MPLAB® Harmony Help - Bluetooth Stack Library

MPLAB Harmony Integrated Software Framework v1.11

© 2013-2017 Microchip Technology Inc. All rights reserved.

Bluetooth Stack Library Help

This section provides information on the PIC32 Bluetooth Stack Library that is available in MPLAB Harmony.

Description

The PIC32 Bluetooth Stack Library consists of two distinct libraries:

- PIC32 Bluetooth Basic Stack Library - this library provides "Core" Bluetooth functionality, which was previously referred to as "SPP-only"
- PIC32 Bluetooth Audio Stack Library - this library, which must be purchased, provides audio features

The PIC32 Bluetooth Stack Library is provided in binary form with associated APIs to enable connection, control and data to and from Bluetooth-enabled devices with PIC32 microcontrollers.

Introduction

Describes the PIC32 Bluetooth Stack Library.

Description

Important! The PIC32 Bluetooth Stack Library is created under license from Searan LLC, and uses the dotstack™ framework. This library has been qualified for use and a QDID is available from the Bluetooth SIG. More information about the qualification process can be found in the *"PIC32 Bluetooth Audio Development Kit Reference Guide"* (DS70005140), which is available from the Microchip website: www.microchip.com.

The PIC32 Bluetooth Stack Library is available only in binary format. When a license is acquired it may be for SPP (data) only, or with access to SBC or AAC decoders. The different licenses come with a number of specific applications. For more information on the license options please see http://www.microchip.com/pic32btsuites.

PIC32 Bluetooth Stack Library

This library provides all of the functionality of the Bluetooth Stack interfaces including profiles and protocols at the lower levels. These are utilized in a number of demonstration applications used in the PIC32 Bluetooth Audio Development Kit. Specific demonstrations to illustrate the various connection options, data and audio communications and control interfaces are available for purchase. Please refer to the Microchip Premium MPLAB Harmony Audio web page (www.microchip.com/pic32harmonypremiumaudio) for more information. This library interface is provided in the C language.

This library provides all of the functionality of the Bluetooth Stack interfaces including profiles and protocols at the lower levels. These are utilized in a number of demonstration applications used in the PIC32 Bluetooth Audio Development Kit, PIC32 Bluetooth Starter Kit and PIC32MZ EC Starter Kit development board attached to Multimedia Expansion Board II (MEB II).

PIC32 Bluetooth Audio Stack Library

Note: The PIC32 Bluetooth Audio Stack Library is a Premium product, and is not included in the standard release of MPLAB Harmony and must be purchased. For information on purchasing the Premium PIC32 Bluetooth Audio Stack Library, please refer to the MPLAB Harmony Premium Products page by visiting:

http://www.microchip.com/pagehandler/en-us/devtools/mplabharmony/home.html

This library provides full functionality of the PIC32 Bluetooth Stack Library that includes Serial Port Profile (SPP) and Audio protocols and profiles such as AVDTP, AVCTP, A2DP, and AVRCP. The library interface is provided in the C language.

The PIC32 Bluetooth Audio Stack Library is a premium product and is not included in the standard release of MPLAB Harmony and must be purchased separately. For information on purchasing the premium PIC32 Bluetooth Audio Stack Library, please refer to the Microchip Premium MPLAB Harmony Audio web page (www.microchip.com/pic32harmonypremiumaudio).

This library provides all of the functionality of the Bluetooth Stack interfaces including profiles and protocols at the lower levels. These are utilized in a number of demonstration applications used in the PIC32 Bluetooth Audio Development Kit, PIC32 Bluetooth Starter Kit and PIC32MZ EC Starter Kit development board attached to Multimedia Expansion Board II (MEB II).

Demonstrations

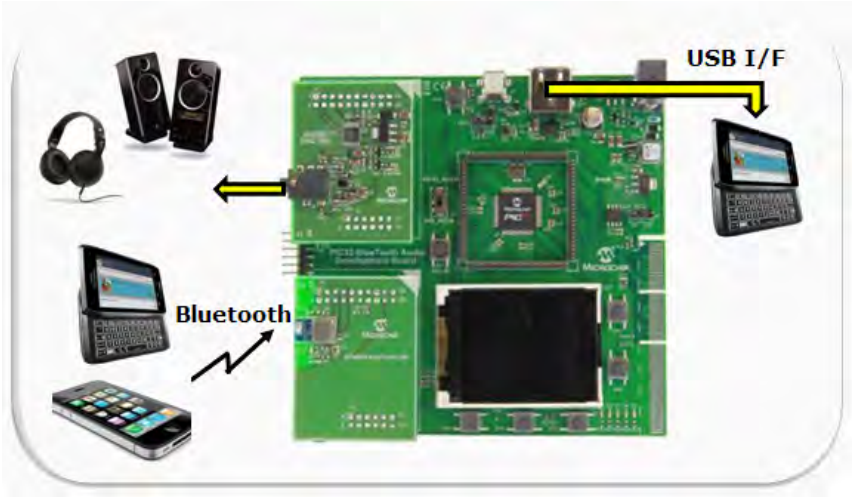
Refer to PIC32 Bluetooth Stack Library Demonstrations for information on the demonstrations that are available for the PIC32 Bluetooth Stack Library. In addition, other specific demonstrations to illustrate the various connection options, data and audio communications and control interfaces are available for purchase. Please refer to the Microchip Premium MPLAB Harmony Audio web page (www.microchip.com/pic32harmonypremiumaudio) for more information.

Bluetooth Stack Licensing

This section describes how to license the stack.

Description

The license to the stack must be procured from Microchip, and it is specific to the Microchip PIC32 microcontroller architecture. Some licenses require a one-time fee conveyed when a customer purchases the demonstration applications for the PIC32 Bluetooth Audio Development Kit (P/N DV320032).



There are four potential options in the license, each with the benefits of the previous.

- *SPP (data only)* - This is a special version of the library available to carry data only. The decoder options have been removed in this case. The license is available for Microchip PIC products, but is free of charge. Special demonstrations and the library for this product can be found under the SPP section by visiting http://www.microchip.com/pic32btsuites.
- *SBC Decoder* - This Bluetooth license also grants the user the ability to use the baseline audio SBC decoder, which is provided as a separate library. The demonstrations include stand-alone Bluetooth audio streaming using this decoder, and other applications which combine the use of USB and Bluetooth audio.
- *AAC Decoder* - This Bluetooth license also grants the user the ability to use the audio AAC decoder if a connected devices supports this form of streaming. For devices that only support SBC, the SBC decoder is also included. All demonstrations of the (1) license are included. In addition, new demonstrations are included that make specific use of the AAC decoder both with and without USB options.
- *Break-in Feature with SBC Decoder* - The break-in feature enables multi-connection support and a "Party mode" where multiple handset devices can take turns sharing the Bluetooth connection to stream SBC audio. This also supports the single connection SBC audio streaming in SPP.

Note: All licenses are available through Microchip direct. More details on the specific application demonstrations can be found in the *"PIC32 Bluetooth Audio Development Kit Reference Guide"* (DS70005140). Visit http://www.microchip.com/pic32btsuites for the available demonstrations and licensing options.

Using the Library

This topic describes the basic architecture of the PIC32 Bluetooth Stack Library and provides information and examples on its use.

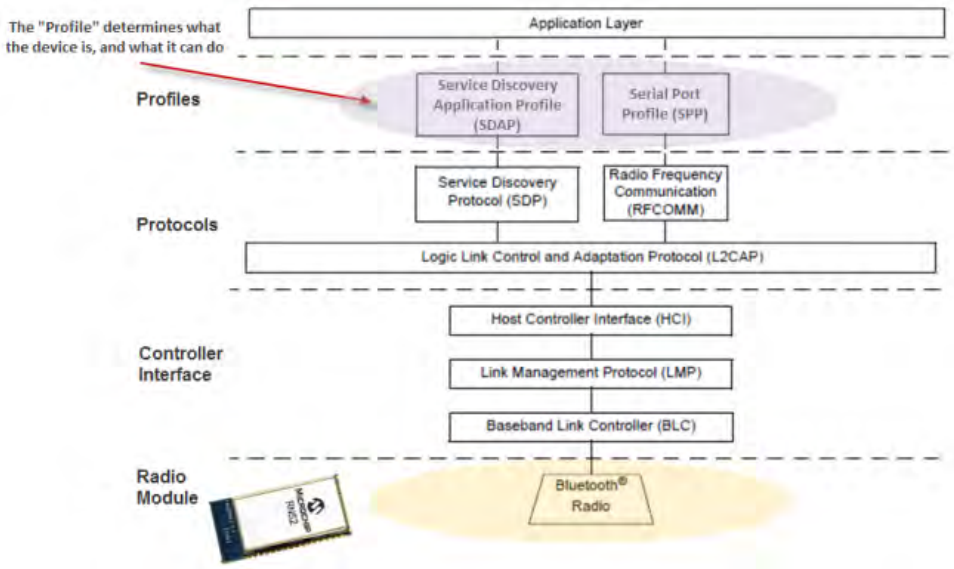
Description

The interface to the Bluetooth stack is driven by a number of header files. The stack header files form a collection of APIs for each protocol and profile of the Bluetooth stack. These files can be found within the following MPLAB Harmony directory, which is included when you license the stack and the example applications that are provided:

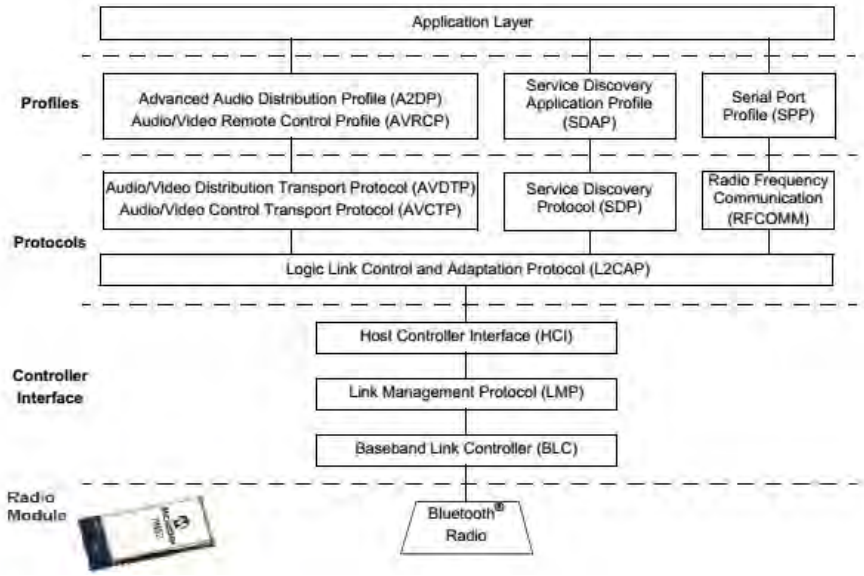
- `<install_dir>/framework/bluetooth/cdbt` for the PIC32 Bluetooth Basic Stack
- `<install_dir>/framework/bluetooth/premium/audio/cdbt` for the PIC32 Bluetooth Audio Stack

The following two figures illustrate the Bluetooth Stacks.

PIC32 Bluetooth Basic Stack



PIC32 Bluetooth Audio Stack



Library Overview

Provides information on the APIs provided in the library.

Description

The library interface routines are divided into various sub-sections, which address one of the profiles or the overall operation of the PIC32

Bluetooth Stack Library.

Library Interface Section	Description
Bluetooth Support	Contains functions for basic Bluetooth support.
A2DP	Contains functions for the Advanced Audio Distribution Profile (see **Note**).
AVCTP	Contains functions for the Audio/Video Control Transport Protocol (see **Note**).
AVDTP	Contains functions for the Audio/Video Distribution Transport Protocol (see **Note**).
AVRCP	Contains functions for the Audio/Video Remote Control Profile (see **Note**).
GAP	Contains functions for the Generic Access Profile.
HCI	Contains functions for the Host Controller Interface.
L2CAP	Contains functions for the Logical Link Control and Adaptation Protocol.
RFCOMM	Contains functions for the RF Communication Lower Layer.
SBC Decoder	Contains functions for the SBC Decoder
SDP	Contains functions for the Service Discovery Protocol.
SPP	Contains functions for the Serial Port Profile.
SSP	Contains functions for Secure Simple Pairing.
Misc.	Contains miscellaneous functions not noted elsewhere in the stack.
Utility	Contains functions that support the stack functions.

Note: The PIC32 Bluetooth Audio Stack is an extension of the PIC32 Bluetooth Basic Stack and provides the additional features to support audio. The following sections cover the audio features of the PIC32 Bluetooth Stack.

- AVDTP Functions
- AVDTP Data Types and Constants
- AVCTP Functions
- AVCTP Data Types and Constants
- A2DP Functions
- A2DP Data Types and Constants
- AVRCP Functions
- AVRCP Data Types and Constants

Configuring the Library

This topic provides information on configuring the library.

AVCTP Configuration Macros

	Name	Description
	__AVCTP_CONFIG_H	This is macro __AVCTP_CONFIG_H.

	AVCTP_ALLOCATE_BUFFERS_RAM_SIZE_VAR	• defgroup avctp_config Configuration
		• ingroup avctp
		*
		• This module describes parameters used to configure AVCTP layer.
		*
		• dotstack is customized using a configuration file. The configuration file tailors the dotstack to the application being built. It has to have the structure shown below.
		* code #include "cdbt/bt/bt_std.h"
		// HCI, L2CAP and SDP must always be present
		// HCI configuration parameters #define HCI_MAX_CMD_BUFFERS ... #define HCI_MAX_DATA_BUFFERS ... #define HCI_MAX_HCI_CONNECTIONS ... #define HCI_RX_BUFFER_LEN ... #define HCI_TX_BUFFER_LEN ... #define HCI_L2CAP_BUFFER_LEN ... #define HCI_MAX_CMD_PARAM_LEN ...
		// L2CAP configuration parameters #define L2CAP_MAX_CMD_BUFFERS ... #define L2CAP_MAX_FRAME_BUFFERS ... #define L2CAP_MAX_PSMS ... #define L2CAP_MAX_CHANNELS ...
		// SDP... more

	AVCTP_ALLOCATE_BUFFERS_VARS	brief Maximum number of message buffers ingroup avctp_config details This parameter defines the maximum number of buffer available for sending message.
	AVCTP_MAX_CHANNELS	brief Maximum number of AVCTP channels ingroup avctp_config details This parameter defines the maximum number of channels a local device can have with remote devices.
	AVCTP_MAX_TRANSPORT_CHANNELS	brief Maximum number of AVCTP transports ingroup avctp_config details This parameter defines the maximum number of transports a local device can have with remote devices. This value should not exceed AVCTP_MAX_CHANNELS.

AVDTP Configuration Macros

	Name	Description
	AVDTP_ALLOCATE_BUFFERS_FUNCTION	This is macro AVDTP_ALLOCATE_BUFFERS_FUNCTION.

	AVDTP_ALLOCATE_BUFFERS_RAM_SIZE_VAR	• defgroup avdtp_config Configuration
		• ingroup avdtp
		*
		• This module describes parameters used to configure AVDTP layer.
		*
		• dotstack is customized using a configuration file. The configuration file tailors the dotstack to the application being built. It has to have the structure shown below.
		* code #include "cdbt/bt/bt_std.h"
		// HCI, L2CAP and SDP must always be present
		// HCI configuration parameters #define HCI_MAX_CMD_BUFFERS ... #define HCI_MAX_DATA_BUFFERS ... #define HCI_MAX_HCI_CONNECTIONS ... #define HCI_RX_BUFFER_LEN ... #define HCI_TX_BUFFER_LEN ... #define HCI_L2CAP_BUFFER_LEN ... #define HCI_MAX_CMD_PARAM_LEN ...
		// L2CAP configuration parameters #define L2CAP_MAX_CMD_BUFFERS ... #define L2CAP_MAX_FRAME_BUFFERS ... #define L2CAP_MAX_PSMS ... #define L2CAP_MAX_CHANNELS ...
		// SDP... more

	AVDTP_ALLOCATE_BUFFERS_VARS	This is macro AVDTP_ALLOCATE_BUFFERS_VARS.
	AVDTP_CMD_ABORT	This is macro AVDTP_CMD_ABORT.

	AVDTP_CMD_CLOSE	This is macro AVDTP_CMD_CLOSE.
	AVDTP_CMD_DISCOVER	This is macro AVDTP_CMD_DISCOVER.
	AVDTP_CMD_GET_CAPABILITIES	This is macro AVDTP_CMD_GET_CAPABILITIES.
	AVDTP_CMD_GET_CONFIGURATION	This is macro AVDTP_CMD_GET_CONFIGURATION.
	AVDTP_CMD_OPEN	This is macro AVDTP_CMD_OPEN.
	AVDTP_CMD_RECONFIGURE	This is macro AVDTP_CMD_RECONFIGURE.
	AVDTP_CMD_SECURITY_CONTROL	This is macro AVDTP_CMD_SECURITY_CONTROL.
	AVDTP_CMD_SET_CONFIGURATION	This is macro AVDTP_CMD_SET_CONFIGURATION.
	AVDTP_CMD_START	This is macro AVDTP_CMD_START.
	AVDTP_CMD_SUSPEND	This is macro AVDTP_CMD_SUSPEND.
	AVDTP_CODEC_CONFIG_BUFFER_LEN	brief Size of the buffer used to store codec specific configuration. ingroup avdtp_config details Each codec uses unique configuration which can take different amount of memory. This parameter defines the size of the buffer for storing codec's configuration. The value of 16 is sufficient for SBC, AAC and MPEG1,2. If vendor specific codec is to be used this value may need to increased.
	AVDTP_MAX_CMD_BUFFERS	brief Maximum number of command buffers. ingroup avdtp_config details This parameter defines the number of buffers reserved for sending commands to a remote device over its control channel. Each channel uses its own buffers so the total number of buffers is AVDTP_MAX_REMOTE_DEVICES * AVDTP_MAX_CMD_BUFFERS. The minimum value is 1. The maximum value is 255. 2 is usually sufficient.
	AVDTP_MAX_CMD_PARAM_LEN	brief Maximum length of control command parameters ingroup avdtp_config details This parameter defines the maximum length of all command parameters. The value should not exceed AVDTP_MAX_TX_BUFFER_LEN - 2 (command header).
	AVDTP_MAX_REMOTE_DEVICES	brief Maximum number of remote devices a local device can be connected to ingroup avdtp_config details This parameter defines the number of remote devices a local device can have simultaneous connections to (i.e. control channels). This value should not exceed AVDTP_MAX_STREAMS. For each remote device AVDTP creates one control channel regardless of the number of streams between the local and the remote devices. Assuming that the local devices wants to have only one channel with each remote device and if AVDTP_MAX_REMOTE_DEVICES > AVDTP_MAX_STREAMS all memory reserved for devices in excess of AVDTP_MAX_STREAMS will be wasted. The minimum value is... more
	AVDTP_MAX_SEP	brief Maximum number of SEPs that can be exposed by a local device ingroup avdtp_config details This parameter defines the number of SEPs an application can expose to remote devices. The minimum value is 1. The maximum value is 255.
	AVDTP_MAX_STREAMS	brief Maximum number of streams that can be exposed by a local device ingroup avdtp_config details This parameter defines the number of streams an application can open between local and remote devices. This value can be different from AVDTP_MAX_SEP but should not exceed it. Since each SEP can only be used once the local device can only have as much streams as there are SEPs. If AVDTP_MAX_STREAMS > AVDTP_MAX_SEP all memory reserved for streams in excess of AVDTP_MAX_SEP will be wasted. The minimum value is 1. The maximum value is 255.
	AVDTP_MAX_TRANSPORT_CHANNELS	brief Maximum number of transport channels. ingroup avdtp_config details Depending on the SEP capabilities (multiplexing, recovery, reporting) each stream may need up to 3 transport channels. E.g., if multiplexing is not supported and recovery and reporting are supported a stream will use 3 transport channels - 1 for media transport session, 1 for recovery transport session and 1 for reporting transport session. If multiplexing is not supported and only reporting is supported the stream will use 2 transport channels - 1 for media transport session and 1 for reporting transport session. If multiplexing is supported the stream will need only... more
	AVDTP_MAX_TX_BUFFER_LEN	brief Size of the transmit buffer. ingroup avdtp_config details This parameter defines the size of the buffer used to send AVDTP control commands to L2CAP layer. Each control channel has it own buffer so the total amount of memory allocated for these buffers is AVDTP_MAX_TX_BUFFER_LEN) * AVDTP_MAX_REMOTE_DEVICES. The minimum value is 1. The maximum value is 255. The value of 32 is usually sufficient.

AVRCP Configuration Macros

	Name	Description
	__AVRCP_CONFIG_H	This is macro __AVRCP_CONFIG_H.

	AVRCP_ALLOCATE_BUFFERS_RAM_SIZE_VAR	• defgroup avrcp_config Configuration
		• ingroup avrcp
		*
		• This module describes parameters used to configure AVRCP layer.
		*
		• dotstack is customized using a configuration file. The configuration file tailors the dotstack to the application being built. It has to have the structure shown below.
		* code #include "cdbt/bt/bt_std.h"
		// HCI, L2CAP and SDP must always be present
		// HCI configuration parameters #define HCI_MAX_CMD_BUFFERS ... #define HCI_MAX_DATA_BUFFERS ... #define HCI_MAX_HCI_CONNECTIONS ... #define HCI_RX_BUFFER_LEN ... #define HCI_TX_BUFFER_LEN ... #define HCI_L2CAP_BUFFER_LEN ... #define HCI_MAX_CMD_PARAM_LEN ...
		// L2CAP configuration parameters #define L2CAP_MAX_CMD_BUFFERS ... #define L2CAP_MAX_FRAME_BUFFERS ... #define L2CAP_MAX_PSMS ... #define L2CAP_MAX_CHANNELS ...
		// SDP... more

	AVRCP_ALLOCATE_BUFFERS_VARS	This is macro AVRCP_ALLOCATE_BUFFERS_VARS.
	AVRCP_CMD_TIMEOUT	brief Command timeout ingroup avrcp_config details This parameter defines the amount of time in milliseconds AVRCP waits for a response to a request. If not defined the default value of 10000 (10 secconds) is used.
	AVRCP_MAX_CHANNELS	brief Maximum number of remote devices a local device can be connected to ingroup avrcp_config details This parameter defines the number of remote devices a local device can have simultaneous connections to (i.e. channels). This value should not exceed AVCTP_MAX_CHANNELS.
	AVRCP_MAX_CMD_BUFFERS	brief Maximum number of command buffers. ingroup avrcp_config details This parameter defines the number of buffers reserved for sending commands to a remote device over its control channel. Each channel uses its own buffers so the total number of buffers is AVRCP_MAX_CHANNELS * AVRCP_MAX_CMD_BUFFERS. The minimum value is 1. The maximum value is 255. If not define one buffer for each channel is reserved.
	AVRCP_MAX_CMD_PARAM_LEN	brief Maximum length of command parameters ingroup avrcp_config details This parameter defines the maximum length of all command parameters. If not defined the default value of 512 is used.
	AVRCP_MAX_DEVICE_NAME_LEN	brief Maximum length of device name ingroup avrcp_config details This parameter defines the size of the buffer used to store device's name while searching for nearby targets with bt_avrcp_find_targets. If the name of the device is longer than AVRCP_MAX_DEVICE_NAME_LEN it is truncated to AVRCP_MAX_DEVICE_NAME_LEN. If not defined the default value of 20 is used.
	AVRCP_MAX_SEARCH_RESULTS	brief Maximum number of devices to find ingroup avrcp_config details This parameter defines the maximum number of devices bt_avrcp_find_targets can find. If not defined the default value of 7 is used.

Bluetooth Support Configuration Macros

	Name	Description
	ATT_ALLOCATE_BUFFERS	This is function ATT_ALLOCATE_BUFFERS.
	ATT_CLIENT_ALLOCATE_BUFFERS	This is function ATT_CLIENT_ALLOCATE_BUFFERS.
	FTP_ALLOCATE_BUFFERS	This is function FTP_ALLOCATE_BUFFERS.
	GATT_CLIENT_ALLOCATE_BUFFERS	This is function GATT_CLIENT_ALLOCATE_BUFFERS.
	HCI_ALLOCATE_BUFFERS	This is function HCI_ALLOCATE_BUFFERS.
	HFP_ALLOCATE_BUFFERS	This is function HFP_ALLOCATE_BUFFERS.
	HID_ALLOCATE_BUFFERS	This is function HID_ALLOCATE_BUFFERS.
	HSP_AG_ALLOCATE_BUFFERS	This is function HSP_AG_ALLOCATE_BUFFERS.
	HSP_ALLOCATE_BUFFERS	This is function HSP_ALLOCATE_BUFFERS.

	IAP_ALLOCATE_BUFFERS	This is function IAP_ALLOCATE_BUFFERS.
	IAP_BT_ALLOCATE_BUFFERS	This is function IAP_BT_ALLOCATE_BUFFERS.
	IAP2_ALLOCATE_BUFFERS	This is function IAP2_ALLOCATE_BUFFERS.
	L2CAP_ALLOCATE_BUFFERS	This is function L2CAP_ALLOCATE_BUFFERS.
	OBEX_ALLOCATE_BUFFERS	This is function OBEX_ALLOCATE_BUFFERS.
	PBAP_ALLOCATE_BUFFERS	This is function PBAP_ALLOCATE_BUFFERS.
	RFCOMM_ALLOCATE_BUFFERS	This is function RFCOMM_ALLOCATE_BUFFERS.
	SDP_ALLOCATE_BUFFERS	This is function SDP_ALLOCATE_BUFFERS.
	SMP_ALLOCATE_BUFFERS	This is function SMP_ALLOCATE_BUFFERS.
	SPP_ALLOCATE_BUFFERS	This is function SPP_ALLOCATE_BUFFERS.
	BT_ENABLE_BLE	This is macro BT_ENABLE_BLE.
	BT_INCLUDE_RFCOMM	This is macro BT_INCLUDE_RFCOMM.
	BT_LOG_LEVEL_MAX	This is macro BT_LOG_LEVEL_MAX.
	BT_LOG_LEVEL_MIN	This is macro BT_LOG_LEVEL_MIN.
	__BT_APP_CONFIG_H	This is macro __BT_APP_CONFIG_H.
	BT_INCLUDE_IAP	This is macro BT_INCLUDE_IAP.
	BT_INCLUDE_IAP2	This is macro BT_INCLUDE_IAP2.
	IAP_BT_MAX_TRANSPORTS	This is macro IAP_BT_MAX_TRANSPORTS.
	IAP_MAX_SESSIONS	This is macro IAP_MAX_SESSIONS.
	IAP_RX_BUFFER_SIZE	This is macro IAP_RX_BUFFER_SIZE.
	IAP2_MAX_PACKET_SIZE	This is macro IAP2_MAX_PACKET_SIZE.
	IAP2_MAX_SESSIONS	This is macro IAP2_MAX_SESSIONS.

HCI Configuration Macros

	Name	Description
	BT_ENABLE_SCO	This is macro BT_ENABLE_SCO.
	BT_HCI_EVT_AUTHENTICATION_COMPLETE_HANDLER	This is macro BT_HCI_EVT_AUTHENTICATION_COMPLETE_HANDLER.
	BT_HCI_EVT_CONNECTION_COMPLETE_HANDLER	This is macro BT_HCI_EVT_CONNECTION_COMPLETE_HANDLER.
	BT_HCI_EVT_CONNECTION_REQUEST_HANDLER	This is macro BT_HCI_EVT_CONNECTION_REQUEST_HANDLER.
	BT_HCI_EVT_ENCRYPTION_CHANGE_HANDLER	This is macro BT_HCI_EVT_ENCRYPTION_CHANGE_HANDLER.
	BT_HCI_EVT_EXTENDED_INQUIRY_RESULT_HANDLER	This is macro BT_HCI_EVT_EXTENDED_INQUIRY_RESULT_HANDLER.
	BT_HCI_EVT_INQUIRY_COMPLETE_HANDLER	This is macro BT_HCI_EVT_INQUIRY_COMPLETE_HANDLER.
	BT_HCI_EVT_INQUIRY_RESULT_HANDLER	This is macro BT_HCI_EVT_INQUIRY_RESULT_HANDLER.
	BT_HCI_EVT_INQUIRY_RESULT_WITH_RSSI_HANDLER	This is macro BT_HCI_EVT_INQUIRY_RESULT_WITH_RSSI_HANDLER.
	BT_HCI_EVT_LINK_KEY_NOTIFICATION_HANDLER	This is macro BT_HCI_EVT_LINK_KEY_NOTIFICATION_HANDLER.
	BT_HCI_EVT_LINK_KEY_REQUEST_HANDLER	This is macro BT_HCI_EVT_LINK_KEY_REQUEST_HANDLER.
	BT_HCI_EVT_MODE_CHANGE_HANDLER	This is macro BT_HCI_EVT_MODE_CHANGE_HANDLER.
	BT_HCI_EVT_PIN_CODE_REQUEST_HANDLER	This is macro BT_HCI_EVT_PIN_CODE_REQUEST_HANDLER.
	BT_HCI_EVT_REMOTE_NAME_REQUEST_COMPLETE_HANDLER	This is macro BT_HCI_EVT_REMOTE_NAME_REQUEST_COMPLETE_HANDLER.
	BT_HCI_EVT_ROLE_CHANGE_HANDLER	This is macro BT_HCI_EVT_ROLE_CHANGE_HANDLER.
	BT_HCI_EVT_SYNCH_CONNECTION_COMPLETE_HANDLER	This is macro BT_HCI_EVT_SYNCH_CONNECTION_COMPLETE_HANDLER.
	BT_LE_EVT_HANDLER	This is macro BT_LE_EVT_HANDLER.

	BT_SSP_EVT_HANDLER	This is macro BT_SSP_EVT_HANDLER.
	HCI_ALLOCATE_BUFFERS_FUNCTION	This is macro HCI_ALLOCATE_BUFFERS_FUNCTION.
	HCI_ALLOCATE_BUFFERS_RAM_SIZE_VAR	This is macro HCI_ALLOCATE_BUFFERS_RAM_SIZE_VAR.
	HCI_ALLOCATE_BUFFERS_VARS	This is macro HCI_ALLOCATE_BUFFERS_VARS.
	HCI_DECLARE_LE_CONN_STATES	This is macro HCI_DECLARE_LE_CONN_STATES.
	HCI_DECLARE_LE_CTRL_STATE	This is macro HCI_DECLARE_LE_CTRL_STATE.
	HCI_ENABLE_CTRL_TO_HOST_FLOW_CONTROL	This is macro HCI_ENABLE_CTRL_TO_HOST_FLOW_CONTROL.
	HCI_INIT_LE_CONN_STATES	This is macro HCI_INIT_LE_CONN_STATES.
	HCI_INIT_LE_CTRL_STATE	This is macro HCI_INIT_LE_CTRL_STATE.
	HCI_INIT_SCO_HANDLERS	This is macro HCI_INIT_SCO_HANDLERS.
	HCI_INIT_SSP_HANDLERS	This is macro HCI_INIT_SSP_HANDLERS.
	HCI_L2CAP_BUFFER_LEN	This is macro HCI_L2CAP_BUFFER_LEN.
	HCI_SIZEOF_LE_CONN_STATES	This is macro HCI_SIZEOF_LE_CONN_STATES.
	HCI_SIZEOF_LE_CTRL_STATE	This is macro HCI_SIZEOF_LE_CTRL_STATE.
	HCI_TX_BUFFER_LEN	This is macro HCI_TX_BUFFER_LEN.
	__HCI_CONFIG_EVENT_HANDLERS_H	This is macro __HCI_CONFIG_EVENT_HANDLERS_H.
	__HCI_CONFIG_H	This is macro __HCI_CONFIG_H.
	HCI_MAX_CONNECT_ATTEMPTS	This is macro HCI_MAX_CONNECT_ATTEMPTS.

L2CAP Configuration Macros

	Name	Description
	L2CAP_ALLOCATE_BUFFERS_FUNCTION	This is macro L2CAP_ALLOCATE_BUFFERS_FUNCTION.
	L2CAP_ALLOCATE_BUFFERS_RAM_SIZE_VAR	This is macro L2CAP_ALLOCATE_BUFFERS_RAM_SIZE_VAR.
	L2CAP_ALLOCATE_BUFFERS_VARS	This is macro L2CAP_ALLOCATE_BUFFERS_VARS.
	L2CAP_FIXED_CHANNELS_DECL	This is macro L2CAP_FIXED_CHANNELS_DECL.
	L2CAP_HCI_PACKET_TYPE	• brief L2CAP_HCI_PACKET_TYPE. • ingroup btconfig * • details Defines a set of packets that link manager is allowed to use when calling bt_l2cap_connect. • The default value is to enable all packet types. enable all packet types
	L2CAP_HCI_PAGE_SCAN_REPETITION_MODE	brief L2CAP_HCI_PAGE_SCAN_REPETITION_MODE. ingroup btconfig details Defines a default value of the page scan repetition mode when calling bt_l2cap_connect. Must be set to one of the following values: HCI_PAGE_SCAN_REPETITION_MODE_R0 HCI_PAGE_SCAN_REPETITION_MODE_R1 HCI_PAGE_SCAN_REPETITION_MODE_R2 The default value is HCI_PAGE_SCAN_REPETITION_MODE_R0.
	L2CAP_HCI_ROLE_SWITCH	brief L2CAP_HCI_ROLE_SWITCH. ingroup btconfig details Defines a default value of the role switch parameter when calling bt_l2cap_connect. Must be set to one of the following values: HCI_ROLE_SWITCH_ALLOW HCI_ROLE_SWITCH_DISALLOW The default value is to allow the role switch.
	L2CAP_IDLE_CONNECTION_TIMEOUT	seconds
	PACK_CONFIG_REQUEST	This is macro PACK_CONFIG_REQUEST.
	PACK_CONFIG_RESPONSE	This is macro PACK_CONFIG_RESPONSE.
	PACK_CONN_REQUEST	This is macro PACK_CONN_REQUEST.
	PACK_CONN_RESPONSE	This is macro PACK_CONN_RESPONSE.
	PACK_DCONN_REQUEST	This is macro PACK_DCONN_REQUEST.
	PACK_DCONN_RESPONSE	This is macro PACK_DCONN_RESPONSE.
	PACK_INFO_REQUEST	This is macro PACK_INFO_REQUEST.
	PACK_INFO_RESPONSE	This is macro PACK_INFO_RESPONSE.
	PROCESS_CONFIG_REQ	This is macro PROCESS_CONFIG_REQ.
	PROCESS_CONFIG_RES	This is macro PROCESS_CONFIG_RES.
	PROCESS_CONN_REQ	This is macro PROCESS_CONN_REQ.

	PROCESS_CONN_RES	This is macro PROCESS_CONN_RES.
	PROCESS_DCONN_REQ	This is macro PROCESS_DCONN_REQ.
	PROCESS_DCONN_RES	This is macro PROCESS_DCONN_RES.
	PROCESS_INFO_REQ	This is macro PROCESS_INFO_REQ.
	PROCESS_INFO_RES	This is macro PROCESS_INFO_RES.
	READ_CONFIG_REQUEST	This is macro READ_CONFIG_REQUEST.
	READ_CONFIG_RESPONSE	This is macro READ_CONFIG_RESPONSE.
	READ_CONN_REQUEST	This is macro READ_CONN_REQUEST.
	READ_CONN_RESPONSE	This is macro READ_CONN_RESPONSE.
	READ_DCONN_REQUEST	This is macro READ_DCONN_REQUEST.
	READ_DCONN_RESPONSE	This is macro READ_DCONN_RESPONSE.
	READ_INFO_REQUEST	This is macro READ_INFO_REQUEST.
	READ_INFO_RESPONSE	This is macro READ_INFO_RESPONSE.
	__L2CAP_CONFIG_H	This is macro __L2CAP_CONFIG_H.
	__L2CAP_CONFIG_HANDLERS_H	This is macro __L2CAP_CONFIG_HANDLERS_H.
	L2CAP_DECL_ERETR_FUNCTIONS	This is macro L2CAP_DECL_ERETR_FUNCTIONS.
	L2CAP_MAX_FIXED_CHANNELS	This is macro L2CAP_MAX_FIXED_CHANNELS.

RFCOMM Configuration Macros

	Name	Description
	RFCOMM_ALLOCATE_BUFFERS_FUNCTION	This is macro RFCOMM_ALLOCATE_BUFFERS_FUNCTION.
	RFCOMM_ALLOCATE_BUFFERS_RAM_SIZE_VAR	This is macro RFCOMM_ALLOCATE_BUFFERS_RAM_SIZE_VAR.
	RFCOMM_ALLOCATE_BUFFERS_VARS	This is macro RFCOMM_ALLOCATE_BUFFERS_VARS.
	RFCOMM_BUFFER_SIZE	This is macro RFCOMM_BUFFER_SIZE.
	RFCOMM_ENABLE_MULTIDEVICE_CHANNELS	brief Enable multi-device server channels. ingroup rfcomm_config details Normally each server channel can be used only once. I.e. if device A connected to channel 1, device B cannot connect to channel 1 until device A disconnects. With this option it is possible to make channels accept connections from several devices at the same time. I.e., if RFCOMM_ENABLE_MULTIDEVICE_CHANNELS is TRUE both device A and device B can connect to channel 1 at the same time.
	RFCOMM_INFO_LEN	brief Maximum size of the data portion of a UIH frame. ingroup rfcomm_config details This parameter defines the maximum size of the data portion of a UIH frame. If CFC is used the actual length of the data portion will be 1 byte less. This value must be less than or equal to HCI_L2CAP_BUFFER_LEN - RFCOMM_FRAME_HEADER_LEN - L2CAP_HEADER_LEN.
	RFCOMM_LOCAL_CREDIT	brief The number of receive buffers. ingroup rfcomm_config details This parameter defines the number of received UIH frames that can be stored on the local device. The flow control mechanism used in RFCOMM ensures that the remote side of the link always knows how many free buffers left on the local device. When the number of free buffers reaches 0, the transmitter stops sending data frames until the receiver frees some buffers. The RFCOMM layer does not actually allocate space for buffers. It uses RFCOMM_LOCAL_CREDIT to keep track of free buffers and report them to the remote side. Actual memory... more
	RFCOMM_MAX_CMD_BUFFERS	brief Maximum number of command buffers. ingroup rfcomm_config details This parameter defines the maximum number of commands that can be sent at the same time. It is usually enough to reserve 2 buffers for each DLC excluding control DLC. Therefore, this value can be defined as rn #define RFCOMM_MAX_CMD_BUFFERS (RFCOMM_MAX_DLCS - 1) * 2
	RFCOMM_MAX_DLCS	brief Maximum number of DLCs ingroup rfcomm_config details This parameter defines the maximum number of DLCs on each session. This value should be at least 2 because each session uses one DLC to convey multiplexer control messages. All other DLCs are used to emulate serial ports.
	RFCOMM_MAX_SERVER_CHANNELS	brief Maximum number of Server channels ingroup rfcomm_config details This parameter defines the maximum number of server channels exposed by the local device. This value should not exceed RFCOMM_MAX_DLCS - 1.

	RFCOMM_MAX_SESSIONS	brief Maximum number of remote devices a local device can be connected to ingroup rfcomm_config details This parameter defines the maximum number of remote devices a local device can have simultaneous connections to. This value should not exceed HCI_MAX_HCI_CONNECTIONS.
	__RFCOMM_CONFIG_H	This is macro __RFCOMM_CONFIG_H.
	RFCOMM_LOCAL_CREDIT_SEND_THRESHOLD_DECL	This is macro RFCOMM_LOCAL_CREDIT_SEND_THRESHOLD_DECL.

SDP Configuration Macros

	Name	Description

	SDP_ALLOCATE_BUFFERS_RAM_SIZE_VAR	• defgroup sdp_config Configuration
		• ingroup sdp
		*
		• This module describes parameters used to configure SDP.
		*
		• dotstack is customized using a configuration file. The configuration file tailors the dotstack to the application being built. It has to have the structure shown below.
		* code #include "cdbt/bt/bt_std.h"
		// HCI and L2CAP must always be present // SDP is required only if stack is running in dual mode. This is the default mode. // To run the stack in single mode (i.e. only BLE is supported) a BT_BLE_SINGLE_MODE symbol // must be defined: // #define BT_BLE_SINGLE_MODE
		// HCI configuration parameters #define... more

	SDP_ALLOCATE_BUFFERS_VARS	This is macro SDP_ALLOCATE_BUFFERS_VARS.
	SDP_MAX_ATTRIBUTE_RESULT_LEN	brief Maximum number of attributes to find ingroup sdp_config details This parameter defines the maximum number of attributes withing a service record the SDP server will return to the client.
	SDP_MAX_PDU_BUFFERS	brief Maximum number of SDP server PDU buffers. ingroup sdp_config details This parameter defines the maximum number of responses the SDP server can send at the same time.
	SDP_MAX_SEARCH_RESULT_LEN	brief Maximum number of service records to find. ingroup sdp_config details This parameter defines the maximum number of service records the SDP server will return to the client.
	__SDP_CONFIG_H	This is macro __SDP_CONFIG_H.

SPP Configuration Macros

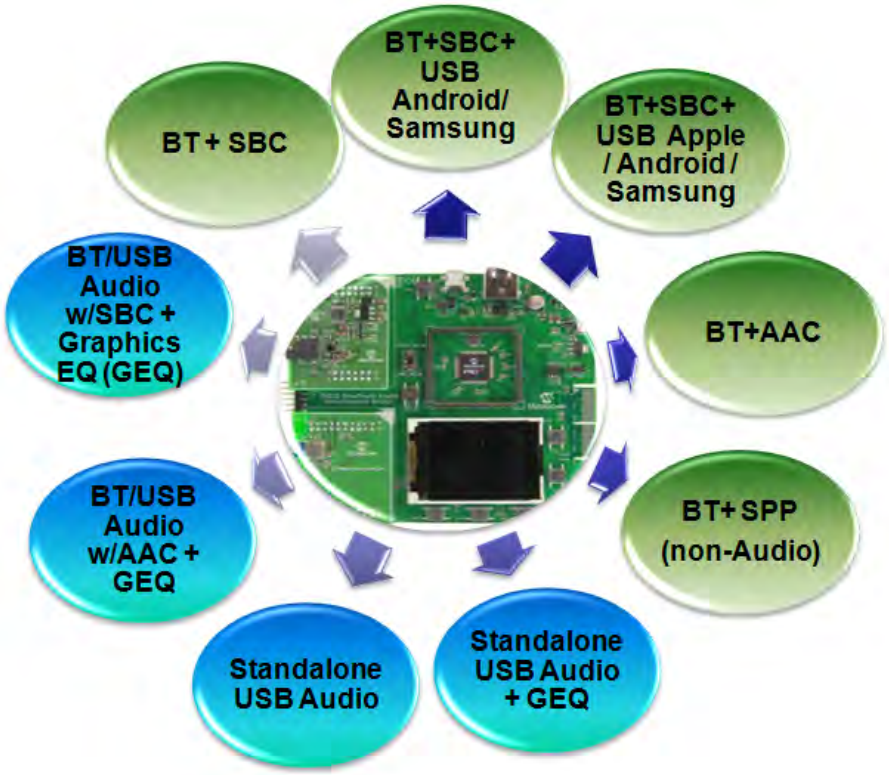
	Name	Description
	SPP_ALLOCATE_BUFFERS_RAM_SIZE_VAR	This is macro SPP_ALLOCATE_BUFFERS_RAM_SIZE_VAR.
	SPP_ALLOCATE_BUFFERS_VARS	This is macro SPP_ALLOCATE_BUFFERS_VARS.
	SPP_DECLARE_FRAME_BUFFERS	This is macro SPP_DECLARE_FRAME_BUFFERS.
	SPP_DISABLE_BUFFERING	This is macro SPP_DISABLE_BUFFERING.
	SPP_FRAME_BUFFERS_RAM_SIZE	This is macro SPP_FRAME_BUFFERS_RAM_SIZE.
	SPP_FRAME_BUFFERS_SIZE	This is macro SPP_FRAME_BUFFERS_SIZE.
	SPP_MAX_PORTS	brief Maximum number of SPP ports. ingroup spp_config details This parameter defines the maximum number of SPP port that can be open between the local and remote devices. If RFCOMM_ENABLE_MULTIDEVICE_CHANNELS is FALSE (default) this value should be equal to RFCOMM_MAX_SERVER_CHANNELS. If RFCOMM_ENABLE_MULTIDEVICE_CHANNELS is TRUE this value should be between RFCOMM_MAX_SERVER_CHANNELS and RFCOMM_MAX_SERVER_CHANNELS * RFCOMM_MAX_SESSIONS.
	__SPP_CONFIG_H	This is macro __SPP_CONFIG_H.

Description

Each major function, protocol and prototype within the library has a configuration header file. Generally these are not modified to utilize the PIC32 Bluetooth Stack with the demonstrations provided. The configuration files for each function are found in the `<install_dir>/framework/bluetooth/cdbt` directory for the PIC32 Bluetooth Basic Stack. The configuration is located in `<sub-directory>_config.h` for that particular function.

In the overall application a number of library controls are exposed to the user. These functions can be found in `user_config.h` in the application folder provided. The user of these configurations is referenced in the *"PIC32 Bluetooth Audio Development Kit Reference Guide*" (DS70005140),

which is available from the Microchip website (www.microchip.com).



Bluetooth Support Configuration Macros

ATT_ALLOCATE_BUFFERS Function

File

bt_oem_config.h

C

```
ATT_ALLOCATE_BUFFERS();
```

Description

This is function ATT_ALLOCATE_BUFFERS.

ATT_CLIENT_ALLOCATE_BUFFERS Function

File

bt_oem_config.h

C

```
ATT_CLIENT_ALLOCATE_BUFFERS();
```

Description

This is function ATT_CLIENT_ALLOCATE_BUFFERS.

FTP_ALLOCATE_BUFFERS Function

File

bt_oem_config.h

C

```
FTP_ALLOCATE_BUFFERS();
```

Description

This is function FTP_ALLOCATE_BUFFERS.

GATT_CLIENT_ALLOCATE_BUFFERS Function

File

bt_oem_config.h

C

```
GATT_CLIENT_ALLOCATE_BUFFERS();
```

Description

This is function GATT_CLIENT_ALLOCATE_BUFFERS.

HCI_ALLOCATE_BUFFERS Function

File

bt_oem_config.h

C

```
HCI_ALLOCATE_BUFFERS();
```

Description

This is function HCI_ALLOCATE_BUFFERS.

HFP_ALLOCATE_BUFFERS Function

File

bt_oem_config.h

C

```
HFP_ALLOCATE_BUFFERS();
```

Description

This is function HFP_ALLOCATE_BUFFERS.

HID_ALLOCATE_BUFFERS Function

File

bt_oem_config.h

C

```
HID_ALLOCATE_BUFFERS();
```

Description

This is function HID_ALLOCATE_BUFFERS.

HSP_AG_ALLOCATE_BUFFERS Function

File

bt_oem_config.h

C

```
HSP_AG_ALLOCATE_BUFFERS();
```

Description

This is function HSP_AG_ALLOCATE_BUFFERS.

HSP_ALLOCATE_BUFFERS Function

File

bt_oem_config.h

C

```
HSP_ALLOCATE_BUFFERS();
```

Description

This is function HSP_ALLOCATE_BUFFERS.

IAP_ALLOCATE_BUFFERS Function

File

bt_oem_config.h

C

```
IAP_ALLOCATE_BUFFERS();
```

Description

This is function IAP_ALLOCATE_BUFFERS.

IAP_BT_ALLOCATE_BUFFERS Function

File

bt_oem_config.h

C

```
IAP_BT_ALLOCATE_BUFFERS();
```

Description

This is function IAP_BT_ALLOCATE_BUFFERS.

IAP2_ALLOCATE_BUFFERS Function

File

bt_oem_config.h

C

```
IAP2_ALLOCATE_BUFFERS();
```

Description

This is function IAP2_ALLOCATE_BUFFERS.

L2CAP_ALLOCATE_BUFFERS Function

File

bt_oem_config.h

C

```
L2CAP_ALLOCATE_BUFFERS();
```

Description

This is function L2CAP_ALLOCATE_BUFFERS.

OBEX_ALLOCATE_BUFFERS Function

File

bt_oem_config.h

C

```
OBEX_ALLOCATE_BUFFERS();
```

Description

This is function OBEX_ALLOCATE_BUFFERS.

PBAP_ALLOCATE_BUFFERS Function

File

bt_oem_config.h

C

```
PBAP_ALLOCATE_BUFFERS();
```

Description

This is function PBAP_ALLOCATE_BUFFERS.

RFCOMM_ALLOCATE_BUFFERS Function

File

bt_oem_config.h

C

```
RFCOMM_ALLOCATE_BUFFERS();
```

Description

This is function RFCOMM_ALLOCATE_BUFFERS.

SDP_ALLOCATE_BUFFERS Function

File

bt_oem_config.h

C

```
SDP_ALLOCATE_BUFFERS();
```

Description

This is function SDP_ALLOCATE_BUFFERS.

SMP_ALLOCATE_BUFFERS Function

File

bt_oem_config.h

C

```
SMP_ALLOCATE_BUFFERS();
```

Description

This is function SMP_ALLOCATE_BUFFERS.

SPP_ALLOCATE_BUFFERS Function

File

bt_oem_config.h

C

```
SPP_ALLOCATE_BUFFERS();
```

Description

This is function SPP_ALLOCATE_BUFFERS.

BT_ENABLE_BLE Macro

File

bt_oem_config.h

C

```
#define BT_ENABLE_BLE
```

Description

This is macro BT_ENABLE_BLE.

BT_INCLUDE_RFCOMM Macro

File

bt_oem_config.h

C

```
#define BT_INCLUDE_RFCOMM
```

Description

This is macro BT_INCLUDE_RFCOMM.

BT_LOG_LEVEL_MAX Macro

File

bt_oem_config.h

C

```
#define BT_LOG_LEVEL_MAX BT_LOG_LEVEL_OFF
```

Description

This is macro BT_LOG_LEVEL_MAX.

BT_LOG_LEVEL_MIN Macro

File

bt_oem_config.h

C

```
#define BT_LOG_LEVEL_MIN BT_LOG_LEVEL
```

Description

This is macro BT_LOG_LEVEL_MIN.

__BT_APP_CONFIG_H Macro

File

bt_oem_config.h

C

```
#define __BT_APP_CONFIG_H
```

Description

This is macro __BT_APP_CONFIG_H.

BT_INCLUDE_IAP Macro

File

bt_oem_config.h

C

```
#define BT_INCLUDE_IAP
```

Description

This is macro BT_INCLUDE_IAP.

BT_INCLUDE_IAP2 Macro

File

bt_oem_config.h

C

```
#define BT_INCLUDE_IAP2
```

Description

This is macro BT_INCLUDE_IAP2.

IAP_BT_MAX_TRANSPORTS Macro

File

bt_oem_config.h

C

```
#define IAP_BT_MAX_TRANSPORTS IAPEA_MAX_SESSIONS
```

Description

This is macro IAP_BT_MAX_TRANSPORTS.

IAP_MAX_SESSIONS Macro

File

bt_oem_config.h

C

```
#define IAP_MAX_SESSIONS IAPEA_MAX_SESSIONS
```

Description

This is macro IAP_MAX_SESSIONS.

IAP_RX_BUFFER_SIZE Macro

File

bt_oem_config.h

C

```
#define IAP_RX_BUFFER_SIZE IAPEA_MAX_PACKET_SIZE
```

Description

This is macro IAP_RX_BUFFER_SIZE.

IAP2_MAX_PACKET_SIZE Macro

File

bt_oem_config.h

C

```
#define IAP2_MAX_PACKET_SIZE IAPEA_MAX_PACKET_SIZE
```

Description

This is macro IAP2_MAX_PACKET_SIZE.

IAP2_MAX_SESSIONS Macro

File

bt_oem_config.h

C

```
#define IAP2_MAX_SESSIONS IAPEA_MAX_SESSIONS
```

Description

This is macro IAP2_MAX_SESSIONS.

AVCTP Configuration Macros

__AVCTP_CONFIG_H Macro

File

avctp_config.h

C

```
#define __AVCTP_CONFIG_H
```

Description

This is macro __AVCTP_CONFIG_H.

AVCTP_ALLOCATE_BUFFERS_RAM_SIZE_VAR Macro

File

avctp_config.h

C

```
#define AVCTP_ALLOCATE_BUFFERS_RAM_SIZE_VAR
```

Description

- defgroup avctp_config Configuration

- ingroup avctp

*

- This module describes parameters used to configure AVCTP layer.

*

- dotstack is customized using a configuration file. The configuration file tailors the dotstack to the application being built. It has to have the structure shown below.

* code #include "cdbt/bt/bt_std.h"

// HCI, L2CAP and SDP must always be present

// HCI configuration parameters #define HCI_MAX_CMD_BUFFERS ... #define HCI_MAX_DATA_BUFFERS ... #define HCI_MAX_HCI_CONNECTIONS ... #define HCI_RX_BUFFER_LEN ... #define HCI_TX_BUFFER_LEN ... #define HCI_L2CAP_BUFFER_LEN ... #define HCI_MAX_CMD_PARAM_LEN ...

// L2CAP configuration parameters #define L2CAP_MAX_CMD_BUFFERS ... #define L2CAP_MAX_FRAME_BUFFERS ... #define L2CAP_MAX_PSMS ... #define L2CAP_MAX_CHANNELS ...

// SDP configuration parameters #define SDP_MAX_SEARCH_RESULT_LEN ... #define SDP_MAX_ATTRIBUTE_RESULT_LEN ...

// Depending on protocols and profiles used below goes configuration parameters // for each used module. E.g., to use and configure AVRCP, // the following values must be defined:

#define BT_INCLUDE_AVCTP // tells dotstack to compile in AVCTP support #define AVCTP_MAX_CHANNELS ... #define AVCTP_MAX_TRANSPORT_CHANNELS ... #define AVCTP_MAX_RX_MESSAGE_LEN ... #define AVCTP_MAX_MESSAGE_BUFFERS ...

#include "cdbt/bt/bt_oem_config.h"

endcode

AVCTP_ALLOCATE_BUFFERS_VARS Macro

File

avctp_config.h

C

```
#define AVCTP_ALLOCATE_BUFFERS_VARS \
    bt_avctp_channel_t                  _avctp_channels[AVCTP_MAX_CHANNELS];    \
    const bt_byte                       _avctp_max_channels = AVCTP_MAX_CHANNELS; \
    bt_avctp_transport_t                _avctp_transports[AVCTP_MAX_TRANSPORT_CHANNELS];    \
    const bt_byte                       _avctp_max_transports = AVCTP_MAX_TRANSPORT_CHANNELS; \
    const bt_uint                       _avctp_max_rx_message_len = AVCTP_MAX_RX_MESSAGE_LEN; \
    bt_byte                             _avctp_rx_buffers[(AVCTP_MAX_RX_MESSAGE_LEN) *
(AVCTP_MAX_TRANSPORT_CHANNELS)]; \
    bt_buffer_header_t                  _avctp_message_buffer_headers[(AVCTP_MAX_MESSAGE_BUFFERS) *
(AVCTP_MAX_TRANSPORT_CHANNELS)];    \
    bt_avctp_message_t                  _avctp_message_buffers[(AVCTP_MAX_MESSAGE_BUFFERS) *
(AVCTP_MAX_TRANSPORT_CHANNELS)];    \
    const bt_byte                       _avctp_max_message_buffers = AVCTP_MAX_MESSAGE_BUFFERS; \
    \
    AVCTP_ALLOCATE_BUFFERS_RAM_SIZE_VAR    \
```

Description

brief Maximum number of message buffers ingroup avctp_config

details This parameter defines the maximum number of buffer available for sending message.

AVCTP_MAX_CHANNELS Macro

File

avctp_config.h

C

```
#define AVCTP_MAX_CHANNELS
```

Description

brief Maximum number of AVCTP channels ingroup avctp_config

details This parameter defines the maximum number of channels a local device can have with remote devices.

AVCTP_MAX_TRANSPORT_CHANNELS Macro

File

avctp_config.h

C

```
#define AVCTP_MAX_TRANSPORT_CHANNELS
```

Description

brief Maximum number of AVCTP transports ingroup avctp_config

details This parameter defines the maximum number of transports a local device can have with remote devices. This value should not exceed AVCTP_MAX_CHANNELS.

AVDTP Configuration Macros

AVDTP_ALLOCATE_BUFFERS_FUNCTION Macro

File

avdtp_config.h

C

```
#define AVDTP_ALLOCATE_BUFFERS_FUNCTION \
    void _bt_avdtp_allocate_buffers(void)    \
    {    \
    } \
```

Description

This is macro AVDTP_ALLOCATE_BUFFERS_FUNCTION.

AVDTP_ALLOCATE_BUFFERS_RAM_SIZE_VAR Macro

File

avdtp_config.h

C

```
#define AVDTP_ALLOCATE_BUFFERS_RAM_SIZE_VAR
```

Description

- defgroup avdtp_config Configuration
- ingroup avdtp

*

- This module describes parameters used to configure AVDTP layer.

*

- dotstack is customized using a configuration file. The configuration file tailors the dotstack to the application being built. It has to have the structure shown below.

* code #include "cdbt/bt/bt_std.h"

// HCI, L2CAP and SDP must always be present

// HCI configuration parameters #define HCI_MAX_CMD_BUFFERS ... #define HCI_MAX_DATA_BUFFERS ... #define HCI_MAX_HCI_CONNECTIONS ... #define HCI_RX_BUFFER_LEN ... #define HCI_TX_BUFFER_LEN ... #define HCI_L2CAP_BUFFER_LEN ... #define HCI_MAX_CMD_PARAM_LEN ...

// L2CAP configuration parameters #define L2CAP_MAX_CMD_BUFFERS ... #define L2CAP_MAX_FRAME_BUFFERS ... #define L2CAP_MAX_PSMS ... #define L2CAP_MAX_CHANNELS ...

// SDP configuration parameters #define SDP_MAX_SEARCH_RESULT_LEN ... #define SDP_MAX_ATTRIBUTE_RESULT_LEN ...

// Depending on protocols and profiles used below goes configuration parameters // for each used module. E.g., to use and configure AVDTP & A2DP (this one does not need configuration), // the following values must be defined:

#define BT_INCLUDE_AVDTP // tells dotstack to compile in AVDTP support #define AVDTP_MAX_SEP ... #define AVDTP_MAX_STREAMS ... #define AVDTP_MAX_REMOTE_DEVICES ... #define AVDTP_MAX_CMD_BUFFERS ... #define AVDTP_MAX_TRANSPORT_CHANNELS ... #define AVDTP_MAX_TX_BUFFER_LEN ... #define AVDTP_MAX_CMD_PARAM_LEN ... #define AVDTP_CODEC_CONFIG_BUFFER_LEN ...

#include "cdbt/bt/bt_oem_config.h"
endcode

AVDTP_ALLOCATE_BUFFERS_VARS Macro

File

avdtp_config.h

C

```
#define AVDTP_ALLOCATE_BUFFERS_VARS \
    bt_avdtp_sep_t                      _avdtp_seps[AVDTP_MAX_SEP];    \
    const bt_byte                       _avdtp_max_seps = AVDTP_MAX_SEP; \
    bt_avdtp_stream_t                   _avdtp_streams[AVDTP_MAX_STREAMS];    \
    const bt_byte                       _avdtp_max_streams = AVDTP_MAX_STREAMS; \
    bt_avdtp_control_channel_t          _avdtp_control_channels[AVDTP_MAX_REMOTE_DEVICES];    \
    const bt_byte                       _avdtp_max_control_channels = AVDTP_MAX_REMOTE_DEVICES; \
    const bt_byte                       _avdtp_max_tx_buffer_len = AVDTP_MAX_TX_BUFFER_LEN; \
    bt_byte                             _avdtp_tx_buffers[(AVDTP_MAX_TX_BUFFER_LEN) *
(AVDTP_MAX_REMOTE_DEVICES)]; \
    bt_avdtp_transport_channel_t        _avdtp_transport_channels[AVDTP_MAX_TRANSPORT_CHANNELS];    \
    const bt_byte                       _avdtp_max_transport_channels = AVDTP_MAX_TRANSPORT_CHANNELS; \
    bt_buffer_header_t                  _avdtp_cmd_buffer_headers[(AVDTP_MAX_CMD_BUFFERS) *
(AVDTP_MAX_REMOTE_DEVICES)];    \
    bt_avdtp_control_cmd_t              _avdtp_cmd_buffers[(AVDTP_MAX_CMD_BUFFERS) *
(AVDTP_MAX_REMOTE_DEVICES)];    \
    const bt_byte                       _avdtp_max_cmd_buffers = AVDTP_MAX_CMD_BUFFERS; \
    bt_byte                             _avdtp_cmd_param_buffers[(AVDTP_MAX_CMD_BUFFERS) *
(AVDTP_MAX_REMOTE_DEVICES) * (AVDTP_MAX_CMD_PARAM_LEN)];    \
    const bt_uint                       _avdtp_max_cmd_param_len = AVDTP_MAX_CMD_PARAM_LEN;    \
    bt_byte                             _avdtp_codec_cfg_buffers[(AVDTP_MAX_STREAMS) * 2 *
(AVDTP_CODEC_CONFIG_BUFFER_LEN)];    \
    const bt_byte                       _avdtp_max_codec_config_buffer_len =
AVDTP_CODEC_CONFIG_BUFFER_LEN;    \
    bt_buffer_header_t                  _avdtp_sep_cfg_buffer_headers[(AVDTP_MAX_STREAMS) * 2];    \
    bt_avdtp_sep_capabilities_t         _avdtp_sep_cfg_buffers[(AVDTP_MAX_STREAMS) * 2];    \
    bt_avdtp_sep_capabilities_t         _avdtp_rcv_sep_caps;    \
    bt_byte                             _avdtp_rcv_sep_codec_cfg_buffer[(AVDTP_CODEC_CONFIG_BUFFER_LEN)];
\
    bt_byte                             _avdtp_listen_sep_buffers[(AVDTP_MAX_STREAMS) * (AVDTP_MAX_SEP)];
\
    bt_byte                             _avdtp_l2cap_media_packet_buffer[HCI_L2CAP_BUFFER_LEN -
L2CAP_HEADER_LEN];    \
    \
    AVDTP_ALLOCATE_BUFFERS_RAM_SIZE_VAR    \
```

Description

This is macro AVDTP_ALLOCATE_BUFFERS_VARS.

AVDTP_CMD_ABORT Macro

File

avdtp_control.h

C

```
#define AVDTP_CMD_ABORT 10
```

Description

This is macro AVDTP_CMD_ABORT.

AVDTP_CMD_CLOSE Macro

File

avdtp_control.h

C

```
#define AVDTP_CMD_CLOSE 8
```

Description

This is macro AVDTP_CMD_CLOSE.

AVDTP_CMD_DISCOVER Macro

File

avdtp_control.h

C

```
#define AVDTP_CMD_DISCOVER 1
```

Description

This is macro AVDTP_CMD_DISCOVER.

AVDTP_CMD_GET_CAPABILITIES Macro

File

avdtp_control.h

C

```
#define AVDTP_CMD_GET_CAPABILITIES 2
```

Description

This is macro AVDTP_CMD_GET_CAPABILITIES.

AVDTP_CMD_GET_CONFIGURATION Macro

File

avdtp_control.h

C

```
#define AVDTP_CMD_GET_CONFIGURATION 4
```

Description

This is macro AVDTP_CMD_GET_CONFIGURATION.

AVDTP_CMD_OPEN Macro

File

avdtp_control.h

C

```
#define AVDTP_CMD_OPEN 6
```

Description

This is macro AVDTP_CMD_OPEN.

AVDTP_CMD_RECONFIGURE Macro

File

avdtp_control.h

C

```
#define AVDTP_CMD_RECONFIGURE 5
```

Description

This is macro AVDTP_CMD_RECONFIGURE.

AVDTP_CMD_SECURITY_CONTROL Macro

File

avdtp_control.h

C

```
#define AVDTP_CMD_SECURITY_CONTROL 11
```

Description

This is macro AVDTP_CMD_SECURITY_CONTROL.

AVDTP_CMD_SET_CONFIGURATION Macro

File

avdtp_control.h

C

```
#define AVDTP_CMD_SET_CONFIGURATION 3
```

Description

This is macro AVDTP_CMD_SET_CONFIGURATION.

AVDTP_CMD_START Macro

File

avdtp_control.h

C

```
#define AVDTP_CMD_START 7
```

Description

This is macro AVDTP_CMD_START.

AVDTP_CMD_SUSPEND Macro

File

avdtp_control.h

C

```
#define AVDTP_CMD_SUSPEND 9
```

Description

This is macro AVDTP_CMD_SUSPEND.

AVDTP_CODEC_CONFIG_BUFFER_LEN Macro

File

avdtp_config.h

C

```
#define AVDTP_CODEC_CONFIG_BUFFER_LEN 16
```

Description

brief Size of the buffer used to store codec specific configuration. ingroup avdtp_config

details Each codec uses unique configuration which can take different amount of memory. This parameter defines the size of the buffer for storing codec's configuration. The value of 16 is sufficient for SBC, AAC and MPEG1,2. If vendor specific codec is to be used this value may need to increased.

AVDTP_MAX_CMD_BUFFERS Macro

File

avdtp_config.h

C

```
#define AVDTP_MAX_CMD_BUFFERS
```

Description

brief Maximum number of command buffers. ingroup avdtp_config

details This parameter defines the number of buffers reserved for sending commands to a remote device over its control channel. Each channel uses its own buffers so the total number of buffers is AVDTP_MAX_REMOTE_DEVICES * AVDTP_MAX_CMD_BUFFERS. The minimum value is 1. The maximum value is 255. 2 is usually sufficient.

AVDTP_MAX_CMD_PARAM_LEN Macro

File

avdtp_config.h

C

```
#define AVDTP_MAX_CMD_PARAM_LEN (AVDTP_MAX_TX_BUFFER_LEN - 2)
```

Description

brief Maximum length of control command parameters ingroup avdtp_config

details This parameter defines the maximum length of all command parameters. The value should not exceed AVDTP_MAX_TX_BUFFER_LEN - 2 (command header).

AVDTP_MAX_REMOTE_DEVICES Macro

File

avdtp_config.h

C

```
#define AVDTP_MAX_REMOTE_DEVICES
```

Description

brief Maximum number of remote devices a local device can be connected to ingroup avdtp_config

details This parameter defines the number of remote devices a local device can have simultaneous connections to (i.e. control channels). This value should not exceed AVDTP_MAX_STREAMS. For each remote device AVDTP creates one control channel regardless of the number of streams between the local and the remote devices. Assuming that the local devices wants to have only one channel with each remote device and if AVDTP_MAX_REMOTE_DEVICES > AVDTP_MAX_STREAMS all memory reserved for devices in excess of AVDTP_MAX_STREAMS will be wasted. The minimum value is 1. The maximum value is 255.

AVDTP_MAX_SEP Macro

File

avdtp_config.h

C

```
#define AVDTP_MAX_SEP
```

Description

brief Maximum number of SEPs that can be exposed by a local device ingroup avdtp_config

details This parameter defines the number of SEPs an application can expose to remote devices. The minimum value is 1. The maximum value is 255.

AVDTP_MAX_STREAMS Macro

File

avdtp_config.h

C

```
#define AVDTP_MAX_STREAMS
```

Description

brief Maximum number of streams that can be exposed by a local device ingroup avdtp_config

details This parameter defines the number of streams an application can open between local and remote devices. This value can be different from AVDTP_MAX_SEP but should not exceed it. Since each SEP can only be used once the local device can only have as much streams as there are SEPs. If AVDTP_MAX_STREAMS > AVDTP_MAX_SEP all memory reserved for streams in excess of AVDTP_MAX_SEP will be wasted. The minimum value is 1. The maximum value is 255.

AVDTP_MAX_TRANSPORT_CHANNELS Macro

File

avdtp_config.h

C

```
#define AVDTP_MAX_TRANSPORT_CHANNELS
```

Description

brief Maximum number of transport channels. ingroup avdtp_config

details Depending on the SEP capabilities (multiplexing, recovery, reporting) each stream may need up to 3 transport channels. E.g., if multiplexing is not supported and recovery and reporting are supported a stream will use 3 transport channels - 1 for media transport session, 1 for recovery transport session and 1 for reporting transport session. If multiplexing is not supported and only reporting is supported the stream will use 2 transport channels - 1 for media transport session and 1 for reporting transport session. If multiplexing is supported the stream will need only 1 transport channel for all supported transport session. With multiplexing even different streams can share the same transport channel. dotstack currently does not support multiplexing, recovery and reporting. So each stream needs its own transport channel for it media transport session. Hence, AVDTP_MAX_TRANSPORT_CHANNELS must be equal to AVDTP_MAX_STREAMS

AVDTP_MAX_TX_BUFFER_LEN Macro

File

avdtp_config.h

C

```
#define AVDTP_MAX_TX_BUFFER_LEN 32
```

Description

brief Size of the transmit buffer. ingroup avdtp_config

details This parameter defines the size of the buffer used to send AVDTP control commands to L2CAP layer. Each control channel has it own buffer so the total amount of memory allocated for these buffers is AVDTP_MAX_TX_BUFFER_LEN) * AVDTP_MAX_REMOTE_DEVICES. The minimum value is 1. The maximum value is 255. The value of 32 is usually sufficient.

AVRCP Configuration Macros

__AVRCP_CONFIG_H Macro

File

avrcp_config.h

C

```
#define __AVRCP_CONFIG_H
```

Description

This is macro __AVRCP_CONFIG_H.

AVRCP_ALLOCATE_BUFFERS_RAM_SIZE_VAR Macro

File

avrcp_config.h

C

```
#define AVRCP_ALLOCATE_BUFFERS_RAM_SIZE_VAR
```

Description

- defgroup avrcp_config Configuration
- ingroup avrcp

*

- This module describes parameters used to configure AVRCP layer.

*

- dotstack is customized using a configuration file. The configuration file tailors the dotstack to the application being built. It has to have the structure shown below.

* code #include "cdbt/bt/bt_std.h"

// HCI, L2CAP and SDP must always be present

// HCI configuration parameters #define HCI_MAX_CMD_BUFFERS ... #define HCI_MAX_DATA_BUFFERS ... #define HCI_MAX_HCI_CONNECTIONS ... #define HCI_RX_BUFFER_LEN ... #define HCI_TX_BUFFER_LEN ... #define HCI_L2CAP_BUFFER_LEN ... #define HCI_MAX_CMD_PARAM_LEN ...

// L2CAP configuration parameters #define L2CAP_MAX_CMD_BUFFERS ... #define L2CAP_MAX_FRAME_BUFFERS ... #define L2CAP_MAX_PSMS ... #define L2CAP_MAX_CHANNELS ...

// SDP configuration parameters #define SDP_MAX_SEARCH_RESULT_LEN ... #define SDP_MAX_ATTRIBUTE_RESULT_LEN ...

// Depending on protocols and profiles used below goes configuration parameters // for each used module. E.g., to use and configure AVRCP, // the following values must be defined:

#define BT_INCLUDE_AVCTP // tells dotstack to compile in AVCTP support #define AVCTP_MAX_CHANNELS ... #define AVCTP_MAX_TRANSPORT_CHANNELS ... #define AVCTP_MAX_RX_MESSAGE_LEN ... #define AVCTP_MAX_MESSAGE_BUFFERS ...

#define BT_INCLUDE_AVRCP // tells dotstack to compile in AVRCP support #define AVRCP_MAX_CHANNELS ... #define AVRCP_MAX_CMD_BUFFERS ... #define AVRCP_MAX_CMD_PARAM_LEN ... #define AVRCP_MAX_SEARCH_RESULTS ... #define AVRCP_MAX_DEVICE_NAME_LEN ... #define AVRCP_CMD_TIMEOUT ...

#include "cdbt/bt/bt_oem_config.h"

endcode

AVRCP_ALLOCATE_BUFFERS_VARS Macro

File

avrcp_config.h

C

```
#define AVRCP_ALLOCATE_BUFFERS_VARS \
    bt_avrcp_channel_t  _avrcp_channels[AVRCP_MAX_CHANNELS];    \
    const bt_byte       _avrcp_max_channels = AVRCP_MAX_CHANNELS; \
    \
    bt_buffer_header_t  _avrcp_cmd_buffer_headers[(AVRCP_MAX_CMD_BUFFERS) * (AVRCP_MAX_CHANNELS)];    \
    bt_av_command_t     _avrcp_cmd_buffers[(AVRCP_MAX_CMD_BUFFERS) * (AVRCP_MAX_CHANNELS)];    \
    const bt_byte       _avrcp_max_cmd_buffers = AVRCP_MAX_CMD_BUFFERS; \
    bt_byte             _avrcp_cmd_param_buffers[(AVRCP_MAX_CMD_BUFFERS) * (AVRCP_MAX_CMD_PARAM_LEN) *
(AVRCP_MAX_CHANNELS)];    \
    const bt_int        _avrcp_max_cmd_param_len = AVRCP_MAX_CMD_PARAM_LEN;    \
    const bt_byte       _avrcp_max_search_results = AVRCP_MAX_SEARCH_RESULTS;    \
    bt_avrcp_device_t   _avrcp_devices_buffer[AVRCP_MAX_SEARCH_RESULTS];    \
    const bt_byte       _avrcp_max_device_name_len = AVRCP_MAX_DEVICE_NAME_LEN;    \
    bt_byte             _avrcp_device_name_buffers[(AVRCP_MAX_SEARCH_RESULTS) * (AVRCP_MAX_DEVICE_NAME_LEN
+ 1)];    \
    const bt_byte       _avrcp_cmd_timeout = (AVRCP_CMD_TIMEOUT) / 100;    \
    \
    void (*_bt_avrcp_response_sent_handler)(bt_avrcp_mgr_t* mgr, bt_avctp_evt_response_sent_t* evt_param) =
AVRCP_RESPONSE_SENT_HANDLER;    \
    void (*_bt_avrcp_command_handler)(bt_avrcp_mgr_t* mgr, bt_avctp_evt_command_received_t* evt_param) =
```

```
AVRCP_COMMAND_HANDLER;    \
    void (*_bt_avrcp_command_sent_handler)(bt_avrcp_mgr_t* mgr, bt_avctp_evt_command_sent_t* evt_param) =
AVRCP_COMMAND_SENT_HANDLER;    \
    void (*_bt_avrcp_response_handler)(bt_avrcp_mgr_t* mgr, bt_avctp_evt_response_received_t* evt_param) =
AVRCP_RESPONSE_HANDLER;    \
    \
    AVRCP_ALLOCATE_BUFFERS_RAM_SIZE_VAR    \
```

Description

This is macro AVRCP_ALLOCATE_BUFFERS_VARS.

AVRCP_CMD_TIMEOUT Macro

File

avrcp_config.h

C

```
#define AVRCP_CMD_TIMEOUT 10000
```

Description

brief Command timeout ingroup avrcp_config

details This parameter defines the amount of time in milliseconds AVRCP waits for a response to a request. If not defined the default value of 10000 (10 secconds) is used.

AVRCP_MAX_CHANNELS Macro

File

avrcp_config.h

C

```
#define AVRCP_MAX_CHANNELS
```

Description

brief Maximum number of remote devices a local device can be connected to ingroup avrcp_config

details This parameter defines the number of remote devices a local device can have simultaneous connections to (i.e. channels). This value should not exceed AVCTP_MAX_CHANNELS.

