTUINT32 Flags);	
Description	The **Btsdk_GATTGetCharacteristicValue** function gets the value of the specified characteristic.	
Parameters	*hDevice*	[in] Handle to the Bluetooth device.
	Characteristic	[in] Pointer to the parent characteristic of the characteristic value to be retrieved.
	CharacteristicValueDataSize	[in] The number of bytes allocated for the CharacteristicValue parameter.
	CharacteristicValue	[out, optional] Pointer to buffer into which to return the characteristic value.
	CharacteristicValueSizeRequired	[out] Pointer to buffer into which to store the number of bytes needed to return data in the buffer pointed to by CharacteristicValue.
	Flags	[in] Flags to modify the behavior of Btsdk_GATTGetCharacteristicValue. Refer to Table 11.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

Before calling *Btsdk_GATTGetCharacteristicValue*, the client must be initialized by a previous successful call to *Btsdk_Init*.

The parent characteristic must be present in the cache, otherwise this function will fail. The parent characteristic must be got by a previous call of *Btsdk_GATTGetCharacteristics*.

When CharacteristicValueDataSize is 0 and CharacteristicValue is NULL, *Btsdk_GATTGetCharacteristicValue* will get he number of bytes needed to return data in the buffer pointed to by CharacteristicValue.

One of the following conditions occurred will return BTSDK_LE_ER_INVALID_PARAMETER:

- hDevice is BTSDK_INVALID_HANDLE.
- Characteristic is NULL.

6.3.11.1.6 Btsdk_GATTGetDescriptorValue

Prototype	BTINT32 Btsdk_GATTGetDescriptorValue(BTDEVHDL hDevice, PBtsdkGATTDescriptorStru Descriptor, BTUINT16 DescriptorValueDataSize, PBtsdkGATTDescriptorValueStru DescriptorValue, BTUINT16* DescriptorValueSizeRequired, BTUINT32 Flags);	
Description	The **Btsdk_GATTGetDescriptorValue** function gets the value of the specified descriptor.	
Parameters	*hDevice*	[in] Handle to the Bluetooth device.
	Descriptor	[in] Pointer to the parent descriptor of the descriptor value to be retrieved.
	DescriptorValueDataSize	[in] The number of bytes allocated for the DescriptorValue parameter.
	DescriptorValue	[out, optional] Pointer to buffer into which to return the descriptor value.
	DescriptorValueSizeRequired	[out] Pointer to buffer into which to store the number of additional bytes needed to return data in the buffer pointed to by DescriptorValue.
	Flags	[out] Flags to modify the behavior of Btsdk_GATTGetDescriptorValue. Refer to Table 11.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

Before calling *Btsdk_GATTGetDescriptorValue*, the client must be initialized by a previous successful call to *Btsdk_Init*.

The parent descriptor must be present in the cache, otherwise this function will fail. The parent descriptor must be got by a previous call of *Btsdk_GATTGetDescriptors.*

When DescriptorValueDataSize is 0 and DescriptorValue is NULL, *Btsdk_GATTGetCharacteristics* will get the actual number of additional bytes needed to return data in the buffer pointed to by DescriptorValue.

One of the following conditions occurred will return BTSDK_LE_ER_INVALID_PARAMETER:

- hDevice is BTSDK_INVALID_HANDLE.
- Descriptor is NULL.

6.3.11.1.7 Btsdk_GATTBeginReliableWrite

Prototype	BTINT32 Btsdk_GATTBeginReliableWrite(BTDEVHDL hDevice, BTSDKHANDLE *ReliableWriteContext, BTUINT32 Flags);	
Description	The **Btsdk_GATTBeginReliableWrite** function specifies that reliable writes are about to begin.	
Parameters	*hDevice*	[in] Handle to the Bluetooth device.
	ReliableWriteContext	[out] Pointer to the context describing the reliable write operation.
	Flags	[in] Flags to modify the behavior of Btsdk_GATTBeginReliableWrite. Refer to Table 11.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

Before calling *Btsdk_GATTBeginReliableWrite*, the client must be initialized by a previous successful call to *Btsdk_Init*.

Unsupport currently.