AVRCP_MAX_CMD_BUFFERS Macro

File

avrcp_config.h

C

```
#define AVRCP_MAX_CMD_BUFFERS 1
```

Description

brief Maximum number of command buffers. ingroup avrcp_config

details This parameter defines the number of buffers reserved for sending commands to a remote device over its control channel. Each channel uses its own buffers so the total number of buffers is AVRCP_MAX_CHANNELS * AVRCP_MAX_CMD_BUFFERS. The minimum value is 1. The maximum value is 255. If not define one buffer for each channel is reserved.

AVRCP_MAX_CMD_PARAM_LEN Macro

File

avrcp_config.h

C

```
#define AVRCP_MAX_CMD_PARAM_LEN 512
```

Description

brief Maximum length of command parameters ingroup avrcp_config

details This parameter defines the maximum length of all command parameters. If not defined the default value of 512 is used.

AVRCP_MAX_DEVICE_NAME_LEN Macro

File

avrcp_config.h

C

```
#define AVRCP_MAX_DEVICE_NAME_LEN 20
```

Description

brief Maximum length of device name ingroup avrcp_config

details This parameter defines the size of the buffer used to store device's name while searching for nearby targets with bt_avrcp_find_targets. If the name of the device is longer than AVRCP_MAX_DEVICE_NAME_LEN it is truncated to AVRCP_MAX_DEVICE_NAME_LEN. If not defined the default value of 20 is used.

AVRCP_MAX_SEARCH_RESULTS Macro

File

avrcp_config.h

C

```
#define AVRCP_MAX_SEARCH_RESULTS 7
```

Description

brief Maximum number of devices to find ingroup avrcp_config

details This parameter defines the maximum number of devices bt_avrcp_find_targets can find. If not defined the default value of 7 is used.

HCI Configuration Macros

BT_ENABLE_SCO Macro

File

hci_config.h

C

```
#define BT_ENABLE_SCO BT_FALSE
```

Description

This is macro BT_ENABLE_SCO.

BT_HCI_EVT_AUTHENTICATION_COMPLETE_HANDLER Macro

File

hci_config_event_handlers.h

C

```
#define BT_HCI_EVT_AUTHENTICATION_COMPLETE_HANDLER bt_hci_evt_default_handler
```

Description

This is macro BT_HCI_EVT_AUTHENTICATION_COMPLETE_HANDLER.

BT_HCI_EVT_CONNECTION_COMPLETE_HANDLER Macro

File

hci_config_event_handlers.h

C

```
#define BT_HCI_EVT_CONNECTION_COMPLETE_HANDLER bt_hci_evt_default_handler
```

Description

This is macro BT_HCI_EVT_CONNECTION_COMPLETE_HANDLER.

BT_HCI_EVT_CONNECTION_REQUEST_HANDLER Macro

File

hci_config_event_handlers.h

C

```
#define BT_HCI_EVT_CONNECTION_REQUEST_HANDLER bt_hci_evt_default_handler
```

Description

This is macro BT_HCI_EVT_CONNECTION_REQUEST_HANDLER.

BT_HCI_EVT_ENCRYPTION_CHANGE_HANDLER Macro

File

hci_config_event_handlers.h

C

```
#define BT_HCI_EVT_ENCRYPTION_CHANGE_HANDLER bt_hci_evt_encryption_change_handler
```

Description

This is macro BT_HCI_EVT_ENCRYPTION_CHANGE_HANDLER.

BT_HCI_EVT_EXTENDED_INQUIRY_RESULT_HANDLER Macro

File

hci_config_event_handlers.h

C

```
#define BT_HCI_EVT_EXTENDED_INQUIRY_RESULT_HANDLER bt_hci_evt_default_handler
```

Description

This is macro BT_HCI_EVT_EXTENDED_INQUIRY_RESULT_HANDLER.

BT_HCI_EVT_INQUIRY_COMPLETE_HANDLER Macro

File

hci_config_event_handlers.h

C

```
#define BT_HCI_EVT_INQUIRY_COMPLETE_HANDLER bt_hci_evt_default_handler
```

Description

This is macro BT_HCI_EVT_INQUIRY_COMPLETE_HANDLER.

BT_HCI_EVT_INQUIRY_RESULT_HANDLER Macro

File

hci_config_event_handlers.h

C

```
#define BT_HCI_EVT_INQUIRY_RESULT_HANDLER bt_hci_evt_default_handler
```

Description

This is macro BT_HCI_EVT_INQUIRY_RESULT_HANDLER.

BT_HCI_EVT_INQUIRY_RESULT_WITH_RSSI_HANDLER Macro

File

hci_config_event_handlers.h

C

```
#define BT_HCI_EVT_INQUIRY_RESULT_WITH_RSSI_HANDLER bt_hci_evt_default_handler
```

Description

This is macro BT_HCI_EVT_INQUIRY_RESULT_WITH_RSSI_HANDLER.

BT_HCI_EVT_LINK_KEY_NOTIFICATION_HANDLER Macro

File

hci_config_event_handlers.h

C

```
#define BT_HCI_EVT_LINK_KEY_NOTIFICATION_HANDLER bt_hci_evt_default_handler
```

Description

This is macro BT_HCI_EVT_LINK_KEY_NOTIFICATION_HANDLER.

BT_HCI_EVT_LINK_KEY_REQUEST_HANDLER Macro

File

hci_config_event_handlers.h

C

```
#define BT_HCI_EVT_LINK_KEY_REQUEST_HANDLER bt_hci_evt_default_handler
```

Description

This is macro BT_HCI_EVT_LINK_KEY_REQUEST_HANDLER.

BT_HCI_EVT_MODE_CHANGE_HANDLER Macro

File

hci_config_event_handlers.h

C

```
#define BT_HCI_EVT_MODE_CHANGE_HANDLER bt_hci_evt_default_handler
```

Description

This is macro BT_HCI_EVT_MODE_CHANGE_HANDLER.

BT_HCI_EVT_PIN_CODE_REQUEST_HANDLER Macro

File

hci_config_event_handlers.h

C

```
#define BT_HCI_EVT_PIN_CODE_REQUEST_HANDLER bt_hci_evt_default_handler
```

Description

This is macro BT_HCI_EVT_PIN_CODE_REQUEST_HANDLER.

BT_HCI_EVT_REMOTE_NAME_REQUEST_COMPLETE_HANDLER Macro

File

hci_config_event_handlers.h

C

```
#define BT_HCI_EVT_REMOTE_NAME_REQUEST_COMPLETE_HANDLER bt_hci_evt_default_handler
```

Description

This is macro BT_HCI_EVT_REMOTE_NAME_REQUEST_COMPLETE_HANDLER.

BT_HCI_EVT_ROLE_CHANGE_HANDLER Macro

File

hci_config_event_handlers.h

C

```
#define BT_HCI_EVT_ROLE_CHANGE_HANDLER bt_hci_evt_default_handler
```

Description

This is macro BT_HCI_EVT_ROLE_CHANGE_HANDLER.

BT_HCI_EVT_SYNCH_CONNECTION_COMPLETE_HANDLER Macro

File

hci_config_event_handlers.h

C

```
#define BT_HCI_EVT_SYNCH_CONNECTION_COMPLETE_HANDLER bt_hci_evt_default_handler
```

Description

This is macro BT_HCI_EVT_SYNCH_CONNECTION_COMPLETE_HANDLER.

BT_LE_EVT_HANDLER Macro

File

hci_config_event_handlers.h

C

```
#define BT_LE_EVT_HANDLER bt_hci_evt_default_handler
```

Description

This is macro BT_LE_EVT_HANDLER.

BT_SSP_EVT_HANDLER Macro

File

hci_config_event_handlers.h

C

```
#define BT_SSP_EVT_HANDLER bt_hci_evt_default_handler
```

Description

This is macro BT_SSP_EVT_HANDLER.

HCI_ALLOCATE_BUFFERS_FUNCTION Macro

File

hci_config.h

C

```
#define HCI_ALLOCATE_BUFFERS_FUNCTION \
    void _hci_allocate_buffers(void)    \
    {    \
        bt_byte i;    \
        _phci_ctrl->connections = _hci_connections;    \
        _zero_memory(_hci_connections, sizeof(bt_hci_conn_state_t) * HCI_MAX_HCI_CONNECTIONS);    \
        for (i = 0; i < HCI_MAX_HCI_CONNECTIONS; i++)    \
        {    \
            _hci_connections[i].recv_data = &_conn_state_recv_buffers[i * (HCI_L2CAP_BUFFER_LEN)];    \
            HCI_INIT_LE_CONN_STATES;    \
        }    \
        HCI_INIT_SCO_HANDLERS    \
        HCI_INIT_SSP_HANDLERS    \
        HCI_INIT_LE_CTRL_STATE    \
    } \
```

Description

This is macro HCI_ALLOCATE_BUFFERS_FUNCTION.

HCI_ALLOCATE_BUFFERS_RAM_SIZE_VAR Macro

File

hci_config.h

C

```
#define HCI_ALLOCATE_BUFFERS_RAM_SIZE_VAR
```

Description

This is macro HCI_ALLOCATE_BUFFERS_RAM_SIZE_VAR.

HCI_ALLOCATE_BUFFERS_VARS Macro

File

hci_config.h

C

```
#define HCI_ALLOCATE_BUFFERS_VARS \
    bt_buffer_header_t  _hci_cmd_buffer_headers[HCI_MAX_CMD_BUFFERS];    \
    bt_hci_command_t    _hci_cmd_buffers[HCI_MAX_CMD_BUFFERS];    \
    const bt_byte       _hci_max_cmd_buffers = HCI_MAX_CMD_BUFFERS; \
    bt_byte             _hci_cmd_param_buffers[HCI_MAX_CMD_BUFFERS * HCI_MAX_CMD_PARAM_LEN];    \
    const bt_byte       _hci_max_cmd_param_len = HCI_MAX_CMD_PARAM_LEN;    \
    \
    bt_buffer_header_t  _hci_send_data_buffer_headers[HCI_MAX_DATA_BUFFERS];    \
    bt_hci_data_t       _hci_send_data_buffers[HCI_MAX_DATA_BUFFERS];    \
    const bt_byte       _hci_max_data_buffers = HCI_MAX_DATA_BUFFERS; \
    \
    bt_hci_conn_state_t _hci_connections[HCI_MAX_HCI_CONNECTIONS];    \
    const bt_byte       _hci_max_hci_connections = HCI_MAX_HCI_CONNECTIONS; \
    \
    bt_byte             _recv_buffer[HCI_RX_BUFFER_LEN];    \
    bt_byte             _send_buffer[HCI_TX_BUFFER_LEN];    \
    const bt_uint       _hci_rcv_buffer_len = HCI_RX_BUFFER_LEN;    \
    const bt_uint       _hci_tx_buffer_len = HCI_TX_BUFFER_LEN;    \
    \
    bt_byte             _conn_state_recv_buffers[(HCI_L2CAP_BUFFER_LEN) * (HCI_MAX_HCI_CONNECTIONS)];    \
    const bt_uint       _hci_l2cap_buffer_len = HCI_L2CAP_BUFFER_LEN;    \
    \
    const bt_bool       _hci_enable_ctrl_to_host_flow_control = HCI_ENABLE_CTRL_TO_HOST_FLOW_CONTROL;    \
```

```
\
const bt_byte       _hci_max_connect_attempts = HCI_MAX_CONNECT_ATTEMPTS;    \
\
const bt_bool       _hci_enable_sco = BT_ENABLE_SCO;    \
void (*_hci_recv_sco_data_packet_fp)(bt_byte* pbuf);    \
\
void (*_bt_ssp_init)(void);    \
void (*_bt_ssp_evt_handler)(bt_hci_event_t* evt);    \
\
void (*_bt_hci_le_init)(bt_hci_le_ctrl_state_t* le_ctrl_state);    \
HCI_DECLARE_LE_CONN_STATES    \
HCI_DECLARE_LE_CTRL_STATE    \
\
HCI_ALLOCATE_BUFFERS_RAM_SIZE_VAR    \
```

Description

This is macro HCI_ALLOCATE_BUFFERS_VARS.

HCI_DECLARE_LE_CONN_STATES Macro

File

hci_config.h

C

```
#define HCI_DECLARE_LE_CONN_STATES
```

Description

This is macro HCI_DECLARE_LE_CONN_STATES.

HCI_DECLARE_LE_CTRL_STATE Macro

File

hci_config.h

C

```
#define HCI_DECLARE_LE_CTRL_STATE
```

Description

This is macro HCI_DECLARE_LE_CTRL_STATE.

HCI_ENABLE_CTRL_TO_HOST_FLOW_CONTROL Macro

File

hci_config.h

C

```
#define HCI_ENABLE_CTRL_TO_HOST_FLOW_CONTROL BT_FALSE
```

Description

This is macro HCI_ENABLE_CTRL_TO_HOST_FLOW_CONTROL.

HCI_INIT_LE_CONN_STATES Macro

File

hci_config.h

C

```
#define HCI_INIT_LE_CONN_STATES \
    _hci_connections[i].le_conn_state = NULL;
```

Description

This is macro HCI_INIT_LE_CONN_STATES.

HCI_INIT_LE_CTRL_STATE Macro

File

hci_config.h

C

```
#define HCI_INIT_LE_CTRL_STATE \
    _bt_hci_le_init = NULL;
```

Description

This is macro HCI_INIT_LE_CTRL_STATE.

HCI_INIT_SCO_HANDLERS Macro

File

hci_config.h

C

```
#define HCI_INIT_SCO_HANDLERS \
    _hci_recv_sco_data_packet_fp = NULL;
```

Description

This is macro HCI_INIT_SCO_HANDLERS.

HCI_INIT_SSP_HANDLERS Macro

File

hci_config.h

C

```
#define HCI_INIT_SSP_HANDLERS \
    _bt_ssp_evt_handler = NULL;    \
    _bt_ssp_init = NULL;
```

Description

This is macro HCI_INIT_SSP_HANDLERS.

HCI_L2CAP_BUFFER_LEN Macro

File

hci_config.h

C

```
#define HCI_L2CAP_BUFFER_LEN (HCI_RX_BUFFER_LEN - HCI_ACL_DATA_HEADER_LEN - HCI_TRANSPORT_HEADER_LEN)
```

Description

This is macro HCI_L2CAP_BUFFER_LEN.

HCI_SIZEOF_LE_CONN_STATES Macro

File

hci_config.h

C

```
#define HCI_SIZEOF_LE_CONN_STATES 0
```

Description

This is macro HCI_SIZEOF_LE_CONN_STATES.

HCI_SIZEOF_LE_CTRL_STATE Macro

File

hci_config.h

C

```
#define HCI_SIZEOF_LE_CTRL_STATE 0
```

Description

This is macro HCI_SIZEOF_LE_CTRL_STATE.

HCI_TX_BUFFER_LEN Macro

File

hci_config.h

C

```
#define HCI_TX_BUFFER_LEN HCI_RX_BUFFER_LEN
```

Description

This is macro HCI_TX_BUFFER_LEN.

__HCI_CONFIG_EVENT_HANDLERS_H Macro

File

hci_config_event_handlers.h

C

```
#define __HCI_CONFIG_EVENT_HANDLERS_H
```

Description

This is macro __HCI_CONFIG_EVENT_HANDLERS_H.

__HCI_CONFIG_H Macro

File

hci_config.h

C

```
#define __HCI_CONFIG_H
```

Description

This is macro __HCI_CONFIG_H.

HCI_MAX_CONNECT_ATTEMPTS Macro

File

hci_config.h

C

```
#define HCI_MAX_CONNECT_ATTEMPTS 4
```

Description

This is macro HCI_MAX_CONNECT_ATTEMPTS.

L2CAP Configuration Macros

L2CAP_ALLOCATE_BUFFERS_FUNCTION Macro

File

l2cap_config.h

C

```
#define L2CAP_ALLOCATE_BUFFERS_FUNCTION \
    void _l2cap_allocate_buffers()    \
    {    \
        bt_byte i;    \
        _l2cap_hci_connect_packet_type = L2CAP_HCI_PACKET_TYPE;    \
        _l2cap_hci_page_scan_repetition_mode = L2CAP_HCI_PAGE_SCAN_REPETITION_MODE;    \
        _l2cap_hci_role_switch = L2CAP_HCI_ROLE_SWITCH;    \
        _l2cap_idle_hci_connection_timeout = L2CAP_IDLE_CONNECTION_TIMEOUT;    \
        \
        _zero_memory(_l2cap_psms, sizeof(bt_l2cap_psm_t) * (L2CAP_MAX_PSMS) * (L2CAP_MAX_MANAGERS));    \
        _zero_memory(_l2cap_channels, sizeof(bt_l2cap_channel_t) * (L2CAP_MAX_CHANNELS) *
(L2CAP_MAX_MANAGERS));    \
        if (_l2cap_channels_ext)    \
            _zero_memory(_l2cap_channels_ext, sizeof(bt_l2cap_channel_ext_t) * (L2CAP_MAX_CHANNELS) *
(L2CAP_MAX_MANAGERS));    \
        if (_l2cap_max_fixed_channels)    \
            _zero_memory(_l2cap_fixed_channels, sizeof(bt_l2cap_fixed_channel_t) *
_l2cap_max_fixed_channels);    \
        \
        for (i = 0; i < L2CAP_MAX_MANAGERS; i++)    \
        {    \
            _mgrs[i]._psms = &_l2cap_psms[i * L2CAP_MAX_PSMS];    \
            _mgrs[i]._channels = &_l2cap_channels[i * (L2CAP_MAX_CHANNELS)];    \
            bt_init_buffer_mgr(&_mgrs[i].connect_params_mgr, L2CAP_MAX_CHANNELS,
sizeof(bt_l2cap_connect_params_t), &_l2cap_connect_params_headers[(L2CAP_MAX_CHANNELS) * i],
&_l2cap_connect_params[(L2CAP_MAX_CHANNELS) * i]);    \
            \
            if (_l2cap_channels_ext)    \
            {    \
                int j;    \
                \
                for (j = 0; j < L2CAP_MAX_CHANNELS; j++)    \
                {    \
                    _mgrs[i]._channels[j].ext = &_l2cap_channels_ext[i * (L2CAP_MAX_CHANNELS) + j];    \
                }    \
            }    \
            if (_l2cap_max_fixed_channels)    \
                _mgrs[i]._fixed_channels = &_l2cap_fixed_channels[i * _l2cap_max_fixed_channels];    \
        }    \
    }    \
```

Description

This is macro L2CAP_ALLOCATE_BUFFERS_FUNCTION.

L2CAP_ALLOCATE_BUFFERS_RAM_SIZE_VAR Macro

File

l2cap_config.h

C

```
#define L2CAP_ALLOCATE_BUFFERS_RAM_SIZE_VAR
```

Description

This is macro L2CAP_ALLOCATE_BUFFERS_RAM_SIZE_VAR.

L2CAP_ALLOCATE_BUFFERS_VARS Macro

File

l2cap_config.h

C

```
#define L2CAP_ALLOCATE_BUFFERS_VARS \
    bt_buffer_header_t  _l2cap_cmd_buffer_headers[L2CAP_MAX_CMD_BUFFERS];    \
    bt_l2cap_command_t  _l2cap_cmd_buffers[L2CAP_MAX_CMD_BUFFERS];    \
    const bt_byte       _l2cap_max_cmd_buffers = L2CAP_MAX_CMD_BUFFERS; \
    \
    bt_l2cap_psm_t      _l2cap_psms[L2CAP_MAX_PSMS * L2CAP_MAX_MANAGERS];    \
    const bt_byte       _l2cap_max_psms = L2CAP_MAX_PSMS; \
    bt_l2cap_channel_t  _l2cap_channels[(L2CAP_MAX_CHANNELS) * L2CAP_MAX_MANAGERS];    \
    const bt_byte       _l2cap_max_channels = L2CAP_MAX_CHANNELS; \
    \
    bt_uint             _l2cap_hci_connect_packet_type;    \
    bt_byte             _l2cap_hci_page_scan_repetition_mode;    \
    bt_byte             _l2cap_hci_role_switch;    \
    bt_long             _l2cap_idle_hci_connection_timeout;    \
    \
    bt_buffer_header_t  _l2cap_connect_params_headers[(L2CAP_MAX_CHANNELS) * (L2CAP_MAX_MANAGERS)];    \
    bt_l2cap_connect_params_t  _l2cap_connect_params[(L2CAP_MAX_CHANNELS) * (L2CAP_MAX_MANAGERS)];    \
    \
    L2CAP_DECL_ERETR_FUNCTIONS    \
    \
    L2CAP_FIXED_CHANNELS_DECL    \
    \
    L2CAP_ALLOCATE_BUFFERS_RAM_SIZE_VAR    \
```

Description

This is macro L2CAP_ALLOCATE_BUFFERS_VARS.

L2CAP_FIXED_CHANNELS_DECL Macro

File

l2cap_config.h

C

```
#define L2CAP_FIXED_CHANNELS_DECL \
    bt_l2cap_fixed_channel_t _l2cap_fixed_channels_buffer[L2CAP_MAX_FIXED_CHANNELS *
L2CAP_MAX_MANAGERS];    \
    bt_l2cap_fixed_channel_t* _l2cap_fixed_channels = &_l2cap_fixed_channels_buffer[0];    \
    bt_byte _l2cap_max_fixed_channels = L2CAP_MAX_FIXED_CHANNELS;
```

Description

This is macro L2CAP_FIXED_CHANNELS_DECL.

L2CAP_HCI_PACKET_TYPE Macro

File

l2cap_config.h

C

```
#define L2CAP_HCI_PACKET_TYPE \
    HCI_BB_PACKET_TYPE_DM1 | \
    HCI_BB_PACKET_TYPE_DH1 | \
    HCI_BB_PACKET_TYPE_DM3 | \
    HCI_BB_PACKET_TYPE_DH3 | \
    HCI_BB_PACKET_TYPE_DM5 | \
    HCI_BB_PACKET_TYPE_DH5
```

Description

- brief L2CAP_HCI_PACKET_TYPE.
- ingroup btconfig

*

- details Defines a set of packets that link manager is allowed to use when calling bt_l2cap_connect.
- The default value is to enable all packet types.

enable all packet types

L2CAP_HCI_PAGE_SCAN_REPETITION_MODE Macro

File

l2cap_config.h

C

```
#define L2CAP_HCI_PAGE_SCAN_REPETITION_MODE HCI_PAGE_SCAN_REPETITION_MODE_R0
```

Description

brief L2CAP_HCI_PAGE_SCAN_REPETITION_MODE. ingroup btconfig

details Defines a default value of the page scan repetition mode when calling bt_l2cap_connect. Must be set to one of the following values: HCI_PAGE_SCAN_REPETITION_MODE_R0 HCI_PAGE_SCAN_REPETITION_MODE_R1 HCI_PAGE_SCAN_REPETITION_MODE_R2

The default value is HCI_PAGE_SCAN_REPETITION_MODE_R0.

L2CAP_HCI_ROLE_SWITCH Macro

File

l2cap_config.h

C

```
#define L2CAP_HCI_ROLE_SWITCH HCI_ROLE_SWITCH_ALLOW
```

Description

brief L2CAP_HCI_ROLE_SWITCH. ingroup btconfig

details Defines a default value of the role switch parameter when calling bt_l2cap_connect. Must be set to one of the following values: HCI_ROLE_SWITCH_ALLOW HCI_ROLE_SWITCH_DISALLOW The default value is to allow the role switch.

L2CAP_IDLE_CONNECTION_TIMEOUT Macro

File

l2cap_config.h

C

```
#define L2CAP_IDLE_CONNECTION_TIMEOUT 5 /* seconds */
```

Description

seconds

PACK_CONFIG_REQUEST Macro

File

l2cap_config_handlers.h

C

```
#define PACK_CONFIG_REQUEST NULL
```

Description

This is macro PACK_CONFIG_REQUEST.

PACK_CONFIG_RESPONSE Macro

File

l2cap_config_handlers.h

C

```
#define PACK_CONFIG_RESPONSE NULL
```

Description

This is macro PACK_CONFIG_RESPONSE.

PACK_CONN_REQUEST Macro

File

l2cap_config_handlers.h

C

```
#define PACK_CONN_REQUEST NULL
```

Description

This is macro PACK_CONN_REQUEST.

PACK_CONN_RESPONSE Macro

File

l2cap_config_handlers.h

C

```
#define PACK_CONN_RESPONSE NULL
```

Description

This is macro PACK_CONN_RESPONSE.

PACK_DCONN_REQUEST Macro

File

l2cap_config_handlers.h

C

```
#define PACK_DCONN_REQUEST NULL
```

Description

This is macro PACK_DCONN_REQUEST.

PACK_DCONN_RESPONSE Macro

File

l2cap_config_handlers.h

C

```
#define PACK_DCONN_RESPONSE NULL
```

Description

This is macro PACK_DCONN_RESPONSE.

PACK_INFO_REQUEST Macro

File

l2cap_config_handlers.h

C

```
#define PACK_INFO_REQUEST NULL
```

Description

This is macro PACK_INFO_REQUEST.

PACK_INFO_RESPONSE Macro

File

l2cap_config_handlers.h

C

```
#define PACK_INFO_RESPONSE NULL
```

Description

This is macro PACK_INFO_RESPONSE.

PROCESS_CONFIG_REQ Macro

File

l2cap_config_handlers.h

C

```
#define PROCESS_CONFIG_REQ _process_unknown_req
```

Description

This is macro PROCESS_CONFIG_REQ.

PROCESS_CONFIG_RES Macro

File

l2cap_config_handlers.h

C

```
#define PROCESS_CONFIG_RES _process_unknown_res
```

Description

This is macro PROCESS_CONFIG_RES.

PROCESS_CONN_REQ Macro

File

l2cap_config_handlers.h

C

```
#define PROCESS_CONN_REQ _process_unknown_req
```

Description

This is macro PROCESS_CONN_REQ.

PROCESS_CONN_RES Macro

File

l2cap_config_handlers.h

C

```
#define PROCESS_CONN_RES _process_unknown_res
```

Description

This is macro PROCESS_CONN_RES.

PROCESS_DCONN_REQ Macro

File

l2cap_config_handlers.h

C

```
#define PROCESS_DCONN_REQ _process_unknown_req
```

Description

This is macro PROCESS_DCONN_REQ.

PROCESS_DCONN_RES Macro

File

l2cap_config_handlers.h

C

```
#define PROCESS_DCONN_RES _process_unknown_res
```

Description

This is macro PROCESS_DCONN_RES.

PROCESS_INFO_REQ Macro

File

l2cap_config_handlers.h

C

```
#define PROCESS_INFO_REQ _process_unknown_req
```

Description

This is macro PROCESS_INFO_REQ.

PROCESS_INFO_RES Macro

File

l2cap_config_handlers.h

C

```
#define PROCESS_INFO_RES _process_unknown_res
```

Description

This is macro PROCESS_INFO_RES.

READ_CONFIG_REQUEST Macro

File

l2cap_config_handlers.h

C

```
#define READ_CONFIG_REQUEST NULL
```

Description

This is macro READ_CONFIG_REQUEST.

READ_CONFIG_RESPONSE Macro

File

l2cap_config_handlers.h

C

```
#define READ_CONFIG_RESPONSE NULL
```

Description

This is macro READ_CONFIG_RESPONSE.

READ_CONN_REQUEST Macro

File

l2cap_config_handlers.h

C

```
#define READ_CONN_REQUEST NULL
```

Description

This is macro READ_CONN_REQUEST.

READ_CONN_RESPONSE Macro

File

l2cap_config_handlers.h

C

```
#define READ_CONN_RESPONSE NULL
```

Description

This is macro READ_CONN_RESPONSE.

READ_DCONN_REQUEST Macro

File

l2cap_config_handlers.h

C

```
#define READ_DCONN_REQUEST NULL
```

Description

This is macro READ_DCONN_REQUEST.

READ_DCONN_RESPONSE Macro

File

l2cap_config_handlers.h

C

```
#define READ_DCONN_RESPONSE NULL
```

Description

This is macro READ_DCONN_RESPONSE.

READ_INFO_REQUEST Macro

File

l2cap_config_handlers.h

C

```
#define READ_INFO_REQUEST NULL
```

Description

This is macro READ_INFO_REQUEST.

READ_INFO_RESPONSE Macro

File

l2cap_config_handlers.h

C

```
#define READ_INFO_RESPONSE NULL
```

Description

This is macro READ_INFO_RESPONSE.

__L2CAP_CONFIG_H Macro

File

l2cap_config.h

C

```
#define __L2CAP_CONFIG_H
```

Description

This is macro __L2CAP_CONFIG_H.

__L2CAP_CONFIG_HANDLERS_H Macro

File

l2cap_config_handlers.h

C

```
#define __L2CAP_CONFIG_HANDLERS_H
```

Description

This is macro __L2CAP_CONFIG_HANDLERS_H.

L2CAP_DECL_ERETR_FUNCTIONS Macro

File

l2cap_config.h

C

```
#define L2CAP_DECL_ERETR_FUNCTIONS \
        bt_l2cap_channel_ext_t*  _l2cap_channels_ext = NULL;    \
        \
        void (*_l2cap_eretr_recv_fp)(bt_l2cap_mgr_p pmgr, bt_l2cap_channel_t* pch, bt_byte* pdata, bt_int
len) = NULL;    \
        bt_bool (*_l2cap_eretr_send_data_fp)(bt_l2cap_channel_t* pch, bt_byte* data, bt_int len,
bt_l2cap_send_data_callback_fp cb, void* cb_param) = NULL;    \
        bt_bool (*_l2cap_eretr_send_smart_data_fp)(bt_l2cap_channel_t* pch, bt_packet_t* packet, bt_int
len, bt_l2cap_send_data_callback_fp cb, void* cb_param) = NULL;    \
```

```
        bt_bool (*_l2cap_eretr_handle_xmit_event_fp)(bt_l2cap_xmit_event_param_t* param) = NULL;    \
        void (*_l2cap_eretr_pack_config_request_fp)(bt_l2cap_channel_t* channel, bt_byte* buffer, bt_int
buffer_len, bt_int* offset) = NULL;
```

Description

This is macro L2CAP_DECL_ERETR_FUNCTIONS.

L2CAP_MAX_FIXED_CHANNELS Macro

File

l2cap_config.h

C

```
#define L2CAP_MAX_FIXED_CHANNELS 0
```

Description

This is macro L2CAP_MAX_FIXED_CHANNELS.

RFCOMM Configuration Macros

RFCOMM_ALLOCATE_BUFFERS_FUNCTION Macro

File

rfcomm_config.h

C

```
#define RFCOMM_ALLOCATE_BUFFERS_FUNCTION \
    void _rfcomm_allocate_buffers(void)    \
    {    \
    }    \
```

Description

This is macro RFCOMM_ALLOCATE_BUFFERS_FUNCTION.

RFCOMM_ALLOCATE_BUFFERS_RAM_SIZE_VAR Macro

File

rfcomm_config.h

C

```
#define RFCOMM_ALLOCATE_BUFFERS_RAM_SIZE_VAR
```

Description

This is macro RFCOMM_ALLOCATE_BUFFERS_RAM_SIZE_VAR.

RFCOMM_ALLOCATE_BUFFERS_VARS Macro

File

rfcomm_config.h

C

```
#define RFCOMM_ALLOCATE_BUFFERS_VARS \
    bt_rfcomm_session_t _rfcomm_sessions[RFCOMM_MAX_SESSIONS];    \
    const bt_byte       _rfcomm_max_sessions = RFCOMM_MAX_SESSIONS;    \
    bt_rfcomm_dlc_t     _rfcomm_dlcs[(RFCOMM_MAX_DLCS) * (RFCOMM_MAX_SESSIONS)];    \
    const bt_byte       _rfcomm_max_dlcs = RFCOMM_MAX_DLCS;    \
    bt_rfcomm_server_channel_t _rfcomm_channels[(RFCOMM_MAX_SERVER_CHANNELS) * (RFCOMM_MAX_SESSIONS)];    \
    const bt_byte       _rfcomm_max_channels = RFCOMM_MAX_SERVER_CHANNELS;    \
    const bt_uint       _rfcomm_pdu_size = (RFCOMM_INFO_LEN) + (RFCOMM_FRAME_HEADER_LEN);    \
    const bt_uint       _rfcomm_info_len = RFCOMM_INFO_LEN;    \
    bt_buffer_header_t  _rfcomm_cmd_buffer_headers[RFCOMM_MAX_CMD_BUFFERS];    \
```

```
bt_rfcomm_command_t _rfcomm_cmd_buffers[RFCOMM_MAX_CMD_BUFFERS];    \
const bt_byte       _rfcomm_max_cmd_buffers = RFCOMM_MAX_CMD_BUFFERS;    \
const bt_byte       _rfcomm_local_credit = RFCOMM_LOCAL_CREDIT;    \
const bt_bool       _rfcomm_enable_multidevice_channels = RFCOMM_ENABLE_MULTIDEVICE_CHANNELS;    \
RFCOMM_LOCAL_CREDIT_SEND_THRESHOLD_DECL    \
RFCOMM_ALLOCATE_BUFFERS_RAM_SIZE_VAR    \
```

Description

This is macro RFCOMM_ALLOCATE_BUFFERS_VARS.

RFCOMM_BUFFER_SIZE Macro

File

rfcomm_config.h

C

```
#define RFCOMM_BUFFER_SIZE (RFCOMM_INFO_LEN)
```

Description

This is macro RFCOMM_BUFFER_SIZE.

RFCOMM_ENABLE_MULTIDEVICE_CHANNELS Macro

File

rfcomm_config.h

C

```
#define RFCOMM_ENABLE_MULTIDEVICE_CHANNELS BT_TRUE
```

Description

brief Enable multi-device server channels. ingroup rfcomm_config

details Normally each server channel can be used only once. I.e. if device A connected to channel 1, device B cannot connect to channel 1 until device A disconnects. With this option it is possible to make channels accept connections from several devices at the same time. I.e., if RFCOMM_ENABLE_MULTIDEVICE_CHANNELS is TRUE both device A and device B can connect to channel 1 at the same time.

RFCOMM_INFO_LEN Macro

File

rfcomm_config.h

C

```
#define RFCOMM_INFO_LEN
```

Description

brief Maximum size of the data portion of a UIH frame. ingroup rfcomm_config

details This parameter defines the maximum size of the data portion of a UIH frame. If CFC is used the actual length of the data portion will be 1 byte less. This value must be less than or equal to HCI_L2CAP_BUFFER_LEN - RFCOMM_FRAME_HEADER_LEN - L2CAP_HEADER_LEN.

RFCOMM_LOCAL_CREDIT Macro

File

rfcomm_config.h

C

```
#define RFCOMM_LOCAL_CREDIT
```

Description

brief The number of receive buffers. ingroup rfcomm_config

details This parameter defines the number of received UIH frames that can be stored on the local device. The flow control mechanism used in RFCOMM ensures that the remote side of the link always knows how many free buffers left on the local device. When the number of free buffers

reaches 0, the transmitter stops sending data frames until the receiver frees some buffers. The RFCOMM layer does not actually allocate space for buffers. It uses RFCOMM_LOCAL_CREDIT to keep track of free buffers and report them to the remote side. Actual memory allocation is done in SPP layer.

RFCOMM_MAX_CMD_BUFFERS Macro

File

rfcomm_config.h

C

```
#define RFCOMM_MAX_CMD_BUFFERS
```

Description

brief Maximum number of command buffers. ingroup rfcomm_config

details This parameter defines the maximum number of commands that can be sent at the same time. It is usually enough to reserve 2 buffers for each DLC excluding control DLC. Therefore, this value can be defined as rn #define RFCOMM_MAX_CMD_BUFFERS (RFCOMM_MAX_DLCS - 1) * 2

RFCOMM_MAX_DLCS Macro

File

rfcomm_config.h

C

```
#define RFCOMM_MAX_DLCS
```

Description

brief Maximum number of DLCs ingroup rfcomm_config

details This parameter defines the maximum number of DLCs on each session. This value should be at least 2 because each session uses one DLC to convey multiplexer control messages. All other DLCs are used to emulate serial ports.

RFCOMM_MAX_SERVER_CHANNELS Macro

File

rfcomm_config.h

C

```
#define RFCOMM_MAX_SERVER_CHANNELS
```

Description

brief Maximum number of Server channels ingroup rfcomm_config

details This parameter defines the maximum number of server channels exposed by the local device. This value should not exceed RFCOMM_MAX_DLCS - 1.

RFCOMM_MAX_SESSIONS Macro

File

rfcomm_config.h

C

```
#define RFCOMM_MAX_SESSIONS
```

Description

brief Maximum number of remote devices a local device can be connected to ingroup rfcomm_config

details This parameter defines the maximum number of remote devices a local device can have simultaneous connections to. This value should not exceed HCI_MAX_HCI_CONNECTIONS.

__RFCOMM_CONFIG_H Macro

File

rfcomm_config.h

C

```
#define __RFCOMM_CONFIG_H
```

Description

This is macro __RFCOMM_CONFIG_H.

RFCOMM_LOCAL_CREDIT_SEND_THRESHOLD_DECL Macro

File

rfcomm_config.h

C

```
#define RFCOMM_LOCAL_CREDIT_SEND_THRESHOLD_DECL const bt_byte _rfcomm_local_credit_send_threshold = 1;
```

Description

This is macro RFCOMM_LOCAL_CREDIT_SEND_THRESHOLD_DECL.

SDP Configuration Macros

SDP_ALLOCATE_BUFFERS_RAM_SIZE_VAR Macro

File

sdp_config.h

C

```
#define SDP_ALLOCATE_BUFFERS_RAM_SIZE_VAR
```

Description

- defgroup sdp_config Configuration
- ingroup sdp

*

- This module describes parameters used to configure SDP.

*

- dotstack is customized using a configuration file. The configuration file tailors the dotstack to the application being built. It has to have the structure shown below.

* code #include "cdbt/bt/bt_std.h"

// HCI and L2CAP must always be present // SDP is required only if stack is running in dual mode. This is the default mode. // To run the stack in single mode (i.e. only BLE is supported) a BT_BLE_SINGLE_MODE symbol // must be defined: // #define BT_BLE_SINGLE_MODE

// HCI configuration parameters #define HCI_MAX_CMD_BUFFERS ... #define HCI_MAX_DATA_BUFFERS ... #define HCI_MAX_HCI_CONNECTIONS ... #define HCI_RX_BUFFER_LEN ... #define HCI_TX_BUFFER_LEN ... #define HCI_L2CAP_BUFFER_LEN ... #define HCI_MAX_CMD_PARAM_LEN ...

// L2CAP configuration parameters #define L2CAP_MAX_CMD_BUFFERS ... #define L2CAP_MAX_FRAME_BUFFERS ... #define L2CAP_MAX_PSMS ... #define L2CAP_MAX_CHANNELS ...

// SDP configuration parameters #define SDP_MAX_SEARCH_RESULT_LEN ... #define SDP_MAX_ATTRIBUTE_RESULT_LEN ...

#include "cdbt/bt/bt_oem_config.h"

endcode

SDP_ALLOCATE_BUFFERS_VARS Macro

File

sdp_config.h

C

```
#define SDP_ALLOCATE_BUFFERS_VARS \
        bt_buffer_header_t           _sdp_packet_buffer_headers[SDP_MAX_PDU_BUFFERS];    \
        bt_sdp_packet_t              _sdp_packet_buffers[SDP_MAX_PDU_BUFFERS];    \
        const bt_byte                _sdp_max_buffers = SDP_MAX_PDU_BUFFERS; \
        bt_buffer_header_t           _sdp_client_packet_buffer_headers[SDP_MAX_PDU_BUFFERS];    \
        bt_sdp_packet_t              _sdp_client_packet_buffers[SDP_MAX_PDU_BUFFERS];    \
        const bt_byte                _sdp_client_max_buffers = SDP_MAX_PDU_BUFFERS; \
        const bt_uint                _sdp_max_search_result_len = SDP_MAX_SEARCH_RESULT_LEN;    \
        const bt_uint                _sdp_max_attribute_result_len = SDP_MAX_ATTRIBUTE_RESULT_LEN;    \
        bt_sr_handle_t               _sdp_found_sr_lists_buffers[SDP_MAX_TRANSACTIONS *
SDP_MAX_SEARCH_RESULT_LEN];    \
        bt_sdp_found_attr_list_t     _sdp_found_attr_lists_buffers[SDP_MAX_TRANSACTIONS *
SDP_MAX_SEARCH_RESULT_LEN];    \
        bt_byte*                     _sdp_found_attr_list_buffers[SDP_MAX_TRANSACTIONS *
SDP_MAX_SEARCH_RESULT_LEN * SDP_MAX_ATTRIBUTE_RESULT_LEN];    \
        \
        bt_buffer_mgr_t              _sdp_tran_buffer_mgr2;    \
        bt_buffer_header_t           _sdp_tran_buffer_headers2[SDP_MAX_TRANSACTIONS];    \
        bt_sdp_transaction_t         _sdp_tran_buffers2[SDP_MAX_TRANSACTIONS];    \
        \
        bt_buffer_mgr_t              _sdp_service_tran_buffer_mgr;    \
        bt_buffer_header_t           _sdp_service_tran_buffer_headers[SDP_MAX_TRANSACTIONS];    \
        bt_sdp_service_transaction_t _sdp_service_tran_buffers[SDP_MAX_TRANSACTIONS];    \
        \
        bt_bool                  (*_sdp_start_fp)(bt_l2cap_mgr_p l2cap_mgr, const bt_byte* sdp_db, bt_uint
sdp_db_len) = &bt_sdp_start;    \
        SDP_ALLOCATE_BUFFERS_RAM_SIZE_VAR    \
```

Description

This is macro SDP_ALLOCATE_BUFFERS_VARS.

SDP_MAX_ATTRIBUTE_RESULT_LEN Macro

File

sdp_config.h

C

```
#define SDP_MAX_ATTRIBUTE_RESULT_LEN
```

Description

brief Maximum number of attributes to find ingroup sdp_config

details This parameter defines the maximum number of attributes withing a service record the SDP server will return to the client.

SDP_MAX_PDU_BUFFERS Macro

File

sdp_config.h

C

```
#define SDP_MAX_PDU_BUFFERS
```

Description

brief Maximum number of SDP server PDU buffers. ingroup sdp_config

details This parameter defines the maximum number of responses the SDP server can send at the same time.

SDP_MAX_SEARCH_RESULT_LEN Macro

File

sdp_config.h

C

```
#define SDP_MAX_SEARCH_RESULT_LEN
```

Description

brief Maximum number of service records to find. ingroup sdp_config

details This parameter defines the maximum number of service records the SDP server will return to the client.

__SDP_CONFIG_H Macro

File

sdp_config.h

C

```
#define __SDP_CONFIG_H
```

Description

This is macro __SDP_CONFIG_H.

SPP Configuration Macros

SPP_ALLOCATE_BUFFERS_RAM_SIZE_VAR Macro

File

spp_config.h

C

```
#define SPP_ALLOCATE_BUFFERS_RAM_SIZE_VAR
```

Description

This is macro SPP_ALLOCATE_BUFFERS_RAM_SIZE_VAR.

SPP_ALLOCATE_BUFFERS_VARS Macro

File

spp_config.h

C

```
#define SPP_ALLOCATE_BUFFERS_VARS \
    bt_spp_port_t     _spp_ports[SPP_MAX_PORTS];    \
    const bt_byte     _spp_max_ports = SPP_MAX_PORTS;    \
    const bt_byte     _spp_disable_buffering = SPP_DISABLE_BUFFERING;    \
    SPP_DECLARE_FRAME_BUFFERS    \
    SPP_ALLOCATE_BUFFERS_RAM_SIZE_VAR    \
```

Description

This is macro SPP_ALLOCATE_BUFFERS_VARS.

SPP_DECLARE_FRAME_BUFFERS Macro

File

spp_config.h

C

```
#define SPP_DECLARE_FRAME_BUFFERS \
    bt_byte*          _spp_frame_buffers = NULL;    \
    bt_int*           _spp_frame_len = NULL;
```

Description

This is macro SPP_DECLARE_FRAME_BUFFERS.

SPP_DISABLE_BUFFERING Macro

File

spp_config.h

C

```
#define SPP_DISABLE_BUFFERING 1
```

Description

This is macro SPP_DISABLE_BUFFERING.

SPP_FRAME_BUFFERS_RAM_SIZE Macro

File

spp_config.h

C

```
#define SPP_FRAME_BUFFERS_RAM_SIZE 0
```

Description

This is macro SPP_FRAME_BUFFERS_RAM_SIZE.

SPP_FRAME_BUFFERS_SIZE Macro

File

spp_config.h

C

```
#define SPP_FRAME_BUFFERS_SIZE
```

Description

This is macro SPP_FRAME_BUFFERS_SIZE.

SPP_MAX_PORTS Macro

File

spp_config.h

C

```
#define SPP_MAX_PORTS
```

Description

brief Maximum number of SPP ports. ingroup spp_config

details This parameter defines the maximum number of SPP port that can be open between the local and remote devices. If RFCOMM_ENABLE_MULTIDEVICE_CHANNELS is FALSE (default) this value should be equal to RFCOMM_MAX_SERVER_CHANNELS. If RFCOMM_ENABLE_MULTIDEVICE_CHANNELS is TRUE this value should be between RFCOMM_MAX_SERVER_CHANNELS and RFCOMM_MAX_SERVER_CHANNELS * RFCOMM_MAX_SESSIONS.

__SPP_CONFIG_H Macro

File

spp_config.h

C

```
#define __SPP_CONFIG_H
```

Description

This is macro __SPP_CONFIG_H.

Building the Library

The library is provided in binary form only, and comes prebuilt when you receive the applications that carry the Bluetooth license.

Library Interface

A2DP Data Types and Constants

	Name	Description
	_bt_a2dp_aac_config_t	This is type bt_a2dp_aac_config_t.
	_bt_a2dp_event_u	brief Parameter to an application callback. ingroup a2dp details This union is used to pass event specific data to the A2DP consumer. Which member of the union points to a valid structure depends on the event reported to the consumer. In general, each event has a corresponding member in the union.
	_bt_a2dp_evt_open_and_start_stream_completed_s	brief Parameter to A2DP_EVT_OPEN_AND_START_STREAM_COMPLETED event ingroup a2dp details A pointer to this structure is passed to the A2DP application callback as a valid member of the bt_a2dp_event_t union - bt_a2dp_event_t::open_and_start_stream_completed - when A2DP completed a "open & start stream" request.
	_bt_a2dp_mgr_t	brief A2DP manager. ingroup a2dp details A structure that glues all pieces together. There is only one instance of this structure allocated by dotstack. A pointer to the instance can be get with ::bt_a2dp_get_mgr.
	_bt_a2dp_mpeg_config_t	This is type bt_a2dp_mpeg_config_t.
	_bt_a2dp_sbc_config_t	This is type bt_a2dp_sbc_config_t.
	_bt_a2dp_sbc_packet_info_t	This is type bt_a2dp_sbc_packet_info_t.
	bt_a2dp_aac_config_t	This is type bt_a2dp_aac_config_t.
	bt_a2dp_event_t	This is type bt_a2dp_event_t.
	bt_a2dp_evt_open_and_start_stream_completed_t	brief Parameter to A2DP_EVT_OPEN_AND_START_STREAM_COMPLETED event ingroup a2dp details A pointer to this structure is passed to the A2DP application callback as a valid member of the bt_a2dp_event_t union - bt_a2dp_event_t::open_and_start_stream_completed - when A2DP completed a "open & start stream" request.
	bt_a2dp_find_server_callback_fp	brief Notify the application of the result of searching for a remote A2DP entity (sourse or sink) ingroup a2dp details This function is called by the A2DP layer when searching for an A2DP entity on a remote device has completed. param supported_features Features supported by a remote A2DP entity. param found c TRUE if an A2DP entity has been found on the remote device. c FALSE otherwise. param param pointer to arbitrary data passed to the bt_a2dp_find_source() or bt_a2dp_find_sink function through its c callback_param parameter.
	bt_a2dp_mgr_callback_fp	brief A2DP application callback. ingroup a2dp details In order to be notified of various events a consumer of the A2DP layer has to register a callback function. The stack will call that function whenever a new event has been generated. param mgr A2DP manager. param evt A2DP event. The event can be one of the following values: @arg A2DP_EVT_CTRL_CHANNEL_CONNECTED: Control channel connected. @arg A2DP_EVT_CTRL_CHANNEL_DISCONNECTED: Control channel disconnected. @arg A2DP_EVT_CTRL_CONNECTION_FAILED: Control channel connection failed (generated only if control connection has been initiated by the local device). @arg A2DP_EVT_DISCOVER_COMPLETED: Local device completed discovering remote SEPs. @arg A2DP_EVT_GET_SEP_CAPABILITIES_COMPLETED: Local... more
	bt_a2dp_mgr_t	This is type bt_a2dp_mgr_t.
	bt_a2dp_mpeg_config_t	This is type bt_a2dp_mpeg_config_t.
	bt_a2dp_sbc_config_t	This is type bt_a2dp_sbc_config_t.
	bt_a2dp_sbc_packet_info_t	This is type bt_a2dp_sbc_packet_info_t.

	__A2DP_CODEC_AAC_H	This is macro __A2DP_CODEC_AAC_H.
	__A2DP_CODEC_MPEG_H	This is macro __A2DP_CODEC_MPEG_H.
	__A2DP_CODEC_SBC_H	This is macro __A2DP_CODEC_SBC_H.
	__A2DP_H	This is macro __A2DP_H.
	__A2DP_PRIVATE_H	This is macro __A2DP_PRIVATE_H.
	A2DP_EVT_ABORT_STREAM_COMPLETED	< This event is generated when a local device received a response (either positive or negative) to a "abort stream" request.
	A2DP_EVT_ABORT_STREAM_REQUESTED	< This event is generated when a local device received "abort stream" request.
	A2DP_EVT_CLOSE_STREAM_COMPLETED	< This event is generated when a local device received a response (either positive or negative) to a "close stream" request.
	A2DP_EVT_CLOSE_STREAM_REQUESTED	< This event is generated when a local device received "close stream" request.
	A2DP_EVT_CTRL_CHANNEL_CONNECTED	< This event is generated when a control channel between two AVDTP entities has been established.
	A2DP_EVT_CTRL_CHANNEL_DISCONNECTED	< This event is generated when a control channel between two AVDTP entities has been terminated.
	A2DP_EVT_CTRL_CONNECTION_FAILED	< This event is generated when a local device failed to create a control channel between two AVDTP entities.
	A2DP_EVT_DISCOVER_SEP_COMPLETED	< This event is generated when a local device received a response (either positive or negative) to a "discover" request.
	A2DP_EVT_GET_SEP_CAPABILITIES_COMPLETED	< This event is generated when a local device received a response (either positive or negative) to a "get SEP capabilities" request.
	A2DP_EVT_GET_STREAM_CONFIGURATION_COMPLETED	< This event is generated when a local device received a response (either positive or negative) to a "get stream configuration" request.
	A2DP_EVT_MEDIA_PACKET_RECEIVED	< This event is generated when a local device received a media packet.
	A2DP_EVT_MEDIA_PACKET_SEND_FAILED	< This event is generated when a local device failed to send a media packet.
	A2DP_EVT_MEDIA_PACKET_SENT	< This event is generated when a local device sent a media packet.
	A2DP_EVT_NOTHING	This is macro A2DP_EVT_NOTHING.
	A2DP_EVT_OPEN_AND_START_STREAM_COMPLETED	< This event is generated when a local device completed "open and start" request.
	A2DP_EVT_OPEN_STREAM_COMPLETED	< This event is generated when a local device received a response (either positive or negative) to a "open stream" request.
	A2DP_EVT_OPEN_STREAM_REQUESTED	< This event is generated when a local device received "open stream" request.
	A2DP_EVT_RECONFIGURE_STREAM_COMPLETED	< This event is generated when a local device received a response (either positive or negative) to a "change stream configuration" request.
	A2DP_EVT_RECONFIGURE_STREAM_REQUESTED	< This event is generated when a local device received "change stream configuration" request.
	A2DP_EVT_SEP_CAPABILITIES_RECEIVED	< This event is generated when a local device received a positive response to a "get SEP capabilities" request.
	A2DP_EVT_SEP_INFO_RECEIVED	< This event is generated for each SEP contained in a positive response to a "discover" request.
	A2DP_EVT_SET_STREAM_CONFIGURATION_COMPLETED	< This event is generated when a local device received a response (either positive or negative) to a "set stream configuration" request.
	A2DP_EVT_SET_STREAM_CONFIGURATION_REQUESTED	< This event is generated when a local device received "set stream configuration" request.
	A2DP_EVT_START_STREAM_COMPLETED	< This event is generated when a local device received a response (either positive or negative) to a "start stream" request.
	A2DP_EVT_START_STREAM_REQUESTED	< This event is generated when a local device received "start stream" request.
	A2DP_EVT_STREAM_ABORTED	< This event is generated when a local device has successfully aborted a stream. < This event follows the A2DP_EVT_ABORT_STREAM_REQUESTED if the upper layer has accepted it. < This event is not generated if stream abortion was initiated by the local device.

	A2DP_EVT_STREAM_CLOSED	< This event is generated when a local device has successfully closed a stream. < This event follows the A2DP_EVT_CLOSE_STREAM_REQUESTED if the upper layer has accepted it. < This event is not generated if stream closing was initiated by the local device.
	A2DP_EVT_STREAM_CONFIGURATION_RECEIVED	< This event is generated when a local device received a positive response to a "get stream configuration" request.
	A2DP_EVT_STREAM_CONFIGURED	< This event is generated when a local device has successfully configured a stream. < This event follows the A2DP_EVT_SET_STREAM_CONFIGURATION_REQUESTED if the upper layer has accepted it. < This event is not generated if stream configuration was initiated by the local device.
	A2DP_EVT_STREAM_OPENED	< This event is generated when a local device has successfully opened a stream. < This event follows the A2DP_EVT_OPEN_STREAM_REQUESTED if the upper layer has accepted it. < This event is not generated if stream opening was initiated by the local device.
	A2DP_EVT_STREAM_RECONFIGURED	< This event is generated when a local device has successfully reconfigured a stream. < This event follows the A2DP_EVT_RECONFIGURE_STREAM_REQUESTED if the upper layer has accepted it. < This event is not generated if stream reconfiguration was initiated by the local device.
	A2DP_EVT_STREAM_SECURITY_CONTROL_COMPLETED	< This event is generated when a local device received a response (either positive or negative) to a "exchange content protection control data" request.
	A2DP_EVT_STREAM_STARTED	< This event is generated when a local device has successfully started a stream. < This event follows the A2DP_EVT_START_STREAM_REQUESTED if the upper layer has accepted it. < This event is not generated if stream starting was initiated by the local device.
	A2DP_EVT_STREAM_SUSPENDED	< This event is generated when a local device has successfully suspended a stream. < This event follows the A2DP_EVT_SUSPEND_STREAM_REQUESTED if the upper layer has accepted it. < This event is not generated if stream suspension was initiated by the local device.
	A2DP_EVT_SUSPEND_STREAM_COMPLETED	< This event is generated when a local device received a response (either positive or negative) to a "suspend stream" request.
	A2DP_EVT_SUSPEND_STREAM_REQUESTED	< This event is generated when a local device received "suspend stream" request.
	A2DP_MANAGER_STATE_CONNECTING	This is macro A2DP_MANAGER_STATE_CONNECTING.
	A2DP_MANAGER_STATE_IDLE	This is macro A2DP_MANAGER_STATE_IDLE.
	A2DP_SINK_FEATURE_AMPLIFIER	< Amplifier
	A2DP_SINK_FEATURE_HEADPHONE	< Headphone
	A2DP_SINK_FEATURE_RECORDER	< Recorder
	A2DP_SINK_FEATURE_SPEAKER	< Speaker
	A2DP_SOURCE_FEATURE_MICROPHONE	< Mic
	A2DP_SOURCE_FEATURE_MIXER	< Mixer
	A2DP_SOURCE_FEATURE_PLAYER	< Player
	A2DP_SOURCE_FEATURE_TUNER	< Tuner
	AAC_CHANNELS_1	This is macro AAC_CHANNELS_1.
	AAC_CHANNELS_2	This is macro AAC_CHANNELS_2.
	AAC_CHANNELS_ALL	This is macro AAC_CHANNELS_ALL.
	AAC_OBJECT_TYPE_MPEG_2_LC	This is macro AAC_OBJECT_TYPE_MPEG_2_LC.
	AAC_OBJECT_TYPE_MPEG_4_LC	This is macro AAC_OBJECT_TYPE_MPEG_4_LC.
	AAC_OBJECT_TYPE_MPEG_4_LTP	This is macro AAC_OBJECT_TYPE_MPEG_4_LTP.
	AAC_OBJECT_TYPE_MPEG_4_SCALABLE	This is macro AAC_OBJECT_TYPE_MPEG_4_SCALABLE.
	AAC_SAMPLING_FREQUENCY_11025	This is macro AAC_SAMPLING_FREQUENCY_11025.
	AAC_SAMPLING_FREQUENCY_12000	This is macro AAC_SAMPLING_FREQUENCY_12000.
	AAC_SAMPLING_FREQUENCY_16000	This is macro AAC_SAMPLING_FREQUENCY_16000.
	AAC_SAMPLING_FREQUENCY_22050	This is macro AAC_SAMPLING_FREQUENCY_22050.
	AAC_SAMPLING_FREQUENCY_24000	This is macro AAC_SAMPLING_FREQUENCY_24000.
	AAC_SAMPLING_FREQUENCY_32000	This is macro AAC_SAMPLING_FREQUENCY_32000.

	AAC_SAMPLING_FREQUENCY_44100	This is macro AAC_SAMPLING_FREQUENCY_44100.
	AAC_SAMPLING_FREQUENCY_48000	This is macro AAC_SAMPLING_FREQUENCY_48000.
	AAC_SAMPLING_FREQUENCY_64000	This is macro AAC_SAMPLING_FREQUENCY_64000.
	AAC_SAMPLING_FREQUENCY_8000	This is macro AAC_SAMPLING_FREQUENCY_8000.
	AAC_SAMPLING_FREQUENCY_88200	This is macro AAC_SAMPLING_FREQUENCY_88200.
	AAC_SAMPLING_FREQUENCY_96000	This is macro AAC_SAMPLING_FREQUENCY_96000.
	AAC_SAMPLING_FREQUENCY_ALL	This is macro AAC_SAMPLING_FREQUENCY_ALL.
	AAC_VBR_NOT_SUPPORTED	This is macro AAC_VBR_NOT_SUPPORTED.
	AAC_VBR_SUPPORTED	This is macro AAC_VBR_SUPPORTED.
	bt_a2dp_abort_stream	brief Suspend a stream. ingroup a2dp details This function tries to suspend a stream by sending a request to the remote party. The stream can be in any state state except AVDTP_STREAM_STATE_IDLE. As a result of this operation the A2DP_EVT_ABORT_STREAM_COMPLETED event will be generated. This operation cannot be rejected. The p evt_param.abort_stream_requested.err_code is always == AVDTP_ERROR_SUCCESS. param mgr A2DP manager. param strm_handle Stream handle. return li c TRUE if the function succeeds, i.e. the actual request has been sent to the remote party. li c FALSE otherwise. No events will be generated.
	bt_a2dp_add_media_rx_buffer	brief Add a media packet buffer to a receive queue ingroup a2dp details The consumer of A2DP is responsible for allocating and supplying A2DP with buffers used to store received packets. A2DP itself only has a queue for storing pointers to buffers supplied by the consumer. When a packet comes in A2DP finds the first buffer large enough to hold the received packet, copies the packet to the buffer and generates a A2DP_EVT_MEDIA_PACKET_RECEIVED event. The consumer then has to process the data in the buffer and return it back to the queue. If there is no buffers in the queue... more
	bt_a2dp_add_media_tx_buffer	brief Add a media packet buffer to a send queue ingroup a2dp details When the consumer of A2DP wants to send a packet to a remote device it calls this function. The function adds the packet to a queue and tells A2DP that it has something to send. The packet will be send as soon as the stream goes to A2DP_STREAM_STATE_STREAMING state. When the packet has been successfully sent a A2DP_EVT_MEDIA_PACKET_SENT is generated. Otherwise a A2DP_EVT_MEDIA_PACKET_SEND_FAILED is generated. Regardless of the event generated the consumer can re-use the buffer as A2DP has removed it from the queue and gave up... more
	bt_a2dp_call_codec	This is macro bt_a2dp_call_codec.
	bt_a2dp_cancel_listen	brief Cancel listening for incoming connections. ingroup a2dp details This function removes a SEP from a list of SEPS which a stream can use for incoming requests. param mgr A2DP manager. param strm_handle Stream handle. param sep_id Local SEP ID. return li c TRUE if the function succeeds. li c FALSE otherwise.
	bt_a2dp_close_stream	brief Close a stream. ingroup a2dp details This function tries to close a stream by sending a request to the remote party. The stream has to be in AVDTP_STREAM_STATE_OPEN or AVDTP_STREAM_STATE_STREAMING state. As a result of this operation the A2DP_EVT_CLOSE_STREAM_COMPLETED event will be generated. If the stream has been closed the p evt_param.bt_avdtp_evt_close_stream_completed_t.err_code == AVDTP_ERROR_SUCCESS. Otherwise, if the remote device for any reason cannot or does not wish to close the stream, the p evt_param.bt_avdtp_evt_close_stream_completed_t.err_code == the error code sent by the remote. param mgr A2DP manager. param strm_handle Stream handle. return li c TRUE if the function succeeds,... more

	bt_a2dp_connect	brief Connect to a remote device. ingroup a2dp details This function opens a control channel connection to a remote device specified by the p remote_addr. If connection cannot be initiated for some reason, for example, there is not enough resources, it returns FALSE and not events are generated. Otherwise the result of an attempt to connect to the remote device is reported via the AVDTP callback. The events generated will either be A2DP_EVT_CTRL_CHANNEL_CONNECTED or A2DP_EVT_CTRL_CHANNEL_CONNECTION_FAILED. param mgr A2DP manager. param remote_addr The address of a remote device. return li c TRUE if connection establishment has been started. li c FALSE... more
	bt_a2dp_connect_ex	brief Connect to a remote device. ingroup a2dp details This function opens a control channel connection to a remote device specified by the p remote_addr. If connection cannot be initiated for some reason, for example, there is not enough resources, it returns FALSE and not events are generated. Otherwise the result of an attempt to connect to the remote device is reported via the AVDTP callback. The events generated will either be A2DP_EVT_CTRL_CHANNEL_CONNECTED or A2DP_EVT_CTRL_CHANNEL_CONNECTION_FAILED. param mgr A2DP manager. param remote_addr The address of a remote device. param acl_config ACL link configuration. This can be a combination of the following... more
	bt_a2dp_create_stream	brief Create a stream. ingroup a2dp details This function allocates memory for storing stream's data and assigns a stream handle. The stream handle is used to manipulate the stream - open, close, configure, suspend, abort. param mgr A2DP manager. return li c Stream handle if the function succeeds. li c 0 otherwise.
	bt_a2dp_destroy_stream	brief Destroy a stream. ingroup a2dp details This function frees memory used by the stream. The stream has to exist and be in the "idle" state for this function to succeed. I.e. the stream has to be closed or aborted before this function can be called. param mgr A2DP manager. param strm_handle Stream handle. return li c TRUE if the function succeeds. li c FALSE otherwise.
	bt_a2dp_disconnect	brief Disconnect from a remote device. ingroup a2dp details This function closes a control and transport channels on all streams associated with the remote device specified by the p remote_addr. As a result of this operation the following events will be generated: @arg A2DP_EVT_MEDIA_PACKET_RECEIVED: if a stream's receive queue is not empty this event is generated for each buffer with bt_media_packet_t::data_len set to 0 @arg A2DP_EVT_MEDIA_PACKET_SENT: if a stream's send queue is not empty this event is generated for each buffer with bt_media_packet_t::data_len set to 0 @arg A2DP_EVT_STREAM_CLOSED: this event is generate if a stream is in... more
	bt_a2dp_discover	brief Discover SEPs on a remote device. ingroup a2dp details This function asks the remote device to send a list of all available SEPs. As a result of this operation the following events will be generated: @arg A2DP_EVT_SEP_INFO_RECEIVED: this event is generated for every SEP received from the remote device. the p evt_param.sep_info_received contains SEP information. @arg A2DP_EVT_DISCOVER_COMPLETED: this event is generated after the last A2DP_EVT_SEP_INFO_RECEIVED if the remote accepted the request and the p evt_param.discover_completed.err_code == AVDTP_ERROR_SUCCESS. if the remote rejected the request the p evt_param.discover_completed.err_code == the error code sent by the remote. param mgr A2DP... more

	bt_a2dp_find_codec	brief Find a codec ingroup a2dp details A2DP in theory can support any type of codec. Each codec uses its own format for exchanging capabilities and configuration information. In order to make our implementation do not care about these formats we use a simple way of telling AVDTP how to parse and serialize codec's configuration. The consumer of A2DP has to register a callback function (one per codec type) for each codec it wishes to support. That callback has to perform two function. The first one is to read the configuration received from the remote device and store it in... more
	bt_a2dp_get_capabilities	brief Get remote SEP capabilities. ingroup a2dp details This function asks the remote device to send capabilities of a SEP specified by the p seid_acp. As a result of this operation the following events will be generated: @arg A2DP_EVT_SEP_CAPABILITIES_RECEIVED: this event is generated if the remote device accepted the request. the p evt_param.sep_capabilities_received contains SEP capabilities. @arg A2DP_EVT_GET_SEP_CAPABILITIES_COMPLETED: this event is generated right after A2DP_EVT_SEP_CAPABILITIES_RECEIVED if the remote accepted the request the p evt_param.get_sep_capabilities_completed.err_code == AVDTP_ERROR_SUCCESS. if the remote rejected the request the p evt_param.get_sep_capabilities_completed.err_code == the error code sent by the remote. param mgr A2DP manager.... more
	bt_a2dp_get_hci_connection	brief Get HCI connection for a stream ingroup a2dp details This function returns a pointer to a structure that describes an HCI connection a stream is open on. The return value can be used to call various function from the HCI layer. For example, if an app wants to force disconnection from a remote device it can call ::bt_hci_disconnect. param mgr A2DP manager. param strm_handle Stream handle. return li c Pointer to a structure that describes an HCI connection if the function succeeds. li c NULL otherwise. The function fails only if a stream specified by the p... more
	bt_a2dp_get_stream_codec_config	brief Get the configuration of the codec currently used with the stream. ingroup a2dp details This function returns a pointer to a structure that contains configuration of the codec currently used with the stream. The structure returned depends on the codec. The dotstack defines structures only for SBC, MPEG-1,2 and MPEG-2,4 AAC codecs: @arg SBC: bt_a2dp_sbc_config_t (defined in a2dp_sbc_codec.h) @arg MPEG-1,2: bt_a2dp_mpeg_config_t (defined in a2dp_mpeg_codec.h) @arg MPEG-2,4 AAC: bt_a2dp_aac_config_t (defined in a2dp_aac_codec.h) param mgr A2DP manager. param strm_handle Stream handle. return li The codec's configuration if strm_handle specifies a valid stream and the stream is in one of the following... more
	bt_a2dp_get_stream_codec_type	brief Get the type of the codec currently used with the stream. ingroup a2dp details This function returns the type of the codec currently used with the stream. param mgr A2DP manager. param strm_handle Stream handle. return @arg The type of the codec if strm_handle specifies a valid stream and the stream is in one of the following states: AVDTP_STREAM_STATE_CONFIGURED AVDTP_STREAM_STATE_OPEN AVDTP_STREAM_STATE_STREAMING @arg The result will be one of the following values: AVDTP_CODEC_TYPE_SBC: SBC AVDTP_CODEC_TYPE_MPEG1_2_AUDIO: MPEG-1,2 (used in MP3 files) AVDTP_CODEC_TYPE_MPEG2_4_AAC: MPEG-2,4 AAC (used in Apple products) AVDTP_CODEC_TYPE_ATRAC: ATRAC (used in Sony products) AVDTP_CODEC_TYPE_NON_A2DP: Non-A2DP... more

	bt_a2dp_get_stream_local_sep_id	brief Get stream's local SEP ID. ingroup a2dp details This function returns the ID of the local SEP associated with the stream. param mgr A2DP manager. param strm_handle Stream handle. return li The ID of the local SEP if strm_handle specifies a valid stream. li 0 otherwise.
	bt_a2dp_get_stream_remote_address	brief Get stream's remote BT address. ingroup a2dp details This function returns the address of the remote device associated with the stream. param mgr A2DP manager. param strm_handle Stream handle. return li The address of the remote device if strm_handle specifies a valid stream. li NULL otherwise.
	bt_a2dp_get_stream_remote_sep_id	brief Get stream's remote SEP ID. ingroup a2dp details This function returns the ID of the remote SEP associated with the stream. param mgr A2DP manager. param strm_handle Stream handle. return li The ID of the remote SEP if strm_handle specifies a valid stream. li 0 otherwise.
	bt_a2dp_get_stream_state	brief Get local stream state. ingroup a2dp details This function returns local state of a stream specified by the p strm_handle. No request is sent to the remote party. param mgr A2DP manager. param strm_handle Stream handle. return The state of the stream. The result will be one of the following values: @arg AVDTP_STREAM_STATE_IDLE: The stream is idle. This can mean two things. The stream specified by p strm_handle does not exist or the stream is closed. @arg AVDTP_STREAM_OPENING_TRANSPORT_CHANNELS: The stream is opening transport channels. @arg AVDTP_STREAM_CLOSING_TRANSPORT_CHANNELS: The stream is closing transport channels. @arg AVDTP_STREAM_STATE_CONFIGURED: The... more
	bt_a2dp_listen	brief Listen for incoming connections. ingroup a2dp details This function tells a stream that it can use a particular SEP to accept incoming requests to open it. The SEP can be associated with multiple streams but used with only one. The stream has to be closed before the SEP can be used with another stream. For outgoing connections this is not needed. Any SEP can be used with any stream given that the SEP is not already in use by another stream. param mgr A2DP manager. param strm_handle Stream handle. param sep_id Local SEP ID. return li c TRUE if... more
	bt_a2dp_reconfigure_stream	brief Reconfigure stream. ingroup a2dp details This function tries to change the stream's configuration. For this function to succeed the stream has to be open. As a result of this operation the A2DP_EVT_STREAM_RECONFIGURE_COMPLETED event will be generated. If reconfiguration was a success the p evt_param.stream_reconfigure_completed.err_code == AVDTP_ERROR_SUCCESS. Otherwise the p evt_param.stream_reconfigure_completed.err_code == the error code sent by the remote. param mgr A2DP manager. param strm_handle Stream handle. param caps New stream configuration. return li c TRUE if the function succeeds, i.e. the actual request has been sent to the remote party. li c FALSE otherwise. No events will be... more
	bt_a2dp_register_sink	brief Register a Sink SEP with the local A2DP manager. ingroup a2dp details This function is used to add a sink SEP to a list of SEPs supported by the local A2DP entity. param mgr A2DP manager. param caps The capabilities of a SEP. return li c ID of a SEP if the function succeeds. li c FALSE otherwise.

	bt_a2dp_register_source	brief Register a Source SEP with the local A2DP manager. ingroup a2dp details This function is used to add a source SEP to a list of SEPs supported by the local A2DP entity. param mgr A2DP manager. param caps The capabilities of a SEP. return li c ID of a SEP if the function succeeds. li c FALSE otherwise.
	bt_a2dp_remove_media_rx_buffer	brief Remove a media packet buffer from a receive queue ingroup a2dp details The consumer of A2DP is responsible for allocating and supplying A2DP with buffers used to store received packets. A2DP itself only has a queue for storing pointers to buffers supplied by the consumer. When a packet comes in A2DP finds the first buffer large enough to hold the received packet, copies the packet to the buffer and generates a A2DP_EVT_MEDIA_PACKET_RECEIVED event. The consumer then has to process the data in the buffer and return it back to the queue. If there is no buffers in the queue... more
	bt_a2dp_remove_media_tx_buffer	brief Remove a media packet buffer from a send queue ingroup a2dp details When the consumer of A2DP wants to send a packet to a remote device it calls bt_avdtp_add_media_tx_buffer function. The function adds the packet to a queue and tells A2DP that it has something to send. The packet will be send as soon as the stream goes to A2DP_STREAM_STATE_STREAMING state. The consumer has a chance to remove a packet from the queue before it has been sent to a remote device by calling ::bt_a2dp_remove_media_tx_buffer. param mgr A2DP manager. param strm_handle Stream handle. param buffer Pointer to a structure... more
	bt_a2dp_start_stream	brief Start a stream. ingroup a2dp details This function tries to start a stream by sending a request to the remote party. The stream has to be in AVDTP_STREAM_STATE_OPEN state. The stream goes to this state as a result of successful configuration or suspension (both can be initiated by either party). As a result of this operation the A2DP_EVT_START_STREAM_COMPLETED event will be generated. If the stream has been open the p evt_param.start_stream_requested.err_code == AVDTP_ERROR_SUCCESS. Otherwise, if the remote device for any reason cannot or does not wish to start the stream, the p evt_param.start_stream_requested.err_code == the error code sent... more
	bt_a2dp_suspend_stream	brief Suspend a stream. ingroup a2dp details This function tries to suspend a stream by sending a request to the remote party. The stream has to be in AVDTP_STREAM_STATE_STREAMING state. As a result of this operation the A2DP_EVT_SUSPEND_STREAM_COMPLETED event will be generated. If the stream has been suspended the p evt_param.bt_avdtp_evt_suspend_stream_requested_t.err_code == AVDTP_ERROR_SUCCESS. Otherwise, if the remote device for any reason cannot or does not wish to suspend the stream, the p evt_param.bt_avdtp_evt_suspend_stream_requested_t.err_code == the error code sent by the remote. param mgr A2DP manager. param strm_handle Stream handle. return li c TRUE if the function succeeds, i.e. the... more
	MPEG_BITRATE_0000	This is macro MPEG_BITRATE_0000.
	MPEG_BITRATE_0001	This is macro MPEG_BITRATE_0001.
	MPEG_BITRATE_0010	This is macro MPEG_BITRATE_0010.
	MPEG_BITRATE_0011	This is macro MPEG_BITRATE_0011.
	MPEG_BITRATE_0100	This is macro MPEG_BITRATE_0100.
	MPEG_BITRATE_0101	This is macro MPEG_BITRATE_0101.
	MPEG_BITRATE_0110	This is macro MPEG_BITRATE_0110.
	MPEG_BITRATE_0111	This is macro MPEG_BITRATE_0111.
	MPEG_BITRATE_1000	This is macro MPEG_BITRATE_1000.
	MPEG_BITRATE_1001	This is macro MPEG_BITRATE_1001.

	MPEG_BITRATE_1010	This is macro MPEG_BITRATE_1010.
	MPEG_BITRATE_1011	This is macro MPEG_BITRATE_1011.
	MPEG_BITRATE_1100	This is macro MPEG_BITRATE_1100.
	MPEG_BITRATE_1101	This is macro MPEG_BITRATE_1101.
	MPEG_BITRATE_1110	This is macro MPEG_BITRATE_1110.
	MPEG_BITRATE_ALL	This is macro MPEG_BITRATE_ALL.
	MPEG_CHANNEL_MODE_ALL	This is macro MPEG_CHANNEL_MODE_ALL.
	MPEG_CHANNEL_MODE_DUAL_CHANNEL	This is macro MPEG_CHANNEL_MODE_DUAL_CHANNEL.
	MPEG_CHANNEL_MODE_JOINT_STEREO	This is macro MPEG_CHANNEL_MODE_JOINT_STEREO.
	MPEG_CHANNEL_MODE_MONO	This is macro MPEG_CHANNEL_MODE_MONO.
	MPEG_CHANNEL_MODE_STEREO	This is macro MPEG_CHANNEL_MODE_STEREO.
	MPEG_CRC_PROTECTION_NOT_SUPPORTED	This is macro MPEG_CRC_PROTECTION_NOT_SUPPORTED.
	MPEG_CRC_PROTECTION_SUPPORTED	This is macro MPEG_CRC_PROTECTION_SUPPORTED.
	MPEG_LAYER_1	This is macro MPEG_LAYER_1.
	MPEG_LAYER_2	This is macro MPEG_LAYER_2.
	MPEG_LAYER_3	This is macro MPEG_LAYER_3.
	MPEG_LAYER_ALL	This is macro MPEG_LAYER_ALL.
	MPEG_MPF_1	This is macro MPEG_MPF_1.
	MPEG_MPF_2	This is macro MPEG_MPF_2.
	MPEG_SAMPLING_FREQUENCY_16000	This is macro MPEG_SAMPLING_FREQUENCY_16000.
	MPEG_SAMPLING_FREQUENCY_22050	This is macro MPEG_SAMPLING_FREQUENCY_22050.
	MPEG_SAMPLING_FREQUENCY_24000	This is macro MPEG_SAMPLING_FREQUENCY_24000.
	MPEG_SAMPLING_FREQUENCY_32000	This is macro MPEG_SAMPLING_FREQUENCY_32000.
	MPEG_SAMPLING_FREQUENCY_44100	This is macro MPEG_SAMPLING_FREQUENCY_44100.
	MPEG_SAMPLING_FREQUENCY_48000	This is macro MPEG_SAMPLING_FREQUENCY_48000.
	MPEG_SAMPLING_FREQUENCY_ALL	This is macro MPEG_SAMPLING_FREQUENCY_ALL.
	MPEG_VBR_NOT_SUPPORTED	This is macro MPEG_VBR_NOT_SUPPORTED.
	MPEG_VBR_SUPPORTED	This is macro MPEG_VBR_SUPPORTED.
	SBC_ALLOCATION_METHOD_ALL	This is macro SBC_ALLOCATION_METHOD_ALL.
	SBC_ALLOCATION_METHOD_LOUDNESS	This is macro SBC_ALLOCATION_METHOD_LOUDNESS.
	SBC_ALLOCATION_METHOD_SNR	This is macro SBC_ALLOCATION_METHOD_SNR.
	SBC_BLOCK_LENGTH_12	This is macro SBC_BLOCK_LENGTH_12.
	SBC_BLOCK_LENGTH_16	This is macro SBC_BLOCK_LENGTH_16.
	SBC_BLOCK_LENGTH_4	This is macro SBC_BLOCK_LENGTH_4.
	SBC_BLOCK_LENGTH_8	This is macro SBC_BLOCK_LENGTH_8.
	SBC_BLOCK_LENGTH_ALL	This is macro SBC_BLOCK_LENGTH_ALL.
	SBC_CHANNEL_MODE_ALL	This is macro SBC_CHANNEL_MODE_ALL.
	SBC_CHANNEL_MODE_DUAL_CHANNEL	This is macro SBC_CHANNEL_MODE_DUAL_CHANNEL.
	SBC_CHANNEL_MODE_JOINT_STEREO	This is macro SBC_CHANNEL_MODE_JOINT_STEREO.
	SBC_CHANNEL_MODE_MONO	This is macro SBC_CHANNEL_MODE_MONO.
	SBC_CHANNEL_MODE_STEREO	This is macro SBC_CHANNEL_MODE_STEREO.
	SBC_SAMPLING_FREQUENCY_16000	This is macro SBC_SAMPLING_FREQUENCY_16000.
	SBC_SAMPLING_FREQUENCY_32000	This is macro SBC_SAMPLING_FREQUENCY_32000.
	SBC_SAMPLING_FREQUENCY_44100	This is macro SBC_SAMPLING_FREQUENCY_44100.
	SBC_SAMPLING_FREQUENCY_48000	This is macro SBC_SAMPLING_FREQUENCY_48000.
	SBC_SAMPLING_FREQUENCY_ALL	This is macro SBC_SAMPLING_FREQUENCY_ALL.
	SBC_SUBBANDS_4	This is macro SBC_SUBBANDS_4.
	SBC_SUBBANDS_8	This is macro SBC_SUBBANDS_8.
	SBC_SUBBANDS_ALL	This is macro SBC_SUBBANDS_ALL.

	bt_a2dp_get_all_capabilities	brief Get remote SEP capabilities. ingroup a2dp details This function asks the remote device to send capabilities of a SEP specified by the p seid_acp. As a result of this operation the following events will be generated: @arg A2DP_EVT_SEP_CAPABILITIES_RECEIVED: this event is generated if the remote device accepted the request. the p evt_param.sep_capabilities_received contains SEP capabilities. @arg A2DP_EVT_GET_SEP_CAPABILITIES_COMPLETED: this event is generated right after A2DP_EVT_SEP_CAPABILITIES_RECEIVED if the remote accepted the request the p evt_param.get_sep_capabilities_completed.err_code == AVDTP_ERROR_SUCCESS. if the remote rejected the request the p evt_param.get_sep_capabilities_completed.err_code == the error code sent by the remote. param mgr A2DP manager.... more
	bt_a2dp_get_configuration	This is macro bt_a2dp_get_configuration.
	bt_a2dp_open_stream	This is macro bt_a2dp_open_stream.
	bt_a2dp_clear_media_tx_queue	brief Clear send queue ingroup a2dp details When the consumer of A2DP wants to send a packet to a remote device it calls bt_avdtp_add_media_tx_buffer function. The function adds the packet to a queue and tells A2DP that it has something to send. The packet will be send as soon as the stream goes to A2DP_STREAM_STATE_STREAMING state. The consumer can remove all packets from the queue before they have been sent to a remote device by calling ::bt_a2dp_clear_media_tx_queue. param mgr A2DP manager. param strm_handle Stream handle. return li c TRUE if the function succeeds. li c FALSE otherwise. The function fails... more
	bt_a2dp_get_stream_config	brief Get stream's configuration. ingroup a2dp details This function returns a pointer to a structure holding current configuration of the stream. param mgr A2DP manager. param strm_handle Stream handle. return li The stream's configuration if strm_handle specifies a valid stream and the stream is in one of the following state: AVDTP_STREAM_STATE_CONFIGURED AVDTP_STREAM_STATE_OPEN AVDTP_STREAM_STATE_STREAMING li NULL otherwise.
	bt_a2dp_get_stream_direction	This is macro bt_a2dp_get_stream_direction.
	bt_a2dp_report_delay	This is macro bt_a2dp_report_delay.
	bt_a2dp_set_media_tx_queue_limit	brief Set limit on the send queue ingroup a2dp details When the consumer of A2DP wants to send a packet to a remote device it calls bt_avdtp_add_media_tx_buffer function. The function adds the packet to a queue and tells A2DP that it has something to send. The packet will be send as soon as the stream goes to A2DP_STREAM_STATE_STREAMING state. By default the send queue can contain unlimited number of packets. The consumer can set a limit on how many packets are held in the queue. In this case when new packet is added to the queue and the length of... more
	A2DP_EVT_SET_STREAM_CONFIGURATION	This is macro A2DP_EVT_SET_STREAM_CONFIGURATION.

A2DP Functions

	Name	Description
	_bt_a2dp_avdtp_mgr_callback	This is function _bt_a2dp_avdtp_mgr_callback.
	bt_a2dp_aac_codec_handler	This is function bt_a2dp_aac_codec_handler.
	bt_a2dp_find_server	This is function bt_a2dp_find_server.

	bt_a2dp_find_sink	brief Find sink ingroup a2dp details This function looks for a sink on a remote device specified by c deviceAddress and, if found, returns features supported by the sink. param deviceAddress The address of a remote device. param callback The callback function that will be called when search has completed. param client_callback The optional callback function that an application can set if it wants to be notified of state changes of the SDP client. The c evt parameter of the callback can be one of the following values: li SDP_CLIENT_STATE_IDLE li SDP_CLIENT_STATE_CONNECTING li SDP_CLIENT_STATE_DISCONNECTING li SDP_CLIENT_STATE_CONNECTED param callback_param A pointer... more
	bt_a2dp_find_source	brief Find source ingroup a2dp details This function looks for a source on a remote device specified by c deviceAddress and, if found, returns features supported by the source. param deviceAddress The address of a remote device. param callback The callback function that will be called when search has completed. param client_callback The optional callback function that an application can set if it wants to be notified of state changes of the SDP client. The c evt parameter of the callback can be one of the following values: li SDP_CLIENT_STATE_IDLE li SDP_CLIENT_STATE_CONNECTING li SDP_CLIENT_STATE_DISCONNECTING li SDP_CLIENT_STATE_CONNECTED param callback_param A pointer... more
	bt_a2dp_get_mgr	brief Return a pointer to an instance of the A2DP manager. ingroup a2dp details This function returns a pointer to an instance of the A@DP manager. There is only one instance of the manager allocated by the stack. The pointer is passed as the first parameter to all A2DP functions.
	bt_a2dp_init	brief Initialize the A2DP layer. ingroup a2dp details This function initializes the A2DP layer of the stack. It must be called prior to any other A2DP function can be called.
	bt_a2dp_mpeg_codec_handler	This is function bt_a2dp_mpeg_codec_handler.
	bt_a2dp_open_and_start_stream	brief Open & start a stream ingroup a2dp details Opening a stream involves sending 3 requests to a remote device - "set configuration", "open stream" and "start stream". Each event generates its own event which must be handled and acted accordingly by the application. To make the use of API easier dotstack combines all these requests in one request called "open & start stream". dotstack sends necessary requests in a proper sequence, handles responses and generates only one event (A2DP_EVT_OPEN_AND_START_STREAM_COMPLETED) at the end. If any of the individual requests has failed the event's parameter bt_a2dp_event_t::open_and_start_stream_completed is populated with... more
	bt_a2dp_register_aac_codec	brief Register default AAC codec ingroup a2dp details This function adds AAC codec implemented by dotstack to the list of known codecs. For more information about codecs see description of ::bt_avdtp_register_codec. The only codec A2DP is mandatory to support is SBC. All other codecs are optional. If an application wants to use AAC codec it must call this function when it is initializing. note dotstack codecs do not do actual encoding/decoding. their function is to parse and serialize codec's configuration. param mgr A2DP manager.
	bt_a2dp_register_callback	brief Register a A2DP application callback. ingroup a2dp details In order to be notified of various events a consumer of the A2DP layer has to register a callback function. The stack will call this function whenever a new event has been generated passing the code of the event as the second parameter. The event can be one of the following values: @arg A2DP_EVT_CTRL_CHANNEL_CONNECTED: Control channel connected. @arg A2DP_EVT_CTRL_CHANNEL_DISCONNECTED: Control channel disconnected. @arg A2DP_EVT_CTRL_CONNECTION_FAILED: Control channel connection failed (generated only if control connection has been initiated by the local device). @arg A2DP_EVT_DISCOVER_COMPLETED: Local device completed discovering remote SEPs.... more
	bt_a2dp_register_mpeg_codec	brief Register default MPEG codec ingroup a2dp details This function adds MPEG codec implemented by dotstack to the list of known codecs. For more information about codecs see description of ::bt_avdtp_register_codec. The only codec A2DP is mandatory to support is SBC. All other codecs are optional. If an application wants to use MPEG-1,2 codec it must call this function when it is initializing. note dotstack codecs do not do actual encoding/decoding. their function is to parse and serialize codec's configuration. param mgr A2DP manager.
	bt_a2dp_sbc_codec_handler	This is function bt_a2dp_sbc_codec_handler.

	bt_a2dp_start	brief Start the A2DP layer. ingroup a2dp details This function makes the A2DP layer ready to accept connection requests from remote device. To make an outgoing connection calling this function is not required. param mgr AVDTP manager. return li c TRUE if the function succeeds. li c FALSE otherwise.
	bt_a2dp_set_configuration	This is function bt_a2dp_set_configuration.

AVCTP Data Types and Constants

	Name	Description
	_bt_avctp_channel_t	brief AVCTP channel description ingroup avctp details This structure is used to hold information about an AVCTP channel.
	_bt_avctp_event_u	brief Parameter to an application callback. ingroup avctp details This union is used to pass event specific data to the AVCTP consumer. Which member of the union points to a valid structure depends on the event reported to the consumer. In general, each event has a corresponding member in the union.
	_bt_avctp_evt_channel_connected_t	brief Parameter to AVCTP_EVT_CHANNEL_CONNECTED event ingroup avctp details A pointer to this structure is passed to the AVCTP application callback as a valid member of the bt_avctp_event_t union - bt_avctp_event_t::channel_connected - when a channel between two devices has been established.
	_bt_avctp_evt_channel_disconnected_t	brief Parameter to AVCTP_EVT_CHANNEL_DISCONNECTED event ingroup avctp details A pointer to this structure is passed to the AVCTP application callback as a valid member of the bt_avctp_event_t union - bt_avctp_event_t::channel_disconnected - when a channel between two devices has been terminated.
	_bt_avctp_evt_command_cancelled_t	brief Parameter to AVCTP_EVT_COMMAND_CANCELLED event ingroup avctp details A pointer to this structure is passed to the AVCTP application callback as a valid member of the bt_avctp_event_t union - bt_avctp_event_t::command_cancelled - when sending a command message has been canceled.
	_bt_avctp_evt_command_received_t	brief Parameter to AVCTP_EVT_COMMAND_RECEIVED event ingroup avctp details A pointer to this structure is passed to the AVCTP application callback as a valid member of the bt_avctp_event_t union - bt_avctp_event_t::command_received - when a local device received a command message.
	_bt_avctp_evt_command_sent_t	brief Parameter to AVCTP_EVT_COMMAND_SENT event ingroup avctp details A pointer to this structure is passed to the AVCTP application callback as a valid member of the bt_avctp_event_t union - bt_avctp_event_t::command_sent - when a local device finished sending a command message.
	_bt_avctp_evt_connection_failed_t	brief Parameter to AVCTP_EVT_CONNECTION_FAILED event ingroup avctp details A pointer to this structure is passed to the AVCTP application callback as a valid member of the bt_avctp_event_t union - bt_avctp_event_t::connection_failed - when a channel between two devices could not be established.
	_bt_avctp_evt_response_cancelled_t	brief Parameter to AVCTP_EVT_RESPONSE_CANCELLED event ingroup avctp details A pointer to this structure is passed to the AVCTP application callback as a valid member of the bt_avctp_event_t union - bt_avctp_event_t::response_cancelled - when sending a response message has been canceled.
	_bt_avctp_evt_response_received_t	brief Parameter to AVCTP_EVT_RESPONSE_RECEIVED event ingroup avctp details A pointer to this structure is passed to the AVCTP application callback as a valid member of the bt_avctp_event_t union - bt_avctp_event_t::response_received - when a local device received a response message.

	_bt_avctp_evt_response_sent_t	brief Parameter to AVCTP_EVT_RESPONSE_SENT event ingroup avctp details A pointer to this structure is passed to the AVCTP application callback as a valid member of the bt_avctp_event_t union - bt_avctp_event_t::response_sent - when a local device finished sending a response message.
	_bt_avctp_message_t	brief AVCTP message description ingroup avctp details This structure is used to hold information about an AVCTP message.
	_bt_avctp_mgr_t	brief AVCTP manager. ingroup avctp details A structure that glues all pieces together. There is only one instance of this structure allocated by dotstack. A pointer to the instance can be get with c bt_avctp_get_mgr().
	_bt_avctp_packet_t	This is type bt_avctp_packet_t.
	_bt_avctp_transport_t	brief AVCTP transport description ingroup avctp details This structure is used to hold information about an AVCTP transport.
	bt_avctp_channel_t	This is type bt_avctp_channel_t.
	bt_avctp_event_t	brief Parameter to an application callback. ingroup avctp details This union is used to pass event specific data to the AVCTP consumer. Which member of the union points to a valid structure depends on the event reported to the consumer. In general, each event has a corresponding member in the union.
	bt_avctp_evt_channel_connected_t	brief Parameter to AVCTP_EVT_CHANNEL_CONNECTED event ingroup avctp details A pointer to this structure is passed to the AVCTP application callback as a valid member of the bt_avctp_event_t union - bt_avctp_event_t::channel_connected - when a channel between two devices has been established.
	bt_avctp_evt_channel_disconnected_t	brief Parameter to AVCTP_EVT_CHANNEL_DISCONNECTED event ingroup avctp details A pointer to this structure is passed to the AVCTP application callback as a valid member of the bt_avctp_event_t union - bt_avctp_event_t::channel_disconnected - when a channel between two devices has been terminated.
	bt_avctp_evt_command_cancelled_t	brief Parameter to AVCTP_EVT_COMMAND_CANCELLED event ingroup avctp details A pointer to this structure is passed to the AVCTP application callback as a valid member of the bt_avctp_event_t union - bt_avctp_event_t::command_cancelled - when sending a command message has been canceled.
	bt_avctp_evt_command_received_t	brief Parameter to AVCTP_EVT_COMMAND_RECEIVED event ingroup avctp details A pointer to this structure is passed to the AVCTP application callback as a valid member of the bt_avctp_event_t union - bt_avctp_event_t::command_received - when a local device received a command message.
	bt_avctp_evt_command_sent_t	brief Parameter to AVCTP_EVT_COMMAND_SENT event ingroup avctp details A pointer to this structure is passed to the AVCTP application callback as a valid member of the bt_avctp_event_t union - bt_avctp_event_t::command_sent - when a local device finished sending a command message.
	bt_avctp_evt_connection_failed_t	brief Parameter to AVCTP_EVT_CONNECTION_FAILED event ingroup avctp details A pointer to this structure is passed to the AVCTP application callback as a valid member of the bt_avctp_event_t union - bt_avctp_event_t::connection_failed - when a channel between two devices could not be established.
	bt_avctp_evt_response_cancelled_t	brief Parameter to AVCTP_EVT_RESPONSE_CANCELLED event ingroup avctp details A pointer to this structure is passed to the AVCTP application callback as a valid member of the bt_avctp_event_t union - bt_avctp_event_t::response_cancelled - when sending a response message has been canceled.

	bt_avctp_evt_response_received_t	brief Parameter to AVCTP_EVT_RESPONSE_RECEIVED event ingroup avctp details A pointer to this structure is passed to the AVCTP application callback as a valid member of the bt_avctp_event_t union - bt_avctp_event_t::response_received - when a local device received a response message.
	bt_avctp_evt_response_sent_t	brief Parameter to AVCTP_EVT_RESPONSE_SENT event ingroup avctp details A pointer to this structure is passed to the AVCTP application callback as a valid member of the bt_avctp_event_t union - bt_avctp_event_t::response_sent - when a local device finished sending a response message.
	bt_avctp_message_t	This is type bt_avctp_message_t.
	bt_avctp_mgr_callback_fp	brief AVCTP application callback. ingroup avctp details In order to be notified of various events a consumer of the AVCTP layer has to register a callback function (done with bt_avctp_set_callback()). The stack will call that function whenever a new event has been generated. param mgr AVCTP manager. param evt AVCTP event. The event can be one of the following values: @arg AVCTP_EVT_CHANNEL_CONNECTED Channel connected. @arg AVCTP_EVT_CHANNEL_DISCONNECTED Channel disconnected. @arg AVCTP_EVT_CONNECTION_FAILED Channel connection failed (generated only if connection has been initiated by the local device). @arg AVCTP_EVT_COMMAND_RECEIVED Command received. @arg AVCTP_EVT_RESPONE_RECEIVED Response received. @arg AVCTP_EVT_COMMAND_SENT Command sent. @arg AVCTP_EVT_RESPONSE_SENT Response... more
	bt_avctp_mgr_t	This is type bt_avctp_mgr_t.
	bt_avctp_packet_t	This is type bt_avctp_packet_t.
	bt_avctp_transport_t	This is type bt_avctp_transport_t.
	__AVCTP_H	This is macro __AVCTP_H.
	__AVCTP_PACKET_H	This is macro __AVCTP_PACKET_H.
	__AVCTP_PRIVATE_H	This is macro __AVCTP_PRIVATE_H.
	AVCTP_CHANNEL_FLAG_LISTENING	This is macro AVCTP_CHANNEL_FLAG_LISTENING.
	AVCTP_CHANNEL_FLAG_SENDING	This is macro AVCTP_CHANNEL_FLAG_SENDING.
	AVCTP_CHANNEL_STATE_CONNECTED	This is macro AVCTP_CHANNEL_STATE_CONNECTED.
	AVCTP_CHANNEL_STATE_CONNECTING	This is macro AVCTP_CHANNEL_STATE_CONNECTING.
	AVCTP_CHANNEL_STATE_DISCONNECTING	This is macro AVCTP_CHANNEL_STATE_DISCONNECTING.
	AVCTP_CHANNEL_STATE_FREE	This is macro AVCTP_CHANNEL_STATE_FREE.
	AVCTP_CHANNEL_STATE_IDLE	This is macro AVCTP_CHANNEL_STATE_IDLE.
	AVCTP_ERROR_BAD_STATE	This is macro AVCTP_ERROR_BAD_STATE.
	AVCTP_ERROR_SUCCESS	This is macro AVCTP_ERROR_SUCCESS.
	AVCTP_EVT_CHANNEL_CONNECTED	< This event is generated when a channel between two AVCTP entities has been established.
	AVCTP_EVT_CHANNEL_DISCONNECTED	< This event is generated when a channel between two AVCTP entities has been terminated.
	AVCTP_EVT_COMMAND_CANCELLED	< This event is generated when a command has been canceled.
	AVCTP_EVT_COMMAND_RECEIVED	< This event is generated when a local device received a command.
	AVCTP_EVT_COMMAND_SENT	< This event is generated when a local device finished sending a command.
	AVCTP_EVT_CONNECTION_FAILED	< This event is generated when a local device failed to create a channel between two AVCTP entities.
	AVCTP_EVT_NOTHING	addtogroup avrcp @{ @name Events details The following is a list of events AVRCP layer generates and can report to the upper layer when it completes executing an operation initiated by either local or remote device.
	AVCTP_EVT_RESPONE_RECEIVED	< This event is generated when a local device received a response.
	AVCTP_EVT_RESPONSE_CANCELLED	< This event is generated when a response has been canceled.
	AVCTP_EVT_RESPONSE_SENT	< This event is generated when a local device finished sending a response.

	AVCTP_MANAGER_STATE_IDLE	This is macro AVCTP_MANAGER_STATE_IDLE.
	AVCTP_MESSAGE_PACKET_TYPE_CONTINUE	This is macro AVCTP_MESSAGE_PACKET_TYPE_CONTINUE.
	AVCTP_MESSAGE_PACKET_TYPE_END	This is macro AVCTP_MESSAGE_PACKET_TYPE_END.
	AVCTP_MESSAGE_PACKET_TYPE_SINGLE	This is macro AVCTP_MESSAGE_PACKET_TYPE_SINGLE.
	AVCTP_MESSAGE_PACKET_TYPE_START	This is macro AVCTP_MESSAGE_PACKET_TYPE_START.
	AVCTP_MESSAGE_TYPE_COMMAND	This is macro AVCTP_MESSAGE_TYPE_COMMAND.
	AVCTP_MESSAGE_TYPE_RESPONSE	This is macro AVCTP_MESSAGE_TYPE_RESPONSE.
	AVCTP_TRANSPORT_FLAG_RX_MESSAGE_STARTED	This is macro AVCTP_TRANSPORT_FLAG_RX_MESSAGE_STARTED.
	AVCTP_TRANSPORT_FLAG_SENDING	This is macro AVCTP_TRANSPORT_FLAG_SENDING.
	AVCTP_TRANSPORT_STATE_CONNECTED	This is macro AVCTP_TRANSPORT_STATE_CONNECTED.
	AVCTP_TRANSPORT_STATE_CONNECTING	This is macro AVCTP_TRANSPORT_STATE_CONNECTING.
	AVCTP_TRANSPORT_STATE_DISCONNECTING	This is macro AVCTP_TRANSPORT_STATE_DISCONNECTING.
	AVCTP_TRANSPORT_STATE_FREE	This is macro AVCTP_TRANSPORT_STATE_FREE.
	AVCTP_TRANSPORT_STATE_IDLE	This is macro AVCTP_TRANSPORT_STATE_IDLE.
	AVDTP_MANAGER_FLAG_SENDING_MEDIA_PACKET	This is macro AVDTP_MANAGER_FLAG_SENDING_MEDIA_PACKET.

AVCTP Functions

	Name	Description
	_bt_avctp_allocate_channel	This is function _bt_avctp_allocate_channel.
	_bt_avctp_allocate_message	This is function _bt_avctp_allocate_message.
	_bt_avctp_allocate_transport	This is function _bt_avctp_allocate_transport.
	_bt_avctp_find_channel	This is function _bt_avctp_find_channel.
	_bt_avctp_find_transport	This is function _bt_avctp_find_transport.
	_bt_avctp_free_channel	This is function _bt_avctp_free_channel.
	_bt_avctp_free_message	This is function _bt_avctp_free_message.
	_bt_avctp_free_transport	This is function _bt_avctp_free_transport.
	_bt_avctp_init_message_buffers	This is function _bt_avctp_init_message_buffers.
	_bt_avctp_init_signal	This is function _bt_avctp_init_signal.
	_bt_avctp_l2cap_read_data_callback	This is function _bt_avctp_l2cap_read_data_callback.
	_bt_avctp_packet_assembler	This is function _bt_avctp_packet_assembler.
	_bt_avctp_send_ipid	This is function _bt_avctp_send_ipid.
	_bt_avctp_set_signal	This is function _bt_avctp_set_signal.
	_bt_avrcp_init_cmd_buffers	This is function _bt_avrcp_init_cmd_buffers.
	bt_avctp_cancel_command	brief Cancel a command message. ingroup avctp details If a message has not yet been sent to the remote device, it can be canceled (i.e. removed from send queue) by calling this function. param channel AVCTP channel. param tran_id Transaction Id. This value is obtained by calling bt_avctp_send_command. return li c TRUE if the function succeeds. li c FALSE otherwise. No events will be generated.
	bt_avctp_cancel_listen	brief Cancel listening for incoming connections. ingroup avctp details This function stops listening for incoming connections on the specified channel. param channel AVCTP channel. return li c TRUE if the function succeeds. li c FALSE otherwise.
	bt_avctp_cancel_response	brief Cancel a response message. ingroup avctp details If a message has not yet been sent to the remote device, it can be canceled (i.e. removed from send queue) by calling this function. param channel AVCTP channel. param tran_id Transaction Id. This value is obtained by calling bt_avctp_send_command. return li c TRUE if the function succeeds. li c FALSE otherwise. No events will be generated.

	bt_avctp_connect	brief Connect to a remote device. ingroup avctp details This function establishes a connection to a remote device specified by the p remote_address. If connection cannot be initiated for some reason, for example, there is not enough resources, it returns FALSE and no events are generated. Otherwise the result of an attempt to connect to the remote device is reported via the AVCTP callback. The events generated will either be AVCTP_EVT_CHANNEL_CONNECTED or AVCTP_EVT_CONNECTION_FAILED. param channel AVCTP channel. param remote_address The address of a remote device. param acl_config ACL link configuration. This can be a combination of the following values:... more
	bt_avctp_create_channel	brief Allocate AVCTP channel ingroup avctp details This function allocates a new incoming AVCTP channel. The channel is intended to be used to accept a connection from a remote device. There can be only one channel for each combination of c profile_id and c psm. param mgr AVCTP manager. param profile_id Profile Id param psm The PSM on which the underlying L2CAP channel will listen and accept incoming connections. param l2cap_mode Underlying L2CAP channel mode. This currently can only be CMODE_BASIC. return li A pointer to the new AVCTP channel if the function succeeds. li c NULL otherwise.
	bt_avctp_create_outgoing_channel	brief Allocate AVCTP channel ingroup avctp details This function allocates a new outgoing AVCTP channel. The channel is intended to be used to create a connection to a remote device. There can be multiple channels with the same c profile_id and c psm. param mgr AVCTP manager. param profile_id Profile Id param psm The PSM on which the underlying L2CAP channel will listen and accept incoming connections. param l2cap_mode Underlying L2CAP channel mode. This currently can only be CMODE_BASIC. return li A pointer to the new AVCTP channel if the function succeeds. li c NULL otherwise.
	bt_avctp_destroy_channel	brief Destroy AVCTP channel. ingroup avctp details This function frees memory used by the channel. The channel has to exist and be in the "idle" state for this function to succeed. I.e. the channel has to be disconnected before this function can be called. param channel AVCTP channel. return li c TRUE if the function succeeds. li c FALSE otherwise.
	bt_avctp_disconnect	brief Disconnect from a remote device. ingroup avctp details This function closes a connection to a remote device. param channel AVCTP channel. return li c TRUE if disconnection has been started. li c FALSE otherwise. No events will be generated.
	bt_avctp_get_channel_remote_address	brief Get channel's remote BT address. ingroup avctp details This function returns the address of the remote device associated with the channel. param channel AVCTP channel. return li The address of the remote device if channel is connected. li NULL otherwise.
	bt_avctp_get_channel_state	brief Get channel state ingroup avctp details This function return current state of the specified channel param channel AVCTP channel. return li AVCTP_CHANNEL_STATE_FREE li AVCTP_CHANNEL_STATE_IDLE li AVCTP_CHANNEL_STATE_CONNECTING li AVCTP_CHANNEL_STATE_CONNECTED li AVCTP_CHANNEL_STATE_DISCONNECTING
	bt_avctp_get_hci_connection	brief Get HCI connection for a channel ingroup avctp details This function returns a pointer to a structure that describes an HCI connection a channel is open on. The return value can be used to call various function from the HCI layer. For example, if an app wants to force disconnection from a remote device it can call ::bt_hci_disconnect. param channel AVCTP channel. return li c Pointer to a structure that describes an HCI connection if the function succeeds. li c NULL otherwise. The function fails only if a channel specified by the p channel parameter li does... more
	bt_avctp_get_mgr	brief Return a pointer to an instance of the AVCTP manager. ingroup avctp details This function returns a pointer to an instance of the AVCTP manager. There is only one instance of the manager allocated by the stack.
	bt_avctp_init	brief Initialize the AVCTP layer. ingroup avctp details This function initializes the AVCTP layer of the stack. It must be called prior to any other AVCTP function can be called.

	bt_avctp_listen	brief Listen for incoming connections. ingroup avctp details This function enables incoming connections on the specified AVCTP channel. param channel AVCTP channel. return li c TRUE if the function succeeds. li c FALSE otherwise.
	bt_avctp_send_command	brief Send a command message to a remote device. ingroup avctp details This function sends a command message to a remote device. param channel AVCTP channel. param data Message body. param data_len Message body length. param tran_id Pointer to a bt_byte where AVRCP will write transaction id assigned to the message. return li c TRUE if the function succeeds. li c FALSE otherwise. No events will be generated.
	bt_avctp_send_response	brief Send a response message to a remote device. ingroup avctp details This function sends a response message to a remote device. param channel AVCTP channel. param tran_id Transaction Id. This value is obtained by calling bt_avctp_send_command. param data Message body. param data_len Message body length. return li c TRUE if the function succeeds. li c FALSE otherwise. No events will be generated.
	bt_avctp_set_callback	brief Register a AVCTP application callback. ingroup avctp details In order to be notified of various events a consumer of the AVCTP layer has to register a callback function. The stack will call this function whenever a new event has been generated passing the code of the event as the second parameter. The event can be one of the following values: @arg AVCTP_EVT_CHANNEL_CONNECTED Channel connected. @arg AVCTP_EVT_CHANNEL_DISCONNECTED Channel disconnected. @arg AVCTP_EVT_CONNECTION_FAILED Channel connection failed (generated only if connection has been initiated by the local device). @arg AVCTP_EVT_COMMAND_RECEIVED Command received. @arg AVCTP_EVT_RESPONE_RECEIVED Response received. @arg AVCTP_EVT_COMMAND_SENT Command sent. @arg AVCTP_EVT_RESPONSE_SENT Response... more
	bt_avctp_start	This is function bt_avctp_start.

AVDTP Data Types and Constants

	Name	Description
	_bt_avdtp_codec_op_decode_t	There is currently no use for this structure.
	_bt_avdtp_codec_op_encode_t	There is currently no use for this structure.
	_bt_avdtp_codec_op_param_u	brief Parameter to a codec handler. ingroup avdtp details This union is used to pass operation specific data to a codec handler. Which member of the union points to a valid structure depends on the operation.
	_bt_avdtp_codec_op_parse_config_t	brief Parameter to AVDTP_CODEC_OPCODE_PARSE_CONFIG operation. ingroup avdtp details A pointer to this structure is passed to the codec handler as a valid member of the bt_avdtp_codec_op_param_t union - bt_avdtp_codec_op_param_t::parse - when ADVDTP needs to parse codec's capabilities/configuration received from the remote device.
	_bt_avdtp_codec_op_parse_packet_t	There is currently no use for this structure.
	_bt_avdtp_codec_op_serialize_config_t	brief Parameter to AVDTP_CODEC_OPCODE_SERIALIZE_CONFIG operation. ingroup avdtp details A pointer to this structure is passed to the codec handler as a valid member of the bt_avdtp_codec_op_param_t union - bt_avdtp_codec_op_param_t::serialize - when ADVDTP needs to serialize codec's capabilities/configuration for sending to the remote device.
	_bt_avdtp_codec_t	brief Codec handler description. ingroup avdtp details This structure is used to register a codec handler for parsing/serializing codec capabilities and configuration. See description of the ::bt_avdtp_register_codec for more details.
	_bt_avdtp_control_channel_t	This is type bt_avdtp_control_channel_t.
	_bt_avdtp_control_cmd_t	This is type bt_avdtp_control_cmd_t.
	_bt_avdtp_ctrl_evt_data_received_t	This is type bt_avdtp_ctrl_evt_data_received_t.

	_bt_avdtp_event_u	brief Parameter to an application callback. ingroup avdtp details This union is used to pass event specific data to the AVDTP consumer. Which member of the union points to a valid structure depends on the event reported to the consumer. In general, each event has a corresponding member in the union.
	_bt_avdtp_evt_abort_stream_requested_t	brief Parameter to AVDTP_EVT_ABORT_STREAM_REQUESTED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::abort_stream_requested - when AVDTP received a "abort stream" request.
	_bt_avdtp_evt_close_stream_completed_t	brief Parameter to AVDTP_EVT_CLOSE_STREAM_COMPLETED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::close_stream_completed - when AVDTP received a response to a "close stream" request.
	_bt_avdtp_evt_close_stream_requested_t	brief Parameter to AVDTP_EVT_CLOSE_STREAM_REQUESTED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::close_stream_requested - when AVDTP received a "close stream" request.
	_bt_avdtp_evt_ctrl_channel_connected_s	brief Parameter to AVDTP_EVT_CTRL_CHANNEL_CONNECTED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::ctrl_channel_connected - when a control channel between two devices has been established.
	_bt_avdtp_evt_ctrl_channel_disconnected_s	brief Parameter to AVDTP_EVT_CTRL_CHANNEL_DISCONNECTED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::ctrl_channel_disconnected - when a control channel between two devices has been terminated.
	_bt_avdtp_evt_discover_completed_t	brief Parameter to AVDTP_EVT_DISCOVER_COMPLETED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::discover_completed - when AVDTP completed discovering SEPs available on a remote device.
	_bt_avdtp_evt_get_sep_capabilities_completed_t	brief Parameter to AVDTP_EVT_GET_SEP_CAPABILITIES_COMPLETED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::get_sep_capabilities_completed - when AVDTP received a response to a "get SEP capabilities" request. AVDTP_EVT_GET_SEP_CAPABILITIES_COMPLETED only informs the status of the request - success or failure. In case of success another event - AVDTP_EVT_SEP_CAPABILITIES_RECEIVED - is generate with a pointer to a structure that holds actual SEP's capabilities.

	_bt_avdtp_evt_get_stream_configuration_completed_t	brief Parameter to AVDTP_EVT_SET_STREAM_CONFIGURATION_COMPLETED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::get_stream_configuration_completed - when AVDTP received a response to a "get stream configuration" request. AVDTP_EVT_SET_STREAM_CONFIGURATION_COMPLETED only informs the status of the request - success or failure. In case of success another event - AVDTP_EVT_STREAM_CONFIGURATION_RECEIVED - is generate with a pointer to a structure that hold actual stream's configuration.
	_bt_avdtp_evt_media_packet_received_t	brief Parameter to AVDTP_EVT_MEDIA_PACKET_RECEIVED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::media_packet_received - when ADVDTP received a media packet from the remote device.
	_bt_avdtp_evt_media_packet_send_failed_t	brief Parameter to AVDTP_EVT_MEDIA_PACKET_SEND_FAILED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::media_packet_send_failed - when ADVDTP failed to send a media packet to the remote device.
	_bt_avdtp_evt_media_packet_sent_t	brief Parameter to AVDTP_EVT_MEDIA_PACKET_SENT event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::media_packet_sent - when ADVDTP sent a media packet to the remote device.
	_bt_avdtp_evt_open_stream_completed_t	brief Parameter to AVDTP_EVT_OPEN_STREAM_COMPLETED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::open_stream_completed - when AVDTP received a response to a "open stream" request.
	_bt_avdtp_evt_open_stream_requested_t	brief Parameter to AVDTP_EVT_OPEN_STREAM_REQUESTED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::open_stream_requested - when AVDTP received a "open stream" request.
	_bt_avdtp_evt_reconfigure_stream_requested_t	brief Parameter to AVDTP_EVT_RECONFIGURE_STREAM_REQUESTED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::reconfigure_stream_requested - when AVDTP received a "change stream configuration" request.

	_bt_avdtp_evt_sep_capabilities_received_t	brief Parameter to AVDTP_EVT_SEP_CAPABILITIES_RECEIVED and AVDTP_EVT_STREAM_CONFIGURATION_RECEIVED events ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::sep_capabilities_received - when AVDTP received a positive response to a "get SEP capabilities" or "get stream configuration" request.
	_bt_avdtp_evt_sep_info_received_t	brief Parameter to AVDTP_EVT_SEP_INFO_RECEIVED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::sep_info_received - when AVDTP received positive result to a "discover" request. AVDTP_EVT_SEP_INFO_RECEIVED is generated for every SEP received from the remote device.
	_bt_avdtp_evt_set_stream_configuration_completed_t	brief Parameter to AVDTP_EVT_SET_STREAM_CONFIGURATION_COMPLETED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::set_stream_configuration_completed - when AVDTP received a response to a "set stream configuration" request.
	_bt_avdtp_evt_set_stream_configuration_requested_t	brief Parameter to AVDTP_EVT_SET_STREAM_CONFIGURATION_REQUESTED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::set_stream_configuration_requested - when AVDTP received a "set stream configuration" request.
	_bt_avdtp_evt_start_stream_completed_t	brief Parameter to AVDTP_EVT_START_STREAM_COMPLETED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::start_stream_completed - when AVDTP received a response to a "start stream" request.
	_bt_avdtp_evt_start_stream_requested_t	brief Parameter to AVDTP_EVT_START_STREAM_REQUESTED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::start_stream_requested - when AVDTP received a "start stream" request.
	_bt_avdtp_evt_stream_aborted_t	brief Parameter to AVDTP_EVT_STREAM_ABORTED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::stream_closed - to notify the AVDTP consumer that a stream has been successfully aborted.
	_bt_avdtp_evt_stream_closed_t	brief Parameter to AVDTP_EVT_STREAM_CLOSED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::stream_closed - to notify the AVDTP consumer that a stream has been successfully closed.

	_bt_avdtp_evt_stream_configured_t	brief Parameter to AVDTP_EVT_STREAM_CONFIGURED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::stream_configured - to notify the AVDTP consumer that a stream configuration has been successfully completed.
	_bt_avdtp_evt_stream_opened_t	brief Parameter to AVDTP_EVT_STREAM_OPENED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::stream_opened - to notify the AVDTP consumer that a stream has been successfully opened.
	_bt_avdtp_evt_stream_reconfigure_completed_t	brief Parameter to AVDTP_EVT_STREAM_RECONFIGURE_COMPLETED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::stream_reconfigure_completed - when AVDTP received a response to a "change stream configuration" request.
	_bt_avdtp_evt_stream_reconfigured_t	brief Parameter to AVDTP_EVT_STREAM_RECONFIGURED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::stream_reconfigured - to notify the AVDTP consumer that a stream configuration has been successfully changed.
	_bt_avdtp_evt_stream_security_control_completed_t	brief Parameter to AVDTP_EVT_STREAM_SECURITY_CONTROL_COMPLETED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::security_control_completed - when AVDTP received a response to a "exchange content protection control data" request.
	_bt_avdtp_evt_stream_started_t	brief Parameter to AVDTP_EVT_STREAM_STARTED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::stream_started - to notify the AVDTP consumer that a stream has been successfully started.
	_bt_avdtp_evt_stream_suspended_t	brief Parameter to AVDTP_EVT_STREAM_SUSPENDED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::stream_suspended - to notify the AVDTP consumer that a stream has been successfully suspended.
	_bt_avdtp_evt_suspend_stream_completed_t	brief Parameter to AVDTP_EVT_SUSPEND_STREAM_COMPLETED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::suspend_stream_completed - when AVDTP received a response to a "suspend stream" request.

	_bt_avdtp_evt_suspend_stream_requested_t	brief Parameter to AVDTP_EVT_SUSPEND_STREAM_REQUESTED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::suspend_stream_requested - when AVDTP received a "suspend stream" request.
	_bt_avdtp_mgr_t	brief AVDTP manager. ingroup avdtp details A structure that glues all pieces together. There is only one instance of this structure allocated by dotstack. A pointer to the instance can be get with c bt_avdtp_get_mgr().
	_bt_avdtp_sep_capabilities_t	brief SEP capabilities ingroup avdtp details This structure is used to hold SEP capabilities.
	_bt_avdtp_sep_t	brief SEP description ingroup avdtp details This structure is used to hold information about SEPs available on a local device.
	_bt_avdtp_stream_t	brief Stream description ingroup avdtp details This structure is used to hold information about streams available on a local device.
	_bt_avdtp_transport_channel_t	brief Transport channel description ingroup avdtp details This structure is used to hold information about transport channels available on a local device.
	_bt_avdtp_transport_session_t	brief Transport session description ingroup avdtp details This structure is used to hold information about transport sessions available on a local device.
	_bt_media_packet_t	brief Media packet buffer ingroup avdtp details This structure is used to receive and send media packet from/to the remote device. See more information about usage of this structure in descriptions of ::bt_avdtp_add_media_rx_buffer and ::bt_avdtp_add_media_tx_buffer.
	bt_avdtp_codec_handler_fp	brief Codec handler. ingroup avdtp details AVDTP in theory can support any type of codec. Each codec uses its own format for exchanging capabilities and configuration information. In order to make out implementation do not care about these formats we use a simple way of telling AVDTP how to parse and serialize codec's configuration. The consumer of AVDTP (e.g. A2DP) for each codec it wishes to support has to register a callback function (one per codec type). That callback has to perform two function. The first one is to read the configuration received from the remote device and store it... more
	bt_avdtp_codec_op_decode_t	There is currently no use for this structure.
	bt_avdtp_codec_op_encode_t	There is currently no use for this structure.
	bt_avdtp_codec_op_param_t	This is type bt_avdtp_codec_op_param_t.
	bt_avdtp_codec_op_parse_config_t	brief Parameter to AVDTP_CODEC_OPCODE_PARSE_CONFIG operation. ingroup avdtp details A pointer to this structure is passed to the codec handler as a valid member of the bt_avdtp_codec_op_param_t union - bt_avdtp_codec_op_param_t::parse - when ADVDTP needs to parse codec's capabilities/configuration received from the remote device.
	bt_avdtp_codec_op_parse_packet_t	There is currently no use for this structure.
	bt_avdtp_codec_op_serialize_config_t	brief Parameter to AVDTP_CODEC_OPCODE_SERIALIZE_CONFIG operation. ingroup avdtp details A pointer to this structure is passed to the codec handler as a valid member of the bt_avdtp_codec_op_param_t union - bt_avdtp_codec_op_param_t::serialize - when ADVDTP needs to serialize codec's capabilities/configuration for sending to the remote device.

	bt_avdtp_codec_t	brief Codec handler description. ingroup avdtp details This structure is used to register a codec handler for parsing/serializing codec capabilities and configuration. See description of the ::bt_avdtp_register_codec for more details.
	bt_avdtp_control_channel_t	This is type bt_avdtp_control_channel_t.
	bt_avdtp_control_cmd_t	This is type bt_avdtp_control_cmd_t.
	bt_avdtp_ctrl_evt_data_received_t	This is type bt_avdtp_ctrl_evt_data_received_t.
	bt_avdtp_event_t	brief Parameter to an application callback. ingroup avdtp details This union is used to pass event specific data to the AVDTP consumer. Which member of the union points to a valid structure depends on the event reported to the consumer. In general, each event has a corresponding member in the union.
	bt_avdtp_evt_abort_stream_requested_t	brief Parameter to AVDTP_EVT_ABORT_STREAM_REQUESTED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::abort_stream_requested - when AVDTP received a "abort stream" request.
	bt_avdtp_evt_close_stream_completed_t	brief Parameter to AVDTP_EVT_CLOSE_STREAM_COMPLETED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::close_stream_completed - when AVDTP received a response to a "close stream" request.
	bt_avdtp_evt_close_stream_requested_t	brief Parameter to AVDTP_EVT_CLOSE_STREAM_REQUESTED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::close_stream_requested - when AVDTP received a "close stream" request.
	bt_avdtp_evt_ctrl_channel_connected_t	brief Parameter to AVDTP_EVT_CTRL_CHANNEL_CONNECTED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::ctrl_channel_connected - when a control channel between two devices has been established.
	bt_avdtp_evt_ctrl_channel_disconnected_t	brief Parameter to AVDTP_EVT_CTRL_CHANNEL_DISCONNECTED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::ctrl_channel_disconnected - when a control channel between two devices has been terminated.
	bt_avdtp_evt_discover_completed_t	brief Parameter to AVDTP_EVT_DISCOVER_COMPLETED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::discover_completed - when AVDTP completed discovering SEPs available on a remote device.

	bt_avdtp_evt_get_sep_capabilities_completed_t	brief Parameter to AVDTP_EVT_GET_SEP_CAPABILITIES_COMPLETED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::get_sep_capabilities_completed - when AVDTP received a response to a "get SEP capabilities" request. AVDTP_EVT_GET_SEP_CAPABILITIES_COMPLETED only informs the status of the request - success or failure. In case of success another event - AVDTP_EVT_SEP_CAPABILITIES_RECEIVED - is generate with a pointer to a structure that holds actual SEP's capabilities.
	bt_avdtp_evt_get_stream_configuration_completed_t	brief Parameter to AVDTP_EVT_SET_STREAM_CONFIGURATION_COMPLETED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::get_stream_configuration_completed - when AVDTP received a response to a "get stream configuration" request. AVDTP_EVT_SET_STREAM_CONFIGURATION_COMPLETED only informs the status of the request - success or failure. In case of success another event - AVDTP_EVT_STREAM_CONFIGURATION_RECEIVED - is generate with a pointer to a structure that hold actual stream's configuration.
	bt_avdtp_evt_media_packet_received_t	brief Parameter to AVDTP_EVT_MEDIA_PACKET_RECEIVED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::media_packet_received - when ADVDTP received a media packet from the remote device.
	bt_avdtp_evt_media_packet_send_failed_t	brief Parameter to AVDTP_EVT_MEDIA_PACKET_SEND_FAILED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::media_packet_send_failed - when ADVDTP failed to send a media packet to the remote device.
	bt_avdtp_evt_media_packet_sent_t	brief Parameter to AVDTP_EVT_MEDIA_PACKET_SENT event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::media_packet_sent - when ADVDTP sent a media packet to the remote device.
	bt_avdtp_evt_open_stream_completed_t	brief Parameter to AVDTP_EVT_OPEN_STREAM_COMPLETED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::open_stream_completed - when AVDTP received a response to a "open stream" request.
	bt_avdtp_evt_open_stream_requested_t	brief Parameter to AVDTP_EVT_OPEN_STREAM_REQUESTED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::open_stream_requested - when AVDTP received a "open stream" request.

	bt_avdtp_evt_reconfigure_stream_requested_t	brief Parameter to AVDTP_EVT_RECONFIGURE_STREAM_REQUESTED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::reconfigure_stream_requested - when AVDTP received a "change stream configuration" request.
	bt_avdtp_evt_sep_capabilities_received_t	brief Parameter to AVDTP_EVT_SEP_CAPABILITIES_RECEIVED and AVDTP_EVT_STREAM_CONFIGURATION_RECEIVED events ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::sep_capabilities_received - when AVDTP received a positive response to a "get SEP capabilities" or "get stream configuration" request.
	bt_avdtp_evt_sep_info_received_t	brief Parameter to AVDTP_EVT_SEP_INFO_RECEIVED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::sep_info_received - when AVDTP received positive result to a "discover" request. AVDTP_EVT_SEP_INFO_RECEIVED is generated for every SEP received from the remote device.
	bt_avdtp_evt_set_stream_configuration_completed_t	brief Parameter to AVDTP_EVT_SET_STREAM_CONFIGURATION_COMPLETED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::set_stream_configuration_completed - when AVDTP received a response to a "set stream configuration" request.
	bt_avdtp_evt_set_stream_configuration_requested_t	brief Parameter to AVDTP_EVT_SET_STREAM_CONFIGURATION_REQUESTED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::set_stream_configuration_requested - when AVDTP received a "set stream configuration" request.
	bt_avdtp_evt_start_stream_completed_t	brief Parameter to AVDTP_EVT_START_STREAM_COMPLETED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::start_stream_completed - when AVDTP received a response to a "start stream" request.
	bt_avdtp_evt_start_stream_requested_t	brief Parameter to AVDTP_EVT_START_STREAM_REQUESTED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::start_stream_requested - when AVDTP received a "start stream" request.
	bt_avdtp_evt_stream_aborted_t	brief Parameter to AVDTP_EVT_STREAM_ABORTED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::stream_closed - to notify the AVDTP consumer that a stream has been successfully aborted.

	bt_avdtp_evt_stream_closed_t	brief Parameter to AVDTP_EVT_STREAM_CLOSED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::stream_closed - to notify the AVDTP consumer that a stream has been successfully closed.
	bt_avdtp_evt_stream_configured_t	brief Parameter to AVDTP_EVT_STREAM_CONFIGURED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::stream_configured - to notify the AVDTP consumer that a stream configuration has been successfully completed.
	bt_avdtp_evt_stream_opened_t	brief Parameter to AVDTP_EVT_STREAM_OPENED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::stream_opened - to notify the AVDTP consumer that a stream has been successfully opened.
	bt_avdtp_evt_stream_reconfigure_completed_t	brief Parameter to AVDTP_EVT_STREAM_RECONFIGURE_COMPLETED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::stream_reconfigure_completed - when AVDTP received a response to a "change stream configuration" request.
	bt_avdtp_evt_stream_reconfigured_t	brief Parameter to AVDTP_EVT_STREAM_RECONFIGURED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::stream_reconfigured - to notify the AVDTP consumer that a stream configuration has been successfully changed.
	bt_avdtp_evt_stream_security_control_completed_t	brief Parameter to AVDTP_EVT_STREAM_SECURITY_CONTROL_COMPLETED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::security_control_completed - when AVDTP received a response to a "exchange content protection control data" request.
	bt_avdtp_evt_stream_started_t	brief Parameter to AVDTP_EVT_STREAM_STARTED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::stream_started - to notify the AVDTP consumer that a stream has been successfully started.
	bt_avdtp_evt_stream_suspended_t	brief Parameter to AVDTP_EVT_STREAM_SUSPENDED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::stream_suspended - to notify the AVDTP consumer that a stream has been successfully suspended.

	bt_avdtp_evt_suspend_stream_completed_t	brief Parameter to AVDTP_EVT_SUSPEND_STREAM_COMPLETED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::suspend_stream_completed - when AVDTP received a response to a "suspend stream" request.
	bt_avdtp_evt_suspend_stream_requested_t	brief Parameter to AVDTP_EVT_SUSPEND_STREAM_REQUESTED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::suspend_stream_requested - when AVDTP received a "suspend stream" request.
	bt_avdtp_mgr_callback_fp	brief AVDTP application callback. ingroup avdtp details In order to be notified of various events a consumer of the AVDTP layer has to register a callback function. The stack will call that function whenever a new event has been generated. param mgr AVDTP manager. param evt AVDTP event. The event can be one of the following values: @arg AVDTP_EVT_CTRL_CHANNEL_CONNECTED: Control channel connected. @arg AVDTP_EVT_CTRL_CHANNEL_DISCONNECTED: Control channel disconnected. @arg AVDTP_EVT_CTRL_CONNECTION_FAILED: Control channel connection failed (generated only if control connection has been initiated by the local device). @arg AVDTP_EVT_DISCOVER_COMPLETED: Local device completed discovering remote SEPs. @arg AVDTP_EVT_GET_SEP_CAPABILITIES_COMPLETED:... more
	bt_avdtp_mgr_t	brief AVDTP manager. ingroup avdtp details A structure that glues all pieces together. There is only one instance of this structure allocated by dotstack. A pointer to the instance can be get with c bt_avdtp_get_mgr().
	bt_avdtp_sep_capabilities_t	brief SEP capabilities ingroup avdtp details This structure is used to hold SEP capabilities.
	bt_avdtp_sep_t	This is type bt_avdtp_sep_t.
	bt_avdtp_stream_t	brief Stream description ingroup avdtp details This structure is used to hold information about streams available on a local device.
	bt_avdtp_transport_channel_t	brief Transport channel description ingroup avdtp details This structure is used to hold information about transport channels available on a local device.
	bt_avdtp_transport_op_callback_fp	This is type bt_avdtp_transport_op_callback_fp.
	bt_avdtp_transport_session_t	brief Transport session description ingroup avdtp details This structure is used to hold information about transport sessions available on a local device.
	bt_media_packet_t	This is type bt_media_packet_t.
	__AVDTP_CONFIG_H	This is macro __AVDTP_CONFIG_H.
	__AVDTP_CONTROL_H	This is macro __AVDTP_CONTROL_H.
	__AVDTP_H	This is macro __AVDTP_H.
	__AVDTP_PRIVATE_H	This is macro __AVDTP_PRIVATE_H.
	AVDTP_CODEC_OPCODE_PARSE_CONFIG	< Parse codec configuration.
	AVDTP_CODEC_OPCODE_PARSE_PACKET	This is macro AVDTP_CODEC_OPCODE_PARSE_PACKET.
	AVDTP_CODEC_OPCODE_SERIALIZE_CONFIG	< Serialize codec configuration.
	AVDTP_CODEC_TYPE_ATRAC	< ATRAC (proprietary codec owned by Sony Corporation).
	AVDTP_CODEC_TYPE_MPEG1_2_AUDIO	< MPEG-1,2 (optional).
	AVDTP_CODEC_TYPE_MPEG2_4_AAC	< MPEG-2,4 AAC (optional, used in Apple's products).
	AVDTP_CODEC_TYPE_NON_A2DP	< Vendor specific.
	AVDTP_CODEC_TYPE_SBC	< SBC (mandatory to support in A2DP profile).

	AVDTP_CTLR_CHANNEL_EVT_CONNECTED	This is macro AVDTP_CTLR_CHANNEL_EVT_CONNECTED.
	AVDTP_CTLR_CHANNEL_EVT_CONNECTION_FAILED	This is macro AVDTP_CTLR_CHANNEL_EVT_CONNECTION_FAILED.
	AVDTP_CTLR_CHANNEL_EVT_DISCONNECTED	This is macro AVDTP_CTLR_CHANNEL_EVT_DISCONNECTED.
	AVDTP_CTLR_CHANNEL_EVT_NOTHING	This is macro AVDTP_CTLR_CHANNEL_EVT_NOTHING.
	AVDTP_CTRL_CHANNEL_EVT_DATA_RECEIVED	This is macro AVDTP_CTRL_CHANNEL_EVT_DATA_RECEIVED.
	AVDTP_CTRL_CHANNEL_STATE_CONNECTED	This is macro AVDTP_CTRL_CHANNEL_STATE_CONNECTED.
	AVDTP_CTRL_CHANNEL_STATE_CONNECTING	This is macro AVDTP_CTRL_CHANNEL_STATE_CONNECTING.
	AVDTP_CTRL_CHANNEL_STATE_DISCONNECTED	This is macro AVDTP_CTRL_CHANNEL_STATE_DISCONNECTED.
	AVDTP_CTRL_MESSAGE_TYPE_ACCEPT	This is macro AVDTP_CTRL_MESSAGE_TYPE_ACCEPT.
	AVDTP_CTRL_MESSAGE_TYPE_COMMAND	This is macro AVDTP_CTRL_MESSAGE_TYPE_COMMAND.
	AVDTP_CTRL_MESSAGE_TYPE_FLD	This is macro AVDTP_CTRL_MESSAGE_TYPE_FLD.
	AVDTP_CTRL_MESSAGE_TYPE_REJECT	This is macro AVDTP_CTRL_MESSAGE_TYPE_REJECT.
	AVDTP_CTRL_PACKET_TYPE_CONTINUE	This is macro AVDTP_CTRL_PACKET_TYPE_CONTINUE.
	AVDTP_CTRL_PACKET_TYPE_END	This is macro AVDTP_CTRL_PACKET_TYPE_END.
	AVDTP_CTRL_PACKET_TYPE_FLD	This is macro AVDTP_CTRL_PACKET_TYPE_FLD.
	AVDTP_CTRL_PACKET_TYPE_SIGNLE	This is macro AVDTP_CTRL_PACKET_TYPE_SIGNLE.
	AVDTP_CTRL_PACKET_TYPE_START	This is macro AVDTP_CTRL_PACKET_TYPE_START.
	AVDTP_ERROR_BAD_ACP_SEID	< The requested command indicates an invalid ACP SEP ID (not addressable)
	AVDTP_ERROR_BAD_CP_FORMAT	< The format of Content Protection Service Capability is not correct.
	AVDTP_ERROR_BAD_HEADER_FORMAT	< The request packet header format is invalid.
	AVDTP_ERROR_BAD_LENGTH	< The request packet length is not match the assumed length.
	AVDTP_ERROR_BAD_MEDIA_TRANSPORT_FORMAT	< The format of Media Transport Capability is not correct.
	AVDTP_ERROR_BAD_MULTIPLEXING_FORMAT	< The format of Multiplexing Service Capability is not correct.
	AVDTP_ERROR_BAD_PAYLOAD_FORMAT	< The requested command has an incorrect payload format.
	AVDTP_ERROR_BAD_RECOVERY_FORMAT	< The format of Recovery Service Capability is not correct.
	AVDTP_ERROR_BAD_RECOVERY_TYPE	< The requested Recovery Type is not defined in AVDTP.
	AVDTP_ERROR_BAD_ROHC_FORMAT	< The format of Header Compression Service Capability is not correct.
	AVDTP_ERROR_BAD_SERV_CATEGORY	< The value of Service Category in the request packet is not defined in AVDTP.
	AVDTP_ERROR_BAD_STATE	< The stream is in state that does not permit executing commands.
	AVDTP_ERROR_FAILED_TO_CONNECT_CONTROL	< An attempt to establish a control channel has failed.
	AVDTP_ERROR_FAILED_TO_CONNECT_TRANSPORT	< An attempt to establish a transport channel has failed.
	AVDTP_ERROR_INVALID_CAPABILITIES	< The reconfigure command is an attempt to reconfigure a transport service capabilities of the SEP. Reconfigure is only permitted for application service capabilities.
	AVDTP_ERROR_NOT_SUPPORTED_COMMAND	< The requested command is not supported by the device.
	AVDTP_ERROR_SEP_IN_USE	< The SEP is in use.
	AVDTP_ERROR_SEP_NOT_IN_USE	< The SEP is not in use.
	AVDTP_ERROR_SUCCESS	< The operation completed with no errors.
	AVDTP_ERROR_UNSUPPORTED_CONFIGURAION	< Configuration not supported.
	AVDTP_EVT_ABORT_STREAM_COMPLETED	< This event is generated when a local device received a response (either positive or negative) to a "abort stream" request.
	AVDTP_EVT_ABORT_STREAM_REQUESTED	< This event is generated when a local device received "abort stream" request.
	AVDTP_EVT_CLOSE_STREAM_COMPLETED	< This event is generated when a local device received a response (either positive or negative) to a "close stream" request.
	AVDTP_EVT_CLOSE_STREAM_REQUESTED	< This event is generated when a local device received "close stream" request.

	AVDTP_EVT_CTRL_CHANNEL_CONNECTED	< This event is generated when a control channel between two AVDTP entities has been established.
	AVDTP_EVT_CTRL_CHANNEL_DISCONNECTED	< This event is generated when a control channel between two AVDTP entities has been terminated.
	AVDTP_EVT_CTRL_CONNECTION_FAILED	< This event is generated when a local device failed to create a control channel between two AVDTP entities.
	AVDTP_EVT_DISCOVER_COMPLETED	< This event is generated when a local device received a response (either positive or negative) to a "discover" request.
	AVDTP_EVT_GET_SEP_CAPABILITIES_COMPLETED	< This event is generated when a local device received a response (either positive or negative) to a "get SEP capabilities" request.
	AVDTP_EVT_GET_STREAM_CONFIGURATION_COMPLETED	< This event is generated when a local device received a response (either positive or negative) to a "get stream configuration" request.
	AVDTP_EVT_LAST	This is macro AVDTP_EVT_LAST.
	AVDTP_EVT_MEDIA_PACKET_RECEIVED	< This event is generated when a local device received a media packet.
	AVDTP_EVT_MEDIA_PACKET_SEND_FAILED	< This event is generated when a local device failed to send a media packet.
	AVDTP_EVT_MEDIA_PACKET_SENT	< This event is generated when a local device sent a media packet.
	AVDTP_EVT_NULL	This is macro AVDTP_EVT_NULL.
	AVDTP_EVT_OPEN_STREAM_COMPLETED	< This event is generated when a local device received a response (either positive or negative) to a "open stream" request.
	AVDTP_EVT_OPEN_STREAM_REQUESTED	< This event is generated when a local device received "open stream" request.
	AVDTP_EVT_RECONFIGURE_STREAM_REQUESTED	< This event is generated when a local device received "change stream configuration" request.
	AVDTP_EVT_SEP_CAPABILITIES_RECEIVED	< This event is generated when a local device received a positive response to a "get SEP capabilities" request.
	AVDTP_EVT_SEP_INFO_RECEIVED	< This event is generated for each SEP contained in a positive response to a "discover" request.
	AVDTP_EVT_SET_STREAM_CONFIGURATION_COMPLETED	< This event is generated when a local device received a response (either positive or negative) to a "set stream configuration" request.
	AVDTP_EVT_SET_STREAM_CONFIGURATION_REQUESTED	< This event is generated when a local device received "set stream configuration" request.
	AVDTP_EVT_START_STREAM_COMPLETED	< This event is generated when a local device received a response (either positive or negative) to a "start stream" request.
	AVDTP_EVT_START_STREAM_REQUESTED	< This event is generated when a local device received "start stream" request.
	AVDTP_EVT_STREAM_ABORTED	< This event is generated when a local device has successfully aborted a stream. < This event follows the AVDTP_EVT_ABORT_STREAM_REQUESTED if the upper layer has accepted it. < This event is not generated if stream abortion was initiated by the local device.
	AVDTP_EVT_STREAM_CLOSED	< This event is generated when a local device has successfully closed a stream. < This event follows the AVDTP_EVT_CLOSE_STREAM_REQUESTED if the upper layer has accepted it. < This event is not generated if stream closing was initiated by the local device.
	AVDTP_EVT_STREAM_CONFIGURATION_RECEIVED	< This event is generated when a local device received a positive response to a "get stream configuration" request.
	AVDTP_EVT_STREAM_CONFIGURED	< This event is generated when a local device has successfully configured a stream. < This event follows the AVDTP_EVT_SET_STREAM_CONFIGURATION_REQUESTED if the upper layer has accepted it. < This event is not generated if stream configuration was initiated by the local device.

	AVDTP_EVT_STREAM_OPENED	< This event is generated when a local device has successfully opened a stream. < This event follows the AVDTP_EVT_OPEN_STREAM_REQUESTED if the upper layer has accepted it. < This event is not generated if stream opening was initiated by the local device.
	AVDTP_EVT_STREAM_RECONFIGURE_COMPLETED	< This event is generated when a local device received a response (either positive or negative) to a "change stream configuration" request.
	AVDTP_EVT_STREAM_RECONFIGURED	< This event is generated when a local device has successfully reconfigured a stream. < This event follows the AVDTP_EVT_RECONFIGURE_STREAM_REQUESTED if the upper layer has accepted it. < This event is not generated if stream reconfiguration was initiated by the local device.
	AVDTP_EVT_STREAM_SECURITY_CONTROL_COMPLETED	< This event is generated when a local device received a response (either positive or negative) to a "exchange content protection control data" request.
	AVDTP_EVT_STREAM_STARTED	< This event is generated when a local device has successfully started a stream. < This event follows the AVDTP_EVT_START_STREAM_REQUESTED if the upper layer has accepted it. < This event is not generated if stream starting was initiated by the local device.
	AVDTP_EVT_STREAM_SUSPENDED	< This event is generated when a local device has successfully suspended a stream. < This event follows the AVDTP_EVT_SUSPEND_STREAM_REQUESTED if the upper layer has accepted it. < This event is not generated if stream suspension was initiated by the local device.
	AVDTP_EVT_SUSPEND_STREAM_COMPLETED	< This event is generated when a local device received a response (either positive or negative) to a "suspend stream" request.
	AVDTP_EVT_SUSPEND_STREAM_REQUESTED	< This event is generated when a local device received "suspend stream" request.
	AVDTP_MANAGER_STATE_CONNECTING	This is macro AVDTP_MANAGER_STATE_CONNECTING.
	AVDTP_MANAGER_STATE_IDLE	This is macro AVDTP_MANAGER_STATE_IDLE.
	AVDTP_MAX_STREAM_TRANSPORT_SESSION	This is macro AVDTP_MAX_STREAM_TRANSPORT_SESSION.
	AVDTP_MEDIA_TYPE_AUDIO	< Audio.
	AVDTP_MEDIA_TYPE_MULTIMEDIA	< Both Audio & Video.
	AVDTP_MEDIA_TYPE_VIDEO	< Video.
	AVDTP_SEP_CAPABILITY_FLAG_CONTENT_PROTECTION	< Content Prortection. A SEP is capable of transferring content protection packets.
	AVDTP_SEP_CAPABILITY_FLAG_HEADER_COMPRESSION	< Header Compression. A SEP can use header compression for transferring Media or Recovery packets.
	AVDTP_SEP_CAPABILITY_FLAG_MEDIA_CODEC	< Media Codec. Defines which codec a SEP supports. A SEP can support only one codec.
	AVDTP_SEP_CAPABILITY_FLAG_MEDIA_TRANSPORT	< Media. A SEP is capable of transferring media (audio, video or both) packets.
	AVDTP_SEP_CAPABILITY_FLAG_MULTIPLEXING	< Multiplexing. Multiple transport sessions, belonging to the same or to a different stream, can share a common transport (L2CAP) channel.
	AVDTP_SEP_CAPABILITY_FLAG_RECOVERY	< Recovery. A SEP is capable of transferring recovery packets.
	AVDTP_SEP_CAPABILITY_FLAG_REPORTING	< Reporting. A SEP is capable of transferring reporting packets.
	AVDTP_SEP_SERVICE_CAPABILITY_CONTENT_PROTECTION	< Content Prortection. A SEP is capable of transferring content protection packets.
	AVDTP_SEP_SERVICE_CAPABILITY_HEADER_COMPRESSION	< Header Compression. A SEP can use header compression for transferring Media or Recovery packets.
	AVDTP_SEP_SERVICE_CAPABILITY_MEDIA_CODEC	< Media Codec. Defines which codec a SEP supports. A SEP can support only one codec.
	AVDTP_SEP_SERVICE_CAPABILITY_MEDIA_TRANSPORT	< Media. A SEP is capable of transferring media (audio, video or both) packets.
	AVDTP_SEP_SERVICE_CAPABILITY_MULTIPLEXING	< Multiplexing. Multiple transport sessions, belonging to the same or to a different stream, can share a common transport (L2CAP) channel.

	AVDTP_SEP_SERVICE_CAPABILITY_RECOVERY	< Recovery. A SEP is capable of transferring recovery packets.
	AVDTP_SEP_SERVICE_CAPABILITY_REPORTING	< Reporting. A SEP is capable of transferring reporting packets.
	AVDTP_SEP_STATE_FREE	This is macro AVDTP_SEP_STATE_FREE.
	AVDTP_SEP_STATE_IDLE	This is macro AVDTP_SEP_STATE_IDLE.
	AVDTP_SEP_TYPE_SINK	< Sink (usually a device like a headphones or BMW).
	AVDTP_SEP_TYPE_SOURCE	< Source (usually a device like a phone, desktop or laptop).
	AVDTP_STREAM_CLOSING_TRANSPORT_CHANNELS	< The stream is closing transport channels.
	AVDTP_STREAM_FLAG_LISTENING	This is macro AVDTP_STREAM_FLAG_LISTENING.
	AVDTP_STREAM_OPENING_TRANSPORT_CHANNELS	< The stream is opening transport channels.
	AVDTP_STREAM_STATE_ABORTING	< The stream is aborting. This means that all transport channels associated with the stream are being closed. < After they have been closed the stream goes to AVDTP_STREAM_STATE_IDLE state.
	AVDTP_STREAM_STATE_CLOSING	< The stream is closing. This means that all transport channels associated with the stream are being closed. < After they have been closed the stream goes to AVDTP_STREAM_STATE_IDLE state.
	AVDTP_STREAM_STATE_CONFIGURED	< The stream has been configured.
	AVDTP_STREAM_STATE_IDLE	< The stream is idle. This can mean two things. The stream specified by strm_handle does not exist < or the stream is closed.
	AVDTP_STREAM_STATE_OPEN	< The stream has been opened.
	AVDTP_STREAM_STATE_STREAMING	< The stream has been started. Depending on the local SEP type (source or sink) it means < that the stream can send or receive media packets.
	AVDTP_TC_OPCODE_CONNECT	This is macro AVDTP_TC_OPCODE_CONNECT.
	AVDTP_TC_OPCODE_DISCONNECT	This is macro AVDTP_TC_OPCODE_DISCONNECT.
	AVDTP_TC_STATUS_ERROR	This is macro AVDTP_TC_STATUS_ERROR.
	AVDTP_TC_STATUS_SUCCESS	This is macro AVDTP_TC_STATUS_SUCCESS.
	AVDTP_TRANSPORT_CHANNEL_TYPE_DEDICATED	This is macro AVDTP_TRANSPORT_CHANNEL_TYPE_DEDICATED.
	AVDTP_TRANSPORT_CHANNEL_TYPE_SHARED	This is macro AVDTP_TRANSPORT_CHANNEL_TYPE_SHARED.
	AVDTP_TRANSPORT_SESSION_TYPE_MEDIA	< Media (audio or video).
	AVDTP_TRANSPORT_SESSION_TYPE_RECOVERY	< Recovery (currently not supported).
	AVDTP_TRANSPORT_SESSION_TYPE_REPORTING	< Reporting (currently not supported).
	bt_avdtp_connect	brief Connect to a remote device. ingroup avdtp details This function opens a control channel connection to a remote device specified by the p remote_addr. If connection cannot be initiated for some reason, for example, there is not enough resources, it returns FALSE and not events are generated. Otherwise the result of an attempt to connect to the remote device is reported via the AVDTP callback. The events generated will either be AVDTP_EVT_CTRL_CHANNEL_CONNECTED or AVDTP_EVT_CTRL_CHANNEL_CONNECTION_FAILED. param mgr AVDTP manager. param remote_addr The address of a remote device. return li c TRUE if connection establishment has been started. li c FALSE... more

	bt_avdtp_connect_ex	brief Connect to a remote device. ingroup avdtp details This function opens a control channel connection to a remote device specified by the p remote_addr. If connection cannot be initiated for some reason, for example, there is not enough resources, it returns FALSE and not events are generated. Otherwise the result of an attempt to connect to the remote device is reported via the AVDTP callback. The events generated will either be AVDTP_EVT_CTRL_CHANNEL_CONNECTED or AVDTP_EVT_CTRL_CHANNEL_CONNECTION_FAILED. param mgr AVDTP manager. param remote_addr The address of a remote device. param acl_config ACL link configuration. This can be a combination of the following... more
	AVDTP_CMD_DELAYREPORT	This is macro AVDTP_CMD_DELAYREPORT.
	AVDTP_CMD_GET_ALL_CAPABILITIES	This is macro AVDTP_CMD_GET_ALL_CAPABILITIES.
	AVDTP_CONTENT_PROTECTION_METHOD_SCMS_T	This is macro AVDTP_CONTENT_PROTECTION_METHOD_SCMS_T.
	AVDTP_CTRL_MESSAGE_TYPE_GENERAL_REJECT	This is macro AVDTP_CTRL_MESSAGE_TYPE_GENERAL_REJECT.
	AVDTP_SCMS_T_CP_BIT	This is macro AVDTP_SCMS_T_CP_BIT.
	AVDTP_SCMS_T_L_BIT	This is macro AVDTP_SCMS_T_L_BIT.
	bt_avdtp_evt_delay_report_completed_t	brief Parameter to AVDTP_EVT_DELAYREPORT_COMPLETED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::delay_report_completed - when AVDTP received a response to a "delay report" request.
	bt_cp_header_t	This is type bt_cp_header_t.
	_bt_avdtp_evt_delay_report_completed_t	brief Parameter to AVDTP_EVT_DELAYREPORT_COMPLETED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::delay_report_completed - when AVDTP received a response to a "delay report" request.
	_bt_cp_header_s	This is record _bt_cp_header_s.
	_bt_avdtp_evt_ctrl_connection_failed_s	brief Parameter to AVDTP_EVT_CTRL_CONNECTION_FAILED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::ctrl_connection_failed - when a control channel between two devices has been established.
	_bt_avdtp_evt_set_stream_configuration_t	brief Parameter to AVDTP_EVT_SET_STREAM_CONFIGURATION event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::set_stream_configuration - when AVDTP received a "set stream configuration" request.
	AVDTP_EVT_DELAYREPORT_COMPLETED	< This event is generated when a local device received a response (either positive or negative) to a "delay report" request.
	AVDTP_EVT_SET_STREAM_CONFIGURATION	This is macro AVDTP_EVT_SET_STREAM_CONFIGURATION.
	AVDTP_SEP_CAPABILITY_FLAG_DELAY_REPORTING	< Delat reporitng.
	AVDTP_SEP_SERVICE_CAPABILITY_DELAY_REPORTING	< Delay Reporting.

	bt_avdtp_evt_ctrl_connection_failed_t	brief Parameter to AVDTP_EVT_CTRL_CONNECTION_FAILED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::ctrl_connection_failed - when a control channel between two devices has been established.
	bt_avdtp_evt_set_stream_configuration_t	brief Parameter to AVDTP_EVT_SET_STREAM_CONFIGURATION event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::set_stream_configuration - when AVDTP received a "set stream configuration" request.