6.3.11.1.8 Btsdk_GATTSetCharacteristicValue

Prototype	BTINT32 Btsdk_GATTSetCharacteristicValue(BTDEVHDL hDevice, PBtsdkGATTCharacteristicStru Characteristic, PBtsdkGATTCharacteristicValueStru CharacteristicValue, BTSDKHANDLE ReliableWriteContext, BTUINT32 Flags);	
Description	The **Btsdk_GATTSetCharacteristicValue** function writes the specified characteristic value to the Bluetooth device.	
Parameters	*hDevice*	[in] Handle to the Bluetooth device.
	Characteristic	[in] Pointer to the parent characteristic.
	CharacteristicValue	[in] Pointer to the characteristic value.
	ReliableWriteContext	[in] The context describing the reliable write operation returned from a previous call to Btsdk_GATTBeginReliableWrite.
	Flags	[in] Flags to modify the behavior of Btsdk_GATTSetCharacteristicValue. Refer to Table 12.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

Before calling *Btsdk_GATTSetCharacteristicValue*, the client must be initialized by a previous successful call to *Btsdk_Init*.

Calling *Btsdk_GATTSetCharacteristicValue* after *Btsdk_GATTBeginReliableWrite*, notifies the remote Bluetooth device to store this request into a prepare queue on the device. Aresponse is required. If a response is not required, then characteristic value must result in a Write Without response PDU on or less than the size of the ATT_MTU. Is signing is required, then the operation must not require a response, and must not occur over a secure channel.

The parent characteristic and characteristic value must be returned from a previous call to *Btsdk_GATTGetCharacteristics*, and must not be alerted. Behavior is undefined if the caller does this.

6.3.11.1.9 Btsdk_GATTEndReliableWrite

Prototype	BTINT32 Btsdk_GATTEndReliableWrite(BTDEVHDL hDevice, BTSDKHANDLE ReliableWriteContext, BTUINT32 Flags);	
Description	The Btsdk_GATTEndReliableWrite function specifies the end of reliable writes, and the writes should be committed.	
Parameters	*hDevice*	[in] Handle to the Bluetooth device.
	ReliableWriteContext	[in] The context describing the reliable write operation returned from a previous call to BluetoothGATTBeginReliableWrite.
	Flags	[in] Flags to modify the behavior of Btsdk_GATTEndReliableWrite. Refer to Table 11.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

Before calling *Btsdk_GATTEndReliableWrite*, the client must be initialized by a previous successful call to *Btsdk_Init*.

Unsupport currently.

6.3.11.1.10 Btsdk_GATTAbortReliableWrite

Prototype	BTINT32 Btsdk_GATTAbortReliableWrite(BTDEVHDL hDevice, BTSDKHANDLE ReliableWriteContext, BTUINT32 Flags);	
Description	The **Btsdk_GATTAbortReliableWrite** function specifies the end of reliable write procedures, and the writes should be aborted.	
Parameters	*hDevice*	[in] Handle to the Bluetooth device.
	ReliableWriteContext	[in] The context describing the reliable write operation returned from a previous call to BluetoothGATTBeginReliableWrite.
	Flags	[in] Flags to modify the behavior of Btsdk_GATTAbortReliableWrite. Refer to Table 11.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

Before calling *Btsdk_GATTAbortReliableWrite*, the client must be initialized by a previous successful call to *Btsdk_Init*.

Unsupport currently.

6.3.11.1.11 Btsdk_GATTSetDescriptorValue

Prototype	BTINT32 Btsdk_GATTSetDescriptorValue(BTDEVHDL hDevice, PBtsdkGATTDescriptorStru Descriptor, PBtsdkGATTDescriptorValueStru DescriptorValue, BTUINT32 Flags);	
Description	The BluetoothGATTSetDescriptorValue function writes the specified descriptor value to the Bluetooth device.	
Parameters	*hDevice*	[in] Handle to the Bluetooth device.
	Descriptor	[in] Pointer to the parent descriptor.
	DescriptorValue	[in] Pointer to the descriptor's value.
	Flags	[in] Flags to modify the behavior of Btsdk_GATTSetDescriptorValue. Refer to Table 13.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

Before calling *Btsdk_GATTSetDescriptorValue*, the client must be initialized by a previous successful call to *Btsdk_Init*.

The parent descriptor must be present in the cache, otherwise this function will fail. The parent descriptor must be got by a previous call of *Btsdk_GATTGetDescriptors.*

One of the following conditions occurred will return BTSDK_LE_ER_INVALID_PARAMETER:

- hDevice is BTSDK_INVALID_HANDLE.
- Descriptor is NULL.