AVDTP Functions

	Name	Description
	_bt_avdtp_add_param_byte	This is function _bt_avdtp_add_param_byte.
	_bt_avdtp_add_param_uint	This is function _bt_avdtp_add_param_uint.
	_bt_avdtp_allocate_buffers	This is function _bt_avdtp_allocate_buffers.
	_bt_avdtp_allocate_cmd	This is function _bt_avdtp_allocate_cmd.
	_bt_avdtp_allocate_sep	This is function _bt_avdtp_allocate_sep.
	_bt_avdtp_allocate_sep_config	This is function _bt_avdtp_allocate_sep_config.
	_bt_avdtp_allocate_stream	This is function _bt_avdtp_allocate_stream.
	_bt_avdtp_allocate_transport_channel	This is function _bt_avdtp_allocate_transport_channel.
	_bt_avdtp_allocate_transport_session_id	This is function _bt_avdtp_allocate_transport_session_id.
	_bt_avdtp_begin_tc_channel_operation	This is function _bt_avdtp_begin_tc_channel_operation.
	_bt_avdtp_commit_tc_channel_operation	This is function _bt_avdtp_commit_tc_channel_operation.
	_bt_avdtp_control_channel_accept_handler	This is function _bt_avdtp_control_channel_accept_handler.
	_bt_avdtp_control_channel_cmd_handler	This is function _bt_avdtp_control_channel_cmd_handler.
	_bt_avdtp_control_channel_event_handler	This is function _bt_avdtp_control_channel_event_handler.
	_bt_avdtp_control_channel_reject_handler	This is function _bt_avdtp_control_channel_reject_handler.
	_bt_avdtp_execute_tc_channel_operation	This is function _bt_avdtp_execute_tc_channel_operation.
	_bt_avdtp_find_listening_stream	This is function _bt_avdtp_find_listening_stream.
	_bt_avdtp_find_stream	This is function _bt_avdtp_find_stream.
	_bt_avdtp_find_stream_by_remote_sep_id	This is function _bt_avdtp_find_stream_by_remote_sep_id.
	_bt_avdtp_find_stream_by_sep_id	This is function _bt_avdtp_find_stream_by_sep_id.
	_bt_avdtp_free_cmd	This is function _bt_avdtp_free_cmd.
	_bt_avdtp_free_sep_config	This is function _bt_avdtp_free_sep_config.
	_bt_avdtp_free_stream	This is function _bt_avdtp_free_stream.
	_bt_avdtp_free_transport_channel	This is function _bt_avdtp_free_transport_channel.
	_bt_avdtp_get_control_channel	This is function _bt_avdtp_get_control_channel.
	_bt_avdtp_init_cmd_buffers	This is function _bt_avdtp_init_cmd_buffers.
	_bt_avdtp_init_sep_config_buffers	This is function _bt_avdtp_init_sep_config_buffers.
	_bt_avdtp_init_signal	This is function _bt_avdtp_init_signal.
	_bt_avdtp_is_sep_inuse	This is function _bt_avdtp_is_sep_inuse.
	_bt_avdtp_open_control_channel_ex	This is function _bt_avdtp_open_control_channel_ex.
	_bt_avdtp_open_control_channel_ex	This is function _bt_avdtp_open_control_channel_ex.
	_bt_avdtp_register_transport_channel_for_operation	This is function _bt_avdtp_register_transport_channel_for_operation.
	_bt_avdtp_send_command	This is function _bt_avdtp_send_command.
	_bt_avdtp_send_media_packet	This is function _bt_avdtp_send_media_packet.
	_bt_avdtp_set_signal	This is function _bt_avdtp_set_signal.
	_bt_avdtp_transport_l2cap_read_data_callback	This is function _bt_avdtp_transport_l2cap_read_data_callback.
	_bt_avdtp_transport_l2cap_state_changed_callback	This is function _bt_avdtp_transport_l2cap_state_changed_callback.
	_bt_avdtp_write_caps	This is function _bt_avdtp_write_caps.

	bt_avdtp_abort_stream	brief Suspend a stream. ingroup avdtp details This function tries to suspend a stream by sending a request to the remote party. The stream can be in any state state except AVDTP_STREAM_STATE_IDLE. As a result of this operation the AVDTP_EVT_ABORT_STREAM_COMPLETED event will be generated. This operation cannot be rejected. The p evt_param.abort_stream_requested.err_code is always == AVDTP_ERROR_SUCCESS. param mgr AVDTP manager. param strm_handle Stream handle. return li c TRUE if the function succeeds, i.e. the actual request has been sent to the remote party. li c FALSE otherwise. No events will be generated.
	bt_avdtp_add_media_rx_buffer	brief Add a media packet buffer to a receive queue ingroup avdtp details The consumer of AVDTP is responsible for allocating and supplying AVDTP with buffers used to store received packets. AVDTP itself only has a queue for storing pointers to buffers supplied by the consumer. When a packet comes in AVDTP finds the first buffer large enough to hold the received packet, copies the packet to the buffer and generates a AVDTP_EVT_MEDIA_PACKET_RECEIVED event. The consumer then has to process the data in the buffer and return it back to the queue. If there is no buffers in the queue... more
	bt_avdtp_add_media_tx_buffer	brief Add a media packet buffer to a send queue ingroup avdtp details When the consumer of AVDTP wants to send a packet to a remote device it calls this function. The function adds the packet to a queue and tells AVDTP that it has something to send. The packet will be send as soon as the stream goes to AVDTP_STREAM_STATE_STREAMING state. When the packet has been successfully sent a AVDTP_EVT_MEDIA_PACKET_SENT is generated. Otherwise a AVDTP_EVT_MEDIA_PACKET_SEND_FAILED is generated. Regardless of the event generated the consumer can re-use the buffer as AVDTP has removed it from the queue and gave up... more
	bt_avdtp_cancel_listen	brief Cancel listening for incoming connections. ingroup avdtp details This function removes a SEP from a list of SEPS which a stream can use for incoming requests. param mgr AVDTP manager. param strm_handle Stream handle. param sep_id Local SEP ID. return li c TRUE if the function succeeds. li c FALSE otherwise.
	bt_avdtp_close_stream	brief Close a stream. ingroup avdtp details This function tries to close a stream by sending a request to the remote party. The stream has to be in AVDTP_STREAM_STATE_OPEN or AVDTP_STREAM_STATE_STREAMING state. As a result of this operation the AVDTP_EVT_CLOSE_STREAM_COMPLETED event will be generated. If the stream has been closed the p evt_param.bt_avdtp_evt_close_stream_completed_t.err_code == AVDTP_ERROR_SUCCESS. Otherwise, if the remote device for any reason cannot or does not wish to close the stream, the p evt_param.bt_avdtp_evt_close_stream_completed_t.err_code == the error code sent by the remote. param mgr AVDTP manager. param strm_handle Stream handle. return li c TRUE if the function succeeds,... more
	bt_avdtp_create_stream	brief Create a stream. ingroup avdtp details This function allocates memory for storing stream's data and assigns a stream handle. The stream handle is used to manipulate the stream - open, close, configure, suspend, abort. param mgr AVDTP manager. return li c Stream handle if the function succeeds. li c 0 otherwise.
	bt_avdtp_destroy_stream	brief Destroy a stream. ingroup avdtp details This function frees memory used by the stream. The stream has to exist and be in the "idle" state for this function to succeed. I.e. the stream has to be closed or aborted before this function can be called. param mgr AVDTP manager. param strm_handle Stream handle. return li c TRUE if the function succeeds. li c FALSE otherwise.

	bt_avdtp_disconnect	brief Disconnect from a remote device. ingroup avdtp details This function closes a control and transport channels on all streams associated with the remote device specified by the p remote_addr. As a result of this operation the following events will be generated: @arg AVDTP_EVT_MEDIA_PACKET_RECEIVED: if a stream's receive queue is not empty this event is generated for each buffer with bt_media_packet_t::data_len set to 0 @arg AVDTP_EVT_MEDIA_PACKET_SENT: if a stream's send queue is not empty this event is generated for each buffer with bt_media_packet_t::data_len set to 0 @arg AVDTP_EVT_STREAM_CLOSED: this event is generate if a stream is in... more
	bt_avdtp_discover	brief Discover SEPs on a remote device. ingroup avdtp details This function asks the remote device to send a list of all available SEPs. As a result of this operation the following events will be generated: @arg AVDTP_EVT_SEP_INFO_RECEIVED: this event is generated for every SEP received from the remote device. the p evt_param.sep_info_received contains SEP information. @arg AVDTP_EVT_DISCOVER_COMPLETED: this event is generated after last AVDTP_EVT_SEP_INFO_RECEIVED if the remote accepted the request and the p evt_param.discover_completed.err_code == AVDTP_ERROR_SUCCESS. if the remote rejected the request the p evt_param.discover_completed.err_code == the error code sent by the remote. param mgr AVDTP... more
	bt_avdtp_find_codec	brief Find a codec ingroup avdtp details AVDTP in theory can support any type of codec. Each codec uses its own format for exchanging capabilities and configuration information. In order to make out implementation do not care about these formats we use a simple way of telling AVDTP how to parse and serialize codec's configuration. The consumer of AVDTP (e.g. A2DP) for each codec it wishes to support has to register a callback function (one per codec type). That callback has to perform two function. The first one is to read the configuration received from the remote device and store... more
	bt_avdtp_get_capabilities	brief Get remote SEP capabilities. ingroup avdtp details This function asks the remote device to send capabilities of a SEP specified by the p seid_acp. As a result of this operation the following events will be generated: @arg AVDTP_EVT_SEP_CAPABILITIES_RECEIVED: this event is generated if the remote device accepted the request. the p evt_param.sep_capabilities_received contains SEP capabilities. @arg AVDTP_EVT_GET_SEP_CAPABILITIES_COMPLETED: this event is generated right after AVDTP_EVT_SEP_CAPABILITIES_RECEIVED if the remote accepted the request the p evt_param.get_sep_capabilities_completed.err_code == AVDTP_ERROR_SUCCESS. if the remote rejected the request the p evt_param.get_sep_capabilities_completed.err_code == the error code sent by the remote. param mgr AVDTP manager.... more
	bt_avdtp_get_configuration	brief Get stream configuration. ingroup avdtp details This function requests stream configuration from a remote device. As a result of this operation the following events will be generated: @arg AVDTP_EVT_STREAM_CONFIGURATION_RECEIVED: this event is generated if the remote accepted the request. the p ebt_para.sep_capabilities_received.caps will contain current stream configuration. @arg AVDTP_EVT_GET_STREAM_CONFIGURATION_COMPLETED: If the remote accepted the request the p evt_param.get_stream_configuration_completed.err_code == AVDTP_ERROR_SUCCESS. if the remote rejected the request the p evt_param.get_stream_configuration_completed.err_code == the error code sent by the remote. param mgr AVDTP manager. param strm_handle Stream handle. return li c TRUE if the function succeeds, i.e. the actual... more

	bt_avdtp_get_hci_connection	brief Get HCI connection for a stream ingroup avdtp details This function returns a pointer to a structure that describes an HCI connection a stream is open on. The return value can be used to call various function from the HCI layer. For example, if an app wants to force disconnection from a remote device it can call bt_hci_disconnect. param mgr AVDTP manager. param strm_handle Stream handle. return li c Pointer to a structure that describes an HCI connection if the function succeeds. li c NULL otherwise. The function fails only if a stream specified by the p strm_handle... more
	bt_avdtp_get_mgr	brief Return a pointer to an instance of the AVDTP manager. ingroup avdtp details This function returns a pointer to an instance of the AVDTP manager. There is only one instance of the manager allocated by the stack. The pointer is passed as the first parameter to all AVDTP functions.
	bt_avdtp_get_sep	brief Get a SEP info by its ID. ingroup avdtp details This function returns a pointer to bt_avdtp_sep_t structure that describes a SEP previously registered with bt_avdtp_register_sep. param mgr AVDTP manager. param sep_id The ID of a SEP. return li c Pointer to bt_avdtp_sep_t if the SEP is in the list of registered SEPs. li c NULL otherwise.
	bt_avdtp_get_stream_codec_config	brief Get the configuration of the codec currently used with the stream. ingroup avdtp details This function returns a pointer to a structure that contains configuration of the codec currently used with the stream. The structure returned depends on the codec. The dotstack defines structures only for SBC, MPEG-1,2 and MPEG-2,4 AAC codecs: @arg SBC: bt_a2dp_sbc_config_t (defined in a2dp_sbc_codec.h) @arg MPEG-1,2: bt_a2dp_mpeg_config_t (defined in a2dp_mpeg_codec.h) @arg MPEG-2,4 AAC: bt_a2dp_aac_config_t (defined in a2dp_aac_codec.h) param mgr AVDTP manager. param strm_handle Stream handle. return li The codec's configuration if strm_handle specifies a valid stream and the stream is in one of the following... more
	bt_avdtp_get_stream_codec_type	brief Get the type of the codec currently used with the stream. ingroup avdtp details This function returns the type of the codec currently used with the stream. param mgr AVDTP manager. param strm_handle Stream handle. return @arg The type of the codec if strm_handle specifies a valid stream and the stream is in one of the following states: AVDTP_STREAM_STATE_CONFIGURED AVDTP_STREAM_STATE_OPEN AVDTP_STREAM_STATE_STREAMING @arg The result will be one of the following values: AVDTP_CODEC_TYPE_SBC: SBC AVDTP_CODEC_TYPE_MPEG1_2_AUDIO: MPEG-1,2 (used in MP3 files) AVDTP_CODEC_TYPE_MPEG2_4_AAC: MPEG-2,4 AAC (used in Apple products) AVDTP_CODEC_TYPE_ATRAC: ATRAC (used in Sony products) AVDTP_CODEC_TYPE_NON_A2DP: Non-A2DP... more
	bt_avdtp_get_stream_config	brief Get stream's configuration. ingroup avdtp details This function returns a pointer to a structure holding the current configuration of stream. param mgr AVDTP manager. param strm_handle Stream handle. return li The stream's configuration if strm_handle specifies a valid stream and the stream is in one of the following state: AVDTP_STREAM_STATE_CONFIGURED AVDTP_STREAM_STATE_OPEN AVDTP_STREAM_STATE_STREAMING li NULL otherwise.
	bt_avdtp_get_stream_local_sep_id	brief Get stream's local SEP ID. ingroup avdtp details This function returns the ID of the local SEP associated with the stream. param mgr AVDTP manager. param strm_handle Stream handle. return li The ID of the local SEP if strm_handle specifies a valid stream. li 0 otherwise.
	bt_avdtp_get_stream_remote_address	brief Get stream's remote BT address. ingroup avdtp details This function returns the address of the remote device associated with the stream. param mgr AVDTP manager. param strm_handle Stream handle. return li The address of the remote device if strm_handle specifies a valid stream. li NULL otherwise.

	bt_avdtp_get_stream_remote_sep_id	brief Get stream's remote SEP ID. ingroup avdtp details This function returns the ID of the remote SEP associated with the stream. param mgr AVDTP manager. param strm_handle Stream handle. return li The ID of the remote SEP if strm_handle specifies a valid stream. li 0 otherwise.
	bt_avdtp_get_stream_state	brief Get local stream state. ingroup avdtp details This function returns local state of a stream specified by the p strm_handle. No request is sent to the remote party. param mgr AVDTP manager. param strm_handle Stream handle. return The state of the stream. The result will be one of the following values: @arg AVDTP_STREAM_STATE_IDLE: The stream is idle. This can mean two things. The stream specified by p strm_handle does not exist or the stream is closed. @arg AVDTP_STREAM_OPENING_TRANSPORT_CHANNELS: The stream is opening transport channels. @arg AVDTP_STREAM_CLOSING_TRANSPORT_CHANNELS: The stream is closing transport channels. @arg AVDTP_STREAM_STATE_CONFIGURED: The... more
	bt_avdtp_init	brief Initialize the AVDTP layer. ingroup avdtp details This function initializes the AVDTP layer of the stack. It must be called prior to any other AVDTP function can be called.
	bt_avdtp_listen	brief Listen for incoming connections. ingroup avdtp details This function tells a stream that it can use a particular SEP to accept incoming requests to open it. The SEP can be associated with multiple streams but used with only one. The stream has to be closed before the SEP can be used with another stream. For outgoing connections this is not needed. Any SEP can be used with any stream given that the SEP is not already in use by another stream. param mgr AVDTP manager. param strm_handle Stream handle. param sep_id Local SEP ID. return li c TRUE if... more
	bt_avdtp_open_stream	brief Open a stream. ingroup avdtp details This function tries to open a stream by sending a request to the remote party. The stream has to be already configured with a bt_avdtp_set_configuration call. As a result of this operation the AVDTP_EVT_OPEN_STREAM_COMPLETED event will be generated. If the stream has been open the p evt_param.open_stream_completed.err_code == AVDTP_ERROR_SUCCESS. Otherwise, if the remote device for any reason cannot or does not wish to open the stream, the p evt_param.open_stream_completed.err_code == the error code sent by the remote. param mgr AVDTP manager. param strm_handle Stream handle. return li c TRUE if the function... more
	bt_avdtp_reconfigure_stream	brief Reconfigure stream. ingroup avdtp details This function tries to change the stream's configuration. For this function to succeed the stream has to be open. As a result of this operation the AVDTP_EVT_STREAM_RECONFIGURE_COMPLETED event will be generated. If reconfiguration was a success the p evt_param.stream_reconfigure_completed.err_code == AVDTP_ERROR_SUCCESS. Otherwise the p evt_param.stream_reconfigure_completed.err_code == the error code sent by the remote. param mgr AVDTP manager. param strm_handle Stream handle. param caps New stream configuration. return li c TRUE if the function succeeds, i.e. the actual request has been sent to the remote party. li c FALSE otherwise. No events will be... more
	bt_avdtp_register_callback	brief Register a AVDTP application callback. ingroup avdtp details In order to be notified of various events a consumer of the AVDTP layer has to register a callback function. The stack will call this function whenever a new event has been generated passing the code of the event as the second parameter. The event can be one of the following values: @arg AVDTP_EVT_CTRL_CHANNEL_CONNECTED: Control channel connected. @arg AVDTP_EVT_CTRL_CHANNEL_DISCONNECTED: Control channel disconnected. @arg AVDTP_EVT_CTRL_CONNECTION_FAILED: Control channel connection failed (generated only if control connection has been initiated by the local device). @arg AVDTP_EVT_DISCOVER_COMPLETED: Local device completed discovering remote... more

	bt_avdtp_register_codec	brief Register a codec ingroup avdtp details AVDTP in theory can support any type of codec. Each codec uses its own format for exchanging capabilities and configuration information. In order to make out implementation do not care about these formats we use a simple way of telling AVDTP how to parse and serialize codec's configuration. The consumer of AVDTP (e.g. A2DP) for each codec it wishes to support has to register a callback function (one per codec type). That callback has to perform two function. The first one is to read the configuration received from the remote device and store... more
	bt_avdtp_register_sep	brief Register a SEP with the local AVDTP manager. ingroup avdtp details This function is used to make a list of SEPs supported by the local ADVTP entity. param mgr AVDTP manager. param type The type of a SEP. The type can be one of the following values: @arg AVDTP_SEP_TYPE_SOURCE: The SEP is a source. @arg AVDTP_SEP_TYPE_SINK: The SEP is a sink. param caps The capabilities of a SEP. return li c ID of a SEP if the function succeeds. li c FALSE otherwise.
	bt_avdtp_remove_media_rx_buffer	brief Remove a media packet buffer from a receive queue ingroup avdtp details The consumer of AVDTP is responsible for allocating and supplying AVDTP with buffers used to store received packets. AVDTP itself only has a queue for storing pointers to buffers supplied by the consumer. When a packet comes in AVDTP finds the first buffer large enough to hold the received packet, copies the packet to the buffer and generates a AVDTP_EVT_MEDIA_PACKET_RECEIVED event. The consumer then has to process the data in the buffer and return it back to the queue. If there is no buffers in the queue... more
	bt_avdtp_remove_media_tx_buffer	brief Remove a media packet buffer from a send queue ingroup avdtp details When the consumer of AVDTP wants to send a packet to a remote device it calls bt_avdtp_add_media_tx_buffer function. The function adds the packet to a queue and tells AVDTP that it has something to send. The packet will be send as soon as the stream goes to AVDTP_STREAM_STATE_STREAMING state. The consumer has a chance to remove a packet from the queue before it has been sent to a remote device by calling bt_avdtp_remove_media_tx_buffer. param mgr AVDTP manager. param strm_handle Stream handle. param buffer Pointer to a structure... more
	bt_avdtp_security_control	brief Exchange content protection control data. ingroup avdtp details This function tries to establish content protection by sending a request to the remote party. The stream can be in any state state except AVDTP_STREAM_STATE_IDLE, AVDTP_STREAM_STATE_CLOSING, AVDTP_STREAM_STATE_ABORTING. As a result of this operation the AVDTP_EVT_STREAM_SECURITY_CONTROL_COMPLETED event will be generated. If the stream's content protection data has been accepted by the remote party the p evt_param.security_control_completed.err_code == AVDTP_ERROR_SUCCESS. Otherwise the p evt_param.security_control_completed.err_code == the error code sent by the remote. note The dotstack does not support content protection. Although the request can be sent it will not affect... more
	bt_avdtp_set_configuration	brief Set stream configuration. ingroup avdtp details This function tries to configure a stream before opening it. As a result of this operation the AVDTP_EVT_SET_STREAM_CONFIGURATION_COMPLETED event will be generated. If configuration was a success the p evt_param.set_stream_configuration_completed.err_code == AVDTP_ERROR_SUCCESS. Otherwise the p evt_param.set_stream_configuration_completed.err_code == the error code sent by the remote and p evt_param.set_stream_configuration_completed.svc_category == the value of the first Service Category to fail. param mgr AVDTP manager. param strm_handle Stream handle. param remote_addr The address of a remote device. param seid_int Local SEP ID. param seid_acp Remote SEP ID. param caps Stream configuration. return li c TRUE if... more
	bt_avdtp_start	brief Start the AVDTP layer. ingroup avdtp details This function makes the AVDTP layer ready to accept connection requests from remote device. To make an outgoing connection calling this function is not required. param mgr AVDTP manager. return li c TRUE if the function succeeds. li c FALSE otherwise.

	bt_avdtp_start_stream	brief Start a stream. ingroup avdtp details This function tries to start a stream by sending a request to the remote party. The stream has to be in AVDTP_STREAM_STATE_OPEN state. The stream goes to this state as a result of successful configuration or suspension (both can be initiated by either party). As a result of this operation the AVDTP_EVT_START_STREAM_COMPLETED event will be generated. If the stream has been open the p evt_param.start_stream_requested.err_code == AVDTP_ERROR_SUCCESS. Otherwise, if the remote device for any reason cannot or does not wish to start the stream, the p evt_param.start_stream_requested.err_code == the error code sent... more
	bt_avdtp_suspend_stream	brief Suspend a stream. ingroup avdtp details This function tries to suspend a stream by sending a request to the remote party. The stream has to be in AVDTP_STREAM_STATE_STREAMING state. As a result of this operation the AVDTP_EVT_SUSPEND_STREAM_COMPLETED event will be generated. If the stream has been suspended the p evt_param.bt_avdtp_evt_suspend_stream_requested_t.err_code == AVDTP_ERROR_SUCCESS. Otherwise, if the remote device for any reason cannot or does not wish to suspend the stream, the p evt_param.bt_avdtp_evt_suspend_stream_requested_t.err_code == the error code sent by the remote. param mgr AVDTP manager. param strm_handle Stream handle. return li c TRUE if the function succeeds, i.e. the... more
	bt_avdtp_unregister_codec	brief Unregister a codec ingroup avdtp details AVDTP in theory can support any type of codec. Each codec uses its own format for exchanging capabilities and configuration information. In order to make out implementation do not care about these formats we use a simple way of telling AVDTP how to parse and serialize codec's configuration. The consumer of AVDTP (e.g. A2DP) for each codec it wishes to support has to register a callback function (one per codec type). That callback has to perform two function. The first one is to read the configuration received from the remote device and store... more
	_bt_avdtp_add_param_uintn	This is function _bt_avdtp_add_param_uintn.
	_bt_avdtp_read_caps	This is function _bt_avdtp_read_caps.
	bt_avdtp_get_l2cap_channel	This is function bt_avdtp_get_l2cap_channel.
	bt_avdtp_clear_media_tx_queue	brief Clear send queue ingroup avdtp details When the consumer of AVDTP wants to send a packet to a remote device it calls bt_avdtp_add_media_tx_buffer function. The function adds the packet to a queue and tells AVDTP that it has something to send. The packet will be send as soon as the stream goes to AVDTP_STREAM_STATE_STREAMING state. The consumer can remove all packets from the queue before they have been sent to a remote device by calling ::bt_avdtp_clear_media_tx_queue. param mgr AVDTP manager. param strm_handle Stream handle. return li c TRUE if the function succeeds. li c FALSE otherwise. The function fails... more
	bt_avdtp_get_all_capabilities	brief Get remote SEP capabilities. ingroup avdtp details This function asks the remote device to send capabilities of a SEP specified by the p seid_acp. As a result of this operation the following events will be generated: @arg AVDTP_EVT_SEP_CAPABILITIES_RECEIVED: this event is generated if the remote device accepted the request. the p evt_param.sep_capabilities_received contains SEP capabilities. @arg AVDTP_EVT_GET_SEP_CAPABILITIES_COMPLETED: this event is generated right after AVDTP_EVT_SEP_CAPABILITIES_RECEIVED if the remote accepted the request the p evt_param.get_sep_capabilities_completed.err_code == AVDTP_ERROR_SUCCESS. if the remote rejected the request the p evt_param.get_sep_capabilities_completed.err_code == the error code sent by the remote. param mgr AVDTP manager.... more
	bt_avdtp_get_stream_direction	This is function bt_avdtp_get_stream_direction.

	bt_avdtp_report_delay	brief Report delay value of a Sink to a Source. ingroup avdtp details This function sends the delay value of a Sink to a Source. This enables synchronous playback of audio and video. Delay reports are always sent from the Sink to the Source. If the Sink's delay report has been accepted by the Source the p evt_param.delay_report_completed.err_code == AVDTP_ERROR_SUCCESS. Otherwise the p evt_param.delay_report_completed.err_code == the error code sent by the Source. param mgr AVDTP manager. param strm_handle Stream handle. param delay The delay value in 1/10 milliseconds. return li c TRUE if the function succeeds, i.e. the actual... more
	bt_avdtp_set_media_tx_queue_limit	brief Set limit on the send queue ingroup avdtp details When the consumer of AVDTP wants to send a packet to a remote device it calls bt_avdtp_add_media_tx_buffer function. The function adds the packet to a queue and tells AVDTP that it has something to send. The packet will be send as soon as the stream goes to AVDTP_STREAM_STATE_STREAMING state. By default the send queue can contain unlimited number of packets. The consumer can set a limit on how many packets are held in the queue. In this case when new packet is added to the queue and the length of... more

AVRCP Data Types and Constants

	Name	Description
	_bt_av_add_to_now_playing_s	brief Parameter to AVRCP_EVT_ADD_TO_NOW_PLAYING_COMPLETED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::add_to_now_playing_status - when a local device received a response to a "add to now playing" request.
	_bt_av_battery_status_of_ct_s	brief Parameter to AVRCP_EVT_BATTERY_STATUS_OF_CT_RECEIVED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::battery_status_of_ct - when a local device received a "battery status of controller" command.
	_bt_av_capability_company_id_s	brief Parameter to AVRCP_EVT_COMPANY_ID_LIST_RECEIVED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::company_id - when a local device received a response to a "get company id" request.
	_bt_av_capability_event_id_s	brief Parameter to AVRCP_EVT_EVENT_ID_LIST_RECEIVED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::supported_event_id - when a local device received a response to a "get supported events" request.
	_bt_av_command_t	This is record _bt_av_command_t.

	_bt_av_displayable_character_set_s	brief Parameter to AVRCP_EVT_DISPLAYABLE_CHARACTER_SET_RECEIVED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::displayable_character_set - when a local device received a "displayable chracter set command" request.
	_bt_av_element_attribute_s	brief Media element attribute ingroup avrcp details This structure is used to store media element attribute.
	_bt_av_element_attributes_s	brief Parameter to AVRCP_EVT_GET_ELEMENT_ATTRIBUTES_RECEIVED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::element_attributes - when a local device received a response to a "get media element attributes" request.
	_bt_av_element_id_s	brief Media element UID ingroup avrcp details This structure is used to store media element UID.
	_bt_av_get_element_attributes_s	brief Parameter to AVRCP_EVT_ELEMENT_ATTRIBUTES_REQUESTED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::get_element_attributes - when a local device received a "get element attributes" request.
	_bt_av_notification_addressed_player_changed_s	brief Parameter to AVRCP_EVT_ADDRESSED_PLAYER_CHANGED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::notification::params:addressed_player - when a local device received a "addressed player changed" notification.
	_bt_av_notification_app_setting_changed_s	This is type bt_av_notification_app_setting_changed_t.
	_bt_av_notification_battery_status_s	brief Parameter to AVRCP_EVT_BATT_STATUS_CHANGED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::notification::params:battery_status - when a local device received a "battery status changed" notification.
	_bt_av_notification_playback_pos_changed_s	brief Parameter to AVRCP_EVT_PLAYBACK_POS_CHANGED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::notification::params:playback_pos - when a local device received a "playback position changed" notification.

	_bt_av_notification_playback_status_changed_s	brief Parameter to AVRCP_EVT_PLAYBACK_STATUS_CHANGED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::notification::params:play_status - when a local device received a "play status changed" notification.
	_bt_av_notification_s	brief Parameter to the following events: li AVRCP_EVT_PLAYBACK_STATUS_CHANGED li AVRCP_EVT_TRACK_CHANGED li AVRCP_EVT_PLAYBACK_POS_CHANGED li AVRCP_EVT_BATT_STATUS_CHANGED li AVRCP_EVT_SYSTEM_STATUS_CHANGED li AVRCP_EVT_NOW_PLAYING_CONTENT_CHANGED li AVRCP_EVT_AVAILABLE_PLAYERS_CHANGED li AVRCP_EVT_ADDRESSED_PLAYER_CHANGED li AVRCP_EVT_UIDS_CHANGED li AVRCP_EVT_VOLUME_CHANGED ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::notification - when a local device received one of the following notifications from the target: li Play status changed li Track changed changed li Playback position changed li Battery status changed li System status changed li Addressed player changed li UIDs changed li Volume changed li Player application setting changed... more
	_bt_av_notification_system_status_changed_s	brief Parameter to AVRCP_EVT_SYSTEM_STATUS_CHANGED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::notification::params:system_status - when a local device received a "system status changed" notification.
	_bt_av_notification_track_changed_s	brief Parameter to AVRCP_EVT_TRACK_CHANGED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::notification::params:track - when a local device received a "track changed" notification.
	_bt_av_notification_uids_changed_s	brief Parameter to AVRCP_EVT_UIDS_CHANGED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::notification::params:uids - when a local device received a "UIDs changed" notification.
	_bt_av_notification_volume_changed_s	brief Parameter to AVRCP_EVT_VOLUME_CHANGED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::notification::params:volume - when a local device received a "UIDs changed" notification.

	_bt_av_play_item_s	brief Parameter to AVRCP_EVT_PLAY_ITEM_COMPLETED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::play_item_status - when a local device received a response to a "play item" request.
	_bt_av_play_status_s	brief Parameter to AVRCP_EVT_GET_PLAY_STATUS_RECEIVED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::play_status - when a local device received a response to a "get play status" request.
	_bt_av_player_setting_current_values_s	brief Parameter to AVRCP_EVT_PLAYER_CURRENT_SETTING_VALUES_RECEIVED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::player_setting_current_values - when a local device received a response to a "get current player setting attribute values" request.
	_bt_av_player_setting_values_s	brief Parameter to AVRCP_EVT_PLAYER_SETTING_VALUES_RECEIVED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::player_setting_values - when a local device received a response to a "get player setting attribute values" request.
	_bt_av_player_setting_values_text_s	brief Parameter to AVRCP_EVT_PLAYER_SETTING_VALUES_TEXT_RECEIVED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::player_setting_values_text - when a local device received a response to a "get player setting attribute values displayable text" request.
	_bt_av_player_settings_s	brief Parameter to AVRCP_EVT_PLAYER_SETTING_ATTRIBUTES_RECEIVED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::player_settings - when a local device received a response to a "get supported player setting attributes" request.
	_bt_av_player_settings_text_s	brief Parameter to AVRCP_EVT_PLAYER_SETTING_ATTRIBUTES_TEXT_RECEIVED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::player_settings_text - when a local device received a response to a "get player setting attributes displayable text" request.
	_bt_av_player_text_s	This is type bt_av_player_text_t.

	_bt_av_register_notification_t	brief Parameter to AVRCP_EVT_REGISTER_NOTIFICATION_REQUESTED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::register_notification - when a local device received a "register notification" request.
	_bt_av_response_t	brief AV/C response header ingroup avrcp details This structure is used to store fields present in every AV/C response.
	_bt_av_set_absolute_volume_s	brief Parameter to AVRCP_EVT_SET_ABSOLUTE_VOLUME_COMPLETED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::absolute_volume - when a local device received a response to a "set absolute volume" request.
	_bt_av_set_addressed_player_s	brief Parameter to AVRCP_EVT_SET_ADDRESSED_PLAYER_COMPLETED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::addressed_player - when a local device received a response to a "set addressed player" request.
	_bt_avrcp_channel_t	brief AVRCP channel description ingroup avrcp details This structure is used to hold information about an AVRCP channel.
	_bt_avrcp_device_s	This is type bt_avrcp_device_t.
	_bt_avrcp_event_u	brief Parameter to an application callback. ingroup avrcp details This union is used to pass event specific data to the AVRCP consumer. Which member of the union points to a valid structure depends on the event reported to the consumer. In general, each event has a corresponding member in the union.
	_bt_avrcp_evt_channel_connected_t	brief Parameter to AVRCP_EVT_CONTROL_CHANNEL_CONNECTED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::channel_connected - when a control channel between two devices has been established.
	_bt_avrcp_evt_channel_disconnected_t	brief Parameter to AVRCP_EVT_CONTROL_CHANNEL_DISCONNECTED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::channel_disconnected - when a control channel between two devices has been terminated.

	_bt_avrcp_evt_connection_failed_t	brief Parameter to AVRCP_EVT_CONTROL_CONNECTION_FAILED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::connection_failed - when a local device failed to create a control channel between two AVRCP entities.
	_bt_avrcp_evt_panel_command_received_t	brief Parameter to AVRCP_EVT_PANEL_COMMAND_RECEIVED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::panel_command_received - when a local device received a PASS THROUGH command.
	_bt_avrcp_evt_panel_response_received_t	brief Parameter to AVRCP_EVT_PANEL_RESPONSE_RECEIVED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::panel_response_received - when a local device received a response to a PASS THROUGH command.
	_bt_avrcp_evt_register_events_completed_t	This is type bt_avrcp_evt_register_events_completed_t.
	_bt_avrcp_evt_search_completed_s	brief Parameter to AVRCP_EVT_SEARCH_COMPLETED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::device_search - when searching for nearby devices has finished.
	_bt_avrcp_mgr_t	brief AVRCP manager. ingroup avrcp details A structure that glues all pieces together. There is only one instance of this structure allocated by dotstack. A pointer to the instance can be get with c bt_avrcp_get_mgr().
	bt_av_add_to_now_playing_t	brief Parameter to AVRCP_EVT_ADD_TO_NOW_PLAYING_COMPLETED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::add_to_now_playing_status - when a local device received a response to a "add to now playing" request.
	bt_av_battery_status_of_ct_t	brief Parameter to AVRCP_EVT_BATTERY_STATUS_OF_CT_RECEIVED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::battery_status_of_ct - when a local device received a "battery status of controller" command.
	bt_av_capability_company_id_t	brief Parameter to AVRCP_EVT_COMPANY_ID_LIST_RECEIVED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::company_id - when a local device received a response to a "get company id" request.

	bt_av_capability_event_id_t	brief Parameter to AVRCP_EVT_EVENT_ID_LIST_RECEIVED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::supported_event_id - when a local device received a response to a "get supported events" request.
	bt_av_command_t	This is type bt_av_command_t.
	bt_av_displayable_character_set_t	brief Parameter to AVRCP_EVT_DISPLAYABLE_CHARACTER_SET_RECEIVED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::displayable_character_set - when a local device received a "displayable chracter set command" request.
	bt_av_element_attribute_t	brief Media element attribute ingroup avrcp details This structure is used to store media element attribute.
	bt_av_element_attributes_t	brief Parameter to AVRCP_EVT_GET_ELEMENT_ATTRIBUTES_RECEIVED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::element_attributes - when a local device received a response to a "get media element attributes" request.
	bt_av_element_id_t	brief Media element UID ingroup avrcp details This structure is used to store media element UID.
	bt_av_get_element_attributes_t	brief Parameter to AVRCP_EVT_ELEMENT_ATTRIBUTES_REQUESTED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::get_element_attributes - when a local device received a "get element attributes" request.
	bt_av_notification_addressed_player_changed_t	brief Parameter to AVRCP_EVT_ADDRESSED_PLAYER_CHANGED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::notification::params:addressed_player - when a local device received a "addressed player changed" notification.
	bt_av_notification_app_setting_changed_t	This is type bt_av_notification_app_setting_changed_t.
	bt_av_notification_battery_status_t	brief Parameter to AVRCP_EVT_BATT_STATUS_CHANGED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::notification::params:battery_status - when a local device received a "battery status changed" notification.

	bt_av_notification_playback_pos_changed_t	brief Parameter to AVRCP_EVT_PLAYBACK_POS_CHANGED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::notification::params:playback_pos - when a local device received a "playback position changed" notification.
	bt_av_notification_playback_status_changed_t	brief Parameter to AVRCP_EVT_PLAYBACK_STATUS_CHANGED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::notification::params:play_status - when a local device received a "play status changed" notification.
	bt_av_notification_system_status_changed_t	brief Parameter to AVRCP_EVT_SYSTEM_STATUS_CHANGED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::notification::params:system_status - when a local device received a "system status changed" notification.
	bt_av_notification_t	brief Parameter to the following events: li AVRCP_EVT_PLAYBACK_STATUS_CHANGED li AVRCP_EVT_TRACK_CHANGED li AVRCP_EVT_PLAYBACK_POS_CHANGED li AVRCP_EVT_BATT_STATUS_CHANGED li AVRCP_EVT_SYSTEM_STATUS_CHANGED li AVRCP_EVT_NOW_PLAYING_CONTENT_CHANGED li AVRCP_EVT_AVAILABLE_PLAYERS_CHANGED li AVRCP_EVT_ADDRESSED_PLAYER_CHANGED li AVRCP_EVT_UIDS_CHANGED li AVRCP_EVT_VOLUME_CHANGED ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::notification - when a local device received one of the following notifications from the target: li Play status changed li Track changed changed li Playback position changed li Battery status changed li System status changed li Addressed player changed li UIDs changed li Volume changed li Player application setting changed... more
	bt_av_notification_track_changed_t	brief Parameter to AVRCP_EVT_TRACK_CHANGED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::notification::params:track - when a local device received a "track changed" notification.
	bt_av_notification_uids_changed_t	brief Parameter to AVRCP_EVT_UIDS_CHANGED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::notification::params:uids - when a local device received a "UIDs changed" notification.

	bt_av_notification_volume_changed_t	brief Parameter to AVRCP_EVT_VOLUME_CHANGED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::notification::params:volume - when a local device received a "UIDs changed" notification.
	bt_av_play_item_t	brief Parameter to AVRCP_EVT_PLAY_ITEM_COMPLETED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::play_item_status - when a local device received a response to a "play item" request.
	bt_av_play_status_t	brief Parameter to AVRCP_EVT_GET_PLAY_STATUS_RECEIVED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::play_status - when a local device received a response to a "get play status" request.
	bt_av_player_setting_current_values_t	brief Parameter to AVRCP_EVT_PLAYER_CURRENT_SETTING_VALUES_RECEIVED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::player_setting_current_values - when a local device received a response to a "get current player setting attribute values" request.
	bt_av_player_setting_values_t	brief Parameter to AVRCP_EVT_PLAYER_SETTING_VALUES_RECEIVED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::player_setting_values - when a local device received a response to a "get player setting attribute values" request.
	bt_av_player_setting_values_text_t	brief Parameter to AVRCP_EVT_PLAYER_SETTING_VALUES_TEXT_RECEIVED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::player_setting_values_text - when a local device received a response to a "get player setting attribute values displayable text" request.
	bt_av_player_settings_t	brief Parameter to AVRCP_EVT_PLAYER_SETTING_ATTRIBUTES_RECEIVED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::player_settings - when a local device received a response to a "get supported player setting attributes" request.

	bt_av_player_settings_text_t	brief Parameter to AVRCP_EVT_PLAYER_SETTING_ATTRIBUTES_TEXT_RECEIVED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::player_settings_text - when a local device received a response to a "get player setting attributes displayable text" request.
	bt_av_player_text_t	This is type bt_av_player_text_t.
	bt_av_register_notification_t	brief Parameter to AVRCP_EVT_REGISTER_NOTIFICATION_REQUESTED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::register_notification - when a local device received a "register notification" request.
	bt_av_response_t	brief AV/C response header ingroup avrcp details This structure is used to store fields present in every AV/C response.
	bt_av_set_absolute_volume_t	brief Parameter to AVRCP_EVT_SET_ABSOLUTE_VOLUME_COMPLETED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::absolute_volume - when a local device received a response to a "set absolute volume" request.
	bt_av_set_addressed_player_t	brief Parameter to AVRCP_EVT_SET_ADDRESSED_PLAYER_COMPLETED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::addressed_player - when a local device received a response to a "set addressed player" request.
	bt_avrcp_channel_t	This is type bt_avrcp_channel_t.
	bt_avrcp_device_t	This is type bt_avrcp_device_t.
	bt_avrcp_event_t	brief Parameter to an application callback. ingroup avrcp details This union is used to pass event specific data to the AVRCP consumer. Which member of the union points to a valid structure depends on the event reported to the consumer. In general, each event has a corresponding member in the union.
	bt_avrcp_evt_channel_connected_t	brief Parameter to AVRCP_EVT_CONTROL_CHANNEL_CONNECTED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::channel_connected - when a control channel between two devices has been established.

	bt_avrcp_evt_channel_disconnected_t	brief Parameter to AVRCP_EVT_CONTROL_CHANNEL_DISCONNECTED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::channel_disconnected - when a control channel between two devices has been terminated.
	bt_avrcp_evt_connection_failed_t	brief Parameter to AVRCP_EVT_CONTROL_CONNECTION_FAILED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::connection_failed - when a local device failed to create a control channel between two AVRCP entities.
	bt_avrcp_evt_panel_command_received_t	brief Parameter to AVRCP_EVT_PANEL_COMMAND_RECEIVED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::panel_command_received - when a local device received a PASS THROUGH command.
	bt_avrcp_evt_panel_response_received_t	brief Parameter to AVRCP_EVT_PANEL_RESPONSE_RECEIVED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::panel_response_received - when a local device received a response to a PASS THROUGH command.
	bt_avrcp_evt_register_events_completed_t	This is type bt_avrcp_evt_register_events_completed_t.
	bt_avrcp_evt_search_completed_t	brief Parameter to AVRCP_EVT_SEARCH_COMPLETED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::device_search - when searching for nearby devices has finished.
	bt_avrcp_find_callback_fp	brief Find Controller/Target callback. ingroup avrcp details This callback is called when search for Controller/Target has finished. param found This can be one of the following values: li c TRUE if Controller/Target was found. li c FALSE otherwise. param callback_param A pointer to an arbitrary data set by a call to bt_avrcp_find_target/bt_avrcp_find_controller.

	bt_avrcp_mgr_callback_fp	brief AVRCP application callback. ingroup avrcp details In order to be notified of various events a consumer of the AVRCP layer has to register a callback function (done with bt_avrcp_start()). The stack will call that function whenever a new event has been generated. param mgr AVRCP manager. param evt AVRCP event. The event can be one of the following values: @arg AVRCP_EVT_CONTROL_CHANNEL_CONNECTED This event is generated when a control channel between two AVRCP entities has been established. @arg AVRCP_EVT_CONTROL_CHANNEL_DISCONNECTED This event is generated when a control channel between two AVRCP entities has been terminated. @arg AVRCP_EVT_CONTROL_CONNECTION_FAILED This event is... more
	bt_avrcp_mgr_t	This is type bt_avrcp_mgr_t.
	__AVRCP_COMMAND_H	This is macro __AVRCP_COMMAND_H.
	__AVRCP_H	This is macro __AVRCP_H.
	__AVRCP_PRIVATE_H	This is macro __AVRCP_PRIVATE_H.
	AVC_BATTERY_STATUS_CRITICAL	This is macro AVC_BATTERY_STATUS_CRITICAL.
	AVC_BATTERY_STATUS_EXTERNAL	This is macro AVC_BATTERY_STATUS_EXTERNAL.
	AVC_BATTERY_STATUS_FULL_CHARGE	This is macro AVC_BATTERY_STATUS_FULL_CHARGE.
	AVC_BATTERY_STATUS_NORMAL	• addtogroup avrcp • @{ • @name Battery status
	AVC_BATTERY_STATUS_WARNING	This is macro AVC_BATTERY_STATUS_WARNING.
	AVC_CAPABILITY_COMPANY_ID	This is macro AVC_CAPABILITY_COMPANY_ID.
	AVC_CAPABILITY_EVENTS_SUPPORTED	This is macro AVC_CAPABILITY_EVENTS_SUPPORTED.
	AVC_CMD_GENERAL_POWER	This is macro AVC_CMD_GENERAL_POWER.
	AVC_CMD_GENERAL_RESERVE	This is macro AVC_CMD_GENERAL_RESERVE.
	AVC_CMD_GENERAL_SUBUNIT_INFO	This is macro AVC_CMD_GENERAL_SUBUNIT_INFO.
	AVC_CMD_GENERAL_UNIT_INFO	This is macro AVC_CMD_GENERAL_UNIT_INFO.
	AVC_CMD_GENERAL_VENDOR_DEPENDENT	This is macro AVC_CMD_GENERAL_VENDOR_DEPENDENT.
	AVC_CMD_GENERAL_VERSION	This is macro AVC_CMD_GENERAL_VERSION.
	AVC_CMD_PANEL_PASS_THROUGH	This is macro AVC_CMD_PANEL_PASS_THROUGH.
	AVC_CTYPE_CONTROL	• addtogroup avrcp • @{ • @name Command types
	AVC_CTYPE_GENERAL_INQUORY	This is macro AVC_CTYPE_GENERAL_INQUORY.
	AVC_CTYPE_NOTIFY	This is macro AVC_CTYPE_NOTIFY.
	AVC_CTYPE_SPECIFIC_IQUIRY	This is macro AVC_CTYPE_SPECIFIC_IQUIRY.
	AVC_CTYPE_STATUS	This is macro AVC_CTYPE_STATUS.
	AVC_EVENT_ADDRESSED_PLAYER_CHANGED	< The Addressed Player has been changed, see 6.9.2.
	AVC_EVENT_AVAILABLE_PLAYERS_CHANGED	< The available players have changed, see 6.9
	AVC_EVENT_BATT_STATUS_CHANGED	< Change in battery status
	AVC_EVENT_FLAG_ADDRESSED_PLAYER_CHANGED	This is macro AVC_EVENT_FLAG_ADDRESSED_PLAYER_CHANGED.

	AVC_EVENT_FLAG_ALL	This is macro AVC_EVENT_FLAG_ALL.
	AVC_EVENT_FLAG_AVAILABLE_PLAYERS_CHANGED	This is macro AVC_EVENT_FLAG_AVAILABLE_PLAYERS_CHANGED.
	AVC_EVENT_FLAG_BATT_STATUS_CHANGED	This is macro AVC_EVENT_FLAG_BATT_STATUS_CHANGED.
	AVC_EVENT_FLAG_NOW_PLAYING_CONTENT_CHANGED	This is macro AVC_EVENT_FLAG_NOW_PLAYING_CONTENT_CHANGED.
	AVC_EVENT_FLAG_PLAYBACK_POS_CHANGED	This is macro AVC_EVENT_FLAG_PLAYBACK_POS_CHANGED.
	AVC_EVENT_FLAG_PLAYBACK_STATUS_CHANGED	• addtogroup avrcp • @{ • @name Notifications mask
	AVC_EVENT_FLAG_PLAYER_APPLICATION_SETTING_CHANGED	This is macro AVC_EVENT_FLAG_PLAYER_APPLICATION_SETTING_CHANGED.
	AVC_EVENT_FLAG_SYSTEM_STATUS_CHANGED	This is macro AVC_EVENT_FLAG_SYSTEM_STATUS_CHANGED.
	AVC_EVENT_FLAG_TRACK_CHANGED	This is macro AVC_EVENT_FLAG_TRACK_CHANGED.
	AVC_EVENT_FLAG_TRACK_REACHED_END	This is macro AVC_EVENT_FLAG_TRACK_REACHED_END.
	AVC_EVENT_FLAG_TRACK_REACHED_START	This is macro AVC_EVENT_FLAG_TRACK_REACHED_START.
	AVC_EVENT_FLAG_UIDS_CHANGED	This is macro AVC_EVENT_FLAG_UIDS_CHANGED.
	AVC_EVENT_FLAG_VOLUME_CHANGED	This is macro AVC_EVENT_FLAG_VOLUME_CHANGED.
	AVC_EVENT_NOW_PLAYING_CONTENT_CHANGED	< The content of the Now Playing list has changed, see 6.9.5.
	AVC_EVENT_PLAYBACK_POS_CHANGED	< Change in playback position. Returned after the specified playback notification change notification interval
	AVC_EVENT_PLAYBACK_STATUS_CHANGED	< Change in playback status of the current track.
	AVC_EVENT_PLAYER_APPLICATION_SETTING_CHANGED	< Change in player application setting
	AVC_EVENT_SYSTEM_STATUS_CHANGED	< Change in system status
	AVC_EVENT_TRACK_CHANGED	< Change of current track
	AVC_EVENT_TRACK_REACHED_END	< Reached end of a track
	AVC_EVENT_TRACK_REACHED_START	< Reached start of a track
	AVC_EVENT_UIDS_CHANGED	< The UIDs have changed, see 6.10.3.3.
	AVC_EVENT_VOLUME_CHANGED	< The volume has been changed locally on the TG, see 6.13.3
	AVC_FLAG_BROWSING_CMD	This is macro AVC_FLAG_BROWSING_CMD.
	AVC_FLAG_PANEL_CLICK	This is macro AVC_FLAG_PANEL_CLICK.
	AVC_FLAG_RESPONSE	This is macro AVC_FLAG_RESPONSE.
	AVC_MAXEVENTS	This is macro AVC_MAXEVENTS.
	AVC_MEDIA_ATTR_FLAG_ALBUM	This is macro AVC_MEDIA_ATTR_FLAG_ALBUM.
	AVC_MEDIA_ATTR_FLAG_ALL	This is macro AVC_MEDIA_ATTR_FLAG_ALL.
	AVC_MEDIA_ATTR_FLAG_ARTIST	This is macro AVC_MEDIA_ATTR_FLAG_ARTIST.
	AVC_MEDIA_ATTR_FLAG_GENRE	This is macro AVC_MEDIA_ATTR_FLAG_GENRE.
	AVC_MEDIA_ATTR_FLAG_NUMBER	This is macro AVC_MEDIA_ATTR_FLAG_NUMBER.
	AVC_MEDIA_ATTR_FLAG_PLAYING_TIME	This is macro AVC_MEDIA_ATTR_FLAG_PLAYING_TIME.
	AVC_MEDIA_ATTR_FLAG_TITLE	• addtogroup avrcp • @{ • @name Media attribute bitmask
	AVC_MEDIA_ATTR_FLAG_TOTAL_NUMBER	This is macro AVC_MEDIA_ATTR_FLAG_TOTAL_NUMBER.

	AVC_MEDIA_ATTR_ID_ALBUM	This is macro AVC_MEDIA_ATTR_ID_ALBUM.
	AVC_MEDIA_ATTR_ID_ARTIST	This is macro AVC_MEDIA_ATTR_ID_ARTIST.
	AVC_MEDIA_ATTR_ID_GENRE	This is macro AVC_MEDIA_ATTR_ID_GENRE.
	AVC_MEDIA_ATTR_ID_NUMBER	This is macro AVC_MEDIA_ATTR_ID_NUMBER.
	AVC_MEDIA_ATTR_ID_PLAYING_TIME	This is macro AVC_MEDIA_ATTR_ID_PLAYING_TIME.
	AVC_MEDIA_ATTR_ID_TITLE	• addtogroup avrcp • @{ • @name Media attribute IDs
	AVC_MEDIA_ATTR_ID_TOTAL_NUMBER	This is macro AVC_MEDIA_ATTR_ID_TOTAL_NUMBER.
	AVC_MEDIA_PLAYER_VIRTUAL_FILESYSTEM	Folder Item, Media Element Item The virtual filesystem containing the Browsed media content of the browsed player
	AVC_NOW_PLAYING	Media Element Item The Now Playing list (or queue) Addressed of the addressed player
	AVC_PACKET_TYPE_CONTINUE	This is macro AVC_PACKET_TYPE_CONTINUE.
	AVC_PACKET_TYPE_END	This is macro AVC_PACKET_TYPE_END.
	AVC_PACKET_TYPE_SINGLE	This is macro AVC_PACKET_TYPE_SINGLE.
	AVC_PACKET_TYPE_START	This is macro AVC_PACKET_TYPE_START.
	AVC_PANEL_BUTTON_PRESSED	This is macro AVC_PANEL_BUTTON_PRESSED.
	AVC_PANEL_BUTTON_RELEASED	This is macro AVC_PANEL_BUTTON_RELEASED.
	AVC_PANEL_OPID_0	define AVC_PANEL_OPID_RESERVER 0x10 define AVC_PANEL_OPID_RESERVER 0x11 define AVC_PANEL_OPID_RESERVER 0x12 define AVC_PANEL_OPID_RESERVER 0x13 define AVC_PANEL_OPID_RESERVER 0x14 define AVC_PANEL_OPID_RESERVER 0x15 define AVC_PANEL_OPID_RESERVER 0x16 define AVC_PANEL_OPID_RESERVER 0x17 define AVC_PANEL_OPID_RESERVER 0x18 define AVC_PANEL_OPID_RESERVER 0x19 define AVC_PANEL_OPID_RESERVER 0x1A define AVC_PANEL_OPID_RESERVER 0x1B define AVC_PANEL_OPID_RESERVER 0x1C define AVC_PANEL_OPID_RESERVER 0x1D define AVC_PANEL_OPID_RESERVER 0x1E define AVC_PANEL_OPID_RESERVER 0x1F
	AVC_PANEL_OPID_1	This is macro AVC_PANEL_OPID_1.
	AVC_PANEL_OPID_2	This is macro AVC_PANEL_OPID_2.
	AVC_PANEL_OPID_3	This is macro AVC_PANEL_OPID_3.
	AVC_PANEL_OPID_4	This is macro AVC_PANEL_OPID_4.
	AVC_PANEL_OPID_5	This is macro AVC_PANEL_OPID_5.
	AVC_PANEL_OPID_6	This is macro AVC_PANEL_OPID_6.
	AVC_PANEL_OPID_7	This is macro AVC_PANEL_OPID_7.
	AVC_PANEL_OPID_8	This is macro AVC_PANEL_OPID_8.
	AVC_PANEL_OPID_9	This is macro AVC_PANEL_OPID_9.
	AVC_PANEL_OPID_A	This is macro AVC_PANEL_OPID_A.
	AVC_PANEL_OPID_ANGLE	define AVC_PANEL_OPID_RESERVED 0x4E define AVC_PANEL_OPID_RESERVED 0x4F
	AVC_PANEL_OPID_APPS_MENU	This is macro AVC_PANEL_OPID_APPS_MENU.
	AVC_PANEL_OPID_B	This is macro AVC_PANEL_OPID_B.
	AVC_PANEL_OPID_BACKWARD	This is macro AVC_PANEL_OPID_BACKWARD.
	AVC_PANEL_OPID_C	This is macro AVC_PANEL_OPID_C.
	AVC_PANEL_OPID_CHANNEL_DOWN	This is macro AVC_PANEL_OPID_CHANNEL_DOWN.
	AVC_PANEL_OPID_CHANNEL_UP	define AVC_PANEL_OPID_RESRVED 0x2D define AVC_PANEL_OPID_RESRVED 0x2E define AVC_PANEL_OPID_RESRVED 0x2F
	AVC_PANEL_OPID_CLEAR	This is macro AVC_PANEL_OPID_CLEAR.

	AVC_PANEL_OPID_CONTENTS_MENU	This is macro AVC_PANEL_OPID_CONTENTS_MENU.
	AVC_PANEL_OPID_D	This is macro AVC_PANEL_OPID_D.
	AVC_PANEL_OPID_DISPLAY_INFORMATION	This is macro AVC_PANEL_OPID_DISPLAY_INFORMATION.
	AVC_PANEL_OPID_DOT	This is macro AVC_PANEL_OPID_DOT.
	AVC_PANEL_OPID_DOWN	This is macro AVC_PANEL_OPID_DOWN.
	AVC_PANEL_OPID_EJECT	This is macro AVC_PANEL_OPID_EJECT.
	AVC_PANEL_OPID_ENTER	This is macro AVC_PANEL_OPID_ENTER.
	AVC_PANEL_OPID_EXIT	This is macro AVC_PANEL_OPID_EXIT.
	AVC_PANEL_OPID_F1	define AVC_PANEL_OPID_RESERVED 0x6D define AVC_PANEL_OPID_RESERVED 0x6E define AVC_PANEL_OPID_RESERVED 0x6F define AVC_PANEL_OPID_RESERVED 0x70
	AVC_PANEL_OPID_F2	This is macro AVC_PANEL_OPID_F2.
	AVC_PANEL_OPID_F3	This is macro AVC_PANEL_OPID_F3.
	AVC_PANEL_OPID_F4	This is macro AVC_PANEL_OPID_F4.
	AVC_PANEL_OPID_F5	This is macro AVC_PANEL_OPID_F5.
	AVC_PANEL_OPID_F6	This is macro AVC_PANEL_OPID_F6.
	AVC_PANEL_OPID_F7	This is macro AVC_PANEL_OPID_F7.
	AVC_PANEL_OPID_F8	This is macro AVC_PANEL_OPID_F8.
	AVC_PANEL_OPID_F9	This is macro AVC_PANEL_OPID_F9.
	AVC_PANEL_OPID_FAST_FORWARD	This is macro AVC_PANEL_OPID_FAST_FORWARD.
	AVC_PANEL_OPID_FAVORITE_MENU	This is macro AVC_PANEL_OPID_FAVORITE_MENU.
	AVC_PANEL_OPID_FORWARD	This is macro AVC_PANEL_OPID_FORWARD.
	AVC_PANEL_OPID_HELP	This is macro AVC_PANEL_OPID_HELP.
	AVC_PANEL_OPID_INPUT_SELECT	This is macro AVC_PANEL_OPID_INPUT_SELECT.
	AVC_PANEL_OPID_KEYBORD_FUNCTION	This is macro AVC_PANEL_OPID_KEYBORD_FUNCTION.
	AVC_PANEL_OPID_LEFT	This is macro AVC_PANEL_OPID_LEFT.
	AVC_PANEL_OPID_LEFT_DOWN	This is macro AVC_PANEL_OPID_LEFT_DOWN.
	AVC_PANEL_OPID_LEFT_UP	This is macro AVC_PANEL_OPID_LEFT_UP.
	AVC_PANEL_OPID_LINKED_CONTENT	This is macro AVC_PANEL_OPID_LINKED_CONTENT.
	AVC_PANEL_OPID_LIST	This is macro AVC_PANEL_OPID_LIST.
	AVC_PANEL_OPID_LIVE_TV	This is macro AVC_PANEL_OPID_LIVE_TV.
	AVC_PANEL_OPID_LOCK	This is macro AVC_PANEL_OPID_LOCK.
	AVC_PANEL_OPID_MUTE	This is macro AVC_PANEL_OPID_MUTE.
	AVC_PANEL_OPID_MUTE_FUNCTION	This is macro AVC_PANEL_OPID_MUTE_FUNCTION.
	AVC_PANEL_OPID_NEXT_DAY	This is macro AVC_PANEL_OPID_NEXT_DAY.
	AVC_PANEL_OPID_ON_DEMAND_MENU	This is macro AVC_PANEL_OPID_ON_DEMAND_MENU.
	AVC_PANEL_OPID_PAGE_DOWN	This is macro AVC_PANEL_OPID_PAGE_DOWN.
	AVC_PANEL_OPID_PAGE_UP	This is macro AVC_PANEL_OPID_PAGE_UP.
	AVC_PANEL_OPID_PAUSE	This is macro AVC_PANEL_OPID_PAUSE.
	AVC_PANEL_OPID_PAUSE_PLAY_FUNCTION	This is macro AVC_PANEL_OPID_PAUSE_PLAY_FUNCTION.
	AVC_PANEL_OPID_PAUSE_RECORD_FUNCTION	This is macro AVC_PANEL_OPID_PAUSE_RECORD_FUNCTION .
	AVC_PANEL_OPID_PIP_DOWN	This is macro AVC_PANEL_OPID_PIP_DOWN.
	AVC_PANEL_OPID_PIP_MOVE	This is macro AVC_PANEL_OPID_PIP_MOVE.
	AVC_PANEL_OPID_PIP_UP	This is macro AVC_PANEL_OPID_PIP_UP.
	AVC_PANEL_OPID_PLAY	This is macro AVC_PANEL_OPID_PLAY.

	AVC_PANEL_OPID_PLAY_FUNCTION	define AVC_PANEL_OPID_RESERVED 0x56 define AVC_PANEL_OPID_RESERVED 0x57 define AVC_PANEL_OPID_RESERVED 0x58 define AVC_PANEL_OPID_RESERVED 0x5A define AVC_PANEL_OPID_RESERVED 0x5B define AVC_PANEL_OPID_RESERVED 0x5C define AVC_PANEL_OPID_RESERVED 0x5D define AVC_PANEL_OPID_RESERVED 0x5E define AVC_PANEL_OPID_RESERVED 0x5F
	AVC_PANEL_OPID_POWER_STATE_FUNCTION	This is macro AVC_PANEL_OPID_POWER_STATE_FUNCTION.
	AVC_PANEL_OPID_POWER_TOGGLE	This is macro AVC_PANEL_OPID_POWER_TOGGLE.
	AVC_PANEL_OPID_PREVIOUS_CHANNEL	This is macro AVC_PANEL_OPID_PREVIOUS_CHANNEL.
	AVC_PANEL_OPID_PREVIOUS_DAY	This is macro AVC_PANEL_OPID_PREVIOUS_DAY.
	AVC_PANEL_OPID_RECORD	This is macro AVC_PANEL_OPID_RECORD.
	AVC_PANEL_OPID_RECORD_FUNCTION	This is macro AVC_PANEL_OPID_RECORD_FUNCTION.
	AVC_PANEL_OPID_RESTORE_FOLUME_FUNCTION	This is macro AVC_PANEL_OPID_RESTORE_FOLUME_FUNCTION.
	AVC_PANEL_OPID_REWIND	This is macro AVC_PANEL_OPID_REWIND.
	AVC_PANEL_OPID_RF_BYPASS	This is macro AVC_PANEL_OPID_RF_BYPASS.
	AVC_PANEL_OPID_RIGHT	This is macro AVC_PANEL_OPID_RIGHT.
	AVC_PANEL_OPID_RIGHT_DOWN	This is macro AVC_PANEL_OPID_RIGHT_DOWN.
	AVC_PANEL_OPID_RIGHT_UP	This is macro AVC_PANEL_OPID_RIGHT_UP.
	AVC_PANEL_OPID_ROOT_MENU	This is macro AVC_PANEL_OPID_ROOT_MENU.
	AVC_PANEL_OPID_SELECT	• addtogroup avrcp • @{ • @name AV/C Panel PASS THROUGH operation IDs
	AVC_PANEL_OPID_SELECT_AUDIO_INPUT_FUNCTION	This is macro AVC_PANEL_OPID_SELECT_AUDIO_INPUT_FUNCTION.
	AVC_PANEL_OPID_SELECT_AV_INPUT_FUNCTION	This is macro AVC_PANEL_OPID_SELECT_AV_INPUT_FUNCTION.
	AVC_PANEL_OPID_SELECT_DISK_FUNCTION	This is macro AVC_PANEL_OPID_SELECT_DISK_FUNCTION.
	AVC_PANEL_OPID_SETUP_MENU	This is macro AVC_PANEL_OPID_SETUP_MENU.
	AVC_PANEL_OPID_SKIP	This is macro AVC_PANEL_OPID_SKIP.
	AVC_PANEL_OPID_SOUND_SELECT	This is macro AVC_PANEL_OPID_SOUND_SELECT.
	AVC_PANEL_OPID_STOP	This is macro AVC_PANEL_OPID_STOP.
	AVC_PANEL_OPID_STOP_FUNCTION	This is macro AVC_PANEL_OPID_STOP_FUNCTION.
	AVC_PANEL_OPID_SUBPICTURE	This is macro AVC_PANEL_OPID_SUBPICTURE.
	AVC_PANEL_OPID_TUNE_FUNCTION	This is macro AVC_PANEL_OPID_TUNE_FUNCTION.
	AVC_PANEL_OPID_UP	This is macro AVC_PANEL_OPID_UP.
	AVC_PANEL_OPID_VENDOR_UNIQUE	This is macro AVC_PANEL_OPID_VENDOR_UNIQUE.
	AVC_PANEL_OPID_VOLUME_DOWN	This is macro AVC_PANEL_OPID_VOLUME_DOWN.
	AVC_PANEL_OPID_VOLUME_UP	This is macro AVC_PANEL_OPID_VOLUME_UP.
	AVC_PANEL_OPID_ZOOM	This is macro AVC_PANEL_OPID_ZOOM.
	AVC_PDUID_ABORT_CONTINUING_RESPONSE	This is macro AVC_PDUID_ABORT_CONTINUING_RESPONSE.
	AVC_PDUID_ADD_TO_NOW_PLAYING	This is macro AVC_PDUID_ADD_TO_NOW_PLAYING.

	AVC_PDUID_CHANGE_PATH	This is macro AVC_PDUID_CHANGE_PATH.
	AVC_PDUID_GENERAL_REJECT	This is macro AVC_PDUID_GENERAL_REJECT.
	AVC_PDUID_GET_CURRENT_PLAYER_APPLICATION_SETTING_VALUE	This is macro AVC_PDUID_GET_CURRENT_PLAYER_APPLICATION_SETTING_VALUE.
	AVC_PDUID_GET_ELEMENT_ATTRIBUTES	This is macro AVC_PDUID_GET_ELEMENT_ATTRIBUTES.
	AVC_PDUID_GET_FOLDER_ITEMS	This is macro AVC_PDUID_GET_FOLDER_ITEMS.
	AVC_PDUID_GET_ITEM_ATTRIBUTES	This is macro AVC_PDUID_GET_ITEM_ATTRIBUTES.
	AVC_PDUID_GET_PLAY_STATUS	This is macro AVC_PDUID_GET_PLAY_STATUS.
	AVC_PDUID_GET_PLAYER_APPLICATION_SETTING_ATTRIBUTE_TEXT	This is macro AVC_PDUID_GET_PLAYER_APPLICATION_SETTING_ATTRIBUTE_TEXT.
	AVC_PDUID_GET_PLAYER_APPLICATION_SETTING_VALUE_TEXT	This is macro AVC_PDUID_GET_PLAYER_APPLICATION_SETTING_VALUE_TEXT.
	AVC_PDUID_GETCAPABILITIES	This is macro AVC_PDUID_GETCAPABILITIES.
	AVC_PDUID_INFORM_BATTERY_STATUS_OF_CT	This is macro AVC_PDUID_INFORM_BATTERY_STATUS_OF_CT.
	AVC_PDUID_INFORM_DISPLAYABLE_CHARACTER_SET	This is macro AVC_PDUID_INFORM_DISPLAYABLE_CHARACTER_SET.
	AVC_PDUID_LIST_PLAYER_APPLICATION_SETTING_ATTRIBUTES	This is macro AVC_PDUID_LIST_PLAYER_APPLICATION_SETTING_ATTRIBUTES.
	AVC_PDUID_LIST_PLAYER_APPLICATION_SETTING_VALUES	This is macro AVC_PDUID_LIST_PLAYER_APPLICATION_SETTING_VALUES.
	AVC_PDUID_PLAY_ITEM	This is macro AVC_PDUID_PLAY_ITEM.
	AVC_PDUID_REGISTER_NOTIFICATION	This is macro AVC_PDUID_REGISTER_NOTIFICATION.
	AVC_PDUID_REQUEST_CONTINUING_RESPONSE	This is macro AVC_PDUID_REQUEST_CONTINUING_RESPONSE.
	AVC_PDUID_SEARCH	This is macro AVC_PDUID_SEARCH.
	AVC_PDUID_SET_ABSOLUTE_VOLUME	This is macro AVC_PDUID_SET_ABSOLUTE_VOLUME.
	AVC_PDUID_SET_ADDRESSED_PLAYER	This is macro AVC_PDUID_SET_ADDRESSED_PLAYER.
	AVC_PDUID_SET_BROWSED_PLAYER	This is macro AVC_PDUID_SET_BROWSED_PLAYER.
	AVC_PDUID_SET_PLAYER_APPLICATION_SETTING_VALUE	This is macro AVC_PDUID_SET_PLAYER_APPLICATION_SETTING_VALUE.
	AVC_PLAY_STATUS_ERROR	This is macro AVC_PLAY_STATUS_ERROR.
	AVC_PLAY_STATUS_FW_SEEK	This is macro AVC_PLAY_STATUS_FW_SEEK.
	AVC_PLAY_STATUS_PAUSED	This is macro AVC_PLAY_STATUS_PAUSED.
	AVC_PLAY_STATUS_PLAYING	This is macro AVC_PLAY_STATUS_PLAYING.
	AVC_PLAY_STATUS_REV_SEEK	This is macro AVC_PLAY_STATUS_REV_SEEK.
	AVC_PLAY_STATUS_STOPPED	• addtogroup avrcp • @{ • @name Play status
	AVC_PLAYER_SETTING_EQUALIZER_OFF	This is macro AVC_PLAYER_SETTING_EQUALIZER_OFF.
	AVC_PLAYER_SETTING_EQUALIZER_ON	This is macro AVC_PLAYER_SETTING_EQUALIZER_ON.
	AVC_PLAYER_SETTING_EQUALIZER_STATUS	This is macro AVC_PLAYER_SETTING_EQUALIZER_STATUS.
	AVC_PLAYER_SETTING_REPEAT_ALL_TRACKS	This is macro AVC_PLAYER_SETTING_REPEAT_ALL_TRACKS.

	AVC_PLAYER_SETTING_REPEAT_GROUP	This is macro AVC_PLAYER_SETTING_REPEAT_GROUP.
	AVC_PLAYER_SETTING_REPEAT_MODE_OFF	This is macro AVC_PLAYER_SETTING_REPEAT_MODE_OFF.
	AVC_PLAYER_SETTING_REPEAT_MODE_STATUS	This is macro AVC_PLAYER_SETTING_REPEAT_MODE_STATUS.
	AVC_PLAYER_SETTING_REPEAT_SINGLE_TRACK	This is macro AVC_PLAYER_SETTING_REPEAT_SINGLE_TRACK.
	AVC_PLAYER_SETTING_SCAN_ALL_TRACKS	This is macro AVC_PLAYER_SETTING_SCAN_ALL_TRACKS.
	AVC_PLAYER_SETTING_SCAN_GROUP	This is macro AVC_PLAYER_SETTING_SCAN_GROUP.
	AVC_PLAYER_SETTING_SCAN_OFF	This is macro AVC_PLAYER_SETTING_SCAN_OFF.
	AVC_PLAYER_SETTING_SCAN_STATUS	This is macro AVC_PLAYER_SETTING_SCAN_STATUS.
	AVC_PLAYER_SETTING_SHUFFLE_ALL_TRACKS	This is macro AVC_PLAYER_SETTING_SHUFFLE_ALL_TRACKS.
	AVC_PLAYER_SETTING_SHUFFLE_GROUP	This is macro AVC_PLAYER_SETTING_SHUFFLE_GROUP.
	AVC_PLAYER_SETTING_SHUFFLE_OFF	This is macro AVC_PLAYER_SETTING_SHUFFLE_OFF.
	AVC_PLAYER_SETTING_SHUFFLE_STATUS	This is macro AVC_PLAYER_SETTING_SHUFFLE_STATUS.
	AVC_RESPONSE_ACCEPTED	This is macro AVC_RESPONSE_ACCEPTED.
	AVC_RESPONSE_CHANGED	This is macro AVC_RESPONSE_CHANGED.
	AVC_RESPONSE_IMPLEMENTED	This is macro AVC_RESPONSE_IMPLEMENTED.
	AVC_RESPONSE_IN_TRANSITION	This is macro AVC_RESPONSE_IN_TRANSITION.
	AVC_RESPONSE_INTERIM	This is macro AVC_RESPONSE_INTERIM.
	AVC_RESPONSE_NOT_IMPLEMENTED	• addtogroup avrcp • @{ • @name Response types
	AVC_RESPONSE_REJECTED	This is macro AVC_RESPONSE_REJECTED.
	AVC_RESPONSE_STABLE	This is macro AVC_RESPONSE_STABLE.
	AVC_RESPONSE_TIMEOUT	This is macro AVC_RESPONSE_TIMEOUT.
	AVC_SCOPE_MEDIA_PLAYER_LIST	Media Player Item Contains all available media players None
	AVC_SEARCH	Media Element Item The results of a search operation Browsed on the browsed player
	AVC_SUBUNIT_ID_EXTENDED_TO_NEXT_BYTE	This is macro AVC_SUBUNIT_ID_EXTENDED_TO_NEXT_BYTE.
	AVC_SUBUNIT_ID_IGNORE	This is macro AVC_SUBUNIT_ID_IGNORE.
	AVC_SUBUNIT_TYPE_AUDIO	This is macro AVC_SUBUNIT_TYPE_AUDIO.
	AVC_SUBUNIT_TYPE_BULLETIN_BOARD	This is macro AVC_SUBUNIT_TYPE_BULLETIN_BOARD.
	AVC_SUBUNIT_TYPE_CA	This is macro AVC_SUBUNIT_TYPE_CA.
	AVC_SUBUNIT_TYPE_CAMERA	This is macro AVC_SUBUNIT_TYPE_CAMERA.
	AVC_SUBUNIT_TYPE_CAMERA_STORAGE	This is macro AVC_SUBUNIT_TYPE_CAMERA_STORAGE.
	AVC_SUBUNIT_TYPE_DISC	This is macro AVC_SUBUNIT_TYPE_DISC.
	AVC_SUBUNIT_TYPE_EXTENDED_TO_NEXT_BYTE	This is macro AVC_SUBUNIT_TYPE_EXTENDED_TO_NEXT_BYTE.
	AVC_SUBUNIT_TYPE_MONITOR	• addtogroup avrcp • @{ • @name Subunit types
	AVC_SUBUNIT_TYPE_PANEL	This is macro AVC_SUBUNIT_TYPE_PANEL.
	AVC_SUBUNIT_TYPE_PRINTER	This is macro AVC_SUBUNIT_TYPE_PRINTER.