6.3.11.1.12 Btsdk_GATTCloseSession

Prototype	BTINT32 Btsdk_GATTCloseSession(BTDEVHDL hDevice, BTUINT32 Flags);	
Description	The **Btsdk_GATTCloseSession** function terminates a GATT session.	
Parameters	*hDevice*	[in] Handle to the Bluetooth device.
	Flags	[in] Flags to modify the behavior of Btsdk_GATTCloseSession. Refer to Table 11.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

Before calling *Btsdk_GATTCloseSession*, the client must be initialized by a previous successful call to *Btsdk_Init*.

A GATT session is created at the time the first direct device call is made. By default the session terminates after 30 seconds. Callers can use *Btsdk_GATTCloseSession* to terminate the session earlier than the 30-second interval to release internal resources held for the duration of the session.

6.3.11.1.13 Btsdk_GATTRegisterEvent

Prototype	BTINT32 Btsdk_GATTRegisterEvent(BTDEVHDL hDevice, BTSDK_GATT_EVENT_TYPE EventType, BTLPVOID EventParameter, FNBLUETOOTH_GATT_NOTIFICATION_CALLBACK *Callback, BTLPVOID CallbackContext, BTSDKHANDLE* pEventHandle, BTUINT32 Flags);	
Description	The **Btsdk_GATTRegisterEvent** function registers a routine to be called back during a characteristic value change event on the given characteristic identified by its characteristic handle.	
Parameters	*hDevice*	[in] Handle to the Bluetooth device.
	EventType	[in] A value from BTSDK_GATT_EVENT_TYPE. Currently, only CharacteristicValueChangedEvent is supported.
	EventParameter	[in] Pointer to a BtsdkGATTCharacteristicStru structure.
	Callback	[in] The routine to call when the Characteristic value changes.
	CallbackContext	[in] Context to pass to Callback.
	pEventHandle	[out] Pointer to buffer to receives a handle for the registration. Profile drivers must pass this handle when calling Btsdk_GATTUnregisterEvent.
	Flags	[in] Flags to modify the behavior of Btsdk_GATTRegisterEvent. Refer to Table 11.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

Before calling *Btsdk_GATTRegisterEvent*, the client must be initialized by a previous successful call to *Btsdk_Init*.

6.3.11.1.14 Btsdk_GATTUnregisterEvent

Prototype	BTINT32 Btsdk_GATTUnregisterEvent(BTSDKHANDLE EventHandle, BTUINT32 Flags);	
Description	The **Btsdk_GATTUnregisterEvent** function unregisters the given characteristic value change event.	
Parameters	*EventHandle*	[in] Handle returned from a previous call to Btsdk_GATTUnregisterEvent.
	Flags	[in] Flags to modify the behavior of Btsdk_GATTUnregisterEvent. Refer to Table 11.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is an error code listed in Table 1.	

Remarks

Before calling *Btsdk_GATTUnregisterEvent*, the client must be initialized by a previous successful call to *Btsdk_Init*.

6.3.11.1.15 FNBLUETOOTH_GATT_NOTIFICATION_CALLBACK

Prototype	typedef void FNBLUETOOTH_GATT_NOTIFICATION_CALLBACK(BTUINT16 ChangedAttributeHandle, BTUINT32 CharacteristicValueDataSize, PBtsdkGATTCharacteristicValueStru CharacteristicValue, BTLPVOID Context);	
Description	User implement a Bluetooth GATT notification callback to be called whenever the value of a specific characteristic changes.	
Parameters	*ChangedAttributeHandle*	[in] Handle to changed attribute.
	CharacteristicValueDataSize	[in] Size of the characteristic.
	CharacteristicValue	[in] Pointer to the characteristic value.
	Context	[in, optional] The context specified by the user in the CallbackContext parameter of the Btsdk_GATTRegisterEvent function when the user registered the GATT callback function.
Return:		

Remarks

6.3.11.2 Function Flag

The **flags** member in Btsdk_GATTGetXXX function can be one or more of those values.

Table 11:

Value	Description
BTSDK_GATT_FLAG_NONE	The client does not have specific GATT requirements (default).
BTSDK _GATT_FLAG_CONNECTION_ENCRYPTED	The client requests the data to be transmitted over an encrypted channel.
BTSDK_GATT_FLAG_CONNECTION_AUTHENTICATED	The client requests the data to be transmitted over an authenticated channel.
BTSDK _GATT_FLAG_FORCE_READ_FROM_DEVICE	The XXX value is to be read directly from the device. This overwrites the one in the cache if one is already present.
BTSDK _GATT_FLAG_FORCE_READ_FROM_CACHE	The XXX value is to be read from the cache (regardless of whether it is present in the cache or not).