	AVC_SUBUNIT_TYPE_TAPE_RECORDER_PLAYER	This is macro AVC_SUBUNIT_TYPE_TAPE_RECORDER_PLAYER.
	AVC_SUBUNIT_TYPE_TUNER	This is macro AVC_SUBUNIT_TYPE_TUNER.
	AVC_SUBUNIT_TYPE_UNIT	This is macro AVC_SUBUNIT_TYPE_UNIT.
	AVC_SUBUNIT_TYPE_VENDOR_UNIQUE	This is macro AVC_SUBUNIT_TYPE_VENDOR_UNIQUE.
	AVC_VOLUME_MAX	100
	AVC_VOLUME_MIN	0
	AVRCP_BTSIG_COMPANY_ID	This is macro AVRCP_BTSIG_COMPANY_ID.
	AVRCP_CHANNEL_FLAG_LISTENING	This is macro AVRCP_CHANNEL_FLAG_LISTENING.
	AVRCP_CHANNEL_FLAG_PLAY_STATUS_REQUESTED	This is macro AVRCP_CHANNEL_FLAG_PLAY_STATUS_REQUESTED.
	AVRCP_CHANNEL_FLAG_REGISTERING_NOTIFICATIONS	This is macro AVRCP_CHANNEL_FLAG_REGISTERING_NOTIFICATIONS.
	AVRCP_CHANNEL_FLAG_SENDING	This is macro AVRCP_CHANNEL_FLAG_SENDING.
	AVRCP_CHANNEL_STATE_CONNECTED	This is macro AVRCP_CHANNEL_STATE_CONNECTED.
	AVRCP_CHANNEL_STATE_CONNECTING	This is macro AVRCP_CHANNEL_STATE_CONNECTING.
	AVRCP_CHANNEL_STATE_DISCONNECTING	This is macro AVRCP_CHANNEL_STATE_DISCONNECTING.
	AVRCP_CHANNEL_STATE_FREE	This is macro AVRCP_CHANNEL_STATE_FREE.
	AVRCP_CHANNEL_STATE_IDLE	This is macro AVRCP_CHANNEL_STATE_IDLE.
	AVRCP_COMMAND_TYPE_BROWSING	This is macro AVRCP_COMMAND_TYPE_BROWSING.
	AVRCP_COMMAND_TYPE_CONTROL	This is macro AVRCP_COMMAND_TYPE_CONTROL.
	AVRCP_EVT_ADD_TO_NOW_PLAYING_COMPLETED	< This event is generated when a local device received a response to a "add to now playing" request.
	AVRCP_EVT_ADDRESSED_PLAYER_CHANGED	< This event is generated when a local device received a "addressed player changed" notification.
	AVRCP_EVT_AVAILABLE_PLAYERS_CHANGED	< This event is generated when a local device received a "available players changed" notification.
	AVRCP_EVT_BATT_STATUS_CHANGED	< This event is generated when a local device received a "battery status changed" notification.
	AVRCP_EVT_BATTERY_STATUS_OF_CT_RECEIVED	< This event is generated when a local device received a "battery status of controller" command.
	AVRCP_EVT_BROWSING_CHANNEL_CONNECTED	< This event is generated when a browsing channel between two AVRCP entities has been established.
	AVRCP_EVT_BROWSING_CHANNEL_DISCONNECTED	< This event is generated when a browsing channel between two AVRCP entities has been terminated.
	AVRCP_EVT_BROWSING_CONNECTION_FAILED	< This event is generated when a local device failed to create a browsing channel between two AVRCP entities.
	AVRCP_EVT_COMPANY_ID_LIST_RECEIVED	< This event is generated when a local device received a response to a "get company id" request.
	AVRCP_EVT_CONTROL_CHANNEL_CONNECTED	< This event is generated when a control channel between two AVRCP entities has been established.
	AVRCP_EVT_CONTROL_CHANNEL_DISCONNECTED	< This event is generated when a control channel between two AVRCP entities has been terminated.
	AVRCP_EVT_CONTROL_CONNECTION_FAILED	< This event is generated when a local device failed to create a control channel between two AVRCP entities.
	AVRCP_EVT_DISPLAYABLE_CHARACTER_SET_RECEIVED	< This event is generated when a local device received a "displayable chracter set command" request.

	AVRCP_EVT_ELEMENT_ATTRIBUTES_REQUESTED	< This event is generated when a local device received a "get element attributes" request.
	AVRCP_EVT_EVENT_ID_LIST_RECEIVED	< This event is generated when a local device received a response to a "get supported events" request.
	AVRCP_EVT_GET_ELEMENT_ATTRIBUTES_RECEIVED	< This event is generated when a local device received a response to a "get media element attributes" request.
	AVRCP_EVT_GET_PLAY_STATUS_RECEIVED	< This event is generated when a local device received a response to a "get play status" request.
	AVRCP_EVT_INFORM_BATTERY_STATUS_OF_CT_COMPLETED	< This event is generated when a local device received a response to a "inform battery status" request.
	AVRCP_EVT_INFORM_DISPLAYABLE_CHARACTER_SET_COMPLETED	< This event is generated when a local device received a response to a "inform displayable character set" request.
	AVRCP_EVT_NOTHING	addtogroup avrcp @{ @name Events details The following is a list of events AVRCP layer generates and can report to the upper layer when it completes executing an operation initiated by either local or remote device.
	AVRCP_EVT_NOW_PLAYING_CONTENT_CHANGED	< This event is generated when a local device received a "now playing content changed" notification.
	AVRCP_EVT_PANEL_COMMAND_RECEIVED	< This event is generated when a local device received a PASS THROUGH command.
	AVRCP_EVT_PANEL_RESPONSE_RECEIVED	< This event is generated when a local device received a response to a PASS THROUGH command.
	AVRCP_EVT_PLAY_ITEM_COMPLETED	< This event is generated when a local device received a response to a "play item" request.
	AVRCP_EVT_PLAYBACK_POS_CHANGED	< This event is generated when a local device received a "playback position changed" notification.
	AVRCP_EVT_PLAYBACK_STATUS_CHANGED	< This event is generated when a local device received a "play status changed" notification.
	AVRCP_EVT_PLAYER_APPLICATION_SETTING_CHANGED	< This event is generated when a local device received a "player application setting changed" notification.
	AVRCP_EVT_PLAYER_CURRENT_SETTING_VALUES_RECEIVED	< This event is generated when a local device received a response to a "get current player setting attribute values" request.
	AVRCP_EVT_PLAYER_SETTING_ATTRIBUTES_RECEIVED	< This event is generated when a local device received a response to a "get supported player setting attributes" request.
	AVRCP_EVT_PLAYER_SETTING_ATTRIBUTES_TEXT_RECEIVED	< This event is generated when a local device received a response to a "get player setting attributes displayable text" request.
	AVRCP_EVT_PLAYER_SETTING_VALUES_RECEIVED	< This event is generated when a local device received a response to a "get player setting attribute values" request.
	AVRCP_EVT_PLAYER_SETTING_VALUES_TEXT_RECEIVED	< This event is generated when a local device received a response to a "get player setting attribute values displayable text" request.
	AVRCP_EVT_REGISTER_NOTIFICATION_REQUESTED	< This event is generated when a local device received a "register notification" request.
	AVRCP_EVT_REGISTER_NOTIFICATIONS_COMPLETED	< This event is generated when a local device received a response to a "register notification" request.
	AVRCP_EVT_SEARCH_COMPLETED	< This event is generated when a local device completed searching for nearby targets.
	AVRCP_EVT_SET_ABSOLUTE_VOLUME_COMPLETED	< This event is generated when a local device received a response to a "set absolute volume" request.
	AVRCP_EVT_SET_ABSOLUTE_VOLUME_REQUESTED	< This event is generated when a local device received a "set absolute volume" request.

	AVRCP_EVT_SET_ADDRESSED_PLAYER_COMPLETED	< This event is generated when a local device received a response to a "set addressed player" request.
	AVRCP_EVT_SET_PLAYER_SETTING_VALUES_COMPLETED	< This event is generated when a local device received a response to a "set player setting attribute values" request.
	AVRCP_EVT_SYSTEM_STATUS_CHANGED	< This event is generated when a local device received a "system status changed" notification.
	AVRCP_EVT_TRACK_CHANGED	< This event is generated when a local device received a "track changed changed" notification.
	AVRCP_EVT_TRACK_REACHED_END	< This event is generated when a local device received a "track reached end" notification.
	AVRCP_EVT_TRACK_REACHED_START	< This event is generated when a local device received a "track reached start" notification.
	AVRCP_EVT_UIDS_CHANGED	< This event is generated when a local device received a "UIDs changed" notification.
	AVRCP_EVT_VOLUME_CHANGED	< This event is generated when a local device received a "volume changed" notification.
	AVRCP_MANAGER_FLAG_SEARCHING	This is macro AVRCP_MANAGER_FLAG_SEARCHING.
	AVRCP_MANAGER_STATE_IDLE	This is macro AVRCP_MANAGER_STATE_IDLE.
	AVRCP_MAX_ELEMENT_ATTRIBUTES	This is macro AVRCP_MAX_ELEMENT_ATTRIBUTES.
	bt_avrcp_0_click	This is macro bt_avrcp_0_click.
	bt_avrcp_0_press	This is macro bt_avrcp_0_press.
	bt_avrcp_1_click	This is macro bt_avrcp_1_click.
	bt_avrcp_1_press	This is macro bt_avrcp_1_press.
	bt_avrcp_2_click	This is macro bt_avrcp_2_click.
	bt_avrcp_2_press	This is macro bt_avrcp_2_press.
	bt_avrcp_3_click	This is macro bt_avrcp_3_click.
	bt_avrcp_3_press	This is macro bt_avrcp_3_press.
	bt_avrcp_4_click	This is macro bt_avrcp_4_click.
	bt_avrcp_4_press	This is macro bt_avrcp_4_press.
	bt_avrcp_5_click	This is macro bt_avrcp_5_click.
	bt_avrcp_5_press	This is macro bt_avrcp_5_press.
	bt_avrcp_6_click	This is macro bt_avrcp_6_click.
	bt_avrcp_6_press	This is macro bt_avrcp_6_press.
	bt_avrcp_7_click	This is macro bt_avrcp_7_click.
	bt_avrcp_7_press	This is macro bt_avrcp_7_press.
	bt_avrcp_8_click	This is macro bt_avrcp_8_click.
	bt_avrcp_8_press	This is macro bt_avrcp_8_press.
	bt_avrcp_9_click	This is macro bt_avrcp_9_click.
	bt_avrcp_9_press	This is macro bt_avrcp_9_press.
	bt_avrcp_angle_click	This is macro bt_avrcp_angle_click.
	bt_avrcp_angle_press	This is macro bt_avrcp_angle_press.
	bt_avrcp_backward_click	This is macro bt_avrcp_backward_click.
	bt_avrcp_backward_press	This is macro bt_avrcp_backward_press.
	bt_avrcp_channel_down_click	This is macro bt_avrcp_channel_down_click.
	bt_avrcp_channel_down_press	This is macro bt_avrcp_channel_down_press.
	bt_avrcp_channel_up_click	This is macro bt_avrcp_channel_up_click.
	bt_avrcp_channel_up_press	This is macro bt_avrcp_channel_up_press.
	bt_avrcp_clear_click	This is macro bt_avrcp_clear_click.
	bt_avrcp_clear_press	This is macro bt_avrcp_clear_press.
	bt_avrcp_content_menu_click	This is macro bt_avrcp_content_menu_click.
	bt_avrcp_contents_menu_press	This is macro bt_avrcp_contents_menu_press.
	bt_avrcp_display_info_click	This is macro bt_avrcp_display_info_click.
	bt_avrcp_display_info_press	This is macro bt_avrcp_display_info_press.
	bt_avrcp_dot_click	This is macro bt_avrcp_dot_click.

	bt_avrcp_dot_press	This is macro bt_avrcp_dot_press.
	bt_avrcp_down_click	This is macro bt_avrcp_down_click.
	bt_avrcp_down_press	This is macro bt_avrcp_down_press.
	bt_avrcp_eject_click	This is macro bt_avrcp_eject_click.
	bt_avrcp_eject_press	This is macro bt_avrcp_eject_press.
	bt_avrcp_enter_click	This is macro bt_avrcp_enter_click.
	bt_avrcp_enter_press	This is macro bt_avrcp_enter_press.
	bt_avrcp_exit_click	This is macro bt_avrcp_exit_click.
	bt_avrcp_exit_press	This is macro bt_avrcp_exit_press.
	bt_avrcp_f1_click	This is macro bt_avrcp_f1_click.
	bt_avrcp_f1_press	This is macro bt_avrcp_f1_press.
	bt_avrcp_f2_click	This is macro bt_avrcp_f2_click.
	bt_avrcp_f2_press	This is macro bt_avrcp_f2_press.
	bt_avrcp_f3_click	This is macro bt_avrcp_f3_click.
	bt_avrcp_f3_press	This is macro bt_avrcp_f3_press.
	bt_avrcp_f4_click	This is macro bt_avrcp_f4_click.
	bt_avrcp_f4_press	This is macro bt_avrcp_f4_press.
	bt_avrcp_f5_click	This is macro bt_avrcp_f5_click.
	bt_avrcp_f5_press	This is macro bt_avrcp_f5_press.
	bt_avrcp_f6_click	This is macro bt_avrcp_f6_click.
	bt_avrcp_f6_press	This is macro bt_avrcp_f6_press.
	bt_avrcp_f7_click	This is macro bt_avrcp_f7_click.
	bt_avrcp_f7_press	This is macro bt_avrcp_f7_press.
	bt_avrcp_f8_click	This is macro bt_avrcp_f8_click.
	bt_avrcp_f8_press	This is macro bt_avrcp_f8_press.
	bt_avrcp_f9_click	This is macro bt_avrcp_f9_click.
	bt_avrcp_f9_press	This is macro bt_avrcp_f9_press.
	bt_avrcp_fast_forward_click	This is macro bt_avrcp_fast_forward_click.
	bt_avrcp_fast_forward_press	This is macro bt_avrcp_fast_forward_press.
	bt_avrcp_favorite_menu_click	This is macro bt_avrcp_favorite_menu_click.
	bt_avrcp_favorive_menu_press	This is macro bt_avrcp_favorive_menu_press.
	bt_avrcp_forward_click	This is macro bt_avrcp_forward_click.
	bt_avrcp_forward_press	This is macro bt_avrcp_forward_press.
	bt_avrcp_help_click	This is macro bt_avrcp_help_click.
	bt_avrcp_help_press	This is macro bt_avrcp_help_press.
	bt_avrcp_input_select_click	This is macro bt_avrcp_input_select_click.
	bt_avrcp_input_select_press	This is macro bt_avrcp_input_select_press.
	bt_avrcp_left_click	This is macro bt_avrcp_left_click.
	bt_avrcp_left_down_click	This is macro bt_avrcp_left_down_click.
	bt_avrcp_left_down_press	This is macro bt_avrcp_left_down_press.
	bt_avrcp_left_press	This is macro bt_avrcp_left_press.
	bt_avrcp_left_up_click	This is macro bt_avrcp_left_up_click.
	bt_avrcp_left_up_press	This is macro bt_avrcp_left_up_press.
	bt_avrcp_mute_click	This is macro bt_avrcp_mute_click.
	bt_avrcp_mute_press	This is macro bt_avrcp_mute_press.
	bt_avrcp_page_down_click	This is macro bt_avrcp_page_down_click.
	bt_avrcp_page_down_press	This is macro bt_avrcp_page_down_press.
	bt_avrcp_page_up_click	This is macro bt_avrcp_page_up_click.
	bt_avrcp_page_up_press	This is macro bt_avrcp_page_up_press.
	bt_avrcp_pause_click	This is macro bt_avrcp_pause_click.
	bt_avrcp_pause_press	This is macro bt_avrcp_pause_press.
	bt_avrcp_play_click	This is macro bt_avrcp_play_click.
	bt_avrcp_play_press	This is macro bt_avrcp_play_press.
	bt_avrcp_power_click	This is macro bt_avrcp_power_click.
	bt_avrcp_power_press	This is macro bt_avrcp_power_press.

	bt_avrcp_previous_channel_click	This is macro bt_avrcp_previous_channel_click.
	bt_avrcp_previous_channel_press	This is macro bt_avrcp_previous_channel_press.
	bt_avrcp_record_click	This is macro bt_avrcp_record_click.
	bt_avrcp_record_press	This is macro bt_avrcp_record_press.
	bt_avrcp_rewind_click	This is macro bt_avrcp_rewind_click.
	bt_avrcp_rewind_press	This is macro bt_avrcp_rewind_press.
	bt_avrcp_right_click	This is macro bt_avrcp_right_click.
	bt_avrcp_right_down_click	This is macro bt_avrcp_right_down_click.
	bt_avrcp_right_down_press	This is macro bt_avrcp_right_down_press.
	bt_avrcp_right_press	This is macro bt_avrcp_right_press.
	bt_avrcp_right_up_click	This is macro bt_avrcp_right_up_click.
	bt_avrcp_right_up_press	This is macro bt_avrcp_right_up_press.
	bt_avrcp_root_menu_click	This is macro bt_avrcp_root_menu_click.
	bt_avrcp_root_menu_press	This is macro bt_avrcp_root_menu_press.
	bt_avrcp_select_click	This is macro bt_avrcp_select_click.
	bt_avrcp_select_press	Panel operations
	bt_avrcp_setup_menu_click	This is macro bt_avrcp_setup_menu_click.
	bt_avrcp_setup_menu_press	This is macro bt_avrcp_setup_menu_press.
	bt_avrcp_sound_select_click	This is macro bt_avrcp_sound_select_click.
	bt_avrcp_sound_select_press	This is macro bt_avrcp_sound_select_press.
	bt_avrcp_stop_click	This is macro bt_avrcp_stop_click.
	bt_avrcp_stop_press	This is macro bt_avrcp_stop_press.
	bt_avrcp_subpicture_click	This is macro bt_avrcp_subpicture_click.
	bt_avrcp_subpicture_press	This is macro bt_avrcp_subpicture_press.
	bt_avrcp_up_click	This is macro bt_avrcp_up_click.
	bt_avrcp_up_press	This is macro bt_avrcp_up_press.
	bt_avrcp_volume_down_click	This is macro bt_avrcp_volume_down_click.
	bt_avrcp_volume_down_press	This is macro bt_avrcp_volume_down_press.
	bt_avrcp_volume_up_click	This is macro bt_avrcp_volume_up_click.
	bt_avrcp_volume_up_press	This is macro bt_avrcp_volume_up_press.
	bt_avrcp_0_release	This is macro bt_avrcp_0_release.
	bt_avrcp_1_release	This is macro bt_avrcp_1_release.
	bt_avrcp_2_release	This is macro bt_avrcp_2_release.
	bt_avrcp_3_release	This is macro bt_avrcp_3_release.
	bt_avrcp_4_release	This is macro bt_avrcp_4_release.
	bt_avrcp_5_release	This is macro bt_avrcp_5_release.
	bt_avrcp_6_release	This is macro bt_avrcp_6_release.
	bt_avrcp_7_release	This is macro bt_avrcp_7_release.
	bt_avrcp_8_release	This is macro bt_avrcp_8_release.
	bt_avrcp_9_release	This is macro bt_avrcp_9_release.
	bt_avrcp_angle_release	This is macro bt_avrcp_angle_release.
	bt_avrcp_backward_release	This is macro bt_avrcp_backward_release.
	bt_avrcp_channel_down_release	This is macro bt_avrcp_channel_down_release.
	bt_avrcp_channel_up_release	This is macro bt_avrcp_channel_up_release.
	bt_avrcp_clear_release	This is macro bt_avrcp_clear_release.
	bt_avrcp_content_menu_release	This is macro bt_avrcp_content_menu_release.
	bt_avrcp_display_info_release	This is macro bt_avrcp_display_info_release.
	bt_avrcp_dot_release	This is macro bt_avrcp_dot_release.
	bt_avrcp_down_release	This is macro bt_avrcp_down_release.
	bt_avrcp_eject_release	This is macro bt_avrcp_eject_release.
	bt_avrcp_enter_release	This is macro bt_avrcp_enter_release.
	bt_avrcp_exit_release	This is macro bt_avrcp_exit_release.
	bt_avrcp_f1_release	This is macro bt_avrcp_f1_release.
	bt_avrcp_f2_release	This is macro bt_avrcp_f2_release.
	bt_avrcp_f3_release	This is macro bt_avrcp_f3_release.

	bt_avrcp_f4_release	This is macro bt_avrcp_f4_release.
	bt_avrcp_f5_release	This is macro bt_avrcp_f5_release.
	bt_avrcp_f6_release	This is macro bt_avrcp_f6_release.
	bt_avrcp_f7_release	This is macro bt_avrcp_f7_release.
	bt_avrcp_f8_release	This is macro bt_avrcp_f8_release.
	bt_avrcp_f9_release	This is macro bt_avrcp_f9_release.
	bt_avrcp_fast_forward_release	This is macro bt_avrcp_fast_forward_release.
	bt_avrcp_favorite_menu_release	This is macro bt_avrcp_favorite_menu_release.
	bt_avrcp_forward_release	This is macro bt_avrcp_forward_release.
	bt_avrcp_help_release	This is macro bt_avrcp_help_release.
	bt_avrcp_input_select_release	This is macro bt_avrcp_input_select_release.
	bt_avrcp_left_down_release	This is macro bt_avrcp_left_down_release.
	bt_avrcp_left_release	This is macro bt_avrcp_left_release.
	bt_avrcp_left_up_release	This is macro bt_avrcp_left_up_release.
	bt_avrcp_mute_release	This is macro bt_avrcp_mute_release.
	bt_avrcp_page_down_release	This is macro bt_avrcp_page_down_release.
	bt_avrcp_page_up_release	This is macro bt_avrcp_page_up_release.
	bt_avrcp_pause_release	This is macro bt_avrcp_pause_release.
	bt_avrcp_play_release	This is macro bt_avrcp_play_release.
	bt_avrcp_power_release	This is macro bt_avrcp_power_release.
	bt_avrcp_previous_channel_release	This is macro bt_avrcp_previous_channel_release.
	bt_avrcp_record_release	This is macro bt_avrcp_record_release.
	bt_avrcp_rewind_release	This is macro bt_avrcp_rewind_release.
	bt_avrcp_right_down_release	This is macro bt_avrcp_right_down_release.
	bt_avrcp_right_release	This is macro bt_avrcp_right_release.
	bt_avrcp_right_up_release	This is macro bt_avrcp_right_up_release.
	bt_avrcp_root_menu_release	This is macro bt_avrcp_root_menu_release.
	bt_avrcp_select_release	This is macro bt_avrcp_select_release.
	bt_avrcp_setup_menu_release	This is macro bt_avrcp_setup_menu_release.
	bt_avrcp_sound_select_release	This is macro bt_avrcp_sound_select_release.
	bt_avrcp_stop_release	This is macro bt_avrcp_stop_release.
	bt_avrcp_subpicture_release	This is macro bt_avrcp_subpicture_release.
	bt_avrcp_up_release	This is macro bt_avrcp_up_release.
	bt_avrcp_volume_down_release	This is macro bt_avrcp_volume_down_release.
	bt_avrcp_volume_up_release	This is macro bt_avrcp_volume_up_release.
	__AVRCP_CONFIG_EVENT_HANDLERS_H	This is macro __AVRCP_CONFIG_EVENT_HANDLERS_H.
	AVRCP_COMMAND_HANDLER	This is macro AVRCP_COMMAND_HANDLER.
	AVRCP_COMMAND_SENT_HANDLER	This is macro AVRCP_COMMAND_SENT_HANDLER.
	AVRCP_RESPONSE_HANDLER	This is macro AVRCP_RESPONSE_HANDLER.
	AVRCP_RESPONSE_SENT_HANDLER	This is macro AVRCP_RESPONSE_SENT_HANDLER.

AVRCP Functions

	Name	Description
	_bt_avrcp_allocate_browsing_cmd	This is function _bt_avrcp_allocate_browsing_cmd.
	_bt_avrcp_allocate_bt_specific_cmd	This is function _bt_avrcp_allocate_bt_specific_cmd.
	_bt_avrcp_allocate_bt_specific_response	This is function _bt_avrcp_allocate_bt_specific_response.
	_bt_avrcp_allocate_channel	void _bt_avctp_l2cap_read_data_callback(struct _channel *pch, bt_byte* pdata, bt_int len); bt_avctp_channel_t* _bt_avctp_find_channel(bt_avctp_mgr_t* mgr, bt_bdaddr_t* remote_addr, bt_uint profile_id);
	_bt_avrcp_allocate_cmd	This is function _bt_avrcp_allocate_cmd.
	_bt_avrcp_allocate_response	This is function _bt_avrcp_allocate_response.
	_bt_avrcp_allocate_simple_panel_cmd	This is function _bt_avrcp_allocate_simple_panel_cmd.
	_bt_avrcp_allocate_simple_panel_response	This is function _bt_avrcp_allocate_simple_panel_response.

	_bt_avrcp_find_channel	This is function _bt_avrcp_find_channel.
	_bt_avrcp_free_channel	This is function _bt_avrcp_free_channel.
	_bt_avrcp_free_cmd	This is function _bt_avrcp_free_cmd.
	_bt_avrcp_get_tick_count	This is function _bt_avrcp_get_tick_count.
	_bt_avrcp_handle_add_to_now_playing	This is function _bt_avrcp_handle_add_to_now_playing.
	_bt_avrcp_handle_command	This is function _bt_avrcp_handle_command.
	_bt_avrcp_handle_command_sent	This is function _bt_avrcp_handle_command_sent.
	_bt_avrcp_handle_get_capabilities	This is function _bt_avrcp_handle_get_capabilities.
	_bt_avrcp_handle_get_current_player_application_setting_value	This is function _bt_avrcp_handle_get_current_player_application_setting_value.
	_bt_avrcp_handle_get_element_attributes	This is function _bt_avrcp_handle_get_element_attributes.
	_bt_avrcp_handle_get_play_status	This is function _bt_avrcp_handle_get_play_status.
	_bt_avrcp_handle_get_player_application_setting_attribute_text	This is function _bt_avrcp_handle_get_player_application_setting_attribute_text.
	_bt_avrcp_handle_get_player_application_setting_value_text	This is function _bt_avrcp_handle_get_player_application_setting_value_text.
	_bt_avrcp_handle_inform_battery_status_of_ct	This is function _bt_avrcp_handle_inform_battery_status_of_ct.
	_bt_avrcp_handle_inform_displayable_character_set	This is function _bt_avrcp_handle_inform_displayable_character_set.
	_bt_avrcp_handle_list_player_application_setting_attributes	This is function _bt_avrcp_handle_list_player_application_setting_attributes.
	_bt_avrcp_handle_list_player_application_setting_values	This is function _bt_avrcp_handle_list_player_application_setting_values.
	_bt_avrcp_handle_play_item	This is function _bt_avrcp_handle_play_item.
	_bt_avrcp_handle_register_notification	This is function _bt_avrcp_handle_register_notification.
	_bt_avrcp_handle_request_continuing_response	This is function _bt_avrcp_handle_request_continuing_response.
	_bt_avrcp_handle_response	This is function _bt_avrcp_handle_response.
	_bt_avrcp_handle_response_sent	This is function _bt_avrcp_handle_response_sent.
	_bt_avrcp_handle_set_absolute_volume	This is function _bt_avrcp_handle_set_absolute_volume.
	_bt_avrcp_handle_set_addressed_player	This is function _bt_avrcp_handle_set_addressed_player.
	_bt_avrcp_handle_set_player_application_setting_value	This is function _bt_avrcp_handle_set_player_application_setting_value.
	_bt_avrcp_init_signal	void _bt_avctp_init_message_buffers();
	_bt_avrcp_init_timer	This is function _bt_avrcp_init_timer.
	_bt_avrcp_register_next_notification	This is function _bt_avrcp_register_next_notification.
	_bt_avrcp_register_pending_notification	This is function _bt_avrcp_register_pending_notification.
	_bt_avrcp_send_notifications	This is function _bt_avrcp_send_notifications.
	_bt_avrcp_set_signal	This is function _bt_avrcp_set_signal.
	_bt_avrcp_start_timer	This is function _bt_avrcp_start_timer.
	_bt_avrcp_tg_handle_get_capabilities	This is function _bt_avrcp_tg_handle_get_capabilities.
	_bt_avrcp_tg_handle_get_element_attributes	This is function _bt_avrcp_tg_handle_get_element_attributes.
	_bt_avrcp_tg_handle_get_play_status	This is function _bt_avrcp_tg_handle_get_play_status.
	_bt_avrcp_tg_handle_inform_battery_status_of_ct	This is function _bt_avrcp_tg_handle_inform_battery_status_of_ct.
	_bt_avrcp_tg_handle_inform_displayable_character_set	This is function _bt_avrcp_tg_handle_inform_displayable_character_set.
	_bt_avrcp_tg_handle_register_notification	This is function _bt_avrcp_tg_handle_register_notification.
	_bt_avrcp_tg_handle_set_absolute_volume	This is function _bt_avrcp_tg_handle_set_absolute_volume.
	_bt_avrcp_write_command_header	This is function _bt_avrcp_write_command_header.
	bt_avrcp_abort_continuing_response	This is function bt_avrcp_abort_continuing_response.

	bt_avrcp_add_to_now_playing	brief Add to "now playing" list. ingroup avrcp details This function adds a media element specified by p element_id to the "now playing" list on the target. param channel AVRCP channel. param scope The scope in which the p element_id is valid. This value can be on the following values: li AVC_SCOPE_MEDIA_PLAYER_LIST li AVC_MEDIA_PLAYER_VIRTUAL_FILESYSTEM li AVC_SEARCH li AVC_NOW_PLAYING param element_id UID of the media element to be added to the "now playing" list. param counter UID counter. return li c TRUE if the function succeeds. li c FALSE otherwise.
	bt_avrcp_avrcpt_request_continuing_response	This is function bt_avrcp_avrcpt_request_continuing_response.
	bt_avrcp_cancel_find	brief Cancel finding Targets ingroup avrcp details This function stops AVRCP layer looking for targets on nearby devices. As a result of this operation the AVRCP_EVT_SEARCH_COMPLETED event will be generated. return li c TRUE if the function succeeds. li c FALSE otherwise. the callback is not called in this case.
	bt_avrcp_cancel_listen	brief Cancel listening for incoming connections. ingroup avrcp details This function stops listening for incoming connections on a specified channel. param channel AVRCP channel. return li c TRUE if the function succeeds. li c FALSE otherwise.
	bt_avrcp_connect	brief Connect to a remote device. ingroup avrcp details This function establishes a connection to a remote device specified by the p remote_address. If connection cannot be initiated for some reason, for example, there is not enough resources, it returns FALSE and no events are generated. Otherwise the result of an attempt to connect to the remote device is reported via the AVRCP callback. The events generated will either be AVRCP_EVT_CONTROL_CHANNEL_CONNECTED or AVRCP_EVT_CONTROL_CONNECTION_FAILED. param channel AVRCP channel. param remote_address The address of a remote device. return li c TRUE if connection establishment has been started. li c FALSE otherwise.... more
	bt_avrcp_create_channel	brief Allocate AVRCP channel ingroup avrcp details This function allocates a new incoming AVRCP channel. The channel is intended to be used to accept a connection from a remote device. There can be only one incoming channel. param mgr AVRCP manager. param create_browsing_channel Defines weather a browsing channel will be created. return li A pointer to the new AVRCP channel if the function succeeds. li c NULL otherwise.
	bt_avrcp_create_outgoing_channel	brief Allocate AVRCP channel ingroup avrcp details This function allocates a new outgoing AVRCP channel. The channel is intended to be used to create a connection to a remote device. There can be multiple outgoing channels. param mgr AVRCP manager. param create_browsing_channel Defines weather a browsing channel will be created. return li A pointer to the new AVRCP channel if the function succeeds. li c NULL otherwise.
	bt_avrcp_destroy_channel	brief Destroy AVRCP channel. ingroup avrcp details This function frees memory used by the channel. The channel has to exist and be in the "idle" state for this function to succeed. I.e. the channel has to be disconnected before this function can be called. param channel AVRCP channel. return li c TRUE if the function succeeds. li c FALSE otherwise.
	bt_avrcp_disconnect	brief Disconnect from a remote device. ingroup avrcp details This function closes a connection to a remote device. param channel AVRCP channel. return li c TRUE if disconnection has been started. li c FALSE otherwise. No events will be generated.

	bt_avrcp_find_controller	brief Find Controller ingroup avrcp details This function looks for a controller on a remote device specified by p deviceAddress. param deviceAddress The address of a remote device. param callback The callback function that will be called when search has completed. param client_callback The optional callback function that an application can set if it wants to be notified of state changes of the SDP client. The c evt parameter of the callback can be one of the following values: li SDP_CLIENT_STATE_IDLE li SDP_CLIENT_STATE_CONNECTING li SDP_CLIENT_STATE_DISCONNECTING li SDP_CLIENT_STATE_CONNECTED param callback_param A pointer to arbitrary data to be passed to the p... more
	bt_avrcp_find_target	brief Find Target ingroup avrcp details This function looks for a target on a remote device specified by p deviceAddress. param deviceAddress The address of a remote device. param callback The callback function that will be called when search has completed. param client_callback The optional callback function that an application can set if it wants to be notified of state changes of the SDP client. The c evt parameter of the callback can be one of the following values: li SDP_CLIENT_STATE_IDLE li SDP_CLIENT_STATE_CONNECTING li SDP_CLIENT_STATE_DISCONNECTING li SDP_CLIENT_STATE_CONNECTED param callback_param A pointer to arbitrary data to be passed to the p... more
	bt_avrcp_find_targets	brief Find Targets ingroup avrcp details This function looks for targets on nearby devices. The AVRCP_EVT_SEARCH_COMPLETED event is generated when the search has completed. param search_length The amount of time the search will be performed for. return li c TRUE if the function succeeds. li c FALSE otherwise. the callback is not called in this case.
	bt_avrcp_get_browsing_channel_state	brief Get AVCTP browsing channel state. ingroup avrcp details This function returns status of the AVCTP browsing channel. param channel AVRCP channel. return Returns of the following values: li AVCTP_CHANNEL_STATE_FREE li AVCTP_CHANNEL_STATE_IDLE li AVCTP_CHANNEL_STATE_CONNECTING li AVCTP_CHANNEL_STATE_CONNECTED li AVCTP_CHANNEL_STATE_DISCONNECTING
	bt_avrcp_get_channel_remote_address	brief Get channel's remote BT address. ingroup avrcp details This function returns the address of the remote device associated with the channel. param channel AVRCP channel. return li The address of the remote device if channel is connected. li NULL otherwise.
	bt_avrcp_get_company_id_list	brief Get Company ID list. ingroup avrcp details This function requests a list of company id's supported by the remote device param channel AVRCP channel. return li c TRUE if the function succeeds. li c FALSE otherwise.
	bt_avrcp_get_control_channel_state	brief Get AVCTP control channel state. ingroup avrcp details This function returns status of the AVCTP control channel. param channel AVRCP channel. return Returns of the following values: li AVCTP_CHANNEL_STATE_FREE li AVCTP_CHANNEL_STATE_IDLE li AVCTP_CHANNEL_STATE_CONNECTING li AVCTP_CHANNEL_STATE_CONNECTED li AVCTP_CHANNEL_STATE_DISCONNECTING

	bt_avrcp_get_current_player_application_setting_value	brief Get current player setting values. ingroup avrcp details This function requests a list of current set values for the player application on the target. The list of attribute ids whose values have to be returned is passed via the p response_buffer parameter. The caller has to set bt_av_player_setting_current_values_t::setting_id_list to a list of player setting attribute ids, bt_av_player_setting_current_values_t::count to the number of entries in the list, bt_av_player_setting_current_values_t::setting_value_id_list to a buffer where returned values will be stored. param channel AVRCP channel. param response_buffer Pointer to bt_av_player_setting_current_values_t structure initialized as stated above. return li c TRUE if the function succeeds. li c... more
	bt_avrcp_get_element_attributes	brief Get media element attributes. ingroup avrcp details This function requests the attributes of the element specified with p element_id. note Currently p element_id is ignored. The AVRCP specification defines that only UID 0 can be used to return the attributes of the current track. param channel AVRCP channel. param element_id UID of the media element whose attributes are requested. param attr_mask Bitmask that defines which attributes are requested. This value can be a combination of the following values: li AVC_MEDIA_ATTR_FLAG_TITLE li AVC_MEDIA_ATTR_FLAG_ARTIST li AVC_MEDIA_ATTR_FLAG_ALBUM li AVC_MEDIA_ATTR_FLAG_NUMBER li AVC_MEDIA_ATTR_FLAG_TOTAL_NUMBER li AVC_MEDIA_ATTR_FLAG_GENRE li AVC_MEDIA_ATTR_FLAG_PLAYING_TIME li AVC_MEDIA_ATTR_FLAG_ALL return li c TRUE if... more
	bt_avrcp_get_hci_connection	brief Get HCI connection for a channel ingroup avrcp details This function returns a pointer to a structure that describes an HCI connection a channel is open on. The return value can be used to call various function from the HCI layer. For example, if an app wants to force disconnection from a remote device it can call ::bt_hci_disconnect. param channel AVRCP channel. return li c Pointer to a structure that describes an HCI connection if the function succeeds. li c NULL otherwise. The function fails only if a channel specified by the p channel parameter li does... more
	bt_avrcp_get_mgr	brief Return a pointer to an instance of the AVRCP manager. ingroup avrcp details This function returns a pointer to an instance of the AVRCP manager. There is only one instance of the manager allocated by the stack.
	bt_avrcp_get_play_status	brief Get playback status. ingroup avrcp details This function requests the status of the currently playing media at the target. param channel AVRCP channel. param repeat_interval Interval in milliseconds at which AVRCP polls the target for playback status. If 0 is passed polling is stopped. return li c TRUE if the function succeeds. li c FALSE otherwise.

	bt_avrcp_get_player_application_setting_attr_text	brief Get player setting attribute text. ingroup avrcp details This function requests the target device to provide supported player application setting attribute displayable text for the provided player application setting attributes. The list of attribute ids whose displayable text have to be returned is passed via the p response_buffer parameter. The caller has to set bt_av_player_settings_text_t::setting_id_list to a list of player setting attribute ids, bt_av_player_settings_text_t::count to the number of entries in the list, bt_av_player_settings_text_t::setting_text_list to a buffer where returned values will be stored. param channel AVRCP channel. param response_buffer Pointer to bt_av_player_settings_text_t structure initialized as stated above. return li c... more
	bt_avrcp_get_player_application_setting_value_text	brief Get player setting value text. ingroup avrcp details This function request the target device to provide target supported player application setting value displayable text for the provided player application setting attribute values. The list of attribute ids whose value displayable text have to be returned is passed via the p response_buffer parameter. The caller has to set bt_av_player_setting_values_text_t::setting_value_id_list to a list of player setting attribute value ids, bt_av_player_setting_values_text_t::count to the number of entries in the list, bt_av_player_setting_values_text_t::setting_value_text_list to a buffer where returned values will be stored. param channel AVRCP channel. param response_buffer Pointer to bt_av_player_setting_values_text_t structure initialized as stated... more
	bt_avrcp_get_subuint_info	brief Get subunit info ingroup avrcp details This function is used to request subunit info from the target. param channel AVRCP channel. return li c TRUE if the function succeeds. li c FALSE otherwise.
	bt_avrcp_get_supported_event_id_list	brief Get supported events. ingroup avrcp details This function requests a list of events supported by the remote device param channel AVRCP channel. return li c TRUE if the function succeeds. li c FALSE otherwise.
	bt_avrcp_get_unit_info	brief Get unit info ingroup avrcp details This function is used to request unit info from the target. param channel AVRCP channel. return li c TRUE if the function succeeds. li c FALSE otherwise.
	bt_avrcp_inform_battery_status	brief Inform controller's battery status. ingroup avrcp details This function is used to inform the target about the controller's battery status. param channel AVRCP channel. param status Battery status. This can be one of the following values: li AVC_BATTERY_STATUS_NORMAL li AVC_BATTERY_STATUS_WARNING li AVC_BATTERY_STATUS_CRITICAL li AVC_BATTERY_STATUS_EXTERNAL li AVC_BATTERY_STATUS_FULL_CHARGE return li c TRUE if the function succeeds. li c FALSE otherwise.
	bt_avrcp_inform_displayable_character_set	brief Inform displayable character set. ingroup avrcp details This function informs the list of character set supported by the controller to the target. param channel AVRCP channel. param charset_list List of displayable character sets. param charset_count Number of entries in the list. return li c TRUE if the function succeeds. li c FALSE otherwise.
	bt_avrcp_init_controller	brief Initialize AVRCP to be used in controller mode. ingroup avrcp details This function initializes the AVRCP layer of the stack in controller mode. It must be called prior to any other AVRCP function can be called.

	bt_avrcp_init_target	brief Initialize AVRCP to be used in target mode. ingroup avrcp details This function initializes the AVRCP layer of the stack in target mode. It must be called prior to any other AVRCP function can be called. param company_id The 24-bit unique ID obtained from the IEEE Registration Authority Committee. If the vendor of a TG device does not have the unique ID, the value 0xFFFFFF may be used. param supported_events Bitmask that specifies events supported by the target. This value can be a combination of the following values: li AVC_EVENT_FLAG_PLAYBACK_STATUS_CHANGED li AVC_EVENT_FLAG_TRACK_CHANGED li AVC_EVENT_FLAG_TRACK_REACHED_END li AVC_EVENT_FLAG_TRACK_REACHED_START li AVC_EVENT_FLAG_PLAYBACK_POS_CHANGED li... more
	bt_avrcp_list_player_application_setting_attributes	brief Get supported player setting attributes. ingroup avrcp details This function request the target device to provide target supported player application setting attributes. The list of attribute ids is stored in the setting_id_list member of the p response_buffer parameter. The caller has to set bt_av_player_settings_t::setting_id_list to a buffer where returned values will be stored and bt_av_player_settings_t::count to the number of entries in the list. param channel AVRCP channel. param response_buffer Pointer to bt_av_player_settings_t structure initialized as stated above. return li c TRUE if the function succeeds. li c FALSE otherwise.
	bt_avrcp_list_player_application_setting_values	brief Get player setting attribute values. ingroup avrcp details This function requests the target device to list the set of possible values for the requested player application setting attribute. The list of attribute value ids is stored in the setting_value_id_list member of the p response_buffer parameter. The caller has to set bt_av_player_setting_values_t::setting_value_id_list to a buffer where returned values will be stored and bt_av_player_setting_values_t::count to the number of entries in the list. param channel AVRCP channel. param response_buffer Pointer to bt_av_player_setting_values_t structure initialized as stated above. return li c TRUE if the function succeeds. li c FALSE otherwise.
	bt_avrcp_listen	brief Listen for incoming connections. ingroup avrcp details This function enables incoming connections on the specified AVRCP channel. param channel AVRCP channel. return li c TRUE if the function succeeds. li c FALSE otherwise.
	bt_avrcp_play_item	brief Play media item. ingroup avrcp details This function starts playing an item indicated by the UID. param channel AVRCP channel. param scope The scope in which the p element_id is valid. This value can be on the following values: li AVC_SCOPE_MEDIA_PLAYER_LIST li AVC_MEDIA_PLAYER_VIRTUAL_FILESYSTEM li AVC_SEARCH li AVC_NOW_PLAYING param element_id UID of the media element to be played. param counter UID counter. return li c TRUE if the function succeeds. li c FALSE otherwise.

	bt_avrcp_register_notification	brief Register notification. ingroup avrcp details This function registers with the target to receive notifications asynchronously based on specific events occurring. param channel AVRCP channel. param event_id Event Id. This value can be one of the following values: li AVC_EVENT_PLAYBACK_STATUS_CHANGED li AVC_EVENT_TRACK_CHANGED li AVC_EVENT_TRACK_REACHED_END li AVC_EVENT_TRACK_REACHED_START li AVC_EVENT_PLAYBACK_POS_CHANGED li AVC_EVENT_BATT_STATUS_CHANGED li AVC_EVENT_SYSTEM_STATUS_CHANGED li AVC_EVENT_PLAYER_APPLICATION_SETTING_CHANGED li AVC_EVENT_NOW_PLAYING_CONTENT_CHANGED li AVC_EVENT_AVAILABLE_PLAYERS_CHANGED li AVC_EVENT_ADDRESSED_PLAYER_CHANGED li AVC_EVENT_UIDS_CHANGED li AVC_EVENT_VOLUME_CHANGED param playback_interval The time interval (in seconds) at which the change in playback position will be notified. Applicable for AVC_EVENT_PLAYBACK_POS_CHANGED event. return li c TRUE if the function succeeds. li c FALSE otherwise.
	bt_avrcp_register_notifications	brief Register notifications. ingroup avrcp details This function registers with the target to receive notifications asynchronously based on specific events occurring. This function is used to register multiple notifications with one call. note This function cannot be used to register for AVC_EVENT_PLAYBACK_POS_CHANGED event. param channel AVRCP channel. param event_mask Bitmask that specifies which events to register for. This value can be a combination of the following values: li AVC_EVENT_FLAG_PLAYBACK_STATUS_CHANGED li AVC_EVENT_FLAG_TRACK_CHANGED li AVC_EVENT_FLAG_TRACK_REACHED_END li AVC_EVENT_FLAG_TRACK_REACHED_START li AVC_EVENT_FLAG_BATT_STATUS_CHANGED li AVC_EVENT_FLAG_SYSTEM_STATUS_CHANGED li AVC_EVENT_FLAG_PLAYER_APPLICATION_SETTING_CHANGED li AVC_EVENT_FLAG_NOW_PLAYING_CONTENT_CHANGED li AVC_EVENT_FLAG_AVAILABLE_PLAYERS_CHANGED li AVC_EVENT_FLAG_ADDRESSED_PLAYER_CHANGED li AVC_EVENT_FLAG_UIDS_CHANGED li AVC_EVENT_FLAG_VOLUME_CHANGED li AVC_EVENT_FLAG_ALL return li c TRUE if the function... more
	bt_avrcp_send_button_click	brief Send AV/C Panel Subunit "click" PASS THROUGH command ingroup avrcp details This function is used to send a button click. Two PATH THROUGTH commands are sent. The first command has button state set to AVC_PANEL_BUTTON_PRESSED. The second command gas button state set to AVC_PANEL_BUTTON_RELEASED param channel AVRCP channel. param button_id Operation Id. This value can be on of the AVC_PANEL_OPID_... constants return li c TRUE if the function succeeds. li c FALSE otherwise.
	bt_avrcp_send_cmd	brief Send AVRCP command ingroup avrcp details This function sends a command to the remote device. param channel AVRCP channel. param command Command to be sent. return li c TRUE if the function succeeds. li c FALSE otherwise.

	bt_avrcp_send_panel_control	brief Send AV/C Panel Subunit "control" PASS THROUGH command ingroup avrcp details This function is used to send AV/C Panel Subunit PASS THROUGH command with command type set to AVC_CTYPE_CONTROL. param channel AVRCP channel. param opid Operation Id. This value can be on of the AVC_PANEL_OPID_... constants param button_state Button state. This can be on of the following values: li AVC_PANEL_BUTTON_PRESSED li AVC_PANEL_BUTTON_RELEASED return li c TRUE if the function succeeds. li c FALSE otherwise.
	bt_avrcp_send_press_panel_control	brief Send AV/C Panel Subunit "pressed" PASS THROUGH command ingroup avrcp details This function is used to send AV/C Panel Subunit PASS THROUGH command with button state set to AVC_PANEL_BUTTON_PRESSED. param channel AVRCP channel. param opid Operation Id. This value can be on of the AVC_PANEL_OPID_... constants return li c TRUE if the function succeeds. li c FALSE otherwise.
	bt_avrcp_send_simple_panel_cmd	brief Send AV/C Panel Subunit PASS THROUGH command ingroup avrcp details This function is used to send AV/C Panel Subunit PASS THROUGH command to the target. param channel AVRCP channel. param ctype Command type. This value can be on of the following values: li AVC_CTYPE_CONTROL 0 li AVC_CTYPE_STATUS 1 li AVC_CTYPE_SPECIFIC_IQUIRY 2 li AVC_CTYPE_NOTIFY 3 li AVC_CTYPE_GENERAL_INQUORY 4 param opid Operation Id. This value can be on of the AVC_PANEL_OPID_... constants param button_state Button state. This can be on of the following values: li AVC_PANEL_BUTTON_PRESSED li AVC_PANEL_BUTTON_RELEASED return li c TRUE if the function succeeds. li c FALSE otherwise.
	bt_avrcp_set_absolute_volume	brief Set absolute volume. ingroup avrcp details This function is used to set an absolute volume to be used by the rendering device. param channel AVRCP channel. param volume Volume return li c TRUE if the function succeeds. li c FALSE otherwise.
	bt_avrcp_set_addressed_player	brief Set addressed player. ingroup avrcp details This function is used to inform the target of which media player the controller wishes to control. param channel AVRCP channel. param player_id Player Id. return li c TRUE if the function succeeds. li c FALSE otherwise.
	bt_avrcp_set_player_application_setting_value	brief Set player setting attribute values. ingroup avrcp details This function requests to set the player application setting list of player application setting values on the target device for the corresponding defined list of setting attributes. for the requested player application setting attribute. The list of attribute value ids is stored in the setting_value_id_list member of the p response_buffer parameter. The caller has to set bt_av_player_setting_values_t::setting_value_id_list to a buffer where returned values will be stored and bt_av_player_setting_values_t::count to the number of entries in the list. param channel AVRCP channel. param attr_id_list List of setting attribute ids. param attr_value_list List of... more
	bt_avrcp_start	brief Start the AVRCP layer. ingroup avrcp details In order to be notified of various events a consumer of the AVRCP layer has to register a callback function. The stack will call the callback function whenever a new event has been generated passing the code of the event as the second parameter. bt_avrcp_start() stores pointers to the c callback and c callback_param in the bt_avrcp_mgr_t structure. param mgr AVRCP manager. param callback The callback function that will be called when the AVRCP generates an event. param callback_param A pointer to arbitrary data to be passed to the c callback callback.... more

	bt_avrcp_tg_send_element_attributes	brief Send media element attributes ingroup avrcp details This function is used to send the system status when AVRCP is running in target mode. If there are active connections with controllers, the ones that registered for AVC_EVENT_SYSTEM_STATUS_CHANGED event will be notified. param status Battery status
	bt_avrcp_tg_set_absolute_volume	brief Set absolute volume ingroup avrcp details This function is used to set the absolute volume when AVRCP is running in target mode. If there are active connections with controllers, the ones that registered for AVC_EVENT_VOLUME_CHANGED event will be notified. param track_id Track Id param song_length The length of the current track. param song_position The position of the current track.
	bt_avrcp_tg_set_battery_status	brief Set battery status ingroup avrcp details This function is used to set the battery status when AVRCP is running in target mode. If there are active connections with controllers, the ones that registered for AVC_EVENT_BATT_STATUS_CHANGED event will be notified. param status Battery status
	bt_avrcp_tg_set_channel_absolute_volume	This is function bt_avrcp_tg_set_channel_absolute_volume.
	bt_avrcp_tg_set_current_track	brief Set current track ingroup avrcp details This function is used to set the current track when AVRCP is running in target mode. If there are active connections with controllers, the ones that registered for AVC_EVENT_TRACK_CHANGED event will be notified. param track_id Track Id param song_length The length of the current track. param song_position The position of the current track.
	bt_avrcp_tg_set_play_status	local target control * • brief Set playback status • ingroup avrcp * • details This function is used to set the playback status when AVRCP is running in target mode. • If there are active connections with controllers, the ones that registered • for AVC_EVENT_PLAYBACK_STATUS_CHANGED event will be notified. * • param song_length The length of the current track. • param song_position The position of the current track. • param play_status Playback status. This value can be one of the following values: • li AVC_PLAY_STATUS_STOPPED • li AVC_PLAY_STATUS_PLAYING • li AVC_PLAY_STATUS_PAUSED • li AVC_PLAY_STATUS_FW_SEEK • li AVC_PLAY_STATUS_REV_SEEK
	bt_avrcp_tg_set_system_status	brief Set system status ingroup avrcp details This function is used to set the system status when AVRCP is running in target mode. If there are active connections with controllers, the ones that registered for AVC_EVENT_SYSTEM_STATUS_CHANGED event will be notified. param status Battery status
	_bt_avrcp_allocate_browsing_response	This is function _bt_avrcp_allocate_browsing_response.
	_bt_avrcp_general_reject	This is function _bt_avrcp_general_reject.
	_bt_avrcp_send_rejected_response	This is function _bt_avrcp_send_rejected_response.

	bt_avrcp_send_release_panel_control	brief Send AV/C Panel Subunit "released" PASS THROUGH command ingroup avrcp details This function is used to send AV/C Panel Subunit PASS THROUGH command with button state set to AVC_PANEL_BUTTON_RELEASED. param channel AVRCP channel. param opid Operation Id. This value can be on of the AVC_PANEL_OPID_... constants return li c TRUE if the function succeeds. li c FALSE otherwise.

Bluetooth Support Data Types and Constants

	Name	Description
	ARG_NOT_USED	This is macro ARG_NOT_USED.
	BT_ASSERT	This is macro BT_ASSERT.
	BT_FALSE	This is macro BT_FALSE.
	BT_LINKKEY_LENGTH	This is macro BT_LINKKEY_LENGTH.
	BT_LOG	This is macro BT_LOG.
	BT_LOG_EX	This is macro BT_LOG_EX.
	BT_LOG_LEVEL_ALL	This is macro BT_LOG_LEVEL_ALL.
	BT_LOG_LEVEL_DEBUG	This is macro BT_LOG_LEVEL_DEBUG.
	BT_LOG_LEVEL_ERROR	This is macro BT_LOG_LEVEL_ERROR.
	BT_LOG_LEVEL_INFO	This is macro BT_LOG_LEVEL_INFO.
	BT_LOG_LEVEL_OFF	This is macro BT_LOG_LEVEL_OFF.
	BT_LOGADDR	This is macro BT_LOGADDR.
	BT_LOGADDR_EX	This is macro BT_LOGADDR_EX.
	BT_LOGINT	This is macro BT_LOGINT.
	BT_LOGINT_EX	This is macro BT_LOGINT_EX.
	BT_LOGLINKKEY	This is macro BT_LOGLINKKEY.
	BT_LOGLINKKEY_EX	This is macro BT_LOGLINKKEY_EX.
	BT_LOGMEMORY	This is macro BT_LOGMEMORY.
	BT_LOGMEMORY_EX	This is macro BT_LOGMEMORY_EX.
	BT_LOGWRITE	This is macro BT_LOGWRITE.
	BT_MAKE_BDADDR	This is macro BT_MAKE_BDADDR.
	BT_MAKE_BDADDR_LE	This is macro BT_MAKE_BDADDR_LE.
	BT_NO	This is macro BT_NO.
	BT_TRUE	This is macro BT_TRUE.
	BT_YES	This is macro BT_YES.
	CDS_LAST_DEVICE_ADDR	This is macro CDS_LAST_DEVICE_ADDR.
	CDS_SIGNATURE	This is macro CDS_SIGNATURE.
	CDS_SIGNATURE_ADDR	This is macro CDS_SIGNATURE_ADDR.
	FALSE	This is macro FALSE.
	HCI_BDADDR_LEN	This is macro HCI_BDADDR_LEN.
	HIDS_LAST_DEVICE_ADDR	This is macro HIDS_LAST_DEVICE_ADDR.
	HIDS_SIGNATURE	This is macro HIDS_SIGNATURE.
	HIDS_SIGNATURE_ADDR	This is macro HIDS_SIGNATURE_ADDR.
	LKS_FIRST_KEY_ADDR	This is macro LKS_FIRST_KEY_ADDR.
	LKS_MAX_LINK_KEYS	This is macro LKS_MAX_LINK_KEYS.
	LKS_SIGNATURE	This is macro LKS_SIGNATURE.
	LKS_SIGNATURE_ADDR	This is macro LKS_SIGNATURE_ADDR.
	LOG	This is macro LOG.
	LOG_EX	This is macro LOG_EX.
	LOGADDR	This is macro LOGADDR.
	LOGADDR_EX	This is macro LOGADDR_EX.
	LOGCLEAR	This is macro LOGCLEAR.
	LOGINT	This is macro LOGINT.
	LOGINT_EX	This is macro LOGINT_EX.
	LOGMEMORY	This is macro LOGMEMORY.

	LOGMEMORY_EX	This is macro LOGMEMORY_EX.
	LOGWRITE	This is macro LOGWRITE.
	NULL	This is macro NULL.
	TRUE	This is macro TRUE.
	_bt_linkkey_notification_t	This is type bt_linkkey_notification_t.
	_bt_linkkey_request_t	This is type bt_linkkey_request_t.
	_bt_linkkey_t	This is type bt_linkkey_t.
	_bt_signal_t	This is record _bt_signal_t.
	_bt_timer_id_enum	This is type bt_timer_id.
	_bt_uuid_s	This is type bt_uuid_t.
	_bt_bdaddr_s	This is type bt_bdaddr_t.
	bt_bdaddr_cp	This is type bt_bdaddr_cp.
	bt_bdaddr_p	This is type bt_bdaddr_p.
	bt_bdaddr_t	This is type bt_bdaddr_t.
	bt_bool	This is type bt_bool.
	bt_byte_cp	This is type bt_byte_cp.
	bt_byte_p	This is type bt_byte_p.
	bt_char	This is type bt_char.
	bt_char_cp	This is type bt_char_cp.
	bt_char_p	This is type bt_char_p.
	bt_id	This is type bt_id.
	bt_int_cp	This is type bt_int_cp.
	bt_int_p	This is type bt_int_p.
	bt_linkkey_cp	This is type bt_linkkey_cp.
	bt_linkkey_notification_t	This is type bt_linkkey_notification_t.
	bt_linkkey_p	This is type bt_linkkey_p.
	bt_linkkey_request_t	This is type bt_linkkey_request_t.
	bt_linkkey_t	This is type bt_linkkey_t.
	bt_long_cp	This is type bt_long_cp.
	bt_long_p	This is type bt_long_p.
	bt_oem_recv_callback_fp	• Receive callback. This callback function is called when a receive operation initiated by bt_oem_recv() has completed. @param len Number of received bytes. The value of this parameter should always be the same as the number of bytes requested in a call to bt_oem_recv().
	bt_oem_send_callback_fp	• Send callback. This callback function is called when a send operation initiated by bt_oem_send() has completed.
	bt_signal_handler_fp	This is type bt_signal_handler_fp.
	bt_signal_t	This is type bt_signal_t.
	bt_storage_callback_fp	brief Storage callback. details This callback is called when a non-volatile storage operation completes.
	bt_sys_callback_fp	• System start callback. This callback function is called when system start initiated by bt_sys_start() has completed. @param success Success of the operation: c BT_TRUE if successfull, c BT_FALSE otherwise. @param param Callback parameter that was specified when bt_sys_start() was called.
	bt_timer_callback_fp	brief Timer callback. details This callback is called when a timer expires.
	bt_timer_id	This is type bt_timer_id.
	bt_uint_cp	This is type bt_uint_cp.
	bt_uint_p	This is type bt_uint_p.
	bt_ulong_cp	This is type bt_ulong_cp.
	bt_ulong_p	This is type bt_ulong_p.
	bt_uuid_cp	This is type bt_uuid_cp.
	bt_uuid_p	This is type bt_uuid_p.

	bt_uuid_t	This is type bt_uuid_t.
	bt_uuid16	This is type bt_uuid16.
	bt_uuid32	This is type bt_uuid32.

Bluetooth Support Functions

	Name	Description
	bt_bdaddr_is_null	This is function bt_bdaddr_is_null.
	bt_bdaddrs_are_equal	This is function bt_bdaddrs_are_equal.
	bt_log_bdaddr	This is function bt_log_bdaddr.
	bt_log_int	This is function bt_log_int.
	bt_log_linkkey	This is function bt_log_linkkey.
	bt_log_memory	This is function bt_log_memory.
	bt_log_msg	This is function bt_log_msg.
	bt_oem_assert	This is function bt_oem_assert.
	bt_oem_get_device_class	This is function bt_oem_get_device_class.
	bt_oem_get_device_name	This is function bt_oem_get_device_name.
	bt_oem_get_pin_code	This is function bt_oem_get_pin_code.
	bt_oem_linkkey_notification	This is function bt_oem_linkkey_notification.
	bt_oem_linkkey_request	This is function bt_oem_linkkey_request.
	bt_oem_log_write	brief Output log message. ingroup log details DotStack calls this function to output its debug information. Implementation should output or store the specified message to whatever device or medium where it can be examined and analyzed.
	bt_oem_recv	• Receive data. This function is called by the HCI layer when it needs more data from the HCI controller. Implementation of this function must receive the specified number of bytes from the HCI controller and call the provided callback function. @param buffer Pointer to a buffer for the received data. The buffer must be long enough to accommodate the number of bytes specified by the par len parameter. @param len Number of bytes to receive. @param callback A callback function that must be called when the requested number of bytes have been received.
	bt_oem_schedule_signals	This is function bt_oem_schedule_signals.
	bt_oem_send	• Send data. This function is called by the HCI layer when it needs to send data to the HCI controller. Implementation of this function must send the specified number of bytes to the HCI controller and call the provided callback function. @param buffer Pointer to the data to be sent . @param len Number of bytes to send. @param callback A callback function that must be called when all data have been sent.
	bt_oem_storage_get_capacity	brief Get non-volatile storage capacity. ingroup stg details Implementation of this function must return the capacity of its non-volatile storage.
	bt_oem_storage_read	brief Read from the non-volatile storage. ingroup stg details This function is called by the stack to read from the non-volatile storage. This function must be implemented by the application. When this function is called the application must start a read operation. When the number of bytes specified by the c len parameter is read, the application must call the callback function specified by the c callback parameter. The application does not have to read the whole number of bytes during the call to this function. It may complete reading later and then call the completion callback. The stack guarantees... more
	bt_oem_storage_start	brief Begin a sequence of non-volatile storage operations. ingroup stg details DotStack calls this function when it starts a sequence of non-volatile storage operations. When the sequence is finished, DotStack will call bt_oem_storage_stop().
	bt_oem_storage_stop	brief End a sequence of non-volatile storage operations. ingroup stg details DotStack calls this function when it finishes executing a sequence of non-volatile storage operations.

	bt_oem_storage_write	brief Write to non-volatile storage. ingroup stg details This function is called by the stack to write data to the non-volatile storage. This function must be implemented by the application. When this function is called, the application must start writing specified data to the non-volatile storage. When all data has been written, the application must call the callback function passed in the c callback parameter. The application does not have to complete the write operation during the call to this function. It may complete the operation later and then call the callback function. In this case, the application does not... more
	bt_oem_timer_clear	brief Clear timer. details This function must be implemented by the application. When this function is called the application must clear the specified timer. If it is already expired and a callback is currently pending, the application should also take measures to cancel the callback. param timerId ID of the timer to clear.
	bt_oem_timer_set	brief Set timer. details This function must be implemented by the application. When it is called, the application must set the specified timer. When the timer expires, the application must call the passed callback function. The function must not wait until the timer expires. It must set the timer and exit immediately. param timerId ID of the timer to set. param milliseconds Timer interval in milliseconds param callback Timer expiration callback function.
	bt_signal_process_pending	This is function bt_signal_process_pending.
	bt_signal_register	This is function bt_signal_register.
	bt_signal_set	This is function bt_signal_set.
	bt_signal_unregister	This is function bt_signal_unregister.
	bt_sys_get_connectable	This is function bt_sys_get_connectable.
	bt_sys_get_discoverable	This is function bt_sys_get_discoverable.
	bt_sys_get_l2cap_manager	• Get the L2CAP manager. This function returns the L2CAP manager. The L2CAP manager is created as part of the start up sequence. @return The L2CAP manager.
	bt_sys_init	• Initialize the Bluetooth system. This function initializes all internal variables of HCI, L2CAP and SDP modules. It must be called by the application before it can access any functionality provided by the library. In addition to this initialization function the application must call initialization functions of all other profile modules the application is intended to use. E.g., if the application is using the SPP module the bt_spp_init() must be called right after calling bt_sys_init(). This function essentially calls bt_sys_init_ex(HCI_LINK_POLICY_ENABLE_ALL) so all link policy setting are enabled.
	bt_sys_set_modes	This is function bt_sys_set_modes.
	bt_sys_start	• Start the Bluetooth system. After all modules used by the application have been initialized this function should be called to start the Bluetooth system operation. During the start up sequence it will reset and initialize the HCI controller and then create the L2CAP manager. The application will be notified when the start up sequence completes by calling the provided callback function. Also, the caller must provide an SDP database. @param discoverable defines whether the device is discoverable after reset. @param connectable defines whether the device is connectable after reset. @param sdp_db SDP database data. @param sdp_db_len Length of SDP database... more
	HCITR_BCSP_ALLOCATE_BUFFERS	This is function HCITR_BCSP_ALLOCATE_BUFFERS.
	IAPEA_ALLOCATE_BUFFERS	This is function IAPEA_ALLOCATE_BUFFERS.
	bt_oem_ssp_callback	This is function bt_oem_ssp_callback.
	bt_signal_init	This is function bt_signal_init.
	bt_sys_init_ex	• Initialize the Bluetooth system. This function initializes all internal variables of HCI, L2CAP and SDP modules. It must be called by the application before it can access any functionality provided by the library. In addition to this initialization function the application must call initialization functions of all other profile modules the application is intended to use. E.g., if the application is using the SPP module the bt_spp_init() must be called right after calling bt_sys_init(). Also, the caller must provide an SDP database. @param default_link_policy default link policy settings. This is a bitmask that defines the initial value of... more

	HCRP_ALLOCATE_BUFFERS	This is function HCRP_ALLOCATE_BUFFERS.
	MAP_ALLOCATE_BUFFERS	This is function MAP_ALLOCATE_BUFFERS.
	bt_hcitr_3wire_cancel_recv_packet	This is function bt_hcitr_3wire_cancel_recv_packet.
	bt_q_contains	This is function bt_q_contains.

GAP Data Types and Constants

	Name	Description
	_bt_device_t	This is type bt_device_t.
	bt_device_t	This is type bt_device_t.
	bt_find_devices_callback_fp	This is type bt_find_devices_callback_fp.

GAP Functions

	Name	Description
	bt_gap_find_devices	This is function bt_gap_find_devices.

HCI Data Types and Constants

	Name	Description
	bt_hci_add_param_hconn	brief Add connection handle parameter to an HCI command ingroup hci
	bt_hci_add_param_lap	This is macro bt_hci_add_param_lap.
	bt_hci_add_param_uint	brief Add unsigned int parameter to an HCI command ingroup hci
	bt_hci_add_param_ulong	brief Add unsigned long parameter to an HCI command ingroup hci
	bt_hci_authenticate	This is macro bt_hci_authenticate.
	bt_hci_exit_sniff_mode	brief Cancel sniff mode
	bt_hci_get_evt_param_hconn	brief Get connection handle parameter from HCI event ingroup hci
	bt_hci_get_evt_param_uint	This is macro bt_hci_get_evt_param_uint.
	bt_hci_get_evt_param_ulong	This is macro bt_hci_get_evt_param_ulong.
	bt_hci_get_param_hconn	brief Get connection handle parameter from HCI command ingroup hci
	bt_hci_le_set_adevrtising_enable	This is macro bt_hci_le_set_adevrtising_enable.
	bt_hci_role_change	This is macro bt_hci_role_change.
	bt_hci_set_encryption	This is macro bt_hci_set_encryption.
	bt_hci_sniff_mode	brief Put local device to sniff mode
	bt_hci_sniff_subrating	This is macro bt_hci_sniff_subrating.
	bt_hcitr_3wire_init	This is macro bt_hcitr_3wire_init.
	bt_hcitr_3wire_reset	This is macro bt_hcitr_3wire_reset.
	bt_hcitr_bcsp_init	This is macro bt_hcitr_bcsp_init.
	bt_hcitr_bcsp_reset	This is macro bt_hcitr_bcsp_reset.
	COD_MAJOR_AUDIO	This is macro COD_MAJOR_AUDIO.
	COD_MAJOR_COMPUTER	This is macro COD_MAJOR_COMPUTER.
	COD_MAJOR_HEALTH	This is macro COD_MAJOR_HEALTH.
	COD_MAJOR_IMAGING	This is macro COD_MAJOR_IMAGING.
	COD_MAJOR_MISC	This is macro COD_MAJOR_MISC.
	COD_MAJOR_NET_ACCESS_POINT	This is macro COD_MAJOR_NET_ACCESS_POINT.
	COD_MAJOR_PERIPHERAL	This is macro COD_MAJOR_PERIPHERAL.
	COD_MAJOR_PHONE	This is macro COD_MAJOR_PHONE.
	COD_MAJOR_TOY	This is macro COD_MAJOR_TOY.
	COD_MAJOR_UNCATEGORIZED	This is macro COD_MAJOR_UNCATEGORIZED.
	COD_MAJOR_WEARABLE	This is macro COD_MAJOR_WEARABLE.
	COD_MINOR_AV_CAMCORDER	This is macro COD_MINOR_AV_CAMCORDER.
	COD_MINOR_AV_CAR_AUDIO	This is macro COD_MINOR_AV_CAR_AUDIO.
	COD_MINOR_AV_GAMING	This is macro COD_MINOR_AV_GAMING.
	COD_MINOR_AV_HANDSFREE	This is macro COD_MINOR_AV_HANDSFREE.

	COD_MINOR_AV_HEADPHONES	This is macro COD_MINOR_AV_HEADPHONES.
	COD_MINOR_AV_HEADSET	This is macro COD_MINOR_AV_HEADSET.
	COD_MINOR_AV_HIFI_AUDIO	This is macro COD_MINOR_AV_HIFI_AUDIO.
	COD_MINOR_AV_LOUDSPEAKER	This is macro COD_MINOR_AV_LOUDSPEAKER.
	COD_MINOR_AV_MICROPHONE	This is macro COD_MINOR_AV_MICROPHONE.
	COD_MINOR_AV_PORTABLE_AUDIO	This is macro COD_MINOR_AV_PORTABLE_AUDIO.
	COD_MINOR_AV_RESERVED	This is macro COD_MINOR_AV_RESERVED.
	COD_MINOR_AV_RESERVERD2	This is macro COD_MINOR_AV_RESERVERD2.
	COD_MINOR_AV_SET_TOP_BOX	This is macro COD_MINOR_AV_SET_TOP_BOX.
	COD_MINOR_AV_UNCATEGORIZED	This is macro COD_MINOR_AV_UNCATEGORIZED.
	COD_MINOR_AV_VCR	This is macro COD_MINOR_AV_VCR.
	COD_MINOR_AV_VIDE_DISPLAY_AND_LOUDSPEAKER	This is macro COD_MINOR_AV_VIDE_DISPLAY_AND_LOUD SPEAKER.
	COD_MINOR_AV_VIDEO_CAMERA	This is macro COD_MINOR_AV_VIDEO_CAMERA.
	COD_MINOR_AV_VIDEO_CONFERENCING	This is macro COD_MINOR_AV_VIDEO_CONFERENCING.
	COD_MINOR_AV_VIDEO_MONITOR	This is macro COD_MINOR_AV_VIDEO_MONITOR.
	COD_MINOR_COMPUTER_DESKTOP	This is macro COD_MINOR_COMPUTER_DESKTOP.
	COD_MINOR_COMPUTER_HANDHELD	This is macro COD_MINOR_COMPUTER_HANDHELD.
	COD_MINOR_COMPUTER_LAPTOP	This is macro COD_MINOR_COMPUTER_LAPTOP.
	COD_MINOR_COMPUTER_PALMSIZED	This is macro COD_MINOR_COMPUTER_PALMSIZED.
	COD_MINOR_COMPUTER_SERVER	This is macro COD_MINOR_COMPUTER_SERVER.
	COD_MINOR_COMPUTER_UNCATEGORIZED	This is macro COD_MINOR_COMPUTER_UNCATEGORIZED.
	COD_MINOR_COMPUTER_WEARABLE	This is macro COD_MINOR_COMPUTER_WEARABLE.
	COD_MINOR_HEALTH_BPM	This is macro COD_MINOR_HEALTH_BPM.
	COD_MINOR_HEALTH_DATA_DISPLAY	This is macro COD_MINOR_HEALTH_DATA_DISPLAY.
	COD_MINOR_HEALTH_GLUCOSE_METER	This is macro COD_MINOR_HEALTH_GLUCOSE_METER.
	COD_MINOR_HEALTH_HEART_MONITOR	This is macro COD_MINOR_HEALTH_HEART_MONITOR.
	COD_MINOR_HEALTH_PULSE_OXIMETER	This is macro COD_MINOR_HEALTH_PULSE_OXIMETER.
	COD_MINOR_HEALTH_THERMOMETER	This is macro COD_MINOR_HEALTH_THERMOMETER.
	COD_MINOR_HEALTH_UNCATEGORIZED	This is macro COD_MINOR_HEALTH_UNCATEGORIZED.
	COD_MINOR_HEALTH_WEIGHING_SCALE	This is macro COD_MINOR_HEALTH_WEIGHING_SCALE.
	COD_MINOR_IMAGING_CAMERA	This is macro COD_MINOR_IMAGING_CAMERA.
	COD_MINOR_IMAGING_DISPLAY	This is macro COD_MINOR_IMAGING_DISPLAY.
	COD_MINOR_IMAGING_PRINTER	This is macro COD_MINOR_IMAGING_PRINTER.
	COD_MINOR_IMAGING_SCANNER	This is macro COD_MINOR_IMAGING_SCANNER.

	COD_MINOR_IMAGING_UNCATEGORIZED	This is macro COD_MINOR_IMAGING_UNCATEGORIZED.
	COD_MINOR_LAN_01_17	This is macro COD_MINOR_LAN_01_17.
	COD_MINOR_LAN_17_33	This is macro COD_MINOR_LAN_17_33.
	COD_MINOR_LAN_33_50	This is macro COD_MINOR_LAN_33_50.
	COD_MINOR_LAN_50_67	This is macro COD_MINOR_LAN_50_67.
	COD_MINOR_LAN_67_83	This is macro COD_MINOR_LAN_67_83.
	COD_MINOR_LAN_83_99	This is macro COD_MINOR_LAN_83_99.
	COD_MINOR_LAN_FULLY_AVAILABLE	This is macro COD_MINOR_LAN_FULLY_AVAILABLE.
	COD_MINOR_LAN_NO_SERVICE	This is macro COD_MINOR_LAN_NO_SERVICE.
	COD_MINOR_LAN_UNCATEGORIZED	This is macro COD_MINOR_LAN_UNCATEGORIZED.
	COD_MINOR_PERIPHERAL_CARD_READER	This is macro COD_MINOR_PERIPHERAL_CARD_READER.
	COD_MINOR_PERIPHERAL_COMBO	This is macro COD_MINOR_PERIPHERAL_COMBO.
	COD_MINOR_PERIPHERAL_DIGITIZER	This is macro COD_MINOR_PERIPHERAL_DIGITIZER.
	COD_MINOR_PERIPHERAL_GAMEPAD	This is macro COD_MINOR_PERIPHERAL_GAMEPAD.
	COD_MINOR_PERIPHERAL_JOYSTICK	This is macro COD_MINOR_PERIPHERAL_JOYSTICK.
	COD_MINOR_PERIPHERAL_KEYBOARD	This is macro COD_MINOR_PERIPHERAL_KEYBOARD.
	COD_MINOR_PERIPHERAL_MOUSE	This is macro COD_MINOR_PERIPHERAL_MOUSE.
	COD_MINOR_PERIPHERAL_OTHER	This is macro COD_MINOR_PERIPHERAL_OTHER.
	COD_MINOR_PERIPHERAL_REMOTE	This is macro COD_MINOR_PERIPHERAL_REMOTE.
	COD_MINOR_PERIPHERAL_SENSING	This is macro COD_MINOR_PERIPHERAL_SENSING.
	COD_MINOR_PERIPHERAL_UNCATEGORIZED	This is macro COD_MINOR_PERIPHERAL_UNCATEGORIZED.
	COD_MINOR_PHONE_CELLULAR	This is macro COD_MINOR_PHONE_CELLULAR.
	COD_MINOR_PHONE_CORDLESS	This is macro COD_MINOR_PHONE_CORDLESS.
	COD_MINOR_PHONE_ISDN	This is macro COD_MINOR_PHONE_ISDN.
	COD_MINOR_PHONE_SMART	This is macro COD_MINOR_PHONE_SMART.
	COD_MINOR_PHONE_UNCATEGORIZED	This is macro COD_MINOR_PHONE_UNCATEGORIZED.
	COD_MINOR_PHONE_WIREDMODEM	This is macro COD_MINOR_PHONE_WIREDMODEM.
	COD_MINOR_TOY_CONTROLLER	This is macro COD_MINOR_TOY_CONTROLLER.
	COD_MINOR_TOY_DOLL	This is macro COD_MINOR_TOY_DOLL.
	COD_MINOR_TOY_GAME	This is macro COD_MINOR_TOY_GAME.
	COD_MINOR_TOY_ROBOT	This is macro COD_MINOR_TOY_ROBOT.
	COD_MINOR_TOY_UNCATEGORIZED	This is macro COD_MINOR_TOY_UNCATEGORIZED.
	COD_MINOR_TOY_VEHICLE	This is macro COD_MINOR_TOY_VEHICLE.
	COD_MINOR_WEARABLE_GLASSES	This is macro COD_MINOR_WEARABLE_GLASSES.
	COD_MINOR_WEARABLE_HELMET	This is macro COD_MINOR_WEARABLE_HELMET.
	COD_MINOR_WEARABLE_JACKET	This is macro COD_MINOR_WEARABLE_JACKET.

	COD_MINOR_WEARABLE_PAGER	This is macro COD_MINOR_WEARABLE_PAGER.
	COD_MINOR_WEARABLE_UNCATEGORIZED	This is macro COD_MINOR_WEARABLE_UNCATEGORIZED.
	COD_MINOR_WEARABLE_WATCH	This is macro COD_MINOR_WEARABLE_WATCH.
	COS_AUDIO	This is macro COS_AUDIO.
	COS_CAPTURING	This is macro COS_CAPTURING.
	COS_INFORMATION	This is macro COS_INFORMATION.
	COS_LIMITED_DISCOVERABLE_MODE	This is macro COS_LIMITED_DISCOVERABLE_MODE.
	COS_NETWORKING	This is macro COS_NETWORKING.
	COS_OBJECTTRANSFER	This is macro COS_OBJECTTRANSFER.
	COS_POSITIONING	This is macro COS_POSITIONING.
	COS_RENDERING	This is macro COS_RENDERING.
	COS_TELEPHONY	This is macro COS_TELEPHONY.
	HCI_ACCEPT_CONNECTION_REQUEST	This is macro HCI_ACCEPT_CONNECTION_REQUEST.
	HCI_ACCEPT_SYNCH_CONNECTION_REQUEST	This is macro HCI_ACCEPT_SYNCH_CONNECTION_REQUEST.
	HCI_ACL_DATA_HEADER_LEN	This is macro HCI_ACL_DATA_HEADER_LEN.
	HCI_AUTHENTICATION_REQUESTED	This is macro HCI_AUTHENTICATION_REQUESTED.
	HCI_BB_PACKET_TYPE_DH1	This is macro HCI_BB_PACKET_TYPE_DH1.
	HCI_BB_PACKET_TYPE_DH3	This is macro HCI_BB_PACKET_TYPE_DH3.
	HCI_BB_PACKET_TYPE_DH5	This is macro HCI_BB_PACKET_TYPE_DH5.
	HCI_BB_PACKET_TYPE_DM1	This is macro HCI_BB_PACKET_TYPE_DM1.
	HCI_BB_PACKET_TYPE_DM3	This is macro HCI_BB_PACKET_TYPE_DM3.
	HCI_BB_PACKET_TYPE_DM5	This is macro HCI_BB_PACKET_TYPE_DM5.
	HCI_BB_PACKET_TYPE_NO_2_DH1	This is macro HCI_BB_PACKET_TYPE_NO_2_DH1.
	HCI_BB_PACKET_TYPE_NO_2_DH3	This is macro HCI_BB_PACKET_TYPE_NO_2_DH3.
	HCI_BB_PACKET_TYPE_NO_2_DH5	This is macro HCI_BB_PACKET_TYPE_NO_2_DH5.
	HCI_BB_PACKET_TYPE_NO_3_DH1	This is macro HCI_BB_PACKET_TYPE_NO_3_DH1.
	HCI_BB_PACKET_TYPE_NO_3_DH3	This is macro HCI_BB_PACKET_TYPE_NO_3_DH3.
	HCI_BB_PACKET_TYPE_NO_3_DH5	This is macro HCI_BB_PACKET_TYPE_NO_3_DH5.
	HCI_C2H_BROADCAST_NOT_PARCKED_SLAVE	This is macro HCI_C2H_BROADCAST_NOT_PARCKED_SLAVE.
	HCI_C2H_BROADCAST_P2P	This is macro HCI_C2H_BROADCAST_P2P.
	HCI_C2H_BROADCAST_PARCKED_SLAVE	This is macro HCI_C2H_BROADCAST_PARCKED_SLAVE.
	HCI_C2H_BROADCAST_RESERVED	This is macro HCI_C2H_BROADCAST_RESERVED.
	HCI_CHANGE_CONNECTION_LINK_KEY	This is macro HCI_CHANGE_CONNECTION_LINK_KEY.
	HCI_CHANGE_CONNECTION_PACKET_TYPE	This is macro HCI_CHANGE_CONNECTION_PACKET_TYPE.
	HCI_CMD_HEADER_LEN	This is macro HCI_CMD_HEADER_LEN.
	HCI_CMD_STATUS_BEING_SENT	This is macro HCI_CMD_STATUS_BEING_SENT.
	HCI_CMD_STATUS_PENDING	This is macro HCI_CMD_STATUS_PENDING.
	HCI_CMD_STATUS_WAITING_RESPONSE	This is macro HCI_CMD_STATUS_WAITING_RESPONSE.

	HCI_CONFIG_BECOME_MASTER	This is macro HCI_CONFIG_BECOME_MASTER.
	HCI_CONFIG_ENABLE_AUTHENTICATION	This is macro HCI_CONFIG_ENABLE_AUTHENTICATION.
	HCI_CONFIG_ENABLE_ENCRYPTION	This is macro HCI_CONFIG_ENABLE_ENCRYPTION.
	HCI_CONN_ROLE_MASTER	This is macro HCI_CONN_ROLE_MASTER.
	HCI_CONN_ROLE_SLAVE	This is macro HCI_CONN_ROLE_SLAVE.
	HCI_CONN_STATE_AUTHENTICATING	This is macro HCI_CONN_STATE_AUTHENTICATING.
	HCI_CONN_STATE_CLOSED	This is macro HCI_CONN_STATE_CLOSED.
	HCI_CONN_STATE_OPEN	This is macro HCI_CONN_STATE_OPEN.
	HCI_CONN_TYPE_ACL	This is macro HCI_CONN_TYPE_ACL.
	HCI_CONN_TYPE_ESCO	This is macro HCI_CONN_TYPE_ESCO.
	HCI_CONN_TYPE_SCO	This is macro HCI_CONN_TYPE_SCO.
	HCI_CONNECTABLE	This is macro HCI_CONNECTABLE.
	HCI_CONTROLLER	This is macro HCI_CONTROLLER.
	HCI_CREATE_CONNECTION	This is macro HCI_CREATE_CONNECTION.
	HCI_CREATE_CONNECTION_CANCEL	This is macro HCI_CREATE_CONNECTION_CANCEL.
	HCI_CREATE_NEW_UNIT_KEY	This is macro HCI_CREATE_NEW_UNIT_KEY.
	HCI_CTRL_LISTENER_ACL_DATA	This is macro HCI_CTRL_LISTENER_ACL_DATA.
	HCI_CTRL_LISTENER_EVENT	This is macro HCI_CTRL_LISTENER_EVENT.
	HCI_CTRL_LISTENER_SCO_DATA	This is macro HCI_CTRL_LISTENER_SCO_DATA.
	HCI_CTRL_STATE_CLOSED	This is macro HCI_CTRL_STATE_CLOSED.
	HCI_CTRL_STATE_CONNECTABLE	This is macro HCI_CTRL_STATE_CONNECTABLE.
	HCI_CTRL_STATE_DISCOVERABLE	This is macro HCI_CTRL_STATE_DISCOVERABLE.
	HCI_CTRL_STATE_INIT	This is macro HCI_CTRL_STATE_INIT.
	HCI_CTRL_STATE_READY	This is macro HCI_CTRL_STATE_READY.
	HCI_CTRL_STATE_SLEEP	This is macro HCI_CTRL_STATE_SLEEP.
	HCI_CTRL_STATE_WAKING_UP	This is macro HCI_CTRL_STATE_WAKING_UP.
	HCI_DATA_BUFFER_STATE_FREE	This is macro HCI_DATA_BUFFER_STATE_FREE.
	HCI_DATA_BUFFER_STATE_USED	This is macro HCI_DATA_BUFFER_STATE_USED.
	HCI_DATA_STATUS_BEING_SENT	This is macro HCI_DATA_STATUS_BEING_SENT.
	HCI_DATA_STATUS_PENDING	This is macro HCI_DATA_STATUS_PENDING.
	HCI_DEFAULT_ACL_CONFIG	Default value for acl_config parameter of bt_hci_connect()
	HCI_DELETE_STORED_LINK_KEY	This is macro HCI_DELETE_STORED_LINK_KEY.
	HCI_DISCONNECT	This is macro HCI_DISCONNECT.
	HCI_DISCOVERABLE	This is macro HCI_DISCOVERABLE.
	HCI_EIR_FEC_NOT_REQUIRED	This is macro HCI_EIR_FEC_NOT_REQUIRED.
	HCI_EIR_FEC_REQUIRED	This is macro HCI_EIR_FEC_REQUIRED.
	HCI_EIR_TYPE_DEVICE_ID	This is macro HCI_EIR_TYPE_DEVICE_ID.
	HCI_EIR_TYPE_FLAGS	This is macro HCI_EIR_TYPE_FLAGS.
	HCI_EIR_TYPE_LOCAL_NAME_COMPLETE	This is macro HCI_EIR_TYPE_LOCAL_NAME_COMPLETE.
	HCI_EIR_TYPE_LOCAL_NAME_SHORTENED	This is macro HCI_EIR_TYPE_LOCAL_NAME_SHORTENED.
	HCI_EIR_TYPE_MANUFACTURER_SPECIFIC	This is macro HCI_EIR_TYPE_MANUFACTURER_SPECIFIC.
	HCI_EIR_TYPE_OOB_COD	This is macro HCI_EIR_TYPE_OOB_COD.

	HCI_EIR_TYPE_OOB_HASH	This is macro HCI_EIR_TYPE_OOB_HASH.
	HCI_EIR_TYPE_OOB_RANDOMIZER	This is macro HCI_EIR_TYPE_OOB_RANDOMIZER.
	HCI_EIR_TYPE_TX_POWER_LEVEL	This is macro HCI_EIR_TYPE_TX_POWER_LEVEL.
	HCI_EIR_TYPE_UUID128_LIST_COMPLETE	This is macro HCI_EIR_TYPE_UUID128_LIST_COMPLETE.
	HCI_EIR_TYPE_UUID128_LIST_MORE_AVAILABLE	This is macro HCI_EIR_TYPE_UUID128_LIST_MORE_AVAILABLE.
	HCI_EIR_TYPE_UUID16_LIST_COMPLETE	This is macro HCI_EIR_TYPE_UUID16_LIST_COMPLETE.
	HCI_EIR_TYPE_UUID16_LIST_MORE_AVAILABLE	This is macro HCI_EIR_TYPE_UUID16_LIST_MORE_AVAILABLE.
	HCI_EIR_TYPE_UUID32_LIST_COMPLETE	This is macro HCI_EIR_TYPE_UUID32_LIST_COMPLETE.
	HCI_EIR_TYPE_UUID32_LIST_MORE_AVAILABLE	This is macro HCI_EIR_TYPE_UUID32_LIST_MORE_AVAILABLE.
	HCI_ENABLE_DEVICE_UNDER_TEST_MODE	This is macro HCI_ENABLE_DEVICE_UNDER_TEST_MODE.
	HCI_ENCRYPTION_OFF	This is macro HCI_ENCRYPTION_OFF.
	HCI_ENCRYPTION_ON	This is macro HCI_ENCRYPTION_ON.
	HCI_ENHANCED_FLUSH	This is macro HCI_ENHANCED_FLUSH.
	HCI_ERR_ACL_CONN_ALREADY_EXISTS	This is macro HCI_ERR_ACL_CONN_ALREADY_EXISTS.
	HCI_ERR_AUTHENTICATION_FAILURE	This is macro HCI_ERR_AUTHENTICATION_FAILURE.
	HCI_ERR_CONN_REJECT_LIMITED_RESOURCES	This is macro HCI_ERR_CONN_REJECT_LIMITED_RESOURCES.
	HCI_ERR_INVALID_PARAMETERS	This is macro HCI_ERR_INVALID_PARAMETERS.
	HCI_ERR_MEMORY_CAPACITY_EXCEEDED	This is macro HCI_ERR_MEMORY_CAPACITY_EXCEEDED.
	HCI_ERR_SCO_CONN_LIMIT_EXCEEDED	This is macro HCI_ERR_SCO_CONN_LIMIT_EXCEEDED.
	HCI_ERR_SIMPLE_PAIRING_NOT_SUPPORTED	This is macro HCI_ERR_SIMPLE_PAIRING_NOT_SUPPORTED.
	HCI_ERR_SUCCESS	• addtogroup hci • @{ • @name Errors
	HCI_ERR_UNSPECIFIED	This is macro HCI_ERR_UNSPECIFIED.
	HCI_EVT_ALL_HCI_EVENTS	Used in bt_hci_ctrl_register_event_listener to indicate that the listener is called for all HCI events (excluding dotstack internal ones)
	HCI_EVT_AUTHENTICATION_COMPLETE	This is macro HCI_EVT_AUTHENTICATION_COMPLETE.
	HCI_EVT_CHANGE_CONN_LINK_COMPLETE	This is macro HCI_EVT_CHANGE_CONN_LINK_COMPLETE.
	HCI_EVT_CMD_SEND_FINISHED	This is macro HCI_EVT_CMD_SEND_FINISHED.
	HCI_EVT_CMD_SEND_STARTED	dotstack internal events
	HCI_EVT_COMMAND_COMPLETE	This is macro HCI_EVT_COMMAND_COMPLETE.
	HCI_EVT_COMMAND_COMPLETE_PARAM_LEN	This is macro HCI_EVT_COMMAND_COMPLETE_PARAM_LEN.
	HCI_EVT_COMMAND_STATUS	This is macro HCI_EVT_COMMAND_STATUS.
	HCI_EVT_CONN_PACKET_TYPE_CHANGED	This is macro HCI_EVT_CONN_PACKET_TYPE_CHANGED.

	HCI_EVT_CONNECTION_COMPLETE	This is macro HCI_EVT_CONNECTION_COMPLETE.
	HCI_EVT_CONNECTION_REQUEST	This is macro HCI_EVT_CONNECTION_REQUEST.
	HCI_EVT_DATA_BUFFER_OVERFLOW	This is macro HCI_EVT_DATA_BUFFER_OVERFLOW.
	HCI_EVT_DISCONNECTION_COMPLETE	This is macro HCI_EVT_DISCONNECTION_COMPLETE.
	HCI_EVT_ENCRYPTION_CHANGE	This is macro HCI_EVT_ENCRYPTION_CHANGE.
	HCI_EVT_ENCRYPTION_KEY_REFRESH_COMPLETE	This is macro HCI_EVT_ENCRYPTION_KEY_REFRESH_COMPLETE.
	HCI_EVT_ENHANCED_FLUSH_COMPLETE	This is macro HCI_EVT_ENHANCED_FLUSH_COMPLETE.
	HCI_EVT_EXTENDED_INQUIRY_RESULT	This is macro HCI_EVT_EXTENDED_INQUIRY_RESULT.
	HCI_EVT_FIRST	This is macro HCI_EVT_FIRST.
	HCI_EVT_FLOW_SPECIFICATION_COMPLETE	This is macro HCI_EVT_FLOW_SPECIFICATION_COMPLETE.
	HCI_EVT_FLUSH_OCCURED	This is macro HCI_EVT_FLUSH_OCCURED.
	HCI_EVT_HARDWARE_ERROR	This is macro HCI_EVT_HARDWARE_ERROR.
	HCI_EVT_INQUIRY_COMPLETE	• addtogroup hci • @{ • @name Events
	HCI_EVT_INQUIRY_RESULT	This is macro HCI_EVT_INQUIRY_RESULT.
	HCI_EVT_INQUIRY_RESULT_WITH_RSSI	This is macro HCI_EVT_INQUIRY_RESULT_WITH_RSSI.
	HCI_EVT_IO_CAPABILITY_REQUEST	This is macro HCI_EVT_IO_CAPABILITY_REQUEST.
	HCI_EVT_IO_CAPABILITY_RESPONSE	This is macro HCI_EVT_IO_CAPABILITY_RESPONSE.
	HCI_EVT_KEYPRESS_NOTIFICATION	This is macro HCI_EVT_KEYPRESS_NOTIFICATION.
	HCI_EVT_LAST	This is macro HCI_EVT_LAST.
	HCI_EVT_LE_META_EVENT	This is macro HCI_EVT_LE_META_EVENT.
	HCI_EVT_LINK_IS_BUSY	This is macro HCI_EVT_LINK_IS_BUSY.
	HCI_EVT_LINK_IS_IDLE	This is macro HCI_EVT_LINK_IS_IDLE.
	HCI_EVT_LINK_KEY_NOTIFICATION	This is macro HCI_EVT_LINK_KEY_NOTIFICATION.
	HCI_EVT_LINK_KEY_REQUEST	This is macro HCI_EVT_LINK_KEY_REQUEST.
	HCI_EVT_LINK_SUPERVISION_TO_CHANGED	This is macro HCI_EVT_LINK_SUPERVISION_TO_CHANGED.
	HCI_EVT_LOOPBACK_COMMAND	This is macro HCI_EVT_LOOPBACK_COMMAND.
	HCI_EVT_MASTER_LINK_KEY_COMPLETE	This is macro HCI_EVT_MASTER_LINK_KEY_COMPLETE.
	HCI_EVT_MAX_SLOTS_CHANGE	This is macro HCI_EVT_MAX_SLOTS_CHANGE.
	HCI_EVT_MODE_CHANGE	This is macro HCI_EVT_MODE_CHANGE.
	HCI_EVT_NUM_OF_COMPLETED_PACKETS	This is macro HCI_EVT_NUM_OF_COMPLETED_PACKETS.
	HCI_EVT_PAGE_SCAN_REPET_MODE_CHANGE	This is macro HCI_EVT_PAGE_SCAN_REPET_MODE_CHANGE.
	HCI_EVT_PIN_CODE_REQUEST	This is macro HCI_EVT_PIN_CODE_REQUEST.
	HCI_EVT_QOS_SETUP_COMPLETE	This is macro HCI_EVT_QOS_SETUP_COMPLETE.
	HCI_EVT_QOS_VIOLATION	This is macro HCI_EVT_QOS_VIOLATION.

	HCI_EVT_READ_CLOCK_OFFSET_COMPLETE	This is macro HCI_EVT_READ_CLOCK_OFFSET_COMPLETE.
	HCI_EVT_READ_RMT_EXT_FEATURES_COMP	This is macro HCI_EVT_READ_RMT_EXT_FEATURES_COMP.
	HCI_EVT_READ_RMT_SUP_FEATURES_COMP	This is macro HCI_EVT_READ_RMT_SUP_FEATURES_COMP.
	HCI_EVT_READ_RMT_VERSION_INFO_COMP	This is macro HCI_EVT_READ_RMT_VERSION_INFO_COMP.
	HCI_EVT_REMOTE_NAME_REQUEST_COMPLETE	This is macro HCI_EVT_REMOTE_NAME_REQUEST_COMPLETE.
	HCI_EVT_REMOTE_OOB_DATA_REQUEST	This is macro HCI_EVT_REMOTE_OOB_DATA_REQUEST.
	HCI_EVT_RETURN_LINK_KEYS	This is macro HCI_EVT_RETURN_LINK_KEYS.
	HCI_EVT_RMT_HOST_SUPP_FEATURES_NTF	This is macro HCI_EVT_RMT_HOST_SUPP_FEATURES_NTF.
	HCI_EVT_ROLE_CHANGE	This is macro HCI_EVT_ROLE_CHANGE.
	HCI_EVT_SIMPLE_PAIRING_COMPLETE	This is macro HCI_EVT_SIMPLE_PAIRING_COMPLETE.
	HCI_EVT_SNIFF_SUBRATING	This is macro HCI_EVT_SNIFF_SUBRATING.
	HCI_EVT_SYNCH_CONNECTION_CHANGED	This is macro HCI_EVT_SYNCH_CONNECTION_CHANGED.
	HCI_EVT_SYNCH_CONNECTION_COMPLETE	This is macro HCI_EVT_SYNCH_CONNECTION_COMPLETE.
	HCI_EVT_USER_CONFIRMATION_REQUEST	This is macro HCI_EVT_USER_CONFIRMATION_REQUEST.
	HCI_EVT_USER_PASSKEY_NOTIFICATION	This is macro HCI_EVT_USER_PASSKEY_NOTIFICATION.
	HCI_EVT_USER_PASSKEY_REQUEST	This is macro HCI_EVT_USER_PASSKEY_REQUEST.
	HCI_EXIT_PARK_STATE	This is macro HCI_EXIT_PARK_STATE.
	HCI_EXIT_PERIODIC_INQUIRY_MODE	This is macro HCI_EXIT_PERIODIC_INQUIRY_MODE.
	HCI_EXIT_SNIFF_MODE	This is macro HCI_EXIT_SNIFF_MODE.
	HCI_FLOW_SPECIFICATION	This is macro HCI_FLOW_SPECIFICATION.
	HCI_FLUSH	This is macro HCI_FLUSH.
	HCI_H2C_BROADCAST_ACTIVE_SLAVE	This is macro HCI_H2C_BROADCAST_ACTIVE_SLAVE.
	HCI_H2C_BROADCAST_NO_BROADCATS	This is macro HCI_H2C_BROADCAST_NO_BROADCATS.
	HCI_H2C_BROADCAST_PARCKED_SLAVE	This is macro HCI_H2C_BROADCAST_PARCKED_SLAVE.
	HCI_H2C_BROADCAST_RESERVED	This is macro HCI_H2C_BROADCAST_RESERVED.
	HCI_HOLD_MODE	addtogroup hci @{ @name Link policy commands details The Link Policy Commands provide methods for the Host to affect how the Link Manager manages the piconet.
	HCI_HOST_BUFFER_SIZE	This is macro HCI_HOST_BUFFER_SIZE.
	HCI_HOST_NUM_OF_COMPLETED_PACKETS	This is macro HCI_HOST_NUM_OF_COMPLETED_PACKETS.
	HCI_INQUIRY	addtogroup hci @{ @name Link control commands details The Link Control commands allow a Controller to control connections to other BR/EDR Controllers.

	HCI_INQUIRY_CANCEL	This is macro HCI_INQUIRY_CANCEL.
	HCI_INQUIRY_MODE_EXTENDED	This is macro HCI_INQUIRY_MODE_EXTENDED.
	HCI_INQUIRY_MODE_STANDARD	This is macro HCI_INQUIRY_MODE_STANDARD.
	HCI_INQUIRY_MODE_WITH_RSSI	This is macro HCI_INQUIRY_MODE_WITH_RSSI.
	HCI_IO_CAPABILITY_REQUEST_NEGATIVE_REPLY	This is macro HCI_IO_CAPABILITY_REQUEST_NEGATIVE_REPLY.
	HCI_IO_CAPABILITY_REQUEST_REPLY	This is macro HCI_IO_CAPABILITY_REQUEST_REPLY.
	HCI_LE_ADD_DEVICE_TO_WHITE_LIST	This is macro HCI_LE_ADD_DEVICE_TO_WHITE_LIST.
	HCI_LE_ADDRESS_TYPE_PUBLIC	This is macro HCI_LE_ADDRESS_TYPE_PUBLIC.
	HCI_LE_ADDRESS_TYPE_RANDOM	This is macro HCI_LE_ADDRESS_TYPE_RANDOM.
	HCI_LE_ADVERTISING_FLAG_BREDR_NOT_SUPPORTED	This is macro HCI_LE_ADVERTISING_FLAG_BREDR_NOT_SUPPORTED.
	HCI_LE_ADVERTISING_FLAG_GENERAL_DISCOVERABLE_MODE	This is macro HCI_LE_ADVERTISING_FLAG_GENERAL_DISCOVERABLE_MODE.
	HCI_LE_ADVERTISING_FLAG_LIMITED_DISCOVERABLE_MODE	This is macro HCI_LE_ADVERTISING_FLAG_LIMITED_DISCOVERABLE_MODE.
	HCI_LE_ADVERTISING_FLAG_SIMULTANEOUS_LE_BREDR_CONTROLLER	This is macro HCI_LE_ADVERTISING_FLAG_SIMULTANEOUS_LE_BREDR_CONTROLLER.
	HCI_LE_ADVERTISING_FLAG_SIMULTANEOUS_LE_BREDR_HOST	This is macro HCI_LE_ADVERTISING_FLAG_SIMULTANEOUS_LE_BREDR_HOST.
	HCI_LE_AVERTISING_DISABLED	This is macro HCI_LE_AVERTISING_DISABLED.
	HCI_LE_AVERTISING_ENABLED	This is macro HCI_LE_AVERTISING_ENABLED.
	HCI_LE_CLEAR_WHITE_LIST	This is macro HCI_LE_CLEAR_WHITE_LIST.
	HCI_LE_CONNECTION_UPDATE	This is macro HCI_LE_CONNECTION_UPDATE.
	HCI_LE_CREATE_CONNECTION	This is macro HCI_LE_CREATE_CONNECTION.
	HCI_LE_CREATE_CONNECTION_CANCEL	This is macro HCI_LE_CREATE_CONNECTION_CANCEL.
	HCI_LE_DISABLED	This is macro HCI_LE_DISABLED.
	HCI_LE_DUPLICATE_FILTERING_DISABLED	This is macro HCI_LE_DUPLICATE_FILTERING_DISABLED.
	HCI_LE_DUPLICATE_FILTERING_ENABLED	This is macro HCI_LE_DUPLICATE_FILTERING_ENABLED.
	HCI_LE_ENABLED	This is macro HCI_LE_ENABLED.
	HCI_LE_ENCRYPT	This is macro HCI_LE_ENCRYPT.
	HCI_LE_EVT_ADVERTISING_REPORT	This is macro HCI_LE_EVT_ADVERTISING_REPORT.
	HCI_LE_EVT_CONNECTION_COMPLETE	This is macro HCI_LE_EVT_CONNECTION_COMPLETE.
	HCI_LE_EVT_CONNECTION_UPDATE_COMPLETE	This is macro HCI_LE_EVT_CONNECTION_UPDATE_COMPLETE.
	HCI_LE_EVT_LONG_TERM_KEY_REQUEST	This is macro HCI_LE_EVT_LONG_TERM_KEY_REQUEST.
	HCI_LE_EVT_READ_REMOTE_USED_FEATURES_COMPLETE	This is macro HCI_LE_EVT_READ_REMOTE_USED_FEATURES_COMPLETE.
	HCI_LE_FILTER_POLICY_NOT_USED	White list is not used to determine which advertiser to connect to. Peer_Address_Type and Peer_Address shall be used.

	HCI_LE_FILTER_POLICY_WHITE_LIST	White list is used to determine which advertiser to connect to. Peer_Address_Type and Peer_Address shall be ignored
	HCI_LE_FLAG_SHARED_ACL_BUFFERS	This is macro HCI_LE_FLAG_SHARED_ACL_BUFFERS.
	HCI_LE_FLAG_SIMULTANEOUS_LE_BREDR	This is macro HCI_LE_FLAG_SIMULTANEOUS_LE_BREDR.
	HCI_LE_LONG_TERM_KEY_REQUEST_NEGATIVE_REPLY	This is macro HCI_LE_LONG_TERM_KEY_REQUEST_NEGATIVE_REPLY.
	HCI_LE_LONG_TERM_KEY_REQUEST_REPLY	This is macro HCI_LE_LONG_TERM_KEY_REQUEST_REPLY.
	HCI_LE_MAX_AD_LEN	This is macro HCI_LE_MAX_AD_LEN.
	HCI_LE_RAND	This is macro HCI_LE_RAND.
	HCI_LE_READ_ADVERTISING_CHANNEL_TX_POWER	This is macro HCI_LE_READ_ADVERTISING_CHANNEL_TX_POWER.
	HCI_LE_READ_BUFFER_SIZE	This is macro HCI_LE_READ_BUFFER_SIZE.
	HCI_LE_READ_CHANNEL_MAP	This is macro HCI_LE_READ_CHANNEL_MAP.
	HCI_LE_READ_LOCAL_SUPPORTED_FEATURES	This is macro HCI_LE_READ_LOCAL_SUPPORTED_FEATURES.
	HCI_LE_READ_REMOTE_USED_FEATURES	This is macro HCI_LE_READ_REMOTE_USED_FEATURES.
	HCI_LE_READ_SUPPORTED_STATES	This is macro HCI_LE_READ_SUPPORTED_STATES.
	HCI_LE_READ_WHITE_LIST_SIZE	This is macro HCI_LE_READ_WHITE_LIST_SIZE.
	HCI_LE_RECEIVE_TEST	This is macro HCI_LE_RECEIVE_TEST.
	HCI_LE_REMOVE_DEVICE_FROM_WHITE_LIST	This is macro HCI_LE_REMOVE_DEVICE_FROM_WHITE_LIST.
	HCI_LE_SCAN_DISABLED	This is macro HCI_LE_SCAN_DISABLED.
	HCI_LE_SCAN_ENABLED	This is macro HCI_LE_SCAN_ENABLED.
	HCI_LE_SCAN_EVT_CANCEL_FAILED	This is macro HCI_LE_SCAN_EVT_CANCEL_FAILED.
	HCI_LE_SCAN_EVT_CANCELLED	This is macro HCI_LE_SCAN_EVT_CANCELLED.
	HCI_LE_SCAN_EVT_DEVICE_FOUND	This is macro HCI_LE_SCAN_EVT_DEVICE_FOUND.
	HCI_LE_SCAN_EVT_START_FAILED	This is macro HCI_LE_SCAN_EVT_START_FAILED.
	HCI_LE_SCAN_EVT_STARTED	This is macro HCI_LE_SCAN_EVT_STARTED.
	HCI_LE_SCAN_FILTER_POLICY_ALL	This is macro HCI_LE_SCAN_FILTER_POLICY_ALL.
	HCI_LE_SCAN_FILTER_POLICY_WHITE_LIST	This is macro HCI_LE_SCAN_FILTER_POLICY_WHITE_LIST.
	HCI_LE_SCAN_TYPE_ACTIVE	This is macro HCI_LE_SCAN_TYPE_ACTIVE.
	HCI_LE_SCAN_TYPE_PASSIVE	This is macro HCI_LE_SCAN_TYPE_PASSIVE.
	HCI_LE_SET_ADVERTISE_ENABLE	This is macro HCI_LE_SET_ADVERTISE_ENABLE.
	HCI_LE_SET_ADVERTISING_DATA	This is macro HCI_LE_SET_ADVERTISING_DATA.
	HCI_LE_SET_ADVERTISING_PARAMETERS	This is macro HCI_LE_SET_ADVERTISING_PARAMETERS.
	HCI_LE_SET_EVENT_MASK	addtogroup hci @{ @name LE controller commands details The LE Controller Commands provide access and control to various capabilities of the Bluetooth hardware, as well as methods for the Host to affect how the Link Layer manages the piconet, and controls connections.

	HCI_LE_SET_HOST_CHANNEL_CLASSIFICATION	This is macro HCI_LE_SET_HOST_CHANNEL_CLASSIFICATION.
	HCI_LE_SET_RANDOM_ADDRESS	This is macro HCI_LE_SET_RANDOM_ADDRESS.
	HCI_LE_SET_SCAN_ENABLE	This is macro HCI_LE_SET_SCAN_ENABLE.
	HCI_LE_SET_SCAN_PARAMETERS	This is macro HCI_LE_SET_SCAN_PARAMETERS.
	HCI_LE_SET_SCAN_RESPONSE_DATA	This is macro HCI_LE_SET_SCAN_RESPONSE_DATA.
	HCI_LE_SIMULTANEOUS_DISABLED	This is macro HCI_LE_SIMULTANEOUS_DISABLED.
	HCI_LE_SIMULTANEOUS_ENABLED	This is macro HCI_LE_SIMULTANEOUS_ENABLED.
	HCI_LE_START_ENCRYPTION	This is macro HCI_LE_START_ENCRYPTION.
	HCI_LE_TEST_END	This is macro HCI_LE_TEST_END.
	HCI_LE_TRANSMITTER_TEST	This is macro HCI_LE_TRANSMITTER_TEST.
	HCI_LINK_KEY_LEN	This is macro HCI_LINK_KEY_LEN.
	HCI_LINK_KEY_REQUEST_NEGATIVE_REPLY	This is macro HCI_LINK_KEY_REQUEST_NEGATIVE_REPLY.
	HCI_LINK_KEY_REQUEST_REPLY	This is macro HCI_LINK_KEY_REQUEST_REPLY.
	HCI_LINK_KEY_TYPE_COMBINATION	This is macro HCI_LINK_KEY_TYPE_COMBINATION.
	HCI_LINK_KEY_TYPE_LOCAL_UNIT	This is macro HCI_LINK_KEY_TYPE_LOCAL_UNIT.
	HCI_LINK_KEY_TYPE_REMOTE_UNIT	This is macro HCI_LINK_KEY_TYPE_REMOTE_UNIT.
	HCI_LINK_POLICY_ENABLE_HOLD_MODE	This is macro HCI_LINK_POLICY_ENABLE_HOLD_MODE.
	HCI_LINK_POLICY_ENABLE_PARK_STATE	This is macro HCI_LINK_POLICY_ENABLE_PARK_STATE.
	HCI_LINK_POLICY_ENABLE_ROLE_SWITCH	This is macro HCI_LINK_POLICY_ENABLE_ROLE_SWITCH.
	HCI_LINK_POLICY_ENABLE_SNIFF_MODE	This is macro HCI_LINK_POLICY_ENABLE_SNIFF_MODE.
	HCI_LINK_TYPE_BD_EDR	This is macro HCI_LINK_TYPE_BD_EDR.
	HCI_LINK_TYPE_LE	This is macro HCI_LINK_TYPE_LE.
	HCI_MASTER_LINK_KEY	This is macro HCI_MASTER_LINK_KEY.
	HCI_MAX_DATA_BUFFERS	This is macro HCI_MAX_DATA_BUFFERS.
	HCI_MAX_EVENT_PARAM_LEN	This is macro HCI_MAX_EVENT_PARAM_LEN.
	HCI_MAX_PARAM_LEN	This is macro HCI_MAX_PARAM_LEN.
	HCI_MAX_PIN_LENGTH	This is macro HCI_MAX_PIN_LENGTH.
	HCI_OPCODE	This is macro HCI_OPCODE.
	HCI_PACKET_BOUNDARY_CONTINUE	This is macro HCI_PACKET_BOUNDARY_CONTINUE.
	HCI_PACKET_BOUNDARY_FIRST	This is macro HCI_PACKET_BOUNDARY_FIRST.
	HCI_PACKET_BOUNDARY_FIRST_AUTO_FLUSH	This is macro HCI_PACKET_BOUNDARY_FIRST_AUTO_FLUSH.
	HCI_PACKET_BOUNDARY_FIRST_NO_AUTO_FLUSH	This is macro HCI_PACKET_BOUNDARY_FIRST_NO_AUTO_FLUSH.
	HCI_PACKET_TYPE_ACL_DATA	This is macro HCI_PACKET_TYPE_ACL_DATA.
	HCI_PACKET_TYPE_COMMAND	This is macro HCI_PACKET_TYPE_COMMAND.
	HCI_PACKET_TYPE_EVENT	This is macro HCI_PACKET_TYPE_EVENT.
	HCI_PACKET_TYPE_NONE	This is macro HCI_PACKET_TYPE_NONE.
	HCI_PACKET_TYPE_SCO_DATA	This is macro HCI_PACKET_TYPE_SCO_DATA.

	HCI_PAGE_SCAN_REPETITION_MODE_R0	This is macro HCI_PAGE_SCAN_REPETITION_MODE_R0.
	HCI_PAGE_SCAN_REPETITION_MODE_R1	This is macro HCI_PAGE_SCAN_REPETITION_MODE_R1.
	HCI_PAGE_SCAN_REPETITION_MODE_R2	This is macro HCI_PAGE_SCAN_REPETITION_MODE_R2.
	HCI_PARAM_LEN_BD_ADDR	This is macro HCI_PARAM_LEN_BD_ADDR.
	HCI_PARAM_LEN_BYTE	This is macro HCI_PARAM_LEN_BYTE.
	HCI_PARAM_LEN_DEV_CLASS	This is macro HCI_PARAM_LEN_DEV_CLASS.
	HCI_PARAM_LEN_HANDLE	This is macro HCI_PARAM_LEN_HANDLE.
	HCI_PARAM_LEN_INT	This is macro HCI_PARAM_LEN_INT.
	HCI_PARAM_LEN_LONG	This is macro HCI_PARAM_LEN_LONG.
	HCI_PARAM_TYPE_BD_ADDR	This is macro HCI_PARAM_TYPE_BD_ADDR.
	HCI_PARAM_TYPE_BYTE	This is macro HCI_PARAM_TYPE_BYTE.
	HCI_PARAM_TYPE_DEV_CLASS	This is macro HCI_PARAM_TYPE_DEV_CLASS.
	HCI_PARAM_TYPE_HANDLE	This is macro HCI_PARAM_TYPE_HANDLE.
	HCI_PARAM_TYPE_INT	This is macro HCI_PARAM_TYPE_INT.
	HCI_PARAM_TYPE_LONG	This is macro HCI_PARAM_TYPE_LONG.
	HCI_PARAM_TYPE_STRING	This is macro HCI_PARAM_TYPE_STRING.
	HCI_PARK_STATE	This is macro HCI_PARK_STATE.
	HCI_PERIODIC_INQUIRY_MODE	This is macro HCI_PERIODIC_INQUIRY_MODE.
	HCI_PIN_CODE_REQUEST_NEGATIVE_REPLY	This is macro HCI_PIN_CODE_REQUEST_NEGATIVE_REPLY.
	HCI_PIN_CODE_REQUEST_REPLY	This is macro HCI_PIN_CODE_REQUEST_REPLY.
	HCI_POWER_MODE_ACTIVE	This is macro HCI_POWER_MODE_ACTIVE.
	HCI_POWER_MODE_HOLD	This is macro HCI_POWER_MODE_HOLD.
	HCI_POWER_MODE_PARK	This is macro HCI_POWER_MODE_PARK.
	HCI_POWER_MODE_SNIFF	This is macro HCI_POWER_MODE_SNIFF.
	HCI_QOS_SETUP	This is macro HCI_QOS_SETUP.
	HCI_READ_AFH_CHANNEL_ASSESSMENT_MODE	This is macro HCI_READ_AFH_CHANNEL_ASSESSMENT_MODE.
	HCI_READ_AFH_CHANNEL_MAP	This is macro HCI_READ_AFH_CHANNEL_MAP.
	HCI_READ_AUTHENTICATION_ENABLE	This is macro HCI_READ_AUTHENTICATION_ENABLE.
	HCI_READ_AUTOMATIC_FLASH_TIMEOUT	This is macro HCI_READ_AUTOMATIC_FLASH_TIMEOUT.
	HCI_READ_BD_ADDR	This is macro HCI_READ_BD_ADDR.
	HCI_READ_BUFFER_SIZE	This is macro HCI_READ_BUFFER_SIZE.
	HCI_READ_CLASS_OF_DEVICE	This is macro HCI_READ_CLASS_OF_DEVICE.
	HCI_READ_CLOCK_COMMAND	This is macro HCI_READ_CLOCK_COMMAND.
	HCI_READ_CLOCK_OFFSET	This is macro HCI_READ_CLOCK_OFFSET.
	HCI_READ_CONNECTION_ACCEPT_TIMEOUT	This is macro HCI_READ_CONNECTION_ACCEPT_TIMEOUT.
	HCI_READ_CURRENT_IAC_LAP	This is macro HCI_READ_CURRENT_IAC_LAP.
	HCI_READ_DEFAULT_ERRONEOUS_DATA_REPORTING	This is macro HCI_READ_DEFAULT_ERRONEOUS_DATA_REPORTING.
	HCI_READ_DEFAULT_POLICY_SETTINGS	This is macro HCI_READ_DEFAULT_POLICY_SETTINGS.
	HCI_READ_ENCRYPTION_MODE	This is macro HCI_READ_ENCRYPTION_MODE.
	HCI_READ_EXTENDED_INQUIRY_RESPONSE	This is macro HCI_READ_EXTENDED_INQUIRY_RESPONSE.

	HCI_READ_FAILED_CONTACT_COUNTER	addtogroup hci @{ @name Status Parameters details The Controller modifies all status parameters. These parameters provide information about the current state of the Link Manager and Baseband in the BR/EDR Controller and the PAL in an AMP Controller. The host device cannot modify any of these parameters other than to reset certain specific parameters.
	HCI_READ_HOLD_MODE_ACTIVITY	This is macro HCI_READ_HOLD_MODE_ACTIVITY.
	HCI_READ_INQUIRY_MODE	This is macro HCI_READ_INQUIRY_MODE.
	HCI_READ_INQUIRY_RESPONSE_TX_POWER_LEVEL	This is macro HCI_READ_INQUIRY_RESPONSE_TX_POWER_LEVEL.
	HCI_READ_INQUIRY_SCAN_ACTIVITY	This is macro HCI_READ_INQUIRY_SCAN_ACTIVITY.
	HCI_READ_INQUIRY_SCAN_TYPE	This is macro HCI_READ_INQUIRY_SCAN_TYPE.
	HCI_READ_LE_HOST_SUPPORT	This is macro HCI_READ_LE_HOST_SUPPORT.
	HCI_READ_LINK_POLICY_SETTINGS	This is macro HCI_READ_LINK_POLICY_SETTINGS.
	HCI_READ_LINK_QUALITY	This is macro HCI_READ_LINK_QUALITY.
	HCI_READ_LINK_SUPERVISION_TIMEOUT	This is macro HCI_READ_LINK_SUPERVISION_TIMEOUT.
	HCI_READ_LMP_HANDLE	This is macro HCI_READ_LMP_HANDLE.
	HCI_READ_LOCAL_EXTENDED_FEATURES	This is macro HCI_READ_LOCAL_EXTENDED_FEATURES.
	HCI_READ_LOCAL_NAME	This is macro HCI_READ_LOCAL_NAME.
	HCI_READ_LOCAL_OOB_DATA	This is macro HCI_READ_LOCAL_OOB_DATA.
	HCI_READ_LOCAL_SUPPORTED_COMMANDS	This is macro HCI_READ_LOCAL_SUPPORTED_COMMANDS.
	HCI_READ_LOCAL_SUPPORTED_FEATURES	This is macro HCI_READ_LOCAL_SUPPORTED_FEATURES.
	HCI_READ_LOCAL_VERSION_INFORMATION	addtogroup hci @{ @name Informational Parameters details The Informational Parameters are fixed by the manufacturer of the Bluetooth hardware. These parameters provide information about the BR/EDR Controller and the capabilities of the Link Manager and Baseband in the BR/EDR Controller and PAL in the AMP Controller. The host device cannot modify any of these parameters.
	HCI_READ_LOOPBACK_MODE	addtogroup hci @{ @name Testing Commands details The Testing commands are used to provide the ability to test various functional capabilities of the Bluetooth hardware.
	HCI_READ_NUM_BROADCST_RETR	This is macro HCI_READ_NUM_BROADCST_RETR.
	HCI_READ_NUM_OF_SUPPORTED_IAC	This is macro HCI_READ_NUM_OF_SUPPORTED_IAC.
	HCI_READ_PAGE_SCAN_ACTIVITY	This is macro HCI_READ_PAGE_SCAN_ACTIVITY.
	HCI_READ_PAGE_SCAN_PERIOD_MODE	This is macro HCI_READ_PAGE_SCAN_PERIOD_MODE.
	HCI_READ_PAGE_SCAN_TYPE	This is macro HCI_READ_PAGE_SCAN_TYPE.
	HCI_READ_PAGE_TIMEOUT	This is macro HCI_READ_PAGE_TIMEOUT.
	HCI_READ_PIN_TYPE	This is macro HCI_READ_PIN_TYPE.

	HCI_READ_REFRESH_ENCRYPTION_KEY	This is macro HCI_READ_REFRESH_ENCRYPTION_KEY.
	HCI_READ_REMOTE_EXTENDED_FEATURES	This is macro HCI_READ_REMOTE_EXTENDED_FEATURES.
	HCI_READ_REMOTE_SUPPORTED_FEATURES	This is macro HCI_READ_REMOTE_SUPPORTED_FEATURES.
	HCI_READ_REMOTE_VERSION_INFORMATION	This is macro HCI_READ_REMOTE_VERSION_INFORMATION.
	HCI_READ_RSSI	This is macro HCI_READ_RSSI.
	HCI_READ_SCAN_ENABLE	This is macro HCI_READ_SCAN_ENABLE.
	HCI_READ_SIMPLE_PAIRING_MODE	This is macro HCI_READ_SIMPLE_PAIRING_MODE.
	HCI_READ_STORED_LINK_KEY	This is macro HCI_READ_STORED_LINK_KEY.
	HCI_READ_SYNC_FLOW_CONTROL_ENABLE	This is macro HCI_READ_SYNC_FLOW_CONTROL_ENABLE.
	HCI_READ_TRANSMIT_POWER_LEVEL	This is macro HCI_READ_TRANSMIT_POWER_LEVEL.
	HCI_READ_VOICE_SETTING	This is macro HCI_READ_VOICE_SETTING.
	HCI_REJECT_CONNECTION_REQUEST	This is macro HCI_REJECT_CONNECTION_REQUEST.
	HCI_REJECT_SYNCH_CONNECTION_REQUEST	This is macro HCI_REJECT_SYNCH_CONNECTION_REQUEST.
	HCI_REMOTE_NAME_REQUEST	This is macro HCI_REMOTE_NAME_REQUEST.
	HCI_REMOTE_NAME_REQUEST_CANCEL	This is macro HCI_REMOTE_NAME_REQUEST_CANCEL.
	HCI_REMOTE_OOB_DATA_REQUEST_NEGATIVE_REPLY	This is macro HCI_REMOTE_OOB_DATA_REQUEST_NEGATIVE_REPLY.
	HCI_REMOTE_OOB_DATA_REQUEST_REPLY	This is macro HCI_REMOTE_OOB_DATA_REQUEST_REPLY.
	HCI_RESET	This is macro HCI_RESET.
	HCI_RESET_FAILED_CONTACT_COUNTER	This is macro HCI_RESET_FAILED_CONTACT_COUNTER.
	HCI_ROLE_DISCOVERY	This is macro HCI_ROLE_DISCOVERY.
	HCI_ROLE_SWITCH_ALLOW	This is macro HCI_ROLE_SWITCH_ALLOW.
	HCI_ROLE_SWITCH_DISALLOW	This is macro HCI_ROLE_SWITCH_DISALLOW.
	HCI_SCAN_INQUIRY	Flags for HCI_READ_SCAN_ENABLE/HCI_WRITE_SCAN_ENABLE
	HCI_SCAN_PAGE	This is macro HCI_SCAN_PAGE.
	HCI_SCO_CONTENT_FORMAT_AIR_CODING_A_LAW	This is macro HCI_SCO_CONTENT_FORMAT_AIR_CODING_A_LAW.
	HCI_SCO_CONTENT_FORMAT_AIR_CODING_CSVD	This is macro HCI_SCO_CONTENT_FORMAT_AIR_CODING_CSVD.
	HCI_SCO_CONTENT_FORMAT_AIR_CODING_N_LAW	This is macro HCI_SCO_CONTENT_FORMAT_AIR_CODING_N_LAW.
	HCI_SCO_CONTENT_FORMAT_AIR_CODING_TRANSPARENT	This is macro HCI_SCO_CONTENT_FORMAT_AIR_CODING_TRANSPARENT.
	HCI_SCO_CONTENT_FORMAT_INPUT_CODING_A_LAW	This is macro HCI_SCO_CONTENT_FORMAT_INPUT_CODING_A_LAW.
	HCI_SCO_CONTENT_FORMAT_INPUT_CODING_LINEAR	This is macro HCI_SCO_CONTENT_FORMAT_INPUT_CODING_LINEAR.

	HCI_SCO_CONTENT_FORMAT_INPUT_CODING_N_LAW	This is macro HCI_SCO_CONTENT_FORMAT_INPUT_CODING_N_LAW.
	HCI_SCO_CONTENT_FORMAT_INPUT_FORMAT_1_COMPLEMENT	This is macro HCI_SCO_CONTENT_FORMAT_INPUT_FORMAT_1_COMPLEMENT.
	HCI_SCO_CONTENT_FORMAT_INPUT_FORMAT_2_COMPLEMENT	This is macro HCI_SCO_CONTENT_FORMAT_INPUT_FORMAT_2_COMPLEMENT.
	HCI_SCO_CONTENT_FORMAT_INPUT_FORMAT_SIGN_MAGNITUDE	This is macro HCI_SCO_CONTENT_FORMAT_INPUT_FORMAT_SIGN_MAGNITUDE.
	HCI_SCO_CONTENT_FORMAT_INPUT_FORMAT_UNSIGNED	This is macro HCI_SCO_CONTENT_FORMAT_INPUT_FORMAT_UNSIGNED.
	HCI_SCO_CONTENT_FORMAT_INPUT_SAMPLE_SIZE_16	This is macro HCI_SCO_CONTENT_FORMAT_INPUT_SAMPLE_SIZE_16.
	HCI_SCO_CONTENT_FORMAT_INPUT_SAMPLE_SIZE_8	This is macro HCI_SCO_CONTENT_FORMAT_INPUT_SAMPLE_SIZE_8.
	HCI_SCO_DATA_HEADER_LEN	This is macro HCI_SCO_DATA_HEADER_LEN.
	HCI_SCO_MAX_DATA_LEN	This is macro HCI_SCO_MAX_DATA_LEN.
	HCI_SCO_MAX_LATENCY_DONTCARE	This is macro HCI_SCO_MAX_LATENCY_DONTCARE.
	HCI_SCO_PACKET_TYPE_ALL	This is macro HCI_SCO_PACKET_TYPE_ALL.
	HCI_SCO_PACKET_TYPE_EV3	This is macro HCI_SCO_PACKET_TYPE_EV3.
	HCI_SCO_PACKET_TYPE_EV4	This is macro HCI_SCO_PACKET_TYPE_EV4.
	HCI_SCO_PACKET_TYPE_EV5	This is macro HCI_SCO_PACKET_TYPE_EV5.
	HCI_SCO_PACKET_TYPE_HV1	This is macro HCI_SCO_PACKET_TYPE_HV1.
	HCI_SCO_PACKET_TYPE_HV2	This is macro HCI_SCO_PACKET_TYPE_HV2.
	HCI_SCO_PACKET_TYPE_HV3	This is macro HCI_SCO_PACKET_TYPE_HV3.
	HCI_SCO_PACKET_TYPE_NO_2_EV3	This is macro HCI_SCO_PACKET_TYPE_NO_2_EV3.
	HCI_SCO_PACKET_TYPE_NO_2_EV5	This is macro HCI_SCO_PACKET_TYPE_NO_2_EV5.
	HCI_SCO_PACKET_TYPE_NO_3_EV3	This is macro HCI_SCO_PACKET_TYPE_NO_3_EV3.
	HCI_SCO_PACKET_TYPE_NO_3_EV5	This is macro HCI_SCO_PACKET_TYPE_NO_3_EV5.
	HCI_SCO_RTX_EFFORT_DONTCARE	This is macro HCI_SCO_RTX_EFFORT_DONTCARE.
	HCI_SCO_RTX_EFFORT_NO_RETRANSMISSION	This is macro HCI_SCO_RTX_EFFORT_NO_RETRANSMISSION.
	HCI_SCO_RTX_EFFORT_OPTIMIZE_LINK_QUALITY	This is macro HCI_SCO_RTX_EFFORT_OPTIMIZE_LINK_QUALITY.
	HCI_SCO_RTX_EFFORT_OPTIMIZE_POWER_CONSUMPTION	This is macro HCI_SCO_RTX_EFFORT_OPTIMIZE_POWER_CONSUMPTION.
	HCI_SCO_RX_BANDWIDTH_DONTCARE	This is macro HCI_SCO_RX_BANDWIDTH_DONTCARE.
	HCI_SCO_TX_BANDWIDTH_DONTCARE	This is macro HCI_SCO_TX_BANDWIDTH_DONTCARE.
	HCI_SEND_DATA_STATUS_INTERRUPTED	This is macro HCI_SEND_DATA_STATUS_INTERRUPTED.
	HCI_SEND_DATA_STATUS_SUCCESS	This is macro HCI_SEND_DATA_STATUS_SUCCESS.
	HCI_SEND_KEY_PRESS_NOTIFICATION	This is macro HCI_SEND_KEY_PRESS_NOTIFICATION.
	HCI_SET_AFH_HOST_CHANNEL_CLASSIFICATION	This is macro HCI_SET_AFH_HOST_CHANNEL_CLASSIFICATION.

	HCI_SET_CONNECTION_ENCRYPTION	This is macro HCI_SET_CONNECTION_ENCRYPTION.
	HCI_SET_CTRL_TO_HOST_FLOW_CONTROL	This is macro HCI_SET_CTRL_TO_HOST_FLOW_CONTROL.
	HCI_SET_EVENT_FILTER	This is macro HCI_SET_EVENT_FILTER.
	HCI_SET_EVENT_MASK	addtogroup hci @{ @name Controller & Baseband commands details The Controller & Baseband Commands provide access and control to various capabilities of the Bluetooth hardware.
	HCI_SETUP_SYNCHRONOUS_CONNECTION	This is macro HCI_SETUP_SYNCHRONOUS_CONNECTION.
	HCI_SNIFF_MODE	This is macro HCI_SNIFF_MODE.
	HCI_SNIFF_SUBRATING	This is macro HCI_SNIFF_SUBRATING.
	HCI_SWITCH_ROLE	This is macro HCI_SWITCH_ROLE.
	HCI_TRANSPORT_HEADER_LEN	This is macro HCI_TRANSPORT_HEADER_LEN.
	HCI_UART_PACKET_TYPE_ACL_DATA	This is macro HCI_UART_PACKET_TYPE_ACL_DATA.
	HCI_UART_PACKET_TYPE_COMMAND	defgroup hcitr_uart HCI UART (H4) transport protocol ingroup hcitr details This module describes functions used to initialize and start HCI UART transport protocol. The transport uses common interface for exchanging data between the host CPU and HCI controller defined in bt_hcitr.h. This interface consist of two functions that must be implemented by the application: li bt_oem_send() li bt_oem_recv()
	HCI_UART_PACKET_TYPE_EVENT	This is macro HCI_UART_PACKET_TYPE_EVENT.
	HCI_UART_PACKET_TYPE_SCO_DATA	This is macro HCI_UART_PACKET_TYPE_SCO_DATA.
	HCI_USER_CONFIRMATION_REQ_NEGATIVE_REPLY	This is macro HCI_USER_CONFIRMATION_REQ_NEGATIVE_REPLY.
	HCI_USER_CONFIRMATION_REQUEST_REPLY	This is macro HCI_USER_CONFIRMATION_REQUEST_REPLY.
	HCI_USER_PASSKEY_REQUEST_NEGATIVE_REPLY	This is macro HCI_USER_PASSKEY_REQUEST_NEGATIVE_REPLY.
	HCI_USER_PASSKEY_REQUEST_REPLY	This is macro HCI_USER_PASSKEY_REQUEST_REPLY.
	HCI_WRITE_AFH_CHANNEL_ASSESSMENT_MODE	This is macro HCI_WRITE_AFH_CHANNEL_ASSESSMENT_MODE.
	HCI_WRITE_AUTHENTICATION_ENABLE	This is macro HCI_WRITE_AUTHENTICATION_ENABLE.
	HCI_WRITE_AUTOMATIC_FLASH_TIMEOUT	This is macro HCI_WRITE_AUTOMATIC_FLASH_TIMEOUT.
	HCI_WRITE_CLASS_OF_DEVICE	This is macro HCI_WRITE_CLASS_OF_DEVICE.
	HCI_WRITE_CONNECTION_ACCEPT_TIMEOUT	This is macro HCI_WRITE_CONNECTION_ACCEPT_TIMEOUT.
	HCI_WRITE_CURRENT_IAC_LAP	This is macro HCI_WRITE_CURRENT_IAC_LAP.
	HCI_WRITE_DEFAULT_ERRONEOUS_DATA_REPORTING	This is macro HCI_WRITE_DEFAULT_ERRONEOUS_DATA_REPORTING.
	HCI_WRITE_DEFAULT_POLICY_SETTINGS	This is macro HCI_WRITE_DEFAULT_POLICY_SETTINGS.
	HCI_WRITE_ENCRYPTION_MODE	This is macro HCI_WRITE_ENCRYPTION_MODE.

	HCI_WRITE_EXTENDED_INQUIRY_RESPONSE	This is macro HCI_WRITE_EXTENDED_INQUIRY_RESPONSE.
	HCI_WRITE_EXTENDED_INQUIRY_RESPONSE_PARAM_LEN	This is macro HCI_WRITE_EXTENDED_INQUIRY_RESPONSE_PARAM_LEN.
	HCI_WRITE_HOLD_MODE_ACTIVITY	This is macro HCI_WRITE_HOLD_MODE_ACTIVITY.
	HCI_WRITE_INQUIRY_MODE	This is macro HCI_WRITE_INQUIRY_MODE.
	HCI_WRITE_INQUIRY_SCAN_ACTIVITY	This is macro HCI_WRITE_INQUIRY_SCAN_ACTIVITY.
	HCI_WRITE_INQUIRY_SCAN_TYPE	This is macro HCI_WRITE_INQUIRY_SCAN_TYPE.
	HCI_WRITE_INQUIRY_TX_POWER_LEVEL	This is macro HCI_WRITE_INQUIRY_TX_POWER_LEVEL.
	HCI_WRITE_LE_HOST_SUPPORT	This is macro HCI_WRITE_LE_HOST_SUPPORT.
	HCI_WRITE_LINK_POLICY_SETTINGS	This is macro HCI_WRITE_LINK_POLICY_SETTINGS.
	HCI_WRITE_LINK_SUPERVISION_TIMEOUT	This is macro HCI_WRITE_LINK_SUPERVISION_TIMEOUT.
	HCI_WRITE_LOCAL_NAME	This is macro HCI_WRITE_LOCAL_NAME.
	HCI_WRITE_LOCAL_NAME_PARAM_LEN	This is macro HCI_WRITE_LOCAL_NAME_PARAM_LEN.
	HCI_WRITE_LOOPBACK_MODE	This is macro HCI_WRITE_LOOPBACK_MODE.
	HCI_WRITE_NUM_BROADCST_RETR	This is macro HCI_WRITE_NUM_BROADCST_RETR.
	HCI_WRITE_PAGE_SCAN_ACTIVITY	This is macro HCI_WRITE_PAGE_SCAN_ACTIVITY.
	HCI_WRITE_PAGE_SCAN_PERIOD_MODE	This is macro HCI_WRITE_PAGE_SCAN_PERIOD_MODE.
	HCI_WRITE_PAGE_SCAN_TYPE	This is macro HCI_WRITE_PAGE_SCAN_TYPE.
	HCI_WRITE_PAGE_TIMEOUT	This is macro HCI_WRITE_PAGE_TIMEOUT.
	HCI_WRITE_PIN_TYPE	This is macro HCI_WRITE_PIN_TYPE.
	HCI_WRITE_SCAN_ENABLE	This is macro HCI_WRITE_SCAN_ENABLE.
	HCI_WRITE_SIMPLE_PAIRING_DEBUG_MODE	This is macro HCI_WRITE_SIMPLE_PAIRING_DEBUG_MODE.
	HCI_WRITE_SIMPLE_PAIRING_MODE	This is macro HCI_WRITE_SIMPLE_PAIRING_MODE.
	HCI_WRITE_STORED_LINK_KEY	This is macro HCI_WRITE_STORED_LINK_KEY.
	HCI_WRITE_SYNC_FLOW_CONTROL_ENABLE	This is macro HCI_WRITE_SYNC_FLOW_CONTROL_ENABLE.
	HCI_WRITE_VOICE_SETTING	This is macro HCI_WRITE_VOICE_SETTING.
	HCITR_3WIRE_DEFAULT_ACK_TIMEOUT	This is macro HCITR_3WIRE_DEFAULT_ACK_TIMEOUT.
	HCITR_BCSP_DEFAULT_ACK_TIMEOUT	This is macro HCITR_BCSP_DEFAULT_ACK_TIMEOUT.
	LM_PACKET_TYPE_DH1	This is macro LM_PACKET_TYPE_DH1.
	LM_PACKET_TYPE_DH3	This is macro LM_PACKET_TYPE_DH3.
	LM_PACKET_TYPE_DH5	This is macro LM_PACKET_TYPE_DH5.
	LM_PACKET_TYPE_DM1	This is macro LM_PACKET_TYPE_DM1.
	LM_PACKET_TYPE_DM3	This is macro LM_PACKET_TYPE_DM3.
	LM_PACKET_TYPE_DM5	This is macro LM_PACKET_TYPE_DM5.
	OGF_CTRL_BASEBAND	This is macro OGF_CTRL_BASEBAND.
	OGF_INFORMATION	This is macro OGF_INFORMATION.
	OGF_LE	This is macro OGF_LE.
	OGF_LINK_CONTROL	This is macro OGF_LINK_CONTROL.
	OGF_LINK_POLICY	This is macro OGF_LINK_POLICY.

	OGF_STATUS	This is macro OGF_STATUS.
	OGF_TESTING	This is macro OGF_TESTING.
	OGF_VENDOR	This is macro OGF_VENDOR.
	bt_hci_cmd_callback_fp	This is type bt_hci_cmd_callback_fp.
	bt_hci_cmd_listener_fp	This is type bt_hci_cmd_listener_fp.
	bt_hci_command_p	This is type bt_hci_command_p.
	bt_hci_command_t	This is type bt_hci_command_t.
	bt_hci_conn_state_p	This is type bt_hci_conn_state_p.
	bt_hci_conn_state_t	This is type bt_hci_conn_state_t.
	bt_hci_connect_callback_fp	brief HCI connect callback. ingroup hci details This typedef defines a type for the callback function that is called when HCI connect operation initiated by a call to bt_hci_connect() is complete. param status Operation status. c It is 0 if connection was successfully established. param pconn pointer to a structure representing the established connection. param param pointer to arbitrary data passed to the bt_hci_connect() function through its c param parameter.
	bt_hci_ctrl_listener_t	This is type bt_hci_ctrl_listener_t.
	bt_hci_ctrl_state_t	This is type bt_hci_ctrl_state_t.
	bt_hci_data_buffer_p	This is type bt_hci_data_buffer_p.
	bt_hci_data_buffer_t	This is type bt_hci_data_buffer_t.
	bt_hci_data_callback_fp	This is type bt_hci_data_callback_fp.
	bt_hci_data_listener_fp	This is type bt_hci_data_listener_fp.
	bt_hci_data_p	This is type bt_hci_data_p.
	bt_hci_data_t	This is type bt_hci_data_t.
	bt_hci_disconnect_callback_fp	brief HCI disconnect callback. ingroup hci details This typedef defines a type for the callback function that is called when an HCI connection has been terminated. param status Operation status. c It is 0 if connection has been successfully terminated. param reason Reason for disconnection. param pconn pointer to a structure representing the connection. param param pointer to arbitrary data associated with an event listener.
	bt_hci_event_e	This is type bt_hci_event_e.
	bt_hci_event_handler_ex_fp	This is type bt_hci_event_handler_ex_fp.
	bt_hci_event_handler_fp	This is type bt_hci_event_handler_fp.
	bt_hci_event_listener_fp	This is type bt_hci_event_listener_fp.
	bt_hci_event_p	This is type bt_hci_event_p.
	bt_hci_event_t	This is type bt_hci_event_t.
	bt_hci_evt_authentication_complete_t	This is type bt_hci_evt_authentication_complete_t.
	bt_hci_evt_command_complete_t	This is type bt_hci_evt_command_complete_t.
	bt_hci_evt_command_status_t	This is type bt_hci_evt_command_status_t.
	bt_hci_evt_connection_complete_t	This is type bt_hci_evt_connection_complete_t.
	bt_hci_evt_connection_request_t	This is type bt_hci_evt_connection_request_t.
	bt_hci_evt_disconnection_complete_t	This is type bt_hci_evt_disconnection_complete_t.
	bt_hci_evt_encryption_change_t	This is type bt_hci_evt_encryption_change_t.
	bt_hci_evt_mode_change_t	This is type bt_hci_evt_mode_change_t.
	bt_hci_evt_role_change_t	This is type bt_hci_evt_role_change_t.
	bt_hci_hconn_p	This is type bt_hci_hconn_p.
	bt_hci_hconn_t	This is type bt_hci_hconn_t.
	bt_hci_inquiry_callback_fp	This is type bt_hci_inquiry_callback_fp.
	bt_hci_inquiry_response_t	This is type bt_hci_inquiry_response_t.
	bt_hci_le_advertising_report_t	This is type bt_hci_le_advertising_report_t.

	bt_hci_le_conn_state_t	This is type bt_hci_le_conn_state_t.
	bt_hci_le_connect_parameters_t	This is type bt_hci_le_connect_parameters_t.
	bt_hci_le_ctrl_state_t	This is type bt_hci_le_ctrl_state_t.
	bt_hci_le_evt_connection_updated_t	This is type bt_hci_le_evt_connection_updated_t.
	bt_hci_le_evt_read_remote_used_features_completed_t	This is type bt_hci_le_evt_read_remote_used_features_completed_t.
	bt_hci_le_evt_read_support_params_t	This is type bt_hci_le_evt_read_support_params_t.
	bt_hci_le_scan_callback_fp	This is type bt_hci_le_scan_callback_fp.
	bt_hci_link_key_t	This is type bt_hci_link_key_t.
	bt_hci_listener_t	This is type bt_hci_listener_t.
	bt_hci_read_inquiry_mode_callback_fp	This is type bt_hci_read_inquiry_mode_callback_fp.
	bt_hci_read_inquiry_scan_activity_callback_fp	This is type bt_hci_read_inquiry_scan_activity_callback_fp.
	bt_hci_read_inquiry_scan_type_callback_fp	This is type bt_hci_read_inquiry_scan_type_callback_fp.
	bt_hci_read_page_scan_activity_callback_fp	This is type bt_hci_read_page_scan_activity_callback_fp.
	bt_hci_read_page_scan_period_mode_callback_fp	This is type bt_hci_read_page_scan_period_mode_callback_fp.
	bt_hci_read_page_scan_type_callback_fp	This is type bt_hci_read_page_scan_type_callback_fp.
	bt_hci_read_page_timeout_callback_fp	This is type bt_hci_read_page_timeout_callback_fp.
	bt_hci_request_remote_name_callback_fp	This is type bt_hci_request_remote_name_callback_fp.
	bt_hci_sco_read_data_callback_fp	This is type bt_hci_sco_read_data_callback_fp.
	bt_hci_start_callback_fp	brief HCI initialization callback. ingroup hci details This typedef defines a function pointer type for the HCI initialization callback functions. Such a function must be passed to the bt_hci_start() function. param success Specifies whether HCI initialization succeeded or not.
	bt_hci_stop_callback_fp	brief HCI stop callback. ingroup hci details This typedef defines a function pointer type for the HCI stop callback functions. Such a function must be passed to the bt_hci_stop() function.
	bt_hci_transport_recv_packet_callback_fp	This is type bt_hci_transport_recv_packet_callback_fp.
	bt_hci_transport_send_packet_callback_fp	This is type bt_hci_transport_send_packet_callback_fp.
	bt_hcitr_tih4_power_callback_fp	This is type bt_hcitr_tih4_power_callback_fp.
	bt_hcitr_tih4_power_event_e	This is type bt_hcitr_tih4_power_event_e.
	bt_le_evt_handler	This is type bt_le_evt_handler.
	hci_transport_t	This is type hci_transport_t.
	pf_hci_sleep_callback	This is type pf_hci_sleep_callback.
	pf_hci_wakeup_callback	This is type pf_hci_wakeup_callback.
	pf_l2cap_receive_callback	This is type pf_l2cap_receive_callback.
	_bt_hci_command_s	This is type bt_hci_command_t.
	_bt_hci_conn_state_s	This is record _bt_hci_conn_state_s.
	_bt_hci_ctrl_listener_t	This is record _bt_hci_ctrl_listener_t.
	_bt_hci_ctrl_state_s	This is type bt_hci_ctrl_state_t.
	_bt_hci_data_buffer_s	This is type bt_hci_data_buffer_t.
	_bt_hci_data_s	This is type bt_hci_data_t.
	_bt_hci_event_e	This is type bt_hci_event_e.
	_bt_hci_event_s	This is type bt_hci_event_t.

	_bt_hci_evt_authentication_complete_t	This is type bt_hci_evt_authentication_complete_t.
	_bt_hci_evt_command_complete_t	This is type bt_hci_evt_command_complete_t.
	_bt_hci_evt_command_status_t	This is type bt_hci_evt_command_status_t.
	_bt_hci_evt_connection_complete_t	This is type bt_hci_evt_connection_complete_t.
	_bt_hci_evt_connection_request_t	This is type bt_hci_evt_connection_request_t.
	_bt_hci_evt_disconnection_complete_t	This is type bt_hci_evt_disconnection_complete_t.
	_bt_hci_evt_encryption_change_s	This is type bt_hci_evt_encryption_change_t.
	_bt_hci_evt_mode_change_t	This is type bt_hci_evt_mode_change_t.
	_bt_hci_evt_role_change_t	This is type bt_hci_evt_role_change_t.
	_bt_hci_inquiry_response_t	This is type bt_hci_inquiry_response_t.
	_bt_hci_le_advertising_report_t	This is record _bt_hci_le_advertising_report_t.
	_bt_hci_le_conn_state_t	This is type bt_hci_le_conn_state_t.
	_bt_hci_le_connect_parameters_t	This is type bt_hci_le_connect_parameters_t.
	_bt_hci_le_ctrl_state_t	This is type bt_hci_le_ctrl_state_t.
	_bt_hci_le_evt_connection_updated_t	This is type bt_hci_le_evt_connection_updated_t.
	_bt_hci_le_evt_read_remote_used_features_completed_t	This is type bt_hci_le_evt_read_remote_used_features_completed_t.
	_bt_hci_le_evt_read_support_params_t	This is type bt_hci_le_evt_read_support_params_t.
	_bt_hci_link_key_s	This is type bt_hci_link_key_t.
	_bt_hci_listener_t	This is record _bt_hci_listener_t.
	_hci_transport_t	This is type hci_transport_t.
	_hcitr_tih4_power_event_e	This is type bt_hcitr_tih4_power_event_e.
	HCI_DISCOVERABLE_MODE_GENERAL	This is macro HCI_DISCOVERABLE_MODE_GENERAL.
	HCI_DISCOVERABLE_MODE_LIMITED	This is macro HCI_DISCOVERABLE_MODE_LIMITED.
	HCI_EIR_TYPE_3D_Information_Data	This is macro HCI_EIR_TYPE_3D_Information_Data.
	HCI_EIR_TYPE_ADVERTISING_INTERVAL	This is macro HCI_EIR_TYPE_ADVERTISING_INTERVAL.
	HCI_EIR_TYPE_APPEARANCE	This is macro HCI_EIR_TYPE_APPEARANCE.
	HCI_EIR_TYPE_LE_BLUETOOTH_DEVICE_ADDRESS	This is macro HCI_EIR_TYPE_LE_BLUETOOTH_DEVICE_ADDRESS.
	HCI_EIR_TYPE_LE_ROLE	This is macro HCI_EIR_TYPE_LE_ROLE.
	HCI_EIR_TYPE_LE_SECURE_CONNECTIONS_CONFIRMATION_VALUE	This is macro HCI_EIR_TYPE_LE_SECURE_CONNECTIONS_CONFIRMATION_VALUE.
	HCI_EIR_TYPE_LE_SECURE_CONNECTIONS_RANDOM_VALUE	This is macro HCI_EIR_TYPE_LE_SECURE_CONNECTIONS_RANDOM_VALUE.
	HCI_EIR_TYPE_PUBLIC_TARGET_ADDRESS	This is macro HCI_EIR_TYPE_PUBLIC_TARGET_ADDRESS.
	HCI_EIR_TYPE_RANDOM_TARGET_ADDRESS	This is macro HCI_EIR_TYPE_RANDOM_TARGET_ADDRESS.
	HCI_EIR_TYPE_SERVICE_DATA	This is macro HCI_EIR_TYPE_SERVICE_DATA.
	HCI_EIR_TYPE_SERVICE_DATA_UUID128	This is macro HCI_EIR_TYPE_SERVICE_DATA_UUID128.
	HCI_EIR_TYPE_SERVICE_DATA_UUID16	This is macro HCI_EIR_TYPE_SERVICE_DATA_UUID16.
	HCI_EIR_TYPE_SERVICE_DATA_UUID32	This is macro HCI_EIR_TYPE_SERVICE_DATA_UUID32.
	HCI_EIR_TYPE_SIMPLE_PAIRING_HASH_C_256	This is macro HCI_EIR_TYPE_SIMPLE_PAIRING_HASH_C_256.