The flags member values for Btsdk_GATTSetCharacteristicValue function.
Table12:

Value	Description
BTSDK_GATT_FLAG_NONE	The client does not have specific GATT requirements (default).
BTSDK_GATT_FLAG_CONNECTION_ENCRYPTED	The client requests the data to be transmitted over an encrypted channel.
BTSDK_GATT_FLAG_CONNECTION_AUTHENTICATED	The client requests the data to be transmitted over an authenticated channel.
BTSDK_GATT_FLAG_WRITE_WITHOUT_RESPONSE	Write without response.
BTSDK_GATT_FLAG_SIGNED_WRITE	Signed write. Bluetooth stack must use with BTSDK_GATT_FLAG_WRITE_WITHOUT_RESPONSE in order to produce signed write without a response.

The flags member values for Btsdk_GATTSetDescriptorValue function.
Table13:

Value	Description
BTSDK_GATT_FLAG_NONE	The client does not have specific GATT requirements (default).
BTSDK_GATT_FLAG_CONNECTION_ENCRYPTED	The client requests the data to be transmitted over an encrypted channel.
BTSDK_GATT_FLAG_CONNECTION_AUTHENTICATED	The client requests the data to be transmitted over an authenticated channel.

7. Local Service Specific API Reference

7.1 Constant Reference

7.1.1 Error Codes

The following table provides a list of local service specific error codes. They are returned by many BlueSoleil functions when they fail.

Name	Value	Description
BTSDK_ER_SWRAPINDX	0X0700	This is the swrap error.
BTSDK_ER_COM_INUSED	0X0701	This com port is in used by other operation.
BTSDK_ER_COM_OPENNOTCOMPLETED	0X0702	Can not open the com port.

Table 11: Local Service Specific Error Codes.

7.2 Data Structures

7.2.1 BtSdkLocalServerAttrStru

Definition	typedef struct _BtSdkLocalServerAttrStru { BTUINT16 mask; BTUINT16 service_class; BTUINT8 svc_name[BTSDK_SERVICENAME_MAXLENGTH]; BTUINT16 security_level; BTUINT16 author_method; BTLPVOID ext_attributes; BTUINT32 app_param; } BtSdkLocalServerAttrStru, * PBtSdkLocalServerAttrStru;	
Description	The structure BtSdkLocalServerAttrStru contains information about a local service record.	
Members	*mask*	A set of flags which specify members to retrieve or set.
	Service_class	Type of the service record. It can be one of the values listed in the Table 2.
	svc_name	User-friendly name of this service record. This string is coded in UTF-8 format. Set *mask* to BTSDK_RSAM_SERVICENAME to use *svc_name*.
	security_level	Specify whether this local service's authorization mode. It can be Authorization, Authentication, Encryption, None
	author_method	Specify local service's authorization way. This value is combined with security level "Authorization". It can be Accept, Prompt, Reject (untrusted device)
	ext_attributes	Profile specific attributes. It must be cast to a pointer to a structure decided by the service type. See following table. Set *mask* to BTSDK_LSAM_SECURITYLEVEL to use *ext_attributes*.
	app_param	User defined parameters

The ***mask*** member can be one or more of these values

Value	Description
BTSDK_LSAM_SERVICENAME	Retrieves or set the *svc_name* member.
BTSDK_LSAM_SECURITYLEVEL	Retrieves or set the *security_level* member.
BTSDK_LSAM_AUTHORMETHOD	Retrieves or set the *author_method* member.
BTSDK_RSAM_EXTATTRIBUTES	Retrieves or set the *ext_attributes* member.
BTSDK_LSAM_APPPARAM	Retrieves or set the *app_param* member.

The *security_level* member can be one or more of these values

Value	Description
BTSDK_SSL_NO_SECURITY	Local service has no security.
BTSDK_SSL_AUTHENTICATION	Local service needs authentication, for OBEX service only.
BTSDK_SSL_AUTHORIZATION	Local service needs authorization.
BTSDK_SSL_ENCRYPTION	Local service is encrypted for data transfer.
BTSDK_SSL_AUTHENTICATION_MITM	Authentication against MITM
BTSDK_DEFAULT_SECURITY	This is combined of BTSDK_SSL_AUTHORIZATION BTSDK_SSL_AUTHENTICATION and BTSDK_SSL_ENCRYPTION

The *author_method* member can be one or more of these values.