	HCI_EIR_TYPE_SIMPLE_PAIRING_RANDOMIZER_R_256	This is macro HCI_EIR_TYPE_SIMPLE_PAIRING_RANDOMIZER_R_256.
	HCI_EIR_TYPE_SLAVE_CONN_INTERVAL_RANGE	This is macro HCI_EIR_TYPE_SLAVE_CONN_INTERVAL_RANGE.
	HCI_EIR_TYPE_SM_OOB_FLAGS	This is macro HCI_EIR_TYPE_SM_OOB_FLAGS.
	HCI_EIR_TYPE_SM_TK_VALUE	This is macro HCI_EIR_TYPE_SM_TK_VALUE.
	HCI_EIR_TYPE_SOLICITATION_UUID128_LIST	This is macro HCI_EIR_TYPE_SOLICITATION_UUID128_LIST.
	HCI_EIR_TYPE_SOLICITATION_UUID16_LIST	This is macro HCI_EIR_TYPE_SOLICITATION_UUID16_LIST.
	HCI_EIR_TYPE_SOLICITATION_UUID32_LIST	This is macro HCI_EIR_TYPE_SOLICITATION_UUID32_LIST.
	HCI_ERR_CONNECTION_TIMEOUT	This is macro HCI_ERR_CONNECTION_TIMEOUT.
	HCI_INIT_FLAG_IGNORE_TOTAL_NUM_ACL_DATA_PACKETS	The total number of ACL data packets read from the controller will be ignored. The stack will assume that he controller can accept only 1 ACL packet and sends HCI_EVT_NUM_OF_COMPLETED_PACKETS for each packet it has processed. This seems to be for controller working over SDIO (at least for Marvell's 88W8777).
	HCI_INIT_FLAG_SEND_HCI_RESET	HCI reset will be sent when bt_hci_init is called
	HCI_LE_ADV_CHANNEL_MAP_ENABLE_37	This is macro HCI_LE_ADV_CHANNEL_MAP_ENABLE_37.
	HCI_LE_ADV_CHANNEL_MAP_ENABLE_38	This is macro HCI_LE_ADV_CHANNEL_MAP_ENABLE_38.
	HCI_LE_ADV_CHANNEL_MAP_ENABLE_39	This is macro HCI_LE_ADV_CHANNEL_MAP_ENABLE_39.
	HCI_LE_ADV_CHANNEL_MAP_ENABLE_ALL	This is macro HCI_LE_ADV_CHANNEL_MAP_ENABLE_ALL.
	HCI_LE_ADV_CHANNEL_MAP_RESERVED	This is macro HCI_LE_ADV_CHANNEL_MAP_RESERVED.
	HCI_LE_ADV_TYPE_CONN_UNDIRECT	Connectable undirected advertising (ADV_IND) (default)
	HCI_LE_ADV_TYPE_DIRECT_HIGH	Connectable high duty cycle directed advertising (ADV_DIRECT_IND, high duty cycle)
	HCI_LE_ADV_TYPE_DIRECT_LOW	Connectable low duty cycle directed advertising (ADV_DIRECT_IND, low duty cycle)
	HCI_LE_ADV_TYPE_NONCONN	Non connectable undirected advertising (ADV_NONCONN_IND)
	HCI_LE_ADV_TYPE_SCAN	Scannable undirected advertising (ADV_SCAN_IND)
	HCI_LE_RANDOM_ADDRESS_TYPE_NON_RESOLVABLE	This is macro HCI_LE_RANDOM_ADDRESS_TYPE_NON_RESOLVABLE.
	HCI_LE_RANDOM_ADDRESS_TYPE_RESOLVABLE	This is macro HCI_LE_RANDOM_ADDRESS_TYPE_RESOLVABLE.
	HCI_LE_RANDOM_ADDRESS_TYPE_STATIC	This is macro HCI_LE_RANDOM_ADDRESS_TYPE_STATIC.
	HCI_LINK_POLICY_ENABLE_ALL	This is macro HCI_LINK_POLICY_ENABLE_ALL.
	bt_hci_le_cancel_connect	This is macro bt_hci_le_cancel_connect.
	bt_hci_send_pin_code	This is macro bt_hci_send_pin_code.
	HCI_ERR_PAIRING_NOT_ALLOWED	This is macro HCI_ERR_PAIRING_NOT_ALLOWED.
	HCI_ERR_PIN_OR_KEY_MISSING	This is macro HCI_ERR_PIN_OR_KEY_MISSING.

	HCI_PACKET_BOUNDARY_COMPLETE	This is macro HCI_PACKET_BOUNDARY_COMPLETE.
	HCI_SUCCESS	This is macro HCI_SUCCESS.

HCI Functions

	Name	Description
	bt_hci_add_param_bdaddr	brief Add BD address parameter to an HCI command ingroup hci
	bt_hci_add_param_byte	brief Add byte parameter to an HCI command ingroup hci
	bt_hci_add_param_cod	brief Add class of device parameter to an HCI command ingroup hci
	bt_hci_add_param_int	brief Add int parameter to an HCI command ingroup hci
	bt_hci_add_param_linkkey	brief Add link key parameter to an HCI command ingroup hci
	bt_hci_add_param_long	brief Add long parameter to an HCI command ingroup hci
	bt_hci_add_param_string	brief Add string parameter to an HCI command ingroup hci
	bt_hci_alloc_canned_command	brief Allocate and initialize an HCI command structure for a canned (pre-formatted) command. ingroup hci
	bt_hci_alloc_command	brief Allocate and initialize an HCI command structure. ingroup hci
	bt_hci_alloc_data_buffer	This is function bt_hci_alloc_data_buffer.
	bt_hci_allocate_write_eir_command	This is function bt_hci_allocate_write_eir_command.
	bt_hci_authenticate_ex	This is function bt_hci_authenticate_ex.
	bt_hci_cancel_find_devices	brief Stop inquiry
	bt_hci_cancel_request_remote_name	brief Cancel remote device name request
	bt_hci_cancel_send_acl_data	brief Cancel sending data over ACL connection
	bt_hci_connect	brief Connect to a remote device. ingroup hci details This function tries to establish an HCI connection with a remote device specified by the Bluetooth address c dest. Upon completion, the callback function specified by the c callback parameter is called. param dest Bluetooth address of the remote device. param packet_type param role_switch param acl_config param callback Pointer to a callback function that is called when the connect operation completes. param param Pointer to arbitrary data that is to be passed to the callback function. return li c TRUE when the function succeeds. li c FALSE otherwise. The callback function... more
	bt_hci_connect_sco	This is function bt_hci_connect_sco.
	bt_hci_ctrl_register_data_listener	This is function bt_hci_ctrl_register_data_listener.
	bt_hci_ctrl_register_listener	This is function bt_hci_ctrl_register_listener.
	bt_hci_ctrl_unregister_listener	This is function bt_hci_ctrl_unregister_listener.
	bt_hci_disconnect	brief Abort connection ingroup hci
	bt_hci_evt_authentication_complete_handler	This is function bt_hci_evt_authentication_complete_handler.
	bt_hci_evt_change_conn_link_complete_handler	This is function bt_hci_evt_change_conn_link_complete_handler.
	bt_hci_evt_command_complete_handler	This is function bt_hci_evt_command_complete_handler.
	bt_hci_evt_command_status_handler	This is function bt_hci_evt_command_status_handler.
	bt_hci_evt_conn_packet_type_changed_handler	This is function bt_hci_evt_conn_packet_type_changed_handler.
	bt_hci_evt_connection_complete_handler	This is function bt_hci_evt_connection_complete_handler.
	bt_hci_evt_connection_request_handler	This is function bt_hci_evt_connection_request_handler.
	bt_hci_evt_data_buffer_overflow_handler	This is function bt_hci_evt_data_buffer_overflow_handler.
	bt_hci_evt_default_handler	This is function bt_hci_evt_default_handler.
	bt_hci_evt_disconnection_complete_handler	This is function bt_hci_evt_disconnection_complete_handler.
	bt_hci_evt_encryption_change_handler	This is function bt_hci_evt_encryption_change_handler.
	bt_hci_evt_extended_inquiry_result_handler	This is function bt_hci_evt_extended_inquiry_result_handler.
	bt_hci_evt_flow_specification_complete_handler	This is function bt_hci_evt_flow_specification_complete_handler.
	bt_hci_evt_flush_occured_handler	This is function bt_hci_evt_flush_occured_handler.
	bt_hci_evt_hardware_error_handler	This is function bt_hci_evt_hardware_error_handler.
	bt_hci_evt_inquiry_complete_handler	This is function bt_hci_evt_inquiry_complete_handler.
	bt_hci_evt_inquiry_result_handler	This is function bt_hci_evt_inquiry_result_handler.
	bt_hci_evt_inquiry_result_with_rssi_handler	This is function bt_hci_evt_inquiry_result_with_rssi_handler.
	bt_hci_evt_link_key_notification_handler	This is function bt_hci_evt_link_key_notification_handler.
	bt_hci_evt_link_key_request_handler	This is function bt_hci_evt_link_key_request_handler.

	bt_hci_evt_loopback_command_handler	This is function bt_hci_evt_loopback_command_handler.
	bt_hci_evt_master_link_key_complete_handler	This is function bt_hci_evt_master_link_key_complete_handler.
	bt_hci_evt_max_slots_change_handler	This is function bt_hci_evt_max_slots_change_handler.
	bt_hci_evt_mode_change_handler	This is function bt_hci_evt_mode_change_handler.
	bt_hci_evt_num_of_completed_packets_handler	This is function bt_hci_evt_num_of_completed_packets_handler.
	bt_hci_evt_page_scan_repet_mode_change_handler	This is function bt_hci_evt_page_scan_repet_mode_change_handler.
	bt_hci_evt_pin_code_request_handler	This is function bt_hci_evt_pin_code_request_handler.
	bt_hci_evt_qos_setup_complete_handler	This is function bt_hci_evt_qos_setup_complete_handler.
	bt_hci_evt_qos_violation_handler	This is function bt_hci_evt_qos_violation_handler.
	bt_hci_evt_read_clock_offset_complete_handler	This is function bt_hci_evt_read_clock_offset_complete_handler.
	bt_hci_evt_read_rmt_ext_features_comp_handler	This is function bt_hci_evt_read_rmt_ext_features_comp_handler.
	bt_hci_evt_read_rmt_sup_features_comp_handler	This is function bt_hci_evt_read_rmt_sup_features_comp_handler.
	bt_hci_evt_read_rmt_version_info_comp_handler	This is function bt_hci_evt_read_rmt_version_info_comp_handler.
	bt_hci_evt_remote_name_request_complete_handler	This is function bt_hci_evt_remote_name_request_complete_handler.
	bt_hci_evt_return_link_keys_handler	This is function bt_hci_evt_return_link_keys_handler.
	bt_hci_evt_role_change_handler	This is function bt_hci_evt_role_change_handler.
	bt_hci_evt_synch_connection_changed_handler	This is function bt_hci_evt_synch_connection_changed_handler.
	bt_hci_evt_synch_connection_complete_handler	This is function bt_hci_evt_synch_connection_complete_handler.
	bt_hci_exit_park_state	brief Exit park state
	bt_hci_exit_sniff_mode_ex	This is function bt_hci_exit_sniff_mode_ex.
	bt_hci_find_devices	brief Start inquiry
	bt_hci_find_devices_ex	This is function bt_hci_find_devices_ex.
	bt_hci_free_command	brief Free HCI command. ingroup hci
	bt_hci_free_data_buffer	This is function bt_hci_free_data_buffer.
	bt_hci_get_evt_param_bdaddr	brief Get bd address parameter from HCI event ingroup hci
	bt_hci_get_evt_param_byte	brief Get byte parameter from HCI event ingroup hci
	bt_hci_get_evt_param_devclass	brief Get class of device parameter from HCI event ingroup hci
	bt_hci_get_evt_param_int	brief Get int parameter from HCI event ingroup hci
	bt_hci_get_evt_param_linkkey	brief Get link key parameter from HCI event ingroup hci
	bt_hci_get_evt_param_long	brief Get long parameter from HCI event ingroup hci
	bt_hci_get_evt_param_uint	brief Get unsigned int parameter from HCI event ingroup hci
	bt_hci_get_evt_param_ulong	brief Get unsigned long parameter from HCI event ingroup hci
	bt_hci_get_last_cmd_status	This is function bt_hci_get_last_cmd_status.
	bt_hci_get_param_bdaddr	brief Get BD address parameter from HCI command ingroup hci
	bt_hci_get_param_byte	brief Get byte parameter from HCI command ingroup hci
	bt_hci_get_param_int	brief Get int parameter from HCI command ingroup hci
	bt_hci_get_param_linkkey	brief Get link key parameter from HCI command ingroup hci
	bt_hci_get_param_long	brief Get long parameter from HCI command ingroup hci
	bt_hci_get_recv_buffer	This is function bt_hci_get_recv_buffer.
	bt_hci_get_recv_buffer_len	This is function bt_hci_get_recv_buffer_len.
	bt_hci_get_send_buffer	This is function bt_hci_get_send_buffer.
	bt_hci_get_send_buffer_len	This is function bt_hci_get_send_buffer_len.
	bt_hci_init	brief Initialize the HCI layer. ingroup hci This function initializes all internal variables of the HCI layer. The application, unless it's going to use only HCI layer, does not need to call this function as it is implicitly called by bt_sys_init. This function essentially calls bt_hci_init_ex(HCI_LINK_POLICY_ENABLE_ALL) so all link policy setting are enabled.
	bt_hci_init_data_buffers	This is function bt_hci_init_data_buffers.
	bt_hci_init_data_queues	This is function bt_hci_init_data_queues.
	bt_hci_init_linkkey_buffers	This is function bt_hci_init_linkkey_buffers.
	bt_hci_le_add_device_to_white_list	This is function bt_hci_le_add_device_to_white_list.
	bt_hci_le_advertising_add	This is function bt_hci_le_advertising_add.
	bt_hci_le_advertising_add_local_name	This is function bt_hci_le_advertising_add_local_name.
	bt_hci_le_advertising_device_id_add	This is function bt_hci_le_advertising_device_id_add.

	bt_hci_le_advertising_flags_add	This is function bt_hci_le_advertising_flags_add.
	bt_hci_le_advertising_get_local_name	This is function bt_hci_le_advertising_get_local_name.
	bt_hci_le_advertising_tx_power_level_add	This is function bt_hci_le_advertising_tx_power_level_add.
	bt_hci_le_advertising_uuid128_add	This is function bt_hci_le_advertising_uuid128_add.
	bt_hci_le_advertising_uuid16_add	This is function bt_hci_le_advertising_uuid16_add.
	bt_hci_le_advertising_uuid32_add	This is function bt_hci_le_advertising_uuid32_add.
	bt_hci_le_advertising_vendor_add	This is function bt_hci_le_advertising_vendor_add.
	bt_hci_le_allocate_set_advertising_data_command	This is function bt_hci_le_allocate_set_advertising_data_command.
	bt_hci_le_cancel_find_devices	This is function bt_hci_le_cancel_find_devices.
	bt_hci_le_clear_white_list	This is function bt_hci_le_clear_white_list.
	bt_hci_le_connect_ex	This is function bt_hci_le_connect_ex.
	bt_hci_le_enable	This is function bt_hci_le_enable.
	bt_hci_le_encrypt	This is function bt_hci_le_encrypt.
	bt_hci_le_find_devices	This is function bt_hci_le_find_devices.
	bt_hci_le_get_connect_parameters	This is function bt_hci_le_get_connect_parameters.
	bt_hci_le_init	This is function bt_hci_le_init.
	bt_hci_le_ltk_negative_reply	This is function bt_hci_le_ltk_negative_reply.
	bt_hci_le_ltk_reply	This is function bt_hci_le_ltk_reply.
	bt_hci_le_rand	This is function bt_hci_le_rand.
	bt_hci_le_read_channel_map	This is function bt_hci_le_read_channel_map.
	bt_hci_le_read_remote_used_features	This is function bt_hci_le_read_remote_used_features.
	bt_hci_le_read_support	This is function bt_hci_le_read_support.
	bt_hci_le_read_white_list_size	This is function bt_hci_le_read_white_list_size.
	bt_hci_le_receiver_test	This is function bt_hci_le_receiver_test.
	bt_hci_le_remove_device_from_white_list	This is function bt_hci_le_remove_device_from_white_list.
	bt_hci_le_set_adevrtising_enable_ex	This is function bt_hci_le_set_adevrtising_enable_ex.
	bt_hci_le_set_advertising_parameters	This is function bt_hci_le_set_advertising_parameters.
	bt_hci_le_set_connect_parameters	This is function bt_hci_le_set_connect_parameters.
	bt_hci_le_set_host_channel_classification	This is function bt_hci_le_set_host_channel_classification.
	bt_hci_le_set_random_address	This is function bt_hci_le_set_random_address.
	bt_hci_le_set_scan_enable	This is function bt_hci_le_set_scan_enable.
	bt_hci_le_set_scan_parameters	This is function bt_hci_le_set_scan_parameters.
	bt_hci_le_start_encryption	This is function bt_hci_le_start_encryption.
	bt_hci_le_supported	This is function bt_hci_le_supported.
	bt_hci_le_test_end	This is function bt_hci_le_test_end.
	bt_hci_le_transmitter_test	This is function bt_hci_le_transmitter_test.
	bt_hci_le_update_connection	This is function bt_hci_le_update_connection.
	bt_hci_le_write_support	This is function bt_hci_le_write_support.
	bt_hci_listen	brief Listen for incoming connections. ingroup hci details param callback Pointer to a callback function that is called when a new incoming connection has been established. param param Pointer to arbitrary data that is to be passed to the callback function. return li c TRUE when the function succeeds. li c FALSE otherwise. The callback function is not called in this case.
	bt_hci_listen_sco	This is function bt_hci_listen_sco.
	bt_hci_param_eir_add	This is function bt_hci_param_eir_add.
	bt_hci_param_eir_device_id_add	This is function bt_hci_param_eir_device_id_add.
	bt_hci_param_eir_local_name_add	This is function bt_hci_param_eir_local_name_add.
	bt_hci_param_eir_uuid128_add	This is function bt_hci_param_eir_uuid128_add.
	bt_hci_param_eir_uuid16_add	This is function bt_hci_param_eir_uuid16_add.
	bt_hci_param_eir_uuid32_add	This is function bt_hci_param_eir_uuid32_add.
	bt_hci_param_eir_vendor_add	This is function bt_hci_param_eir_vendor_add.
	bt_hci_param_tx_power_level_add	This is function bt_hci_param_tx_power_level_add.
	bt_hci_park_state	brief Put local device to park state
	bt_hci_read_inquiry_mode	brief Get current inquiry mode

	bt_hci_read_inquiry_scan_activity	brief Get current inquiry scan activity configuration
	bt_hci_read_inquiry_scan_type	brief Get current inquiry scan type
	bt_hci_read_page_scan_activity	brief Get current page scan activity configuration
	bt_hci_read_page_scan_period_mode	brief Get current page scan period mode
	bt_hci_read_page_scan_type	brief Get current page scan type
	bt_hci_read_page_timeout	brief Get current page timeout
	bt_hci_register_listener	This is function bt_hci_register_listener.
	bt_hci_reject_pin_code	This is function bt_hci_reject_pin_code.
	bt_hci_request_remote_name	brief Request remote device's name
	bt_hci_role_change_ex	brief Change local device's role
	bt_hci_send_acl_data	brief Send data over ACL connection
	bt_hci_send_cmd	brief Send a command to local device
	bt_hci_send_linkkey	This is function bt_hci_send_linkkey.
	bt_hci_send_sco_data	brief Send data over SCO connection
	bt_hci_set_encryption_ex	brief Set connection encryption
	bt_hci_set_event_listener	listener is triggered by the following events: HCI_EVT_AUTHENTICATION_COMPLETE HCI_EVT_CONNECTION_COMPLETE HCI_EVT_DISCONNECTION_COMPLETE HCI_EVT_ROLE_CHANGE HCI_EVT_MODE_CHANGE
	bt_hci_set_incoming_connection_role	This is function bt_hci_set_incoming_connection_role.
	bt_hci_set_scan	This is function bt_hci_set_scan.
	bt_hci_sniff_mode_ex	This is function bt_hci_sniff_mode_ex.
	bt_hci_sniff_subrating_ex	This is function bt_hci_sniff_subrating_ex.
	bt_hci_start	brief Start HCI layer. ingroup hci details This function starts the HCI layer of the stack. Starting the HCI layer consists essentially of two steps: 1. Make the HCI transport receive packets from the controller. This results in a call to bt_oem_recv. 2. Reset and configure the controller. Upon completion of controller initialization the callback function passed in the c callback parameter is called. param callback Completion callback. Called when controller initialization is complete. param callback_param A pointer to arbitrary data to be passed to the c callback callback. param enable_scan This is a bitmask that defines which scans are enabled... more
	bt_hci_start_no_init	brief Start HCI layer without controller configuration. ingroup hci details This function is similar to bt_hci_start but unlike the former it does not perform the controller configuration. I.e., bt_hci_start_no_init simply calls the HCI transport and makes it receive packets from the controller. The main purpose of this function is make the HCI transport ready to exchange packets if controller needs some vendor specific configuration before it can be used with the stack. E.g., controllers based on CRS8811 chip need loading various values that configure its operating mode using CSR's proprietary protocol. So the application after configuring the HCI transport would... more
	bt_hci_stop	brief Stop HCI layer. ingroup hci details This function makes the HCI layer inoperable. After this call the application must perform the full reset of the HCI transport and stack. param callback Completion callback. Called when the HCI layer has been stopped. param callback_param A pointer to arbitrary data to be passed to the c callback callback.
	bt_hci_transport_recv_packet	This is function bt_hci_transport_recv_packet.
	bt_hci_transport_send_cmd	deprecated
	bt_hci_transport_send_data	This is function bt_hci_transport_send_data.
	bt_hci_transport_send_packet	This is function bt_hci_transport_send_packet.
	bt_hci_transport_set_transport	This is function bt_hci_transport_set_transport.
	bt_hci_unregister_listener	This is function bt_hci_unregister_listener.
	bt_hci_write_default_link_policy_settings	This is function bt_hci_write_default_link_policy_settings.
	bt_hci_write_eir	This is function bt_hci_write_eir.

	bt_hci_write_inquiry_mode	brief Configure inquiry mode
	bt_hci_write_inquiry_scan_activity	brief Configure inquiry scan activity
	bt_hci_write_inquiry_scan_type	brief Configure inquiry scan type
	bt_hci_write_local_name	brief Write local device name ingroup hci details The Write_Local_Name command provides the ability to modify the userfriendly name for the BR/EDR Controller.
	bt_hci_write_page_scan_activity	brief Configure page scan activity
	bt_hci_write_page_scan_period_mode	brief Configure page scan period mode
	bt_hci_write_page_scan_type	brief Configure page scan type
	bt_hci_write_page_timeout	brief Configure page timeout
	bt_hcitr_3wire_start	This is function bt_hcitr_3wire_start.
	bt_hcitr_bcsp_init_ex	This is function bt_hcitr_bcsp_init_ex.
	bt_hcitr_bcsp_reset_ex	This is function bt_hcitr_bcsp_reset_ex.
	bt_hcitr_bcsp_start	This is function bt_hcitr_bcsp_start.
	bt_hcitr_packet_init	This is function bt_hcitr_packet_init.
	bt_hcitr_packet_reset	This is function bt_hcitr_packet_reset.
	bt_hcitr_packet_start	This is function bt_hcitr_packet_start.
	bt_hcitr_tih4_init	This is function bt_hcitr_tih4_init.
	bt_hcitr_tih4_reset	This is function bt_hcitr_tih4_reset.
	bt_hcitr_tih4_start	This is function bt_hcitr_tih4_start.
	bt_hcitr_tih4_wake_up	This is function bt_hcitr_tih4_wake_up.
	bt_hcitr_uart_init	brief Initialize HCI UART (H4) transport protocol ingroup hcitr_uart details This function initializes internal structures of the transport. The application must call it as early as possible before bt_hcitr_uart_start and before the stack is initialized and started with bt_sys_init and bt_sys_start.
	bt_hcitr_uart_reset	brief Re-initialize HCI UART (H4) transport protocol ingroup hcitr_uart details This function re-initializes the transport. Currently it simply calls bt_hcitr_uart_init. After calling this function the application must perform the full initialization of the stack by calling bt_sys_init, bt_sys_start and initialization functions of all other profile modules the application is intending to use.
	bt_hcitr_uart_start	brief Start HCI UART (H4) transport protocol ingroup hcitr_uart details This function starts the transport, i.e., makes it able to receive and send packets.
	hci_allocate_conn_state	This is function hci_allocate_conn_state.
	hci_check_aux_info	From hci_aux_info.c
	hci_cq_find_by_bdaddr_and_opcode	This is function hci_cq_find_by_bdaddr_and_opcode.
	hci_cq_find_by_hconn	This is function hci_cq_find_by_hconn.
	hci_cq_find_by_hconn_and_opcode	This is function hci_cq_find_by_hconn_and_opcode.
	hci_cq_find_by_opcode	This is function hci_cq_find_by_opcode.
	hci_get_conn_state	This is function hci_get_conn_state.
	hci_sleep	brief Put local device to sleep
	hci_wakeup	brief Awaken local device
	is_bdaddr_command	This is function is_bdaddr_command.
	is_hconn_command	This is function is_hconn_command.
	_bt_hci_ctrl_notify_data_listeners	This is function _bt_hci_ctrl_notify_data_listeners.
	_bt_hci_ctrl_notify_listeners	This is function _bt_hci_ctrl_notify_listeners.
	_bt_hci_get_tick_count	This is function _bt_hci_get_tick_count.
	_bt_hci_init_signal	This is function _bt_hci_init_signal.
	_bt_hci_init_timer	From hci_timer.c
	_bt_hci_init_transport	This is function _bt_hci_init_transport.
	_bt_hci_le_command_complete_handler	This is function _bt_hci_le_command_complete_handler.
	_bt_hci_notify_listeners	This is function _bt_hci_notify_listeners.
	_bt_hci_set_signal	This is function _bt_hci_set_signal.
	_hci_allocate_buffers	Defined by OEM through library configuration
	_hci_allocate_cmd	This is function _hci_allocate_cmd.
	_hci_free_cmd	This is function _hci_free_cmd.

	_hci_init_cmd_buffers	This is function _hci_init_cmd_buffers.
	_hci_init_cmd_queues	This is function _hci_init_cmd_queues.
	_hci_receive_start	From hci_receive.c
	_hci_recv_sco_data_packet	From hci_sco.c
	_hci_send_commands_from_queue	From hci_send.c
	_hci_send_data	This is function _hci_send_data.
	_hci_send_data_fragment	This is function _hci_send_data_fragment.
	_hci_send_data_from_queue	This is function _hci_send_data_from_queue.
	_bt_le_evt_handler	This is function _bt_le_evt_handler.
	bt_hci_le_allocate_set_scan_response_data_command	This is function bt_hci_le_allocate_set_scan_response_data_command.
	bt_hci_le_cancel_connect_ex	This is function bt_hci_le_cancel_connect_ex.
	bt_hci_le_ibeacon_add	This is function bt_hci_le_ibeacon_add.
	bt_hci_set_vendor_specific_event_handler	This is function bt_hci_set_vendor_specific_event_handler.
	bt_hcitr_3wire_init_ex	This is function bt_hcitr_3wire_init_ex.
	bt_hcitr_3wire_reset_ex	This is function bt_hcitr_3wire_reset_ex.
	_bt_hci_set_init_flags	This is function _bt_hci_set_init_flags.
	hci_get_conn_state_by_bdaddr	This is function hci_get_conn_state_by_bdaddr.
	bt_hci_cancel_find_devices_ex	brief Stop inquiry
	bt_hci_get_inquiry_response_tx_power_level	This is function bt_hci_get_inquiry_response_tx_power_level.
	bt_hci_init_ex	brief Initialize the HCI layer. ingroup hci This function initializes all internal variables of the HCI layer. The application, unless it's going to use only HCI layer, does not need to call this function as it is implicitly called by bt_sys_init_ex. @param default_link_policy default link policy settings. This is a bitmask that defines the initial value of the link policy settings for all new BR/EDR connections. This value can be a combination of the following values: li HCI_LINK_POLICY_ENABLE_ROLE_SWITCH li HCI_LINK_POLICY_ENABLE_HOLD_MODE li HCI_LINK_POLICY_ENABLE_SNIFF_MODE li HCI_LINK_POLICY_ENABLE_PARK_STATE To enable all settings pass HCI_LINK_POLICY_ENABLE_ALL.
	bt_hci_reset	brief Reset controller. ingroup hci details This function resets the BT controller. param callback Completion callback. Called when the controller has been reset. param callback_param A pointer to arbitrary data to be passed to the c callback callback.
	bt_hci_send_pin_code_ex	This is function bt_hci_send_pin_code_ex.
	bt_hci_set_scan_ex	This is function bt_hci_set_scan_ex.
	bt_hci_write_link_policy_settings	This is function bt_hci_write_link_policy_settings.
	bt_hci_write_local_name_ex	This is function bt_hci_write_local_name_ex.

L2CAP Data Types and Constants

	Name	Description
	bt_l2cap_connect	This is macro bt_l2cap_connect.
	bt_l2cap_listen	This is macro bt_l2cap_listen.
	bt_l2cap_test_enable_local_config	This is macro bt_l2cap_test_enable_local_config.
	bt_l2cap_test_enable_remote_config	This is macro bt_l2cap_test_enable_remote_config.
	CHANNEL_SIGNAL_CMD_DISCONNECT_FIXED	This is macro CHANNEL_SIGNAL_CMD_DISCONNECT_FIXED.
	CID_ATT	This is macro CID_ATT.
	CID_LE_SIG	This is macro CID_LE_SIG.
	CID_MAX	This is macro CID_MAX.
	CID_MAX_FIXED	This is macro CID_MAX_FIXED.
	CID_NULL	This is macro CID_NULL.
	CID_RECV	This is macro CID_RECV.
	CID_SIG	This is macro CID_SIG.
	CID_SM	This is macro CID_SM.
	CMODE_BASIC	This is macro CMODE_BASIC.
	CMODE_ERETR	This is macro CMODE_ERETR.

	CMODE_FLOW	This is macro CMODE_FLOW.
	CMODE_RETR	This is macro CMODE_RETR.
	CMODE_STRM	This is macro CMODE_STRM.
	CSTATE_CLOSED	This is macro CSTATE_CLOSED.
	CSTATE_FREE	This is macro CSTATE_FREE.
	CSTATE_OPEN	This is macro CSTATE_OPEN.
	CSTATE_WAIT_CONFIG	This is macro CSTATE_WAIT_CONFIG.
	CSTATE_WAIT_CONFIG_REQ	This is macro CSTATE_WAIT_CONFIG_REQ.
	CSTATE_WAIT_CONFIG_RSP	This is macro CSTATE_WAIT_CONFIG_RSP.
	CSTATE_WAIT_CONNECT	This is macro CSTATE_WAIT_CONNECT.
	CSTATE_WAIT_CONNECT_RSP	This is macro CSTATE_WAIT_CONNECT_RSP.
	CSTATE_WAIT_DISCONNECT	This is macro CSTATE_WAIT_DISCONNECT.
	CTYPE_CL	connectionless bt_l2cap_channel
	CTYPE_CO	connection-oriented bt_l2cap_channel
	GET_F_BIT	This is macro GET_F_BIT.
	GET_FRAME_TYPE	This is macro GET_FRAME_TYPE.
	GET_P_BIT	This is macro GET_P_BIT.
	GET_REQ_SEQ	This is macro GET_REQ_SEQ.
	GET_S_FUNCTION	This is macro GET_S_FUNCTION.
	GET_SAR	This is macro GET_SAR.
	GET_TX_SEQ	This is macro GET_TX_SEQ.
	L2CAP_CHANNEL_FLAG_INCOMING	This is macro L2CAP_CHANNEL_FLAG_INCOMING.
	L2CAP_CHANNEL_FLAG_SENDING	This is macro L2CAP_CHANNEL_FLAG_SENDING.
	L2CAP_CMD_CONFIG_REQUEST	This is macro L2CAP_CMD_CONFIG_REQUEST.
	L2CAP_CMD_CONFIG_RESPONSE	This is macro L2CAP_CMD_CONFIG_RESPONSE.
	L2CAP_CMD_CONN_PARAM_UPDATE_REQUEST	This is macro L2CAP_CMD_CONN_PARAM_UPDATE_REQUEST.
	L2CAP_CMD_CONN_PARAM_UPDATE_RESPONSE	This is macro L2CAP_CMD_CONN_PARAM_UPDATE_RESPONSE.
	L2CAP_CMD_CONN_REQUEST	This is macro L2CAP_CMD_CONN_REQUEST.
	L2CAP_CMD_CONN_RESPONSE	This is macro L2CAP_CMD_CONN_RESPONSE.
	L2CAP_CMD_DATA_LEN_CMD_REJECT	This is macro L2CAP_CMD_DATA_LEN_CMD_REJECT.
	L2CAP_CMD_DATA_LEN_CONFIG_REQUEST	This is macro L2CAP_CMD_DATA_LEN_CONFIG_REQUEST.
	L2CAP_CMD_DATA_LEN_CONFIG_RESPONSE	This is macro L2CAP_CMD_DATA_LEN_CONFIG_RESPONSE.
	L2CAP_CMD_DATA_LEN_CONN_REQUEST	This is macro L2CAP_CMD_DATA_LEN_CONN_REQUEST.
	L2CAP_CMD_DATA_LEN_CONN_RESPONSE	This is macro L2CAP_CMD_DATA_LEN_CONN_RESPONSE.
	L2CAP_CMD_DATA_LEN_DCONN_REQUEST	This is macro L2CAP_CMD_DATA_LEN_DCONN_REQUEST.
	L2CAP_CMD_DATA_LEN_DCONN_RESPONSE	This is macro L2CAP_CMD_DATA_LEN_DCONN_RESPONSE.
	L2CAP_CMD_DATA_LEN_ECHO_REQUEST	This is macro L2CAP_CMD_DATA_LEN_ECHO_REQUEST.
	L2CAP_CMD_DATA_LEN_ECHO_RESPONSE	This is macro L2CAP_CMD_DATA_LEN_ECHO_RESPONSE.
	L2CAP_CMD_DATA_LEN_INFO_REQUEST	This is macro L2CAP_CMD_DATA_LEN_INFO_REQUEST.
	L2CAP_CMD_DATA_LEN_INFO_RESPONSE	This is macro L2CAP_CMD_DATA_LEN_INFO_RESPONSE.
	L2CAP_CMD_DCONN_REQUEST	This is macro L2CAP_CMD_DCONN_REQUEST.
	L2CAP_CMD_DCONN_RESPONSE	This is macro L2CAP_CMD_DCONN_RESPONSE.
	L2CAP_CMD_ECHO_REQUEST	This is macro L2CAP_CMD_ECHO_REQUEST.
	L2CAP_CMD_ECHO_RESPONSE	This is macro L2CAP_CMD_ECHO_RESPONSE.
	L2CAP_CMD_HEADER_LEN	This is macro L2CAP_CMD_HEADER_LEN.
	L2CAP_CMD_INFO_REQUEST	This is macro L2CAP_CMD_INFO_REQUEST.
	L2CAP_CMD_INFO_RESPONSE	This is macro L2CAP_CMD_INFO_RESPONSE.
	L2CAP_CMD_LAST	This is macro L2CAP_CMD_LAST.

	L2CAP_CMD_REJECT	This is macro L2CAP_CMD_REJECT.
	L2CAP_CMD_RESERVED	This is macro L2CAP_CMD_RESERVED.
	L2CAP_CMD_STATUS_BEING_SENT	This is macro L2CAP_CMD_STATUS_BEING_SENT.
	L2CAP_CMD_STATUS_PENDING	This is macro L2CAP_CMD_STATUS_PENDING.
	L2CAP_CMD_STATUS_WAITING_RESPONSE	This is macro L2CAP_CMD_STATUS_WAITING_RESPONSE.
	L2CAP_CONFIG_RESULT_REJECTED	This is macro L2CAP_CONFIG_RESULT_REJECTED.
	L2CAP_CONFIG_RESULT_SUCCESS	This is macro L2CAP_CONFIG_RESULT_SUCCESS.
	L2CAP_CONFIG_RESULT_UNACCEPTABLE_PARAMETER	This is macro L2CAP_CONFIG_RESULT_UNACCEPTABLE_PARAMETER.
	L2CAP_CONFIG_RESULT_UNKNOWN_OPTION	This is macro L2CAP_CONFIG_RESULT_UNKNOWN_OPTION.
	L2CAP_CONN_REQ_RESULT_INVALID_PSM	This is macro L2CAP_CONN_REQ_RESULT_INVALID_PSM.
	L2CAP_CONN_REQ_RESULT_NO_RESOURCES	This is macro L2CAP_CONN_REQ_RESULT_NO_RESOURCES.
	L2CAP_CONN_REQ_RESULT_PENDING	This is macro L2CAP_CONN_REQ_RESULT_PENDING.
	L2CAP_CONN_REQ_RESULT_SECURITY_BLOCK	This is macro L2CAP_CONN_REQ_RESULT_SECURITY_BLOCK.
	L2CAP_CONN_REQ_RESULT_SUCCESS	This is macro L2CAP_CONN_REQ_RESULT_SUCCESS.
	L2CAP_CONN_REQ_STATUS_AUTHENTICATION_PENDING	This is macro L2CAP_CONN_REQ_STATUS_AUTHENTICATION_PENDING.
	L2CAP_CONN_REQ_STATUS_AUTHORIZATION_PENDING	This is macro L2CAP_CONN_REQ_STATUS_AUTHORIZATION_PENDING.
	L2CAP_CONN_REQ_STATUS_NO_INFO	This is macro L2CAP_CONN_REQ_STATUS_NO_INFO.
	L2CAP_DEFAULT_ERTX	seconds
	L2CAP_DEFAULT_RTX	seconds
	L2CAP_ECHO_MAX_DATA_LEN	echo request and response
	L2CAP_ERETR_RECV_STATE_RECV	This is macro L2CAP_ERETR_RECV_STATE_RECV.
	L2CAP_ERETR_RECV_STATE_REJ_SENT	This is macro L2CAP_ERETR_RECV_STATE_REJ_SENT.
	L2CAP_ERETR_RECV_STATE_SREJ_SENT	This is macro L2CAP_ERETR_RECV_STATE_SREJ_SENT.
	L2CAP_ERETR_XMIT_STATE_WAIT_ACK	This is macro L2CAP_ERETR_XMIT_STATE_WAIT_ACK.
	L2CAP_ERETR_XMIT_STATE_WAIT_F	This is macro L2CAP_ERETR_XMIT_STATE_WAIT_F.
	L2CAP_ERETR_XMIT_STATE_XMIT	This is macro L2CAP_ERETR_XMIT_STATE_XMIT.
	L2CAP_EXT_BI_QOS	This is macro L2CAP_EXT_BI_QOS.
	L2CAP_EXT_ENHANCED_RETRANSMISSION	This is macro L2CAP_EXT_ENHANCED_RETRANSMISSION.
	L2CAP_EXT_FCS_OPTION	This is macro L2CAP_EXT_FCS_OPTION.
	L2CAP_EXT_FLOW_CONTROL	This is macro L2CAP_EXT_FLOW_CONTROL.
	L2CAP_EXT_RETRANSMISSION	This is macro L2CAP_EXT_RETRANSMISSION.
	L2CAP_EXT_STREAMING	This is macro L2CAP_EXT_STREAMING.
	L2CAP_FRAME_TYPE_I	This is macro L2CAP_FRAME_TYPE_I.
	L2CAP_FRAME_TYPE_S	This is macro L2CAP_FRAME_TYPE_S.
	L2CAP_HCI_CONNECT_PACKET_TYPE	This is macro L2CAP_HCI_CONNECT_PACKET_TYPE.
	L2CAP_HCI_CONNECT_PAGE_SCAN_REPETITION_MODE	This is macro L2CAP_HCI_CONNECT_PAGE_SCAN_REPETITION_MODE.
	L2CAP_HCI_CONNECT_ROLE_SWITCH	This is macro L2CAP_HCI_CONNECT_ROLE_SWITCH.
	L2CAP_HEADER_LEN	This is macro L2CAP_HEADER_LEN.
	L2CAP_IDLE_HCI_CONNECTION_TIMEOUT	This is macro L2CAP_IDLE_HCI_CONNECTION_TIMEOUT.
	L2CAP_INFO_NOT_SUPPORTED	REDFLAG: should it be non zero?
	L2CAP_INFO_RESULT_SUCCESS	This is macro L2CAP_INFO_RESULT_SUCCESS.
	L2CAP_INFO_TYPE_CONNECTIONLESS_MTU	information request
	L2CAP_INFO_TYPE_EXTENDED_SUPPORT	This is macro L2CAP_INFO_TYPE_EXTENDED_SUPPORT.
	L2CAP_LINK_TYPE_BD_EDR	This is macro L2CAP_LINK_TYPE_BD_EDR.

	L2CAP_LINK_TYPE_LE	This is macro L2CAP_LINK_TYPE_LE.
	L2CAP_MAX_CMD_QUEUE_LEN	This is macro L2CAP_MAX_CMD_QUEUE_LEN.
	L2CAP_MAX_FRAME_BUFFERS	This is macro L2CAP_MAX_FRAME_BUFFERS.
	L2CAP_MAX_MANAGERS	This is macro L2CAP_MAX_MANAGERS.
	L2CAP_MAX_MTU	This is macro L2CAP_MAX_MTU.
	L2CAP_MAX_OPTIONS	configuration request/response
	L2CAP_MAX_RTX	This is macro L2CAP_MAX_RTX.
	L2CAP_MGR_STATE_FREE	This is macro L2CAP_MGR_STATE_FREE.
	L2CAP_MGR_STATE_LISTENING_L2CAP	This is macro L2CAP_MGR_STATE_LISTENING_L2CAP.
	L2CAP_MGR_STATE_USED	This is macro L2CAP_MGR_STATE_USED.
	L2CAP_MIN_MTU	This is macro L2CAP_MIN_MTU.
	L2CAP_MONITOR_TIMEOUT	This is macro L2CAP_MONITOR_TIMEOUT.
	L2CAP_OPTION_LEN_FLASH_QOS	This is macro L2CAP_OPTION_LEN_FLASH_QOS.
	L2CAP_OPTION_LEN_FLASH_RFC	This is macro L2CAP_OPTION_LEN_FLASH_RFC.
	L2CAP_OPTION_LEN_FLASH_TIMEOUT	This is macro L2CAP_OPTION_LEN_FLASH_TIMEOUT.
	L2CAP_OPTION_LEN_MAX_MTU	This is macro L2CAP_OPTION_LEN_MAX_MTU.
	L2CAP_OPTION_LEN_UNKNOWN	This is macro L2CAP_OPTION_LEN_UNKNOWN.
	L2CAP_OPTION_LEN_UNKNOWN_DATA	21
	L2CAP_OPTION_TYPE_FLASH_QOS	This is macro L2CAP_OPTION_TYPE_FLASH_QOS.
	L2CAP_OPTION_TYPE_FLASH_QOS_FLAG	This is macro L2CAP_OPTION_TYPE_FLASH_QOS_FLAG.
	L2CAP_OPTION_TYPE_FLASH_RFC	This is macro L2CAP_OPTION_TYPE_FLASH_RFC.
	L2CAP_OPTION_TYPE_FLASH_RFC_FLAG	This is macro L2CAP_OPTION_TYPE_FLASH_RFC_FLAG.
	L2CAP_OPTION_TYPE_FLASH_TIMEOUT	This is macro L2CAP_OPTION_TYPE_FLASH_TIMEOUT.
	L2CAP_OPTION_TYPE_FLASH_TIMEOUT_FLAG	This is macro L2CAP_OPTION_TYPE_FLASH_TIMEOUT_FLAG.
	L2CAP_OPTION_TYPE_MAX_MTU	This is macro L2CAP_OPTION_TYPE_MAX_MTU.
	L2CAP_OPTION_TYPE_MAX_MTU_FLAG	This is macro L2CAP_OPTION_TYPE_MAX_MTU_FLAG.
	L2CAP_OPTION_TYPE_UNKNOWN_FLAG	This is macro L2CAP_OPTION_TYPE_UNKNOWN_FLAG.
	L2CAP_PACKET_DATA_TYPE_RAW	This is macro L2CAP_PACKET_DATA_TYPE_RAW.
	L2CAP_PACKET_DATA_TYPE_SMART	This is macro L2CAP_PACKET_DATA_TYPE_SMART.
	L2CAP_REJECT_REASON_INVALID_CHANNEL	This is macro L2CAP_REJECT_REASON_INVALID_CHANNEL.
	L2CAP_REJECT_REASON_MTU_EXCEEDED	This is macro L2CAP_REJECT_REASON_MTU_EXCEEDED.
	L2CAP_REJECT_REASON_NOT_UNDERSTOOD	This is macro L2CAP_REJECT_REASON_NOT_UNDERSTOOD.
	L2CAP_RETR_TIMEOUT	This is macro L2CAP_RETR_TIMEOUT.
	L2CAP_RFC_BASIC	This is macro L2CAP_RFC_BASIC.
	L2CAP_RFC_ENHANCED_RETRANSMISSION	This is macro L2CAP_RFC_ENHANCED_RETRANSMISSION.
	L2CAP_RFC_ERETR_TIMEOUT	3 secs
	L2CAP_RFC_ERETR_TX_WINDOW	This is macro L2CAP_RFC_ERETR_TX_WINDOW.
	L2CAP_RFC_FLOW_CONTROL	This is macro L2CAP_RFC_FLOW_CONTROL.
	L2CAP_RFC_MONITOR_TIMEOUT	12 secs
	L2CAP_RFC_RETRANSMISSION	This is macro L2CAP_RFC_RETRANSMISSION.
	L2CAP_RFC_STREAMING	This is macro L2CAP_RFC_STREAMING.
	L2CAP_SAR_SDU_CONTINUE	This is macro L2CAP_SAR_SDU_CONTINUE.
	L2CAP_SAR_SDU_START	This is macro L2CAP_SAR_SDU_START.
	L2CAP_SAR_SDU_STOP	This is macro L2CAP_SAR_SDU_STOP.
	L2CAP_SAR_UNSEGMENTED	This is macro L2CAP_SAR_UNSEGMENTED.
	L2CAP_SERVICE_TYPE_BEST_EFFORT	This is macro L2CAP_SERVICE_TYPE_BEST_EFFORT.
	L2CAP_SERVICE_TYPE_GUARANTEED	This is macro L2CAP_SERVICE_TYPE_GUARANTEED.
	L2CAP_SERVICE_TYPE_NO_TRAFFIC	This is macro L2CAP_SERVICE_TYPE_NO_TRAFFIC.
	L2CAP_SFUNCTION_REJ	This is macro L2CAP_SFUNCTION_REJ.
	L2CAP_SFUNCTION_RNR	This is macro L2CAP_SFUNCTION_RNR.
	L2CAP_SFUNCTION_RR	This is macro L2CAP_SFUNCTION_RR.

	L2CAP_SFUNCTION_SREJ	This is macro L2CAP_SFUNCTION_SREJ.
	PSM_ATT	This is macro PSM_ATT.
	PSM_AVCTP	This is macro PSM_AVCTP.
	PSM_AVCTP_Browsing	This is macro PSM_AVCTP_Browsing.
	PSM_AVDTP	This is macro PSM_AVDTP.
	PSM_BNEP	This is macro PSM_BNEP.
	PSM_HID_Control	This is macro PSM_HID_Control.
	PSM_HID_Interrupt	This is macro PSM_HID_Interrupt.
	PSM_RFCOMM	This is macro PSM_RFCOMM.
	PSM_SDP	This is macro PSM_SDP.
	SET_F_BIT	This is macro SET_F_BIT.
	SET_FRAME_TYPE	This is macro SET_FRAME_TYPE.
	SET_P_BIT	This is macro SET_P_BIT.
	SET_REQ_SEQ	This is macro SET_REQ_SEQ.
	SET_S_FUNCTION	This is macro SET_S_FUNCTION.
	SET_SAR	This is macro SET_SAR.
	SET_TX_SEQ	This is macro SET_TX_SEQ.
	_bt_l2cap_start_monitor_timer	This is macro _bt_l2cap_start_monitor_timer.
	_bt_l2cap_start_monitor_timer_if_not_running	This is macro _bt_l2cap_start_monitor_timer_if_not_running.
	_bt_l2cap_start_retr_timer	This is macro _bt_l2cap_start_retr_timer.
	_bt_l2cap_start_retr_timer_if_not_running	This is macro _bt_l2cap_start_retr_timer_if_not_running.
	_bt_l2cap_stop_monitor_timer	This is macro _bt_l2cap_stop_monitor_timer.
	_bt_l2cap_stop_retr_timer	This is macro _bt_l2cap_stop_retr_timer.
	_bt_l2cap_cfg_option_t	This is type bt_l2cap_cfg_option_t.
	_bt_l2cap_channel_t	This is record _bt_l2cap_channel_t.
	_bt_l2cap_cmd_config_req_t	This is type bt_l2cap_cmd_config_req_t.
	_bt_l2cap_cmd_config_res_t	This is type bt_l2cap_cmd_config_res_t.
	_bt_l2cap_cmd_conn_param_update_req_t	connection parameter update
	_bt_l2cap_cmd_conn_param_update_res_t	This is type bt_l2cap_cmd_conn_param_update_res_t.
	_bt_l2cap_cmd_connection_req_t	This is type bt_l2cap_cmd_connection_req_t.
	_bt_l2cap_cmd_connection_res_t	This is type bt_l2cap_cmd_connection_res_t.
	_bt_l2cap_cmd_disconnection_req_t	This is type bt_l2cap_cmd_disconnection_req_t.
	_bt_l2cap_cmd_echo_req_t	This is type bt_l2cap_cmd_echo_req_t.
	_bt_l2cap_cmd_echo_res_t	This is type bt_l2cap_cmd_echo_res_t.
	_bt_l2cap_cmd_header_t	This is type bt_l2cap_cmd_header_t.
	_bt_l2cap_cmd_info_req_t	This is type bt_l2cap_cmd_info_req_t.
	_bt_l2cap_cmd_info_res_t	This is type bt_l2cap_cmd_info_res_t.
	_bt_l2cap_cmd_reject_param_t	This is type bt_l2cap_cmd_reject_param_t.
	_bt_l2cap_cmd_reject_t	This is type bt_l2cap_cmd_reject_t.
	_bt_l2cap_command_t	This is type bt_l2cap_command_t.
	_bt_l2cap_eretr_xmit_event_e	This is type bt_l2cap_eretr_xmit_event_e.
	_bt_l2cap_fixed_channel_s	This is type bt_l2cap_fixed_channel_t.
	_bt_l2cap_frame_desc_s	typedef union _bt_l2cap_frame_control_s { struct { bt_byte frame_type:1; bt_byte txSeq:6; bt_byte f:1; bt_byte reqSeq:6; bt_byte sar:2; } iframe; struct { bt_byte frame_type:1; bt_byte reserved:1; bt_byte s:2; bt_byte p:1; bt_byte reserved2:2; bt_byte f:1; bt_byte reqSeq:6; bt_byte reserved3:2; } sframe; } bt_l2cap_frame_control_t;
	_bt_l2cap_mgr_s	This is type bt_l2cap_mgr_t.
	_bt_l2cap_option_flash_timeout_t	This is type bt_l2cap_option_flash_timeout_t.
	_bt_l2cap_option_max_mtu_t	This is type bt_l2cap_option_max_mtu_t.
	_bt_l2cap_option_qos_t	This is type bt_l2cap_option_qos_t.
	_bt_l2cap_option_rfc_t	This is type bt_l2cap_option_rfc_t.
	_bt_l2cap_option_unknown_t	This is type bt_l2cap_option_unknown_t.
	_bt_l2cap_packet_t	This is type bt_l2cap_packet_t.
	_bt_l2cap_psm_s	This is type bt_l2cap_psm_t.

	_bt_l2cap_xmit_event_param_t	This is type bt_l2cap_xmit_event_param_t.
	bt_l2cap_cfg_option_p	This is type bt_l2cap_cfg_option_p.
	bt_l2cap_cfg_option_t	This is type bt_l2cap_cfg_option_t.
	bt_l2cap_channel_t	This is type bt_l2cap_channel_t.
	bt_l2cap_cmd_assembler_fp	This is type bt_l2cap_cmd_assembler_fp.
	bt_l2cap_cmd_config_req_p	This is type bt_l2cap_cmd_config_req_p.
	bt_l2cap_cmd_config_req_t	This is type bt_l2cap_cmd_config_req_t.
	bt_l2cap_cmd_config_res_p	This is type bt_l2cap_cmd_config_res_p.
	bt_l2cap_cmd_config_res_t	This is type bt_l2cap_cmd_config_res_t.
	bt_l2cap_cmd_conn_param_update_req_t	connection parameter update
	bt_l2cap_cmd_conn_param_update_res_t	This is type bt_l2cap_cmd_conn_param_update_res_t.
	bt_l2cap_cmd_connection_req_p	This is type bt_l2cap_cmd_connection_req_p.
	bt_l2cap_cmd_connection_req_t	This is type bt_l2cap_cmd_connection_req_t.
	bt_l2cap_cmd_connection_res_p	This is type bt_l2cap_cmd_connection_res_p.
	bt_l2cap_cmd_connection_res_t	This is type bt_l2cap_cmd_connection_res_t.
	bt_l2cap_cmd_disconnection_req_p	This is type bt_l2cap_cmd_disconnection_req_p.
	bt_l2cap_cmd_disconnection_req_t	This is type bt_l2cap_cmd_disconnection_req_t.
	bt_l2cap_cmd_echo_req_p	This is type bt_l2cap_cmd_echo_req_p.
	bt_l2cap_cmd_echo_req_t	This is type bt_l2cap_cmd_echo_req_t.
	bt_l2cap_cmd_echo_res_p	This is type bt_l2cap_cmd_echo_res_p.
	bt_l2cap_cmd_echo_res_t	This is type bt_l2cap_cmd_echo_res_t.
	bt_l2cap_cmd_header_p	This is type bt_l2cap_cmd_header_p.
	bt_l2cap_cmd_header_t	This is type bt_l2cap_cmd_header_t.
	bt_l2cap_cmd_info_req_p	This is type bt_l2cap_cmd_info_req_p.
	bt_l2cap_cmd_info_req_t	This is type bt_l2cap_cmd_info_req_t.
	bt_l2cap_cmd_info_res_p	This is type bt_l2cap_cmd_info_res_p.
	bt_l2cap_cmd_info_res_t	This is type bt_l2cap_cmd_info_res_t.
	bt_l2cap_cmd_parser_fp	This is type bt_l2cap_cmd_parser_fp.
	bt_l2cap_cmd_reject_p	This is type bt_l2cap_cmd_reject_p.
	bt_l2cap_cmd_reject_param_t	This is type bt_l2cap_cmd_reject_param_t.
	bt_l2cap_cmd_reject_t	This is type bt_l2cap_cmd_reject_t.
	bt_l2cap_command_t	This is type bt_l2cap_command_t.
	bt_l2cap_connect_callback_fp	This is type bt_l2cap_connect_callback_fp.
	bt_l2cap_eretr_xmit_event_e	This is type bt_l2cap_eretr_xmit_event_e.
	bt_l2cap_fixed_channel_t	This is type bt_l2cap_fixed_channel_t.
	bt_l2cap_frame_desc_t	typedef union _bt_l2cap_frame_control_s { struct { bt_byte frame_type:1; bt_byte txSeq:6; bt_byte f:1; bt_byte reqSeq:6; bt_byte sar:2; } iframe; struct { bt_byte frame_type:1; bt_byte reserved:1; bt_byte s:2; bt_byte p:1; bt_byte reserved2:2; bt_byte f:1; bt_byte reqSeq:6; bt_byte reserved3:2; } sframe; } bt_l2cap_frame_control_t;
	bt_l2cap_listen_callback_fp	This is type bt_l2cap_listen_callback_fp.
	bt_l2cap_mgr_p	This is type bt_l2cap_mgr_p.
	bt_l2cap_mgr_t	This is type bt_l2cap_mgr_t.
	bt_l2cap_option_flash_timeout_t	This is type bt_l2cap_option_flash_timeout_t.
	bt_l2cap_option_max_mtu_t	This is type bt_l2cap_option_max_mtu_t.
	bt_l2cap_option_qos_t	This is type bt_l2cap_option_qos_t.
	bt_l2cap_option_rfc_t	This is type bt_l2cap_option_rfc_t.
	bt_l2cap_option_unknown_t	This is type bt_l2cap_option_unknown_t.
	bt_l2cap_packet_t	This is type bt_l2cap_packet_t.
	bt_l2cap_psm_t	This is type bt_l2cap_psm_t.
	bt_l2cap_read_data_callback_fp	This is type bt_l2cap_read_data_callback_fp.
	bt_l2cap_request_handler_fp	This is type bt_l2cap_request_handler_fp.
	bt_l2cap_response_handler_fp	This is type bt_l2cap_response_handler_fp.
	bt_l2cap_send_data_callback_fp	This is type bt_l2cap_send_data_callback_fp.
	bt_l2cap_state_changed_callback_fp	This is type bt_l2cap_state_changed_callback_fp.

	bt_l2cap_xmit_event_param_t	This is type bt_l2cap_xmit_event_param_t.
	cmd_disconnection_res	This is type cmd_disconnection_res.
	pcmd_disconnection_res	This is type pcmd_disconnection_res.
	pf_l2cap_cmd_callback	This is type pf_l2cap_cmd_callback.
	L2CAP_CHANNEL_FLAG_FORCE_HCI_DISCONNECT	This is macro L2CAP_CHANNEL_FLAG_FORCE_HCI_DISCONNECT.
	L2CAP_EXT_EXT_FLOW_SPEC_BR_EDR	This is macro L2CAP_EXT_EXT_FLOW_SPEC_BR_EDR.
	L2CAP_EXT_EXT_WINDOW_SIZE	This is macro L2CAP_EXT_EXT_WINDOW_SIZE.
	L2CAP_EXT_FEATURES_ENABLED	This is macro L2CAP_EXT_FEATURES_ENABLED.
	L2CAP_EXT_FIXED_CHANNELS	This is macro L2CAP_EXT_FIXED_CHANNELS.
	L2CAP_EXT_UNICAST_CONNLESS_DATA_RCPT	This is macro L2CAP_EXT_UNICAST_CONNLESS_DATA_RCPT.
	L2CAP_INFO_TYPE_FIXED_CHANNELS	This is macro L2CAP_INFO_TYPE_FIXED_CHANNELS.
	bt_l2cap_channel_ext_t	This is type bt_l2cap_channel_ext_t.
	_bt_l2cap_channel_ext_t	This is record _bt_l2cap_channel_ext_t.
	_bt_l2cap_connect_params_s	This is record _bt_l2cap_connect_params_s.
	L2CAP_CONN_REQ_RESULT_INVALID_SOURCE_CID	This is macro L2CAP_CONN_REQ_RESULT_INVALID_SOURCE_CID.
	L2CAP_CONN_REQ_RESULT_SRC_CID_ALREADY_ALLOCATED	This is macro L2CAP_CONN_REQ_RESULT_SRC_CID_ALREADY_ALLOCATED.
	bt_l2cap_connect_params_t	This is type bt_l2cap_connect_params_t.

L2CAP Functions

	Name	Description
	bt_l2cap_alloc_cmd_buffer	This is function bt_l2cap_alloc_cmd_buffer.
	bt_l2cap_alloc_cmd_config_req	This is function bt_l2cap_alloc_cmd_config_req.
	bt_l2cap_alloc_cmd_config_res	This is function bt_l2cap_alloc_cmd_config_res.
	bt_l2cap_alloc_cmd_conn_param_update_req	This is function bt_l2cap_alloc_cmd_conn_param_update_req.
	bt_l2cap_alloc_cmd_conn_param_update_res	This is function bt_l2cap_alloc_cmd_conn_param_update_res.
	bt_l2cap_alloc_cmd_connection_req	This is function bt_l2cap_alloc_cmd_connection_req.
	bt_l2cap_alloc_cmd_connection_res	This is function bt_l2cap_alloc_cmd_connection_res.
	bt_l2cap_alloc_cmd_disconnection_req	This is function bt_l2cap_alloc_cmd_disconnection_req.
	bt_l2cap_alloc_cmd_disconnection_res	This is function bt_l2cap_alloc_cmd_disconnection_res.
	bt_l2cap_alloc_cmd_echo_req	This is function bt_l2cap_alloc_cmd_echo_req.
	bt_l2cap_alloc_cmd_echo_res	This is function bt_l2cap_alloc_cmd_echo_res.
	bt_l2cap_alloc_cmd_info_req	This is function bt_l2cap_alloc_cmd_info_req.
	bt_l2cap_alloc_cmd_info_res	This is function bt_l2cap_alloc_cmd_info_res.
	bt_l2cap_alloc_cmd_reject	This is function bt_l2cap_alloc_cmd_reject.
	bt_l2cap_alloc_frame_buffer	This is function bt_l2cap_alloc_frame_buffer.
	bt_l2cap_allocate_channel	This is function bt_l2cap_allocate_channel.
	bt_l2cap_allocate_fixed_channel	This is function bt_l2cap_allocate_fixed_channel.
	bt_l2cap_allocate_mgr	brief Allocate L2CAP manager. ingroup l2cap details This function allocates and initializes an L2CAP manager structure. One L2CAP manager manages all L2CAP connections for a particular local device. The local device is specified by the c hci_ctrl parameter. param hci_ctrl Pointer to the hci_ctrl_state structure that represents the local device (HCI controller) for which a L2CAP manager is to be allocated. return A pointer to the allocated L2CAP manager structure. The returned L2CAP manager should be freed by a call to bt_l2cap_free_mgr when it is no longer needed.
	bt_l2cap_allocate_psm	This is function bt_l2cap_allocate_psm.

	bt_l2cap_connect_ext	brief Connect to a remote device. ingroup l2cap details This function establishes an L2CAP connection with a remote device on a specific PSM. When connect operation completes a callback function is called. param mgr The L2CAP manager. param remote_addr The address of the remote device. param psm The PSM for the connection. param connect_cb The Callback function that is called when the connect operation completes. param param A pointer to arbitrary data to be passed to the c connect_cb callback. param state_cb The callback function that is called when the state of the established connection changes. return li c TRUE... more
	bt_l2cap_connect_fixed_channel	This is function bt_l2cap_connect_fixed_channel.
	bt_l2cap_disconnect	brief Close L2CAP channel. ingroup l2cap details This function closes an L2CAP channel. The channel can be established either by a call to bt_l2cap_connect() or by an incoming connection request. param ch The L2CAP channel to be closed. return li c TRUE when the function succeeds. li c FALSE otherwise. The callback function is not called in this case.
	bt_l2cap_echo	brief Send echo command
	bt_l2cap_find_fixed_channel	This is function bt_l2cap_find_fixed_channel.
	bt_l2cap_find_psm	This is function bt_l2cap_find_psm.
	bt_l2cap_free_cmd_buffer	This is function bt_l2cap_free_cmd_buffer.
	bt_l2cap_free_fixed_channel	This is function bt_l2cap_free_fixed_channel.
	bt_l2cap_free_frame_buffer	This is function bt_l2cap_free_frame_buffer.
	bt_l2cap_free_mgr	brief Release L2CAP manger. ingroup l2cap details This function releases the L2CAP manager structure. param mgr The L2CAP manager structure to be released.
	bt_l2cap_free_psm	This is function bt_l2cap_free_psm.
	bt_l2cap_get_channel	This is function bt_l2cap_get_channel.
	bt_l2cap_get_channel_by_bdaddr_cid	This is function bt_l2cap_get_channel_by_bdaddr_cid.
	bt_l2cap_get_channel_by_hconn_cid	This is function bt_l2cap_get_channel_by_hconn_cid.
	bt_l2cap_get_channel_by_hconn_dest_cid	This is function bt_l2cap_get_channel_by_hconn_dest_cid.
	bt_l2cap_get_channel_by_psm	This is function bt_l2cap_get_channel_by_psm.
	bt_l2cap_get_mgr	This is function bt_l2cap_get_mgr.
	bt_l2cap_init	brief Initialize the L2CAP layer. ingroup l2cap details This function initializes the L2CAP layer of the stack. It must be called prior to any other L2CAP function can be called.
	bt_l2cap_init_channels	This is function bt_l2cap_init_channels.
	bt_l2cap_init_cmd_buffers	This is function bt_l2cap_init_cmd_buffers.
	bt_l2cap_init_frame_buffers	This is function bt_l2cap_init_frame_buffers.
	bt_l2cap_init_psms	This is function bt_l2cap_init_psms.
	bt_l2cap_listen_ext	brief Listen for incoming connections. ingroup l2cap details This function tells the L2CAP manager to listen for incoming connections on a specific PSM. When a connection is established a callback function is called. param mgr The L2CAP manager. param psm The PSM on which the manager will listen and accept incoming connections. param callback The callback function that will be called when an incoming connection is established. param param An arbitrary data pointer that will be passed to the callback function specified by the c callback parameter. return li c TRUE when the function succeeds. li c FALSE otherwise. The... more
	bt_l2cap_listen_fixed_channel	This is function bt_l2cap_listen_fixed_channel.
	bt_l2cap_packet_cmd_assembler	This is function bt_l2cap_packet_cmd_assembler.
	bt_l2cap_packet_data_assembler	This is function bt_l2cap_packet_data_assembler.
	bt_l2cap_read_data	This is function bt_l2cap_read_data.
	bt_l2cap_reject	brief Send reject command (used to reject unknown or invalid commands)
	bt_l2cap_send_cmd	This is function bt_l2cap_send_cmd.

	bt_l2cap_send_data	brief Send data over an L2CAP channel ingroup l2cap details This function sends data over the specified L2CAP channel. param channel The L2CAP channel to send data over. param data The pointer to the data. param len The length of the data. param callback The callback function that is called when sending the data has been completed. return li c TRUE if the function succeeds. li c FALSE otherwise. The callback function is not called in this case.
	bt_l2cap_update_conn_parameters	This is function bt_l2cap_update_conn_parameters.
	_bt_l2cap_clear_channel_cmd_queue	This is function _bt_l2cap_clear_channel_cmd_queue.
	_bt_l2cap_eretr_handle_xmit_event	This is function _bt_l2cap_eretr_handle_xmit_event.
	_bt_l2cap_eretr_retr_frames	This is function _bt_l2cap_eretr_retr_frames.
	_bt_l2cap_fcs	This is function _bt_l2cap_fcs.
	_bt_l2cap_get_tick_count	This is function _bt_l2cap_get_tick_count.
	_bt_l2cap_init_signal	This is function _bt_l2cap_init_signal.
	_bt_l2cap_init_timer	This is function _bt_l2cap_init_timer.
	_bt_l2cap_notify_and_remove	This is function _bt_l2cap_notify_and_remove.
	_bt_l2cap_send_commands_from_queue	This is function _bt_l2cap_send_commands_from_queue.
	_bt_l2cap_set_signal	This is function _bt_l2cap_set_signal.
	_l2cap_allocate_buffers	Defined by OEM through library configuration
	_l2cap_data_receive_callback	From l2cap_recv.c
	_pack_cmd_reject	This is function _pack_cmd_reject.
	_pack_config_request	This is function _pack_config_request.
	_pack_config_response	This is function _pack_config_response.
	_pack_conn_param_update_request	This is function _pack_conn_param_update_request.
	_pack_conn_param_update_response	This is function _pack_conn_param_update_response.
	_pack_conn_request	This is function _pack_conn_request.
	_pack_conn_response	This is function _pack_conn_response.
	_pack_dconn_request	This is function _pack_dconn_request.
	_pack_dconn_response	This is function _pack_dconn_response.
	_pack_echo_request	This is function _pack_echo_request.
	_pack_echo_response	This is function _pack_echo_response.
	_pack_info_request	This is function _pack_info_request.
	_pack_info_response	This is function _pack_info_response.
	_process_config_req	From cmd_config.c
	_process_config_res	This is function _process_config_res.
	_process_conn_param_update_req	This is function _process_conn_param_update_req.
	_process_conn_param_update_res	This is function _process_conn_param_update_res.
	_process_conn_req	From cmd_connect.c
	_process_conn_res	This is function _process_conn_res.
	_process_dconn_req	From cmd_disconnect.c
	_process_dconn_res	This is function _process_dconn_res.
	_process_echo_req	From cmd_echo.c
	_process_echo_res	This is function _process_echo_res.
	_process_info_req	From cmd_info.c
	_process_info_res	This is function _process_info_res.
	_process_reject	From cmd_reject.c
	_process_unknown_req	From cmd_unknown.c
	_process_unknown_res	This is function _process_unknown_res.
	_read_cmd_reject	This is function _read_cmd_reject.
	_read_config_request	This is function _read_config_request.
	_read_config_response	This is function _read_config_response.
	_read_conn_param_update_request	This is function _read_conn_param_update_request.
	_read_conn_param_update_response	This is function _read_conn_param_update_response.
	_read_conn_request	This is function _read_conn_request.
	_read_conn_response	This is function _read_conn_response.
	_read_dconn_request	This is function _read_dconn_request.

	_read_dconn_response	This is function _read_dconn_response.
	_read_echo_request	This is function _read_echo_request.
	_read_echo_response	This is function _read_echo_response.
	_read_info_request	This is function _read_info_request.
	_read_info_response	This is function _read_info_response.
	_bt_l2cap_eretr_pack_config_request	This is function _bt_l2cap_eretr_pack_config_request.
	_bt_l2cap_eretr_recv	This is function _bt_l2cap_eretr_recv.
	_bt_l2cap_eretr_send_data	This is function _bt_l2cap_eretr_send_data.
	_bt_l2cap_eretr_send_pending_frames	This is function _bt_l2cap_eretr_send_pending_frames.
	_bt_l2cap_eretr_send_smart_data	This is function _bt_l2cap_eretr_send_smart_data.
	_bt_l2cap_recv_req_seq_and_fbit	This is function _bt_l2cap_recv_req_seq_and_fbit.
	_bt_l2cap_send_ack	This is function _bt_l2cap_send_ack.
	_bt_l2cap_send_i_or_rr_or_rnr	This is function _bt_l2cap_send_i_or_rr_or_rnr.
	_bt_l2cap_send_rej	This is function _bt_l2cap_send_rej.
	_bt_l2cap_send_rnr	This is function _bt_l2cap_send_rnr.
	_bt_l2cap_send_rr	This is function _bt_l2cap_send_rr.
	_bt_l2cap_send_rr_or_rnr	This is function _bt_l2cap_send_rr_or_rnr.
	bt_l2cap_disconnect_ex	This is function bt_l2cap_disconnect_ex.
	bt_l2cap_free_channel	This is function bt_l2cap_free_channel.
	bt_l2cap_hci_has_open_channels	This is function bt_l2cap_hci_has_open_channels.
	bt_l2cap_is_channel_open	This is function bt_l2cap_is_channel_open.
	bt_l2cap_send_config	This is function bt_l2cap_send_config.
	bt_l2cap_send_smart_data	This is function bt_l2cap_send_smart_data.
	_bt_l2cap_process_connect_signal	This is function _bt_l2cap_process_connect_signal.

Misc. Data Types and Constants

	Name	Description
	CSR_SNK_ADC	This is macro CSR_SNK_ADC.
	CSR_SNK_FASTPIPE	This is macro CSR_SNK_FASTPIPE.
	CSR_SNK_FM	This is macro CSR_SNK_FM.
	CSR_SNK_I2S	This is macro CSR_SNK_I2S.
	CSR_SNK_L2CAP	This is macro CSR_SNK_L2CAP.
	CSR_SNK_PCM	This is macro CSR_SNK_PCM.
	CSR_SNK_SCO	This is macro CSR_SNK_SCO.
	CSR_SNK_SPDIF	This is macro CSR_SNK_SPDIF.
	CSR_SRC_ADC	This is macro CSR_SRC_ADC.
	CSR_SRC_FASTPIPE	This is macro CSR_SRC_FASTPIPE.
	CSR_SRC_FM	This is macro CSR_SRC_FM.
	CSR_SRC_I2S	This is macro CSR_SRC_I2S.
	CSR_SRC_L2CAP	This is macro CSR_SRC_L2CAP.
	CSR_SRC_MIC	This is macro CSR_SRC_MIC.
	CSR_SRC_PCM	This is macro CSR_SRC_PCM.
	CSR_SRC_SCO	This is macro CSR_SRC_SCO.
	CSR_SRC_SPDIF	This is macro CSR_SRC_SPDIF.
	CSR_VARID_CACHED_TEMPERATURE	This is macro CSR_VARID_CACHED_TEMPERATURE.
	CSR_VARID_ENABLE_SCO_STREAMS	This is macro CSR_VARID_ENABLE_SCO_STREAMS.
	CSR_VARID_MAP_SCO_AUDIO	This is macro CSR_VARID_MAP_SCO_AUDIO.
	CSR_VARID_PIO	This is macro CSR_VARID_PIO.
	CSR_VARID_PIO_DIRECTION_MASK	This is macro CSR_VARID_PIO_DIRECTION_MASK.
	CSR_VARID_PIO_PROTECT_MASK	This is macro CSR_VARID_PIO_PROTECT_MASK.
	CSR_VARID_RSSI_ACL	This is macro CSR_VARID_RSSI_ACL.
	CSR_VARID_STREAM_CLOSE_SINK	This is macro CSR_VARID_STREAM_CLOSE_SINK.
	CSR_VARID_STREAM_CLOSE_SOURCE	This is macro CSR_VARID_STREAM_CLOSE_SOURCE.
	CSR_VARID_STREAM_CONFIGURE	This is macro CSR_VARID_STREAM_CONFIGURE.
	CSR_VARID_STREAM_CONNECT	This is macro CSR_VARID_STREAM_CONNECT.