Value	Description
BTSDK_AUTHORIZATION_ACCEPT	
BTSDK_AUTHORIZATION_REJECT	
BTSDK_AUTHORIZATION_PROMPT	

The ***ext_attributes*** member can be a pointer to one of these structures

Value of ***service_class***	Type of ***ext_attributes***
BTSDK_CLS_SERIAL_PORT	BtSdkLocalSPPServerAttrStru

7.2.2 BtSdkLocalSPPServerAttrStru

Definition	typedef struct _ BtSdkLocalSPPServerAttrStru { BTUINT32 size; BTUINT16 mask; BTUINT8 com_index; } BtSdkLocalSPPServerAttrStru, *PBtSdkLocalSPPServerAttrStru;	
Description	The structure BtSdkLocalSPPServerAttrStru describes the attribute information of local SPP service.	
Members	*size*	Size of the structure, in bytes.
	mask	A set of flags which specify members to retrieve or set..
	com_index	The COM port number assigned to this local SPP service.

The ***mask*** member should be set to this value

Value	Description
BTSDK_LSPPSAM_COMINDEX	

7.3 API Functions

7.3.1 Btsdk_AddServer

Prototype	BTSVCHDL Btsdk_AddServer (BtSdkLocalServerAttrStru * attribute);	
Description	The **Btsdk_AddServer** function adds a service record to SDK service database. Remote client cannot access the new record until it is activated.	
Parameters	attribute	[in] Attributes of the new service record..
Return:	Handle assigned to the new service record if it is added successfully. BTSDK_INVALID_HANDLE if the new record can't be created.	

Remarks

7.3.2 Btsdk_ RemoveServer

Prototype	BTINT32 Btsdk_RemoveServer (BTSVCHDL svc_hdl);	
Description	The **Btsdk_RemoveServer** function removes a service record from SDK service database.	
Parameters	svc_hdl	[in] handle to the service record to be removed.
Return:	BTSDK_OK for success, other for error code.	

Remarks

7.3.3 Btsdk_ UpdateServerAttributes

Prototype	BTINT32 Btsdk_UpdateServerAttributes (BTSVCHDL svc_hdl, BtSdkLocalServerAttrStru *attribute);	
Description	The **Btsdk_UpdateServerAttributes** function modifies the attributes of a service record. It will restart an active service record with the new attributes.	
Parameters	svc_hdl	[in] handle to the service record to be modified.
	attribute	[in] The new attributes. attribute->service_class is ignored when calling this function. That is, modification of the service class is disallowed.
Return:	BTSDK_OK for success other for error code.	

Remarks

7.3.4 Btsdk_ StartServer

Prototype	BTINT32 Btsdk_StartServer (BTSVCHDL svc_hdl);	
Description	The **Btsdk_StartServer** function activates a service record so that a remote client can access it.	
Parameters	svc_hdl	[in] Handle to the service record to be activated.
Return:	BTSDK_OK for success other for error code.	

Remarks

7.3.5 Btsdk_ StopServer

Prototype	BTINT32 Btsdk_StopServer (BTSVCHDL svc_hdl);	
Description	The **Btsdk_SopServer** function deactivates a service record so that remote client cannot access it.	
Parameters	svc_hdl	[in] Handle to the service record to be activated.
Return:	BTSDK_OK for success other for error code.	

Remarks

7.3.6 Btsdk_ GetServerStatus

Prototype	BTINT32 Btsdk_GetServerStatus (BTSVCHDL svc_hdl, BTUINT16 *status);	
Description	The **Btsdk_GetServerStatus** function gets the current status of a service record.	
Parameters	svc_hdl	[in] Handle to the service record to be activated.
	status	[out] Pointer of server status. The return value can be one or more of the values listed below.
Return:	BTSDK_OK for success other for error code.	

Remarks

The following table provides a list of flags that specify the Bluetooth service status.

Name	Description
BTSDK_SERVER_STARTED	
BTSDK_SERVER_STOPPED	
BTSDK_SERVER_CONNECTED	

Table 12: Bluetooth service Status

7.3.7 Btsdk_ GetServerAttributes

Prototype	BTINT32 Btsdk_GetServerAttributes (BTSVCHDL svc_hdl, BtSdkLocalServerAttrStru * attribute);	
Description	The **Btsdk_GetServerAttributes** function gets the current attributes of a service record.	
Parameters	svc_hdl	[in] Handle to the service record to be activated.
	attribute	[out] Pointer to BtSdkLocalServerAttrStru to receive the attributes. .
Return:	BTSDK_OK for success other for error code.	

Remarks

7.3.8 Btsdk_ StartEnumLocalServer

Prototype	BTSDKHANDLE Btsdk_StartEnumLocalServer (void);	
Description	The **Btsdk_StartEnumLocalServeS** function starts to search all local services.	
Parameters		
Return:	If the function succeeds, the return value is a search handle used in a subsequent call to Btsdk_EnumLocalServer and Btsdk_EndEnumLocalServer. If the function fails, the return value is BTSDK_INVALID_HANDLE.	

Remarks

7.3.9 Btsdk_ EnumLocalServer

Prototype	BTSDKHANDLE Btsdk_EnumLocalServer (BTSDKHANDLE enum_handle, BtSdkLocalServerAttrStru *attribute);	
Description	The **Btsdk_StartEnumLocalServes** function continues to search the SDK service database for a local service.	
Parameters	*enum_handle*	[in] Search handle returned by a previous call to Btsdk_StartEnumLocalServer.
	attribute	[out] Pointer to the BtSdkLocalServerAttrStru structure that receives information about the found service.
Return:	If the function succeeds, the return value is the handle specified the found service.If no matching service can be found, the return value is BTSDK_INVALID_HANDLE.	

Remarks

7.3.10 Btsdk_ EndEnumLocalServer

Prototype	BTSDKHANDLE Btsdk_EndEnumLocalServer (BTSDKHANDLE enum_handle);	
Description	The **Btsdk_EndEnumLocalServes** function closes the specified search handle. The Btsdk_EnumLocalServer function uses the search handle to find local servers.	
Parameters	*enum_handle*	[in] Search handle returned by a previous call to Btsdk_StartEnumLocalServer.
Return:	If the function succeeds, the return value is BTSDK_OK. If the function fails, the return value is the error code.	

Remarks

7.3.11 Btsdk_ GetPrivateProfileString

Prototype	BTUINT32 Btsdk_GetPrivateProfileString (BTINT8 *lpAppName, BTINT8 *lpKeyName, BTINT8 *lpDefault, BTINT8 *lpReturnedString, BTUINT32 nSize, BTINT8 *lpFileName);	
Description	The **Btsdk_GetPrivateProfileString** function retrieves a string from the specified section in an initialization file.	
Parameters	*lpAppName*	[in] Pointer to a null-terminated string that specifies the name of the section containing the key name. If this parameter is NULL, the function copies all section names in the file to the supplied buffer.
	lpKeyName	[in] Pointer to the null-terminated string specifying the name of the key whose associated string is to be retrieved. If this parameter is NULL, all key names in the section specified by the *lpAppName* parameter are copied to the buffer specified by the *lpReturnedString* parameter.
	lpDefault	[in] Pointer to a null-terminated default string. If the *lpKeyName* key cannot be found in the initialization file, this function copies the default string to the *lpReturnedString* buffer. If this parameter is NULL, the default is an empty string, "".
	lpReturnedString	[out] Pointer to the buffer that receives the retrieved string.
	nSize	[in] Size of the buffer pointed to by the *lpReturnedString* parameter, in characters.
	lpFileName	[in] Pointer to a null-terminated string that specifies the name of the initialization file. If this parameter does not contain a full path to the file, the system searches for the file in the Windows directory.
Return:	The return value is the number of characters copied to the buffer, not including the terminating null character.	

Remarks

7.3.12 Btsdk_ WritePrivateProfileString

Prototype	BOOL Btsdk_GetPrivateProfileString (BTINT8 *lpAppName, BTINT8 *lpKeyName, BTINT8 *lpString, BTINT8 *lpFileName);	
Description	The **Btsdk_WritePrivateProfileString** function retrieves a string from the specified section in an initialization file.	
Parameters	*lpAppName*	[in] Pointer to a null-terminated string containing the name of the section to which the string will be copied. If the section does not exist, it is created. The name of the section is case-independent; the string can be any combination of uppercase and lowercase letters.
	lpKeyName	[in] Pointer to the null-terminated string containing the name of the key to be associated with a string. If the key does not exist in the specified section, it is created. If this parameter is NULL, the entire section, including all entries within the section, is deleted.
	lpString	[in] Pointer to a null-terminated string to be written to the file. If this parameter is NULL, the key pointed to by the *lpKeyName* parameter is deleted.
	lpFileName	[in] Pointer to a null-terminated string that specifies the name of the initialization file.
Return:	If the function successfully copies the string to the initialization file, the return value is nonzero.	

Remarks

7.3.13 Btsdk_ GetPrivateProfileInt

Prototype	BTINT32 Btsdk_GetPrivateProfileInt (BTINT8 *lpAppName, BTINT8 *lpKeyName, BTINT32 nDefault, BTINT8 *lpFileName);	
Description	The **Btsdk_GetPrivateProfileInt** function retrieves an integer associated with a key in the specified section of an initialization file.	
Parameters	*lpAppName*	[in] Pointer to a null-terminated string specifying the name of the section in the initialization file.
	lpKeyName	[in] Pointer to the null-terminated string specifying the name of the key whose value is to be retrieved. This value is in the form of a string; the function converts the string into an integer and returns the integer.
	nDefault	[in] Default value to return if the key name cannot be found in the initialization file.
	lpFileName	[in] Pointer to a null-terminated string that specifies the name of the initialization file. If this parameter does not contain a full path to the file, the system searches for the file in the Windows directory.
Return:	The return value is the integer equivalent of the string following the specified key name in the specified initialization file. If the key is not found, the return value is the specified default value.	

Remarks

7.3.14 Btsdk_ WritePrivateProfileInt

Prototype	BTINT32 Btsdk_WritePrivateProfileInt (BTINT8 *lpAppName, BTINT8 *lpKeyName, BTINT32 nNumber, BTINT8 *lpFileName);	
Description	The **Btsdk_WritePrivateProfileInt** function replaces the keys and values for the specified section in an initialization file.	
Parameters	*lpAppName*	[in] Pointer to a null-terminated string specifying the name of the section in which data is written. This section name is typically the name of the calling application.
	lpKeyName	[in] Pointer to a buffer containing the key names and whose value to be writed.
	nNumber	[in]Value to be writed.
	lpFileName	[in] Pointer to a null-terminated string that specifies the name of the initialization file. If this parameter does not contain a full path to the file, the system searches for the file in the Windows directory.
Return:	The value had been written in to the file.	

Remarks

7.3.15 Btsdk_ SetServiceSecurityLevel

Prototype	BTUINT32 Btsdk_SetServiceSecurityLevel (BTSVCHDL svc_hdl, BTUINT8 level);	
Description	The **Btsdk_SetServiceSecurityLevel** function sets the security level for a specified local service, and is only valid when the local device is in security mode 2	
Parameters	svc_hdl	[in] Handle to the service record to be queried.
	level	[in] specify what a security procedure or a combination of security procedures should be initiated when the service is accessed, including: BTSDK_SSL_AUTHENTICATION: authentication. BTSDK_SSL_AUTHORIZATION: authorization, implicitly covering BTSDK_SSL_AUTHENTICATION. BTSDK_SSL_ENCRYPTION: encryption, implicitly covering BTSDK_SSL_AUTHENTICATION. .
Return:	BTSDK_OK for success other for error code.	

Remarks

7.3.16 Btsdk_ GetServiceSecurityLevel

Prototype	BTUINT32 Btsdk_GetServiceSecurityLevel (BTSVCHDL svc_hdl, BTUINT8 *level);	
Description	The **Btsdk_GetServiceSecurityLevel** function gets the security level for a specified local service, and is only valid when the local device is in security mode 2.	
Parameters	svc_hdl	[in] Handle to the service record to be queried.
	level	[out] pointer to the storage of the security level, which specify what a security procedure or a combination of security procedures should be initiated when the service is accessed, including: BTSDK_SSL_AUTHENTICATION: authentication BTSDK_SSL_AUTHORIZATION: authorization, implicitly covering SM_AUTHEN BTSDK_SSL_ENCRYPTION: encryption, implicitly covering SM_AUTHEN
Return:	BTSDK_OK for success other for error code.	

Remarks