	CSR_VARID_STREAM_GET_SINK	This is macro CSR_VARID_STREAM_GET_SINK.
	CSR_VARID_STREAM_GET_SOURCE	This is macro CSR_VARID_STREAM_GET_SOURCE.
	GETRESP_BAD_REQ	This is macro GETRESP_BAD_REQ.
	GETRESP_ERROR	This is macro GETRESP_ERROR.
	GETRESP_NO_ACCESS	This is macro GETRESP_NO_ACCESS.
	GETRESP_NO_SUCH_VARID	This is macro GETRESP_NO_SUCH_VARID.
	GETRESP_NO_VALUE	This is macro GETRESP_NO_VALUE.
	GETRESP_OK	BCCMD GETRESP status codes
	GETRESP_PERMISSION_DENIED	This is macro GETRESP_PERMISSION_DENIED.
	GETRESP_READ_ONLY	This is macro GETRESP_READ_ONLY.
	GETRESP_TOO_BIG	This is macro GETRESP_TOO_BIG.
	GETRESP_WRITE_ONLY	This is macro GETRESP_WRITE_ONLY.
	H	PS key store (0 = use default)
	PS_DEFAULT	This is macro PS_DEFAULT.
	PS_F	This is macro PS_F.
	PS_I	This is macro PS_I.
	PS_RAM	This is macro PS_RAM.
	PS_ROM	This is macro PS_ROM.
	PSKEY_ANA_FREQ	This is macro PSKEY_ANA_FREQ.
	PSKEY_BDADDR	Select PS key definitions
	PSKEY_CLOCK_REQUEST_ENABLE	This is macro PSKEY_CLOCK_REQUEST_ENABLE.
	PSKEY_DEEP_SLEEP_CLEAR_RTS	This is macro PSKEY_DEEP_SLEEP_CLEAR_RTS.
	PSKEY_DEEP_SLEEP_STATE	This is macro PSKEY_DEEP_SLEEP_STATE.
	PSKEY_DEEP_SLEEP_USE_EXTERNAL_CLOCK	This is macro PSKEY_DEEP_SLEEP_USE_EXTERNAL_CLOCK.
	PSKEY_DEEP_SLEEP_WAKE_CTS	This is macro PSKEY_DEEP_SLEEP_WAKE_CTS.
	PSKEY_DIGITAL_AUDIO_BITS_PER_SAMPLE	This is macro PSKEY_DIGITAL_AUDIO_BITS_PER_SAMPLE.
	PSKEY_DIGITAL_AUDIO_CONFIG	This is macro PSKEY_DIGITAL_AUDIO_CONFIG.
	PSKEY_HCI_NOP_DISABLE	This is macro PSKEY_HCI_NOP_DISABLE.
	PSKEY_HOST_INTERFACE	This is macro PSKEY_HOST_INTERFACE.
	PSKEY_HOSTIO_MAP_SCO_PCM	This is macro PSKEY_HOSTIO_MAP_SCO_PCM.
	PSKEY_HOSTIO_UART_RESET_TIMEOUT	This is macro PSKEY_HOSTIO_UART_RESET_TIMEOUT.
	PSKEY_PCM_PULL_CONTROL	This is macro PSKEY_PCM_PULL_CONTROL.
	PSKEY_PIO_DEEP_SLEEP_EITHER_LEVEL	This is macro PSKEY_PIO_DEEP_SLEEP_EITHER_LEVEL.
	PSKEY_UART_BAUDRATE	This is macro PSKEY_UART_BAUDRATE.
	PSKEY_UART_BITRATE	This is macro PSKEY_UART_BITRATE.
	PSKEY_UART_CONFIG_BCSP	This is macro PSKEY_UART_CONFIG_BCSP.
	PSKEY_UART_CONFIG_H4	This is macro PSKEY_UART_CONFIG_H4.
	PSKEY_UART_CONFIG_H5	This is macro PSKEY_UART_CONFIG_H5.
	PSKEY_UART_TX_WINDOW_SIZE	This is macro PSKEY_UART_TX_WINDOW_SIZE.
	PSKEY_VM_DISABLE	This is macro PSKEY_VM_DISABLE.
	SET_PS_VALUE_BDADDR	This is macro SET_PS_VALUE_BDADDR.
	SET_PS_VALUE_UINT16	Macros for defining lists of PS values for use with btx_csr_set_ps_vars().
	SET_PS_VALUE_UINT32	This is macro SET_PS_VALUE_UINT32.
	W	This is macro W.
	BTX_TI_EXEC_SCRIPT_OEM_RX_BUFFER_SIZE	This is macro BTX_TI_EXEC_SCRIPT_OEM_RX_BUFFER_SIZE.
	btx_ti_get_script__CC2560_ServicePack	Alias for the latest script
	btx_ti_get_script__CC2564_BLE_Init	This is macro btx_ti_get_script__CC2564_BLE_Init.
	btx_ti_get_script__CC2564_ServicePack	Aliases for latest scripts
	btx_ti_get_script__CC2564B_BLE_Init	This is macro btx_ti_get_script__CC2564B_BLE_Init.
	btx_ti_get_script__CC2564B_ServicePack	Aliases for latest scripts
	BTX_TI_MODULATION_SCHEME_8_DPSK	This is macro BTX_TI_MODULATION_SCHEME_8_DPSK.
	BTX_TI_MODULATION_SCHEME_CW	This is macro BTX_TI_MODULATION_SCHEME_CW.
	BTX_TI_MODULATION_SCHEME_GFSK	This is macro BTX_TI_MODULATION_SCHEME_GFSK.
	BTX_TI_MODULATION_SCHEME_P4_DPSK	This is macro BTX_TI_MODULATION_SCHEME_P4_DPSK.

	BTX_TI_MODULATION_TYPE_EDR2	This is macro BTX_TI_MODULATION_TYPE_EDR2.
	BTX_TI_MODULATION_TYPE_EDR3	This is macro BTX_TI_MODULATION_TYPE_EDR3.
	BTX_TI_MODULATION_TYPE_GFSK	This is macro BTX_TI_MODULATION_TYPE_GFSK.
	BTX_TI_TEST_PATTERN_ALL_0	This is macro BTX_TI_TEST_PATTERN_ALL_0.
	BTX_TI_TEST_PATTERN_ALL_1	This is macro BTX_TI_TEST_PATTERN_ALL_1.
	BTX_TI_TEST_PATTERN_F0F0	This is macro BTX_TI_TEST_PATTERN_F0F0.
	BTX_TI_TEST_PATTERN_FF00	This is macro BTX_TI_TEST_PATTERN_FF00.
	BTX_TI_TEST_PATTERN_PN15	This is macro BTX_TI_TEST_PATTERN_PN15.
	BTX_TI_TEST_PATTERN_PN9	This is macro BTX_TI_TEST_PATTERN_PN9.
	BTX_TI_TEST_PATTERN_USER	This is macro BTX_TI_TEST_PATTERN_USER.
	BTX_TI_TEST_PATTERN_Z0Z0	This is macro BTX_TI_TEST_PATTERN_Z0Z0.
	_W	This is macro _W.
	_btx_csr_autobaud_buffer_t	This is record _btx_csr_autobaud_buffer_t.
	_btx_csr_bccmd_header_s	This is type btx_csr_bccmd_header_t.
	_btx_csr_cached_temperature_s	This is type btx_csr_cached_temperature_t.
	_btx_csr_exec_script_buffer_t	This is record _btx_csr_exec_script_buffer_t.
	_btx_csr_pio_direction_mask_s	This is type btx_csr_pio_direction_mask_t.
	_btx_csr_pio_protection_mask_s	This is type btx_csr_pio_protection_mask_t.
	_btx_csr_pio_s	This is type btx_csr_pio_t.
	_btx_csr_rssi_acl_s	This is type btx_csr_rssi_acl_t.
	_btx_csr_script_t	This is type btx_csr_script_t.
	_btx_csr_set_ps_vars_buffer_t	This is record _btx_csr_set_ps_vars_buffer_t.
	_btx_csr_strm_get_sink_s	This is type btx_csr_strm_get_sink_t.
	_btx_csr_strm_get_source_s	This is type btx_csr_strm_get_source_t.
	_btx_csr_var_u	This is type btx_csr_var_t.
	_btx_ti_exec_script_buffer_t	This is record _btx_ti_exec_script_buffer_t.
	_btx_ti_exec_script_oem_buffer_t	This is record _btx_ti_exec_script_oem_buffer_t.
	_btx_ti_script_t	This is type btx_ti_script_t.
	btx_csr_autobaud_buffer_t	This is type btx_csr_autobaud_buffer_t.
	btx_csr_autobaud_callback_fp	This is type btx_csr_autobaud_callback_fp.
	btx_csr_bccmd_header_t	This is type btx_csr_bccmd_header_t.
	btx_csr_cached_temperature_t	This is type btx_csr_cached_temperature_t.
	btx_csr_exec_script_buffer_t	This is type btx_csr_exec_script_buffer_t.
	btx_csr_exec_script_callback_fp	This is type btx_csr_exec_script_callback_fp.
	btx_csr_get_ps_var_callback_fp	This is type btx_csr_get_ps_var_callback_fp.
	btx_csr_get_var_callback_fp	This is type btx_csr_get_var_callback_fp.
	btx_csr_pio_direction_mask_t	This is type btx_csr_pio_direction_mask_t.
	btx_csr_pio_protection_mask_t	This is type btx_csr_pio_protection_mask_t.
	btx_csr_pio_t	This is type btx_csr_pio_t.
	btx_csr_rssi_acl_t	This is type btx_csr_rssi_acl_t.
	btx_csr_script_t	This is type btx_csr_script_t.
	btx_csr_set_ps_vars_buffer_t	This is type btx_csr_set_ps_vars_buffer_t.
	btx_csr_set_ps_vars_callback_fp	This is type btx_csr_set_ps_vars_callback_fp.
	btx_csr_set_var_callback_fp	This is type btx_csr_set_var_callback_fp.
	btx_csr_strm_get_sink_t	This is type btx_csr_strm_get_sink_t.
	btx_csr_strm_get_source_t	This is type btx_csr_strm_get_source_t.
	btx_csr_var_t	This is type btx_csr_var_t.
	btx_ti_completion_callback_fp	This is type btx_ti_completion_callback_fp.
	btx_ti_exec_script_buffer_t	This is type btx_ti_exec_script_buffer_t.
	btx_ti_exec_script_oem_buffer_t	This is type btx_ti_exec_script_oem_buffer_t.
	btx_ti_exec_script_oem_callback_fp	This is type btx_ti_exec_script_oem_callback_fp.
	btx_ti_script_t	This is type btx_ti_script_t.
	BTX_TI_A3DP_ROLE_SINK	This is macro BTX_TI_A3DP_ROLE_SINK.
	BTX_TI_A3DP_ROLE_SOURCE	This is macro BTX_TI_A3DP_ROLE_SOURCE.
	BTX_TI_AVPR_DISABLE	This is macro BTX_TI_AVPR_DISABLE.

	BTX_TI_AVPR_DO_NOT_LOAD_A3DP_CODE	This is macro BTX_TI_AVPR_DO_NOT_LOAD_A3DP_CODE.
	BTX_TI_AVPR_ENABLE	This is macro BTX_TI_AVPR_ENABLE.
	BTX_TI_AVPR_LOAD_A3DP_CODE	This is macro BTX_TI_AVPR_LOAD_A3DP_CODE.
	btx_ti_get_script__CC2564B_AVPR_Init	This is macro btx_ti_get_script__CC2564B_AVPR_Init.
	BTX_TI_SBC_ALLOCATION_METHOD_LOUDNESS	This is macro BTX_TI_SBC_ALLOCATION_METHOD_LOUDNESS.
	BTX_TI_SBC_ALLOCATION_METHOD_SNR	This is macro BTX_TI_SBC_ALLOCATION_METHOD_SNR.
	BTX_TI_SBC_BLOCKS_12	This is macro BTX_TI_SBC_BLOCKS_12.
	BTX_TI_SBC_BLOCKS_16	This is macro BTX_TI_SBC_BLOCKS_16.
	BTX_TI_SBC_BLOCKS_4	This is macro BTX_TI_SBC_BLOCKS_4.
	BTX_TI_SBC_BLOCKS_8	This is macro BTX_TI_SBC_BLOCKS_8.
	BTX_TI_SBC_CHANNEL_MODE_DUAL_CHANNEL	This is macro BTX_TI_SBC_CHANNEL_MODE_DUAL_CHANNEL.
	BTX_TI_SBC_CHANNEL_MODE_JOINT_STEREO	This is macro BTX_TI_SBC_CHANNEL_MODE_JOINT_STEREO.
	BTX_TI_SBC_CHANNEL_MODE_MONO	This is macro BTX_TI_SBC_CHANNEL_MODE_MONO.
	BTX_TI_SBC_CHANNEL_MODE_STEREO	This is macro BTX_TI_SBC_CHANNEL_MODE_STEREO.
	BTX_TI_SBC_SAMPLE_FREQUENCY_16000	This is macro BTX_TI_SBC_SAMPLE_FREQUENCY_16000.
	BTX_TI_SBC_SAMPLE_FREQUENCY_32000	This is macro BTX_TI_SBC_SAMPLE_FREQUENCY_32000.
	BTX_TI_SBC_SAMPLE_FREQUENCY_44100	This is macro BTX_TI_SBC_SAMPLE_FREQUENCY_44100.
	BTX_TI_SBC_SAMPLE_FREQUENCY_48000	This is macro BTX_TI_SBC_SAMPLE_FREQUENCY_48000.
	CSR_I2S_STREAM_CFG_KEY_AUDIO_ATTEN	This is macro CSR_I2S_STREAM_CFG_KEY_AUDIO_ATTEN.
	CSR_I2S_STREAM_CFG_KEY_AUDIO_ATTEN_ENABLE	This is macro CSR_I2S_STREAM_CFG_KEY_AUDIO_ATTEN_ENABLE.
	CSR_I2S_STREAM_CFG_KEY_BITS_PER_SAMPLE	This is macro CSR_I2S_STREAM_CFG_KEY_BITS_PER_SAMPLE.
	CSR_I2S_STREAM_CFG_KEY_CROP_ENABLE	This is macro CSR_I2S_STREAM_CFG_KEY_CROP_ENABLE.
	CSR_I2S_STREAM_CFG_KEY_JUSTIFY_DELAY	This is macro CSR_I2S_STREAM_CFG_KEY_JUSTIFY_DELAY.
	CSR_I2S_STREAM_CFG_KEY_JUSTIFY_FORMAT	This is macro CSR_I2S_STREAM_CFG_KEY_JUSTIFY_FORMAT.
	CSR_I2S_STREAM_CFG_KEY_JUSTIFY_RESOLUTION	This is macro CSR_I2S_STREAM_CFG_KEY_JUSTIFY_RESOLUTION.
	CSR_I2S_STREAM_CFG_KEY_MASTER	This is macro CSR_I2S_STREAM_CFG_KEY_MASTER.
	CSR_I2S_STREAM_CFG_KEY_MCLK	This is macro CSR_I2S_STREAM_CFG_KEY_MCLK.
	CSR_I2S_STREAM_CFG_KEY_POLARITY	This is macro CSR_I2S_STREAM_CFG_KEY_POLARITY.
	CSR_I2S_STREAM_CFG_KEY_RX_START_SAMPLE	This is macro CSR_I2S_STREAM_CFG_KEY_RX_START_SAMPLE.
	CSR_I2S_STREAM_CFG_KEY_SYNC_RATE	This is macro CSR_I2S_STREAM_CFG_KEY_SYNC_RATE.
	CSR_I2S_STREAM_CFG_KEY_TX_START_SAMPLE	This is macro CSR_I2S_STREAM_CFG_KEY_TX_START_SAMPLE.
	CSR_MAX_HQ_PACKET_LEN	This is macro CSR_MAX_HQ_PACKET_LEN.
	CSR_MSG_TYPE_GETREQ	This is macro CSR_MSG_TYPE_GETREQ.
	CSR_MSG_TYPE_GETRESP	This is macro CSR_MSG_TYPE_GETRESP.
	CSR_MSG_TYPE_SETREQ	This is macro CSR_MSG_TYPE_SETREQ.
	CSR_PCM_STREAM_CFG_KEY_LSB_FIRST	This is macro CSR_PCM_STREAM_CFG_KEY_LSB_FIRST.
	CSR_PCM_STREAM_CFG_KEY_MANCHESTER	This is macro CSR_PCM_STREAM_CFG_KEY_MANCHESTER.
	CSR_PCM_STREAM_CFG_KEY_MANCHESTER_SLAVE	This is macro CSR_PCM_STREAM_CFG_KEY_MANCHESTER_SLAVE.
	CSR_PCM_STREAM_CFG_KEY_MASTER	This is macro CSR_PCM_STREAM_CFG_KEY_MASTER.
	CSR_PCM_STREAM_CFG_KEY_MCLK	This is macro CSR_PCM_STREAM_CFG_KEY_MCLK.
	CSR_PCM_STREAM_CFG_KEY_SHORT_SYNC	This is macro CSR_PCM_STREAM_CFG_KEY_SHORT_SYNC.
	CSR_PCM_STREAM_CFG_KEY_SIGN_EXTEND	This is macro CSR_PCM_STREAM_CFG_KEY_SIGN_EXTEND.
	CSR_PCM_STREAM_CFG_KEY_SLOT_COUNT	This is macro CSR_PCM_STREAM_CFG_KEY_SLOT_COUNT.
	CSR_PCM_STREAM_CFG_KEY_SYNC_RATE	This is macro CSR_PCM_STREAM_CFG_KEY_SYNC_RATE.
	CSR_PCM_STREAM_CFG_KEY_TX_TRISTATE	This is macro CSR_PCM_STREAM_CFG_KEY_TX_TRISTATE.
	CSR_VARID_CAPABILITY_DOWNLOAD_INDICATION	This is macro CSR_VARID_CAPABILITY_DOWNLOAD_INDICATION.
	CSR_VARID_CREATE_OPERATOR_C	This is macro CSR_VARID_CREATE_OPERATOR_C.
	CSR_VARID_CREATE_OPERATOR_P	This is macro CSR_VARID_CREATE_OPERATOR_P.
	CSR_VARID_DESTROY_OPERATORS	This is macro CSR_VARID_DESTROY_OPERATORS.
	CSR_VARID_DSPMANAGER_CONFIG_REQUEST	This is macro CSR_VARID_DSPMANAGER_CONFIG_REQUEST.

	CSR_VARID_MESSAGE_FROM_OPERATOR	Select Variable definitions
	CSR_VARID_START_OPERATORS	This is macro CSR_VARID_START_OPERATORS.
	CSR_VARID_STOP_OPERATORS	This is macro CSR_VARID_STOP_OPERATORS.
	CSR_VARID_STREAM_DRAINED_NOTIFICATION	This is macro CSR_VARID_STREAM_DRAINED_NOTIFICATION.
	CSR_VARID_STREAM_TRANSFORM_DISCONNECT	This is macro CSR_VARID_STREAM_TRANSFORM_DISCONNECT.
	PSKEY_BLE_DEFAULT_TX_POWER	This is macro PSKEY_BLE_DEFAULT_TX_POWER.
	PSKEY_DEEP_SLEEP_EXTERNAL_CLOCK_SOURCE_PIO	This is macro PSKEY_DEEP_SLEEP_EXTERNAL_CLOCK_SOURCE_PIO.
	PSKEY_H_HC_FC_MAX_SCO_PKT_LEN	This is macro PSKEY_H_HC_FC_MAX_SCO_PKT_LEN.
	PSKEY_H_HC_FC_MAX_SCO_PKTS	This is macro PSKEY_H_HC_FC_MAX_SCO_PKTS.
	PSKEY_PCM_CLOCK_RATE	This is macro PSKEY_PCM_CLOCK_RATE.
	PSKEY_PCM_CONFIG32	This is macro PSKEY_PCM_CONFIG32.
	PSKEY_PCM_FORMAT	This is macro PSKEY_PCM_FORMAT.
	PSKEY_PCM_SYNC_RATE	This is macro PSKEY_PCM_SYNC_RATE.
	PSKEY_PCM_USE_LOW_JITTER_MODE	This is macro PSKEY_PCM_USE_LOW_JITTER_MODE.
	PSKEY_UART_CONFIG_H4DS	This is macro PSKEY_UART_CONFIG_H4DS.
	BT_LOG_LEVEL_A2DP_PAKCET	This is macro BT_LOG_LEVEL_A2DP_PAKCET.
	DCRH	Number of configuration blocks
	btx_csr_bccmd_callback_fp	This is type btx_csr_bccmd_callback_fp.
	btx_csr_bccmd_listener_t	This is type btx_csr_bccmd_listener_t.
	btx_csr_create_operator_c_t	This is type btx_csr_create_operator_c_t.
	btx_csr_exec_hq_script_buffer_t	This is type btx_csr_exec_hq_script_buffer_t.
	btx_csr_exec_hq_script_callback_fp	This is type btx_csr_exec_hq_script_callback_fp.
	btx_csr_get_script_fp	This is type btx_csr_get_script_fp.
	btx_csr_strm_connect_t	This is type btx_csr_strm_connect_t.
	btx_ti_codec_config_t	This is type btx_ti_codec_config_t.
	_btx_csr_bccmd_listener_t	This is record _btx_csr_bccmd_listener_t.
	_btx_csr_create_operator_c_s	This is type btx_csr_create_operator_c_t.
	_btx_csr_exec_hq_script_buffer_t	This is record _btx_csr_exec_hq_script_buffer_t.
	_btx_csr_strm_connect_s	This is type btx_csr_strm_connect_t.
	_btx_ti_codec_config_s	This is type btx_ti_codec_config_t.
	PSKEY_LC_DEFAULT_TX_POWER	This is macro PSKEY_LC_DEFAULT_TX_POWER.
	PSKEY_LC_MAX_TX_POWER	This is macro PSKEY_LC_MAX_TX_POWER.
	PSKEY_LC_MAX_TX_POWER_NO_RSSI	This is macro PSKEY_LC_MAX_TX_POWER_NO_RSSI.

Misc. Functions

	Name	Description
	btx_csr_alloc_bccmd_getreq	This is function btx_csr_alloc_bccmd_getreq.
	btx_csr_alloc_bccmd_setreq	This is function btx_csr_alloc_bccmd_setreq.
	btx_csr_autobaud	brief Configure controller's UART speed. ingroup btx_csr details This function makes the controller auto-configure its UART speed. The host transport must be set to H4. This function works only with BC6 controllers.
	btx_csr_bc7_sel_host_interface_h4	brief Configure controller's UART speed and host interface. ingroup btx_csr details This function makes the controller auto-configure its UART speed and select H4 as host interface. PS_KEY_HOST_INTERFACE must not be set. PS_KEY_UART_BITRATE must be set to 0. This function works only with BC7 controllers.
	btx_csr_enable_tx	brief Enable/disable transmitter ingroup btx_csr param enable Specifies whether the transmitter should be enable or disabled. param callback The callback function that will be called after the command has completed. param callback_param A pointer to arbitrary data to be passed to the c callback callback.

	btx_csr_exec_script	brief Patch controller's firmware ingroup btx_csr details This function executes a script that patches the controller's firmware. The c script must point to a structure that contain a complete patch script for a particular controller model and revision. If the revision specified in the script and revision read from the controller are the same btx_csr_patch_controller() loads the script to the controller and calls the c callback with the first parameter TRUE. Otherwise the c callback is called with the first parameter FALSE. If support for multiple firmware revisions is neede use btx_csr_patch_controller(). param script Array of... more
	btx_csr_get_cached_temperature	brief Get chip's cached temperature ingroup btx_csr param callback The callback function that will be called after the command has completed. param callback_param A pointer to arbitrary data to be passed to the c callback callback.
	btx_csr_get_ps_var	This is function btx_csr_get_ps_var.
	btx_csr_get_ps_var_ex	This is function btx_csr_get_ps_var_ex.
	btx_csr_get_rssi_acl	brief Get RSSI ingroup btx_csr details This function retrieves the RSSI for a given HCI ACL handle. param hconn ACL connection handle. param callback The callback function that will be called after the command has completed. param callback_param A pointer to arbitrary data to be passed to the c callback callback.
	btx_csr_get_script__PB_27_R20_BC6ROM_A04	brief Return script for patching BlueCore 6 ingroup btx_csr
	btx_csr_get_script__PB_90_REV6	brief Return script for patching CSR8810 (BlueCore 7) ingroup btx_csr
	btx_csr_get_var	This is function btx_csr_get_var.
	btx_csr_set_ps_var	This is function btx_csr_set_ps_var.
	btx_csr_set_ps_var_ex	This is function btx_csr_set_ps_var_ex.
	btx_csr_set_ps_vars	brief Write PS variables ingroup btx_csr details param ps_vars PS values param buffer A buffer for storing temporary data during function execution. param callback The callback function that will be called when all PS values have been sent to the controller or error occurred. param callback_param A pointer to arbitrary data to be passed to the c callback callback..
	btx_csr_set_ps_vars_ex	brief Write PS variables ingroup btx_csr param ps_vars PS values param buffer A buffer for storing temporary data during function execution. param store param callback The callback function that will be called when all PS values have been sent to the controller or error occurred. param callback_param A pointer to arbitrary data to be passed to the c callback callback..
	btx_csr_set_var	This is function btx_csr_set_var.
	btx_csr_warm_reset	brief Warm reset ingroup btx_csr details This function performs warm reset of the controller. All patches and configuration parameters sent to the controller before warm reset are kept intact.
	btx_csr_warm_reset_ex	brief Warm reset ingroup btx_csr details This function performs warm reset of the controller. All patches and configuration parameters sent to the controller before warm reset are kept intact. Since the controller does not respond to the warm reset command as it starts resetting immediately upon receiving the command, the c callback is called right after the command packet has been transmitted to the controller. param callback The callback function that will be called after the warm reset command has been sent to the controller. param callback_param A pointer to arbitrary data to be passed to the c callback callback.... more
	btx_ti_drpb_enable_rf_calibration	This is function btx_ti_drpb_enable_rf_calibration.
	btx_ti_drpb_set_power_vector	This is function btx_ti_drpb_set_power_vector.

	btx_ti_drpb_tester_con_tx	This is function btx_ti_drpb_tester_con_tx.
	btx_ti_enable_deep_sleep	This is function btx_ti_enable_deep_sleep.
	btx_ti_enable_fast_clock_crystal	This is function btx_ti_enable_fast_clock_crystal.
	btx_ti_enable_low_power_scan	This is function btx_ti_enable_low_power_scan.
	btx_ti_enable_low_power_scan_default	This is function btx_ti_enable_low_power_scan_default.
	btx_ti_exec_script	This is function btx_ti_exec_script.
	btx_ti_exec_script_oem	This is function btx_ti_exec_script_oem.
	btx_ti_get_script__BL6450_2_0_BT_Service_Pack_2_36	CC2560 Scripts (Service pack 2.36)
	btx_ti_get_script__BL6450_2_0_BT_Service_Pack_2_44	CC2560 Scripts (Service pack 2.44)
	btx_ti_get_script__BL6450L_BT_Service_Pack_2_10	CC2564 Scripts (Service pack 2.10)
	btx_ti_get_script__BL6450L_BT_Service_Pack_2_10_BLE_AddOn	This is function btx_ti_get_script__BL6450L_BT_Service_Pack_2_10_BLE_AddOn.
	btx_ti_get_script__BL6450L_BT_Service_Pack_2_12	CC2564 Scripts (Service pack 2.12)
	btx_ti_get_script__BL6450L_BT_Service_Pack_2_12_BLE_AddOn	This is function btx_ti_get_script__BL6450L_BT_Service_Pack_2_12_BLE_AddOn.
	btx_ti_get_script__BL6450L_BT_Service_Pack_2_7	CC2564 Scripts (Service pack 2.7)
	btx_ti_get_script__BL6450L_BT_Service_Pack_2_8_AVPR_AddOn	This is function btx_ti_get_script__BL6450L_BT_Service_Pack_2_8_AVPR_AddOn.
	btx_ti_get_script__BL6450L_BT_Service_Pack_2_8_BLE_AddOn	This is function btx_ti_get_script__BL6450L_BT_Service_Pack_2_8_BLE_AddOn.
	btx_ti_get_script__BL6450L_BT_Service_Pack_2_8_Short	CC2564 Scripts (Service pack 2.8)
	btx_ti_get_script__BL6450x_BT_Service_Pack_2_7_AVPR_AddOn	This is function btx_ti_get_script__BL6450x_BT_Service_Pack_2_7_AVPR_AddOn.
	btx_ti_get_script__BL6450x_BT_Service_Pack_2_7_BLE_AddOn	This is function btx_ti_get_script__BL6450x_BT_Service_Pack_2_7_BLE_AddOn.
	btx_ti_get_script__CC2564B_BT_Service_Pack_0_1	CC2564B Scripts (Service pack 0.1)
	btx_ti_get_script__CC2564B_BT_Service_Pack_0_1_BLE_AddOn	This is function btx_ti_get_script__CC2564B_BT_Service_Pack_0_1_BLE_AddOn.
	btx_ti_get_script__CC2564B_BT_Service_Pack_0_2	CC2564B Scripts (Service pack 0.2)
	btx_ti_get_script__CC2564B_BT_Service_Pack_0_2_BLE_AddOn	This is function btx_ti_get_script__CC2564B_BT_Service_Pack_0_2_BLE_AddOn.
	btx_ti_get_script__WL127xL_BT_Service_Pack_2_3	CC2564 Scripts (Service pack 2.3)
	btx_ti_get_script__WL127xL_BT_Service_Pack_2_3_AVPR_AddOn	This is function btx_ti_get_script__WL127xL_BT_Service_Pack_2_3_AVPR_AddOn.
	btx_ti_get_script__WL127xL_BT_Service_Pack_2_3_BLE_AddOn	This is function btx_ti_get_script__WL127xL_BT_Service_Pack_2_3_BLE_AddOn.
	btx_ti_get_script__WL127xL_BT_Service_Pack_2_3_DC2DC_AddOn	This is function btx_ti_get_script__WL127xL_BT_Service_Pack_2_3_DC2DC_AddOn.
	btx_ti_get_script__WL127xL_BT_Service_Pack_2_4	CC2564 Scripts (Service pack 2.4)
	btx_ti_get_script__WL127xL_BT_Service_Pack_2_4_AVPR_AddOn	This is function btx_ti_get_script__WL127xL_BT_Service_Pack_2_4_AVPR_AddOn.
	btx_ti_get_script__WL127xL_BT_Service_Pack_2_4_BLE_AddOn	This is function btx_ti_get_script__WL127xL_BT_Service_Pack_2_4_BLE_AddOn.
	btx_ti_get_script__WL127xL_BT_Service_Pack_2_4_DC2DC_AddOn	This is function btx_ti_get_script__WL127xL_BT_Service_Pack_2_4_DC2DC_AddOn.
	btx_ti_get_script__XWL1271L1_BT_ServicePack_1_3	CC2564 Scripts (Service pack 1.3)
	btx_ti_get_script__XWL1271L1_BT_ServicePack_1_3_BLE_Init	This is function btx_ti_get_script__XWL1271L1_BT_ServicePack_1_3_BLE_Init.

	btx_ti_init_ble_controller	This is function btx_ti_init_ble_controller.
	btx_ti_init_controller	This is function btx_ti_init_controller.
	btx_ti_set_uart_baud_rate	This is function btx_ti_set_uart_baud_rate.
	btx_ti_write_bdaddr	This is function btx_ti_write_bdaddr.
	btx_ti_write_hardware_register	This is function btx_ti_write_hardware_register.
	btx_csr_get_script__dsp_script__PB_109_DSP_rev8	brief Return script for patching DSP in CSR8x11 A08 (BlueCore 7) ingroup btx_csr
	btx_csr_get_script__PB_101_CSR8811_CSP28_UART	brief Return script for patching CSR8x11 A06 (BlueCore 7) ingroup btx_csr
	btx_csr_get_script__PB_109_CSR8811_REV16	brief Return script for patching CSR8x11 A08 (BlueCore 7) ingroup btx_csr
	btx_csr_get_script__PB_173_CSR8X11_REV1	brief Return script for patching CSR8x11 A12 (BlueCore 7) ingroup btx_csr
	btx_csr_init	brief Initialize CSR support layer. ingroup btx_csr details This function initializes all internal variables of the CSR support layer. CSR controllers use vendor specific event (0xFF) to carry the BCCMD protocol. They also do not report number of completed packets for BCCMD commands. This function installs a vendor specific event handler that makes sure that callback are called on corresponding vendor specific commands and the number of free command buffers in the controller is kept correct.
	btx_csr_init_hq_script	This is function btx_csr_init_hq_script.
	btx_csr_patch_controller	brief Patch controller's firmware ingroup btx_csr details This function executes a script that patches the controller's firmware. Each entry of the c scripts array must be a complete patch script for a particular controller model and revision. btx_csr_patch_controller() reads the revision number from the controller then tries to find the corresponding script in the c scripts. If there is a matching script it is loaded to the controller and c callback is called with the first parameter TRUE. If no suitable script found c callback is called with the first parameter FALSE. param scripts Array of patch scripts.... more
	btx_csr_register_bccmd_listener	This is function btx_csr_register_bccmd_listener.
	btx_csr_send_dsp_config_data	This is function btx_csr_send_dsp_config_data.
	btx_csr_send_next_hq_script_packet	This is function btx_csr_send_next_hq_script_packet.
	btx_csr_unregister_bccmd_listener	This is function btx_csr_unregister_bccmd_listener.
	btx_ti_a3dp_sink_close_stream	This is function btx_ti_a3dp_sink_close_stream.
	btx_ti_a3dp_sink_codec_config	This is function btx_ti_a3dp_sink_codec_config.
	btx_ti_a3dp_sink_open_stream	This is function btx_ti_a3dp_sink_open_stream.
	btx_ti_a3dp_sink_start_stream	This is function btx_ti_a3dp_sink_start_stream.
	btx_ti_a3dp_sink_stop_stream	This is function btx_ti_a3dp_sink_stop_stream.
	btx_ti_avpr_debug	This is function btx_ti_avpr_debug.
	btx_ti_avpr_enable	This is function btx_ti_avpr_enable.
	btx_ti_get_script__BL6450L_BT_Service_Pack_2_14	CC2564 Scripts (Service pack 2.14)
	btx_ti_get_script__BL6450L_BT_Service_Pack_2_14_BLE_AddOn	This is function btx_ti_get_script__BL6450L_BT_Service_Pack_2_14_BLE_AddOn.
	btx_ti_get_script__CC2564B_BT_Service_Pack_1_0	CC2564B Scripts (Service pack 1.0)
	btx_ti_get_script__CC2564B_BT_Service_Pack_1_0_BLE_AddOn	This is function btx_ti_get_script__CC2564B_BT_Service_Pack_1_0_BLE_AddOn.
	btx_ti_get_script__CC2564B_BT_Service_Pack_1_1	CC2564B Scripts (Service pack 1.1)
	btx_ti_get_script__CC2564B_BT_Service_Pack_1_1_BLE_AddOn	This is function btx_ti_get_script__CC2564B_BT_Service_Pack_1_1_BLE_AddOn.
	btx_ti_get_script__CC2564B_BT_Service_Pack_1_2	CC2564B Scripts (Service pack 1.2)
	btx_ti_get_script__CC2564B_BT_Service_Pack_1_2_AVPR_AddOn	This is function btx_ti_get_script__CC2564B_BT_Service_Pack_1_2_AVPR_AddOn.

	btx_ti_get_script__CC2564B_BT_Service_Pack_1_2_BLE_AddOn	This is function btx_ti_get_script__CC2564B_BT_Service_Pack_1_2_BLE_AddOn.
	btx_ti_le_enable	This is function btx_ti_le_enable.
	btx_ti_set_afh_mode	This is function btx_ti_set_afh_mode.
	btx_ti_write_codec_config	This is function btx_ti_write_codec_config.
	btx_csr_exec_hq_script	This is function btx_csr_exec_hq_script.

Platform Data Types and Constants

	Name	Description
	BYTE_SIZE	This is macro BYTE_SIZE.
	bt_byte	This is type bt_byte.
	bt_int	This is type bt_int.
	bt_long	This is type bt_long.
	bt_uint	This is type bt_uint.
	bt_ulong	This is type bt_ulong.

RFCOMM Data Types and Constants

	Name	Description
	MK_CMD_ADDRESS	define RFCOMM_MS_RTC
	MK_DLCI	This is macro MK_DLCI.
	RFCOMM_CFC_ENABLED	This is macro RFCOMM_CFC_ENABLED.
	RFCOMM_CFC_LOCAL_CREDIT	This is macro RFCOMM_CFC_LOCAL_CREDIT.
	RFCOMM_CFC_MAX_INITIAL_CREDIT	This is macro RFCOMM_CFC_MAX_INITIAL_CREDIT.
	RFCOMM_CMD_STATUS_FC_PENDING	This is macro RFCOMM_CMD_STATUS_FC_PENDING.
	RFCOMM_CMD_STATUS_PENDING	This is macro RFCOMM_CMD_STATUS_PENDING.
	RFCOMM_CMD_STATUS_WAITING_RESPONSE	This is macro RFCOMM_CMD_STATUS_WAITING_RESPONSE.
	RFCOMM_COMMAND	This is macro RFCOMM_COMMAND.
	RFCOMM_CTL_MSG_CLD	This is macro RFCOMM_CTL_MSG_CLD.
	RFCOMM_CTL_MSG_FCOFF	This is macro RFCOMM_CTL_MSG_FCOFF.
	RFCOMM_CTL_MSG_FCON	This is macro RFCOMM_CTL_MSG_FCON.
	RFCOMM_CTL_MSG_MSC	This is macro RFCOMM_CTL_MSG_MSC.
	RFCOMM_CTL_MSG_NSC	This is macro RFCOMM_CTL_MSG_NSC.
	RFCOMM_CTL_MSG_PN	This is macro RFCOMM_CTL_MSG_PN.
	RFCOMM_CTL_MSG_PSC	This is macro RFCOMM_CTL_MSG_PSC.
	RFCOMM_CTL_MSG_RLS	This is macro RFCOMM_CTL_MSG_RLS.
	RFCOMM_CTL_MSG_RPN	This is macro RFCOMM_CTL_MSG_RPN.
	RFCOMM_CTL_MSG_SNC	This is macro RFCOMM_CTL_MSG_SNC.
	RFCOMM_CTL_MSG_TEST	This is macro RFCOMM_CTL_MSG_TEST.
	RFCOMM_DLC_CHANGED_CONN_STATE	This is macro RFCOMM_DLC_CHANGED_CONN_STATE.
	RFCOMM_DLC_CHANGED_REMOTE_MSC	This is macro RFCOMM_DLC_CHANGED_REMOTE_MSC.
	RFCOMM_DLC_CONNECTION_FAILED	This is macro RFCOMM_DLC_CONNECTION_FAILED.
	RFCOMM_DLC_STATE_CLOSED	This is macro RFCOMM_DLC_STATE_CLOSED.
	RFCOMM_DLC_STATE_OPEN	This is macro RFCOMM_DLC_STATE_OPEN.
	RFCOMM_DLCI_CONTROL	This is macro RFCOMM_DLCI_CONTROL.
	RFCOMM_DLCI_FREE	This is macro RFCOMM_DLCI_FREE.
	RFCOMM_ERR_DM	This is macro RFCOMM_ERR_DM.
	RFCOMM_ERR_SUCCESS	This is macro RFCOMM_ERR_SUCCESS.
	RFCOMM_ERR_TIMEOUT	This is macro RFCOMM_ERR_TIMEOUT.
	RFCOMM_FC_TYPE_AGREGATE	This is macro RFCOMM_FC_TYPE_AGREGATE.
	RFCOMM_FC_TYPE_CREDIT	This is macro RFCOMM_FC_TYPE_CREDIT.
	RFCOMM_FLAG_CR	This is macro RFCOMM_FLAG_CR.
	RFCOMM_FLAG_EA	This is macro RFCOMM_FLAG_EA.
	RFCOMM_FLAG_PF	This is macro RFCOMM_FLAG_PF.
	RFCOMM_FRAME_HEADER_LEN	This is macro RFCOMM_FRAME_HEADER_LEN.
	RFCOMM_FRAME_TYPE_DISC	This is macro RFCOMM_FRAME_TYPE_DISC.

	RFCOMM_FRAME_TYPE_DM	This is macro RFCOMM_FRAME_TYPE_DM.
	RFCOMM_FRAME_TYPE_SABM	This is macro RFCOMM_FRAME_TYPE_SABM.
	RFCOMM_FRAME_TYPE_UA	This is macro RFCOMM_FRAME_TYPE_UA.
	RFCOMM_FRAME_TYPE_UI	This is macro RFCOMM_FRAME_TYPE_UI.
	RFCOMM_FRAME_TYPE_UIH	This is macro RFCOMM_FRAME_TYPE_UIH.
	RFCOMM_LINE_STATUS_FRAMING	This is macro RFCOMM_LINE_STATUS_FRAMING.
	RFCOMM_LINE_STATUS_OVERRUN	This is macro RFCOMM_LINE_STATUS_OVERRUN.
	RFCOMM_LINE_STATUS_PARITY	This is macro RFCOMM_LINE_STATUS_PARITY.
	RFCOMM_MAX_INFO_LEN	This is macro RFCOMM_MAX_INFO_LEN.
	RFCOMM_MODEM_STATUS_DV	This is macro RFCOMM_MODEM_STATUS_DV.
	RFCOMM_MODEM_STATUS_FC	This is macro RFCOMM_MODEM_STATUS_FC.
	RFCOMM_MODEM_STATUS_IC	This is macro RFCOMM_MODEM_STATUS_IC.
	RFCOMM_MODEM_STATUS_RTC	This is macro RFCOMM_MODEM_STATUS_RTC.
	RFCOMM_MODEM_STATUS_RTR	This is macro RFCOMM_MODEM_STATUS_RTR.
	RFCOMM_MX_MSG_FCOFF	This is macro RFCOMM_MX_MSG_FCOFF.
	RFCOMM_MX_MSG_FCON	This is macro RFCOMM_MX_MSG_FCON.
	RFCOMM_MX_MSG_MSC	This is macro RFCOMM_MX_MSG_MSC.
	RFCOMM_MX_MSG_NSC	This is macro RFCOMM_MX_MSG_NSC.
	RFCOMM_MX_MSG_PN	This is macro RFCOMM_MX_MSG_PN.
	RFCOMM_MX_MSG_RLS	This is macro RFCOMM_MX_MSG_RLS.
	RFCOMM_MX_MSG_RPN	This is macro RFCOMM_MX_MSG_RPN.
	RFCOMM_MX_MSG_TEST	This is macro RFCOMM_MX_MSG_TEST.
	RFCOMM_RESPONSE	This is macro RFCOMM_RESPONSE.
	RFCOMM_ROLE_INITIATOR	This is macro RFCOMM_ROLE_INITIATOR.
	RFCOMM_ROLE_RESPONDER	This is macro RFCOMM_ROLE_RESPONDER.
	RFCOMM_RPN_BAUD_RATE_1152	This is macro RFCOMM_RPN_BAUD_RATE_1152.
	RFCOMM_RPN_BAUD_RATE_192	This is macro RFCOMM_RPN_BAUD_RATE_192.
	RFCOMM_RPN_BAUD_RATE_2304	This is macro RFCOMM_RPN_BAUD_RATE_2304.
	RFCOMM_RPN_BAUD_RATE_24	This is macro RFCOMM_RPN_BAUD_RATE_24.
	RFCOMM_RPN_BAUD_RATE_384	This is macro RFCOMM_RPN_BAUD_RATE_384.
	RFCOMM_RPN_BAUD_RATE_48	This is macro RFCOMM_RPN_BAUD_RATE_48.
	RFCOMM_RPN_BAUD_RATE_576	This is macro RFCOMM_RPN_BAUD_RATE_576.
	RFCOMM_RPN_BAUD_RATE_72	This is macro RFCOMM_RPN_BAUD_RATE_72.
	RFCOMM_RPN_BAUD_RATE_96	default
	RFCOMM_RPN_DATA_BIT_5	This is macro RFCOMM_RPN_DATA_BIT_5.
	RFCOMM_RPN_DATA_BIT_6	This is macro RFCOMM_RPN_DATA_BIT_6.
	RFCOMM_RPN_DATA_BIT_7	This is macro RFCOMM_RPN_DATA_BIT_7.
	RFCOMM_RPN_DATA_BIT_8	default
	RFCOMM_RPN_FLC_N	This is macro RFCOMM_RPN_FLC_N.
	RFCOMM_RPN_FLC_RTC_INPUT	This is macro RFCOMM_RPN_FLC_RTC_INPUT.
	RFCOMM_RPN_FLC_RTC_OUTPUT	This is macro RFCOMM_RPN_FLC_RTC_OUTPUT.
	RFCOMM_RPN_FLC_RTR_INPUT	This is macro RFCOMM_RPN_FLC_RTR_INPUT.
	RFCOMM_RPN_FLC_RTR_OUTPUT	This is macro RFCOMM_RPN_FLC_RTR_OUTPUT.
	RFCOMM_RPN_FLC_XONOFF_INPUT	This is macro RFCOMM_RPN_FLC_XONOFF_INPUT.
	RFCOMM_RPN_FLC_XONOFF_OUTPUT	This is macro RFCOMM_RPN_FLC_XONOFF_OUTPUT.
	RFCOMM_RPN_PARITY_EVEN	This is macro RFCOMM_RPN_PARITY_EVEN.
	RFCOMM_RPN_PARITY_MARK	This is macro RFCOMM_RPN_PARITY_MARK.
	RFCOMM_RPN_PARITY_N	default
	RFCOMM_RPN_PARITY_ODD	This is macro RFCOMM_RPN_PARITY_ODD.
	RFCOMM_RPN_PARITY_SPACE	This is macro RFCOMM_RPN_PARITY_SPACE.
	RFCOMM_RPN_PARITY_Y	This is macro RFCOMM_RPN_PARITY_Y.
	RFCOMM_RPN_STOP_BIT_1	default
	RFCOMM_RPN_STOP_BIT_1_5	This is macro RFCOMM_RPN_STOP_BIT_1_5.
	RFCOMM_RPN_XOFF_DEFAULT	This is macro RFCOMM_RPN_XOFF_DEFAULT.
	RFCOMM_RPN_XON_DEFAULT	This is macro RFCOMM_RPN_XON_DEFAULT.

	RFCOMM_SERIAL_PORT_CH_1	This is macro RFCOMM_SERIAL_PORT_CH_1.
	RFCOMM_SERIAL_PORT_CH_10	This is macro RFCOMM_SERIAL_PORT_CH_10.
	RFCOMM_SERIAL_PORT_CH_11	This is macro RFCOMM_SERIAL_PORT_CH_11.
	RFCOMM_SERIAL_PORT_CH_12	This is macro RFCOMM_SERIAL_PORT_CH_12.
	RFCOMM_SERIAL_PORT_CH_13	This is macro RFCOMM_SERIAL_PORT_CH_13.
	RFCOMM_SERIAL_PORT_CH_14	This is macro RFCOMM_SERIAL_PORT_CH_14.
	RFCOMM_SERIAL_PORT_CH_15	This is macro RFCOMM_SERIAL_PORT_CH_15.
	RFCOMM_SERIAL_PORT_CH_16	This is macro RFCOMM_SERIAL_PORT_CH_16.
	RFCOMM_SERIAL_PORT_CH_17	This is macro RFCOMM_SERIAL_PORT_CH_17.
	RFCOMM_SERIAL_PORT_CH_18	This is macro RFCOMM_SERIAL_PORT_CH_18.
	RFCOMM_SERIAL_PORT_CH_19	This is macro RFCOMM_SERIAL_PORT_CH_19.
	RFCOMM_SERIAL_PORT_CH_2	This is macro RFCOMM_SERIAL_PORT_CH_2.
	RFCOMM_SERIAL_PORT_CH_20	This is macro RFCOMM_SERIAL_PORT_CH_20.
	RFCOMM_SERIAL_PORT_CH_21	This is macro RFCOMM_SERIAL_PORT_CH_21.
	RFCOMM_SERIAL_PORT_CH_22	This is macro RFCOMM_SERIAL_PORT_CH_22.
	RFCOMM_SERIAL_PORT_CH_23	This is macro RFCOMM_SERIAL_PORT_CH_23.
	RFCOMM_SERIAL_PORT_CH_24	This is macro RFCOMM_SERIAL_PORT_CH_24.
	RFCOMM_SERIAL_PORT_CH_25	This is macro RFCOMM_SERIAL_PORT_CH_25.
	RFCOMM_SERIAL_PORT_CH_26	This is macro RFCOMM_SERIAL_PORT_CH_26.
	RFCOMM_SERIAL_PORT_CH_27	This is macro RFCOMM_SERIAL_PORT_CH_27.
	RFCOMM_SERIAL_PORT_CH_28	This is macro RFCOMM_SERIAL_PORT_CH_28.
	RFCOMM_SERIAL_PORT_CH_29	This is macro RFCOMM_SERIAL_PORT_CH_29.
	RFCOMM_SERIAL_PORT_CH_3	This is macro RFCOMM_SERIAL_PORT_CH_3.
	RFCOMM_SERIAL_PORT_CH_30	This is macro RFCOMM_SERIAL_PORT_CH_30.
	RFCOMM_SERIAL_PORT_CH_4	This is macro RFCOMM_SERIAL_PORT_CH_4.
	RFCOMM_SERIAL_PORT_CH_5	This is macro RFCOMM_SERIAL_PORT_CH_5.
	RFCOMM_SERIAL_PORT_CH_6	This is macro RFCOMM_SERIAL_PORT_CH_6.
	RFCOMM_SERIAL_PORT_CH_7	This is macro RFCOMM_SERIAL_PORT_CH_7.
	RFCOMM_SERIAL_PORT_CH_8	This is macro RFCOMM_SERIAL_PORT_CH_8.
	RFCOMM_SERIAL_PORT_CH_9	This is macro RFCOMM_SERIAL_PORT_CH_9.
	RFCOMM_SESSION_CHANGED_AFC	This is macro RFCOMM_SESSION_CHANGED_AFC.
	RFCOMM_SESSION_CHANGED_CONN_STATE	This is macro RFCOMM_SESSION_CHANGED_CONN_STATE.
	RFCOMM_SESSION_STATE_CONNECTED	This is macro RFCOMM_SESSION_STATE_CONNECTED.
	RFCOMM_SESSION_STATE_DISCONNECTED	This is macro RFCOMM_SESSION_STATE_DISCONNECTED.
	RFCOMM_SESSION_STATE_FREE	This is macro RFCOMM_SESSION_STATE_FREE.
	RFCOMM_TIMEOUT	This is macro RFCOMM_TIMEOUT.
	_bt_rfcomm_command_t	This is type bt_rfcomm_command_t.
	_bt_rfcomm_ctl_msg_t	This is type bt_rfcomm_ctl_msg_t.
	_bt_rfcomm_dlc_t	This is type bt_rfcomm_dlc_t.
	_bt_rfcomm_mgr_t	This is type bt_rfcomm_mgr_t.
	_bt_rfcomm_server_channel_t	This is type bt_rfcomm_server_channel_t.
	_bt_rfcomm_session_listener_t	This is type bt_rfcomm_session_listener_t.
	_bt_rfcomm_session_t	This is type bt_rfcomm_session_t.
	_bt_sdp_data_element_t	This is type bt_sdp_data_element_t.
	bt_rfcomm_cmd_callback_fp	This is type bt_rfcomm_cmd_callback_fp.
	bt_rfcomm_command_p	This is type bt_rfcomm_command_p.
	bt_rfcomm_command_t	This is type bt_rfcomm_command_t.
	bt_rfcomm_ctl_msg_p	This is type bt_rfcomm_ctl_msg_p.
	bt_rfcomm_ctl_msg_t	This is type bt_rfcomm_ctl_msg_t.
	bt_rfcomm_dlc_p	This is type bt_rfcomm_dlc_p.
	bt_rfcomm_dlc_state_callback_fp	This is type bt_rfcomm_dlc_state_callback_fp.
	bt_rfcomm_dlc_t	This is type bt_rfcomm_dlc_t.
	bt_rfcomm_mgr_t	This is type bt_rfcomm_mgr_t.
	bt_rfcomm_read_data_callback_fp	This is type bt_rfcomm_read_data_callback_fp.
	bt_rfcomm_send_data_callback_fp	This is type bt_rfcomm_send_data_callback_fp.

	bt_rfcomm_server_channel_t	This is type bt_rfcomm_server_channel_t.
	bt_rfcomm_session_listener_t	This is type bt_rfcomm_session_listener_t.
	bt_rfcomm_session_p	This is type bt_rfcomm_session_p.
	bt_rfcomm_session_t	This is type bt_rfcomm_session_t.
	bt_rfcomm_state_callback_fp	This is type bt_rfcomm_state_callback_fp.
	RFCOMM_ERR_INTERRUPTED	This is macro RFCOMM_ERR_INTERRUPTED.
	RFCOMM_MIX_INFO_LEN	This is macro RFCOMM_MIX_INFO_LEN.
	RFCOMM_MX_MSG_MAX_DATA_LEN	This is macro RFCOMM_MX_MSG_MAX_DATA_LEN.

RFCOMM Functions

	Name	Description
	bt_rfcomm_allocate_dlc	brief Allocate DLC. ingroup rfcomm details This function allocates a new DLC on the specified RFCOMM session. param session The RFCOMM session. param dlci DLCI of the new DLC. return li A pointer to the new DLC if the function succeeds. li c NULL otherwise.
	bt_rfcomm_allocate_session	brief Allocate RFCOMM session. ingroup rfcomm details This function allocates a new RFCOMM session. param l2cap_mgr The L2CAP manager on which the RFCOMM session is to be created. param callback The callback function that is called when session state changes. param param An arbitrary data pointer that will be passed to the callback function specified by the c callback parameter. return li A pointer to the new RFCOMM session structure if the function succeeds. li c NULL otherwise.
	bt_rfcomm_close_dlc	brief Close DLC. ingroup rfcomm details This function closes a DLC. If DLCI = 0, the parent RFCOMM session is also closed.
	bt_rfcomm_connect	brief Connect to a remote device. ingroup rfcomm details This function establishes an RFCOMM connection with a remote device and opens a data DLC. Changes in the session state are reported through a callback function specified when the session has been allocated via call to bt_rfcomm_allocate_session. Changes in the data DLC are reported through a callback function specified in this call. param remote_addr Address of the remote device. param server_channel A server channel of the remote RFCOMM server. param callback The callback function for reporting changes in DLC state opened by this call. param param An arbitrary data pointer... more
	bt_rfcomm_find_dlc	brief Find DLC
	bt_rfcomm_free_dlc	brief Release DLC. ingroup rfcomm details This function releases the specified DLC. param dlc The DLC to be released.
	bt_rfcomm_free_session	brief Release RFCOMM session. ingroup rfcomm details This function deallocates the specified RFCOMM session. This function does not disconnect the session. It just frees the memory used by the bt_rfcomm_session structure. The session has to be disconnected by calling bt_rfcomm_close_dlc with DLCI = 0 first. param session The RFCOMM session to be deallocated.
	bt_rfcomm_init	brief Initialize the RFCOMM layer. ingroup rfcomm details This function initializes the RFCOMM layer of the stack. It must be called prior to any other RFCOMM function can be called.
	bt_rfcomm_listen	brief Listen for incoming connections. ingroup rfcomm details This function enables incoming connections on the specified RFCOMM session. Changes in the session state are reported through a callback function. param server_channel A server channel on which the RFCOMM session will listen and accept incoming connections. param callback The callback function that is called when session state changes. param param An arbitrary data pointer that will be passed to the callback function specified by the c callback parameter. return li c TRUE if the function succeeds. li c FALSE otherwise. The callback function is not called in this case.

	bt_rfcomm_open_dlc	brief Open DLC. ingroup rfcomm details This function opens the specified DLC. Before calling this function the RFCOMM session must be already open. This function is not to be used with DLCI = 0. Changes in DLC state are reported through a callback function. param dlc The DLC to open. param callback The callback function for reporting changes in DLC state. param param An arbitrary data pointer that will be passed to the callback function specified by the c callback parameter. return li c TRUE if the function succeeds. li c FALSE otherwise. The callback function is not called in... more
	bt_rfcomm_register_listener	This is function bt_rfcomm_register_listener.
	bt_rfcomm_send_credit	This is function bt_rfcomm_send_credit.
	bt_rfcomm_send_data	brief Send data over a DLC. ingroup rfcomm details This function sends data over the specified DLC. Operation completion is reported through callback function. param dlc The DLC. param data A pointer to the data to be sent. param len Data length. param callback The callback function that is called when operation completes. return li c TRUE if the function succeeds. li c FALSE otherwise. The callback function is not called in this case.
	bt_rfcomm_unregister_listener	This is function bt_rfcomm_unregister_listener.
	check_fcs	From rfcomm_fcs.c
	read_command	From channel_cmd_recv.c
	rfcomm_cq_ack_cmd	This is function rfcomm_cq_ack_cmd.
	rfcomm_cq_ack_mx_cmd	This is function rfcomm_cq_ack_mx_cmd.
	rfcomm_send_cmd	This is function rfcomm_send_cmd.
	rfcomm_send_commands_from_queue	This is function rfcomm_send_commands_from_queue.
	rfcomm_send_mx_msc_cmd	This is function rfcomm_send_mx_msc_cmd.
	rfcomm_send_mx_pn_cmd	This is function rfcomm_send_mx_pn_cmd.
	_rfcomm_alloc_cmd_buffer	This is function _rfcomm_alloc_cmd_buffer.
	_rfcomm_allocate_buffers	Defined by OEM through library configuration
	_rfcomm_allocate_mx_cmd	This is function _rfcomm_allocate_mx_cmd.
	_rfcomm_free_cmd_buffer	This is function _rfcomm_free_cmd_buffer.
	_rfcomm_get_tick_count	This is function _rfcomm_get_tick_count.
	_rfcomm_init_cmd_buffers	From rfcomm_cmdbuffer.c
	_rfcomm_init_mgr	This is function _rfcomm_init_mgr.
	_rfcomm_init_sessions	From rfcomm_session.c
	_rfcomm_init_timer	This is function _rfcomm_init_timer.
	_rfcomm_l2cap_read_data_callback	From rfcomm_cmd_recv.c
	_rfcomm_mx_process_fc	This is function _rfcomm_mx_process_fc.
	_rfcomm_mx_process_pn	This is function _rfcomm_mx_process_pn.
	_rfcomm_mx_process_rpn	This is function _rfcomm_mx_process_rpn.
	_rfcomm_process_cmd_frame_disc	From frame_disc.c
	_rfcomm_process_cmd_frame_dm	This is function _rfcomm_process_cmd_frame_dm.
	_rfcomm_process_cmd_frame_sabm	This is function _rfcomm_process_cmd_frame_sabm.
	_rfcomm_process_cmd_frame_ua	This is function _rfcomm_process_cmd_frame_ua.
	_rfcomm_process_cmd_frame_uih	From frame_uih.c
	_rfcomm_process_mx_fc_response	This is function _rfcomm_process_mx_fc_response.
	_rfcomm_process_mx_msc_response	This is function _rfcomm_process_mx_msc_response.
	_rfcomm_process_mx_pn_response	This is function _rfcomm_process_mx_pn_response.
	_rfcomm_process_mx_rls_response	This is function _rfcomm_process_mx_rls_response.
	_rfcomm_process_res_frame_disc	This is function _rfcomm_process_res_frame_disc.
	_rfcomm_process_res_frame_dm	This is function _rfcomm_process_res_frame_dm.
	_rfcomm_process_res_frame_sabm	This is function _rfcomm_process_res_frame_sabm.
	_rfcomm_process_res_frame_ua	This is function _rfcomm_process_res_frame_ua.
	_rfcomm_process_res_frame_uih	This is function _rfcomm_process_res_frame_uih.
	_rfcomm_send_command	From rfcomm_cmd_send.c
	_rfcomm_send_dm_response	From frame_dm.c
	_rfcomm_send_mx_fc_cmd	This is function _rfcomm_send_mx_fc_cmd.
	_rfcomm_send_mx_msc_response	This is function _rfcomm_send_mx_msc_response.

	_rfcomm_send_mx_nsc_response	This is function _rfcomm_send_mx_nsc_response.
	_rfcomm_send_mx_rls_cmd	This is function _rfcomm_send_mx_rls_cmd.
	_rfcomm_send_mx_rls_response	This is function _rfcomm_send_mx_rls_response.
	_rfcomm_send_mx_test_response	This is function _rfcomm_send_mx_test_response.
	_rfcomm_send_sabm_cmd	From frame_sabm.c
	_rfcomm_send_ua_response	From frame_ua.c
	_bt_rfcomm_allocate_channel	This is function _bt_rfcomm_allocate_channel.
	_bt_rfcomm_clear_queue	From command_queue.c
	_bt_rfcomm_find_channel	This is function _bt_rfcomm_find_channel.
	_bt_rfcomm_get_mgr	This is function _bt_rfcomm_get_mgr.
	_bt_rfcomm_init_signal	This is function _bt_rfcomm_init_signal.
	_bt_rfcomm_mgr_allocate_session	This is function _bt_rfcomm_mgr_allocate_session.
	_bt_rfcomm_mgr_l2cap_listen_callback	This is function _bt_rfcomm_mgr_l2cap_listen_callback.
	_bt_rfcomm_mgr_notify_listeners	This is function _bt_rfcomm_mgr_notify_listeners.
	_bt_rfcomm_set_signal	This is function _bt_rfcomm_set_signal.
	_calc_fcs	This is function _calc_fcs.
	rfcomm_find_session	This is function rfcomm_find_session.
	_rfcomm_l2cap_state_changed_callback	From rfcomm.c
	bt_rfcomm_cancel_listen	This is function bt_rfcomm_cancel_listen.
	bt_rfcomm_get_frame_length	This is function bt_rfcomm_get_frame_length.
	rfcomm_cq_find_failed_pn	This is function rfcomm_cq_find_failed_pn.

SBC Decoder Data Types and Constants

	Name	Description
	sbc_struct	
	sbc_t	This is type sbc_t.
	__SBC_H	This is macro __SBC_H.
	INP_BUF_SIZE	Bytes
	MAX_SBC_DEC_STATE_SIZE	Bytes
	OUT_BUF_SIZE	Bytes
	SBC_AM_LOUDNESS	Allocation method
	SBC_AM_SNR	This is macro SBC_AM_SNR.
	SBC_BE	This is macro SBC_BE.
	SBC_BLK_12	This is macro SBC_BLK_12.
	SBC_BLK_16	This is macro SBC_BLK_16.
	SBC_BLK_4	Blocks
	SBC_BLK_8	This is macro SBC_BLK_8.
	SBC_FREQ_16000	Sampling frequency
	SBC_FREQ_32000	This is macro SBC_FREQ_32000.
	SBC_FREQ_44100	This is macro SBC_FREQ_44100.
	SBC_FREQ_48000	This is macro SBC_FREQ_48000.
	SBC_LE	Data endianness
	SBC_MODE_DUAL_CHANNEL	This is macro SBC_MODE_DUAL_CHANNEL.
	SBC_MODE_JOINT_STEREO	This is macro SBC_MODE_JOINT_STEREO.
	SBC_MODE_MONO	Channel mode
	SBC_MODE_STEREO	This is macro SBC_MODE_STEREO.
	SBC_SB_4	Sub-bands
	SBC_SB_8	This is macro SBC_SB_8.
	ssize_t	

SBC Decoder Functions

	Name	Description
	sbc_decode	Decodes the encoded SBC frame.
	sbc_get_codesize	Gets the input block size.
	sbc_get_frame_duration	Gets the time one input/output block takes to play in microseconds.

	sbc_get_frame_length	Gets the output block size.
	sbc_get_state_size	Gets the SBC Decoder's state data structure.
	sbc_init	Initializes the SBC Decoder.

SDP Data Types and Constants

	Name	Description
	BEGIN_DE_SEQUENCE	• brief Begin a data element sequence • ingroup sdp * • details BEGIN_DE_SEQUENCE and END_DE_SEQUENCE are used to define a data element sequence which is an array of sdp_data_element structures. • The array is used a search pattern in bt_sdp_request_service_search() and bt_sdp_request_service_attribute(). • For example, to find a AVRCP Target the following code can be used: code const bt_uuid_t AVRCP_AV_REMOTE_CONTROL_CLSID = { 0x5F9B34FB, 0x80000080, 0x00001000, SDP_CLSID_AV_REMOTE_CONTROL }; const bt_uuid_t AVRCP_AV_REMOTE_CONTROL_TARGET_CLSID = { 0x5F9B34FB, 0x80000080, 0x00001000, SDP_CLSID_AV_REMOTE_CONTROL_TARGET }; BEGIN_DE_SEQUENCE(avrcp_target_service_search, 2) DE_UUID128(AVRCP_AV_REMOTE_CONTROL_CLSID) DE_UUID128(AVRCP_AV_REMOTE_CONTROL_TARGET_CLSID) END_DE_SEQUENCE(avrcp_target_service_search) . . . void findAvrcpTarget(void) { INIT_DE_SEQUENCE(avrcp_target_service_search); bt_sdp_request_service_search(channel, &seq_avrcp_target_service_search, &callback, NULL); } endcode * • param... more
	DE_BOOL	brief Declare a boolean data element ingroup sdp details This macro adds a boolean data element to a data element sequence. This macro is to be used between BEGIN_DE_SEQUENCE and END_DE_SEQUENCE. param value The data element value.
	DE_INT	brief Declare a 1-byte signed integer data element ingroup sdp details This macro adds a 1-byte signed integer data element to a data element sequence. This macro is to be used between BEGIN_DE_SEQUENCE and END_DE_SEQUENCE. param value The data element value.
	DE_STRING	brief Declare a text string data element ingroup sdp details This macro adds a text string data element to a data element sequence. The length of the generated data element will be the actual length of the string. This macro is to be used between BEGIN_DE_SEQUENCE and END_DE_SEQUENCE. param value The data element value.
	DE_STRING2	brief Declare a text string data element ingroup sdp details This macro adds a text string data element to a data element sequence. The length of the generated data element will be the value specified by the "len" parameter even if the actual length of the string is not equal to the "len" value. This macro is to be used between BEGIN_DE_SEQUENCE and END_DE_SEQUENCE. param value The data element value. param len The length of the data element value.
	DE_UINT	brief Declare a 1-byte unsigned integer data element ingroup sdp details This macro adds a 1-byte unsigned integer data element to a data element sequence. This macro is to be used between BEGIN_DE_SEQUENCE and END_DE_SEQUENCE. param value The data element value.

	DE_UINT16	brief Declare a 2-byte unsigned integer data element ingroup sdp details This macro adds a 2-byte unsigned integer data element to a data element sequence. This macro is to be used between BEGIN_DE_SEQUENCE and END_DE_SEQUENCE. param value The data element value.
	DE_URL	brief Declare a URL data element ingroup sdp details This macro adds a URL data element to a data element sequence. This macro is to be used between BEGIN_DE_SEQUENCE and END_DE_SEQUENCE. param value The data element value which is a pointer to a string.
	DE_UUID128	brief Declare a 128-bit UUID data element ingroup sdp details This macro adds a 128-bit UUID data element to a data element sequence. This macro is to be used between BEGIN_DE_SEQUENCE and END_DE_SEQUENCE. param value The data element value. The value must be a name of a variable of type bt_uuid.
	DE_UUID16	brief Declare a 16-bit UUID data element ingroup sdp details This macro adds a 16-bit UUID data element to a data element sequence. This macro is to be used between BEGIN_DE_SEQUENCE and END_DE_SEQUENCE. param value The data element value.
	DE_UUID32	brief Declare a 32-bit UUID data element ingroup sdp details This macro adds a 32-bit UUID data element to a data element sequence. This macro is to be used between BEGIN_DE_SEQUENCE and END_DE_SEQUENCE. param value The data element value.
	END_DE_SEQUENCE	brief End a data element sequence ingroup sdp details BEGIN_DE_SEQUENCE and END_DE_SEQUENCE are used to define a data element sequence which is an array of bt_sdp_data_element structures. param id The data element sequence identifier.
	INIT_DE_SEQUENCE	brief Initialize a data element sequence ingroup sdp details This macro calls a function defined in BEGIN_DE_SEQUENCE which initializes the data element sequence. param id The data element sequence identifier.
	SDP_ATTRID_AdditionalProtocolDescriptorLists	This is macro SDP_ATTRID_AdditionalProtocolDescriptorLists.
	SDP_ATTRID_BluetoothProfileDescriptorList	This is macro SDP_ATTRID_BluetoothProfileDescriptorList.
	SDP_ATTRID_BrowseGroupList	This is macro SDP_ATTRID_BrowseGroupList.
	SDP_ATTRID_ClientExecutableURL	This is macro SDP_ATTRID_ClientExecutableURL.
	SDP_ATTRID_DIPrimaryRecord	This is macro SDP_ATTRID_DIPrimaryRecord.
	SDP_ATTRID_DIProductId	This is macro SDP_ATTRID_DIProductId.
	SDP_ATTRID_DISpecificationId	This is macro SDP_ATTRID_DISpecificationId.
	SDP_ATTRID_DIVendorId	This is macro SDP_ATTRID_DIVendorId.
	SDP_ATTRID_DIVendorIdSource	This is macro SDP_ATTRID_DIVendorIdSource.
	SDP_ATTRID_DIVersion	This is macro SDP_ATTRID_DIVersion.
	SDP_ATTRID_DocumentationURL	This is macro SDP_ATTRID_DocumentationURL.
	SDP_ATTRID_GAPRemoteAudioVolumeControl	This is macro SDP_ATTRID_GAPRemoteAudioVolumeControl.
	SDP_ATTRID_GroupID	This is macro SDP_ATTRID_GroupID.
	SDP_ATTRID_HDPDataExchangeSpecification	This is macro SDP_ATTRID_HDPDataExchangeSpecification.
	SDP_ATTRID_HDPMCAPSupportedProcedures	This is macro SDP_ATTRID_HDPMCAPSupportedProcedures.
	SDP_ATTRID_HDPSuportedFeatures	This is macro SDP_ATTRID_HDPSuportedFeatures.
	SDP_ATTRID_HFPAGNetwork	This is macro SDP_ATTRID_HFPAGNetwork.
	SDP_ATTRID_HFPSupportedFeatures	This is macro SDP_ATTRID_HFPSupportedFeatures.
	SDP_ATTRID_HIDBatteryPower	This is macro SDP_ATTRID_HIDBatteryPower.
	SDP_ATTRID_HIDBootDevice	This is macro SDP_ATTRID_HIDBootDevice.
	SDP_ATTRID_HIDCountryCode	This is macro SDP_ATTRID_HIDCountryCode.
	SDP_ATTRID_HIDDescriptorList	This is macro SDP_ATTRID_HIDDescriptorList.
	SDP_ATTRID_HIDDeviceReleaseNumber	This is macro SDP_ATTRID_HIDDeviceReleaseNumber.
	SDP_ATTRID_HIDDeviceSubclass	This is macro SDP_ATTRID_HIDDeviceSubclass.
	SDP_ATTRID_HIDLANGIDBaseList	This is macro SDP_ATTRID_HIDLANGIDBaseList.

	SDP_ATTRID_HIDNormallyConnectable	This is macro SDP_ATTRID_HIDNormallyConnectable.
	SDP_ATTRID_HIDParserVersion	This is macro SDP_ATTRID_HIDParserVersion.
	SDP_ATTRID_HIDProfileVersion	This is macro SDP_ATTRID_HIDProfileVersion.
	SDP_ATTRID_HIDReconnectInitiate	This is macro SDP_ATTRID_HIDReconnectInitiate.
	SDP_ATTRID_HIDRemoteWake	This is macro SDP_ATTRID_HIDRemoteWake.
	SDP_ATTRID_HIDSDPDisable	This is macro SDP_ATTRID_HIDSDPDisable.
	SDP_ATTRID_HIDSupervisionTimeout	This is macro SDP_ATTRID_HIDSupervisionTimeout.
	SDP_ATTRID_HIDVirtualCable	This is macro SDP_ATTRID_HIDVirtualCable.
	SDP_ATTRID_IconURL	This is macro SDP_ATTRID_IconURL.
	SDP_ATTRID_INVALID	This is macro SDP_ATTRID_INVALID.
	SDP_ATTRID_LanguageBaseAttributeIDList	This is macro SDP_ATTRID_LanguageBaseAttributeIDList.
	SDP_ATTRID_OFFSET_ProviderName	This is macro SDP_ATTRID_OFFSET_ProviderName.
	SDP_ATTRID_OFFSET_ServiceDescription	This is macro SDP_ATTRID_OFFSET_ServiceDescription.
	SDP_ATTRID_OFFSET_ServiceName	This is macro SDP_ATTRID_OFFSET_ServiceName.
	SDP_ATTRID_PrimaryLanguageBaseId	This is macro SDP_ATTRID_PrimaryLanguageBaseId.
	SDP_ATTRID_ProtocolDescriptorList	This is macro SDP_ATTRID_ProtocolDescriptorList.
	SDP_ATTRID_ServiceAvailability	This is macro SDP_ATTRID_ServiceAvailability.
	SDP_ATTRID_ServiceClassIDList	This is macro SDP_ATTRID_ServiceClassIDList.
	SDP_ATTRID_ServiceDatabaseState	This is macro SDP_ATTRID_ServiceDatabaseState.
	SDP_ATTRID_ServiceID	This is macro SDP_ATTRID_ServiceID.
	SDP_ATTRID_ServiceInfoTimeToLive	This is macro SDP_ATTRID_ServiceInfoTimeToLive.
	SDP_ATTRID_ServiceRecordHandle	This is macro SDP_ATTRID_ServiceRecordHandle.
	SDP_ATTRID_ServiceRecordState	This is macro SDP_ATTRID_ServiceRecordState.
	SDP_ATTRID_SupportedFeatures	This is macro SDP_ATTRID_SupportedFeatures.
	SDP_ATTRID_VersionNumberList	This is macro SDP_ATTRID_VersionNumberList.
	SDP_CLIENT_EVT_CONNECTED	This is macro SDP_CLIENT_EVT_CONNECTED.
	SDP_CLIENT_EVT_DISCONNECTED	This is macro SDP_CLIENT_EVT_DISCONNECTED.
	SDP_CLIENT_EVT_NULL	This is macro SDP_CLIENT_EVT_NULL.
	SDP_CLSID_ADVANCED_AUDIO_DISTRIBUTION	This is macro SDP_CLSID_ADVANCED_AUDIO_DISTRIBUTION.
	SDP_CLSID_AUDIO_SINK	This is macro SDP_CLSID_AUDIO_SINK.
	SDP_CLSID_AUDIO_SOURCE	This is macro SDP_CLSID_AUDIO_SOURCE.
	SDP_CLSID_AV_REMOTE_CONTROL	This is macro SDP_CLSID_AV_REMOTE_CONTROL.
	SDP_CLSID_AV_REMOTE_CONTROL_CONTROLLER	This is macro SDP_CLSID_AV_REMOTE_CONTROL_CONTROLLER.
	SDP_CLSID_AV_REMOTE_CONTROL_PROFILE_ID	This is macro SDP_CLSID_AV_REMOTE_CONTROL_PROFILE_ID.
	SDP_CLSID_AV_REMOTE_CONTROL_TARGET	This is macro SDP_CLSID_AV_REMOTE_CONTROL_TARGET.
	SDP_CLSID_AVCTP	This is macro SDP_CLSID_AVCTP.
	SDP_CLSID_AVDTP	This is macro SDP_CLSID_AVDTP.
	SDP_CLSID_BrowseGroupDescriptorServiceClassID	This is macro SDP_CLSID_BrowseGroupDescriptorServiceClassID.
	SDP_CLSID_DialupNetworking	This is macro SDP_CLSID_DialupNetworking.
	SDP_CLSID_GENERIC_AUDIO	This is macro SDP_CLSID_GENERIC_AUDIO.
	SDP_CLSID_HDP	This is macro SDP_CLSID_HDP.
	SDP_CLSID_HDP_SINK	This is macro SDP_CLSID_HDP_SINK.
	SDP_CLSID_HDP_SOURCE	This is macro SDP_CLSID_HDP_SOURCE.
	SDP_CLSID_HFP	This is macro SDP_CLSID_HFP.
	SDP_CLSID_HFP_AG	This is macro SDP_CLSID_HFP_AG.
	SDP_CLSID_HID	This is macro SDP_CLSID_HID.
	SDP_CLSID_HIDProtocol	This is macro SDP_CLSID_HIDProtocol.
	SDP_CLSID_HSP	This is macro SDP_CLSID_HSP.
	SDP_CLSID_HSP_AG	This is macro SDP_CLSID_HSP_AG.
	SDP_CLSID_HSP_HS	This is macro SDP_CLSID_HSP_HS.
	SDP_CLSID_L2CAP	This is macro SDP_CLSID_L2CAP.
	SDP_CLSID_MCAP_CONTROL	This is macro SDP_CLSID_MCAP_CONTROL.
	SDP_CLSID_MCAP_DATA	This is macro SDP_CLSID_MCAP_DATA.
	SDP_CLSID_OBEXFileTransfer	This is macro SDP_CLSID_OBEXFileTransfer.
	SDP_CLSID_OBEXObjectPush	This is macro SDP_CLSID_OBEXObjectPush.

	SDP_CLSID_PBAP_PCE	This is macro SDP_CLSID_PBAP_PCE.
	SDP_CLSID_PBAP_PSE	This is macro SDP_CLSID_PBAP_PSE.
	SDP_CLSID_PNPInformation	This is macro SDP_CLSID_PNPInformation.
	SDP_CLSID_PublicBrowseGroup	This is macro SDP_CLSID_PublicBrowseGroup.
	SDP_CLSID_RFCOMM	This is macro SDP_CLSID_RFCOMM.
	SDP_CLSID_SerialPort	This is macro SDP_CLSID_SerialPort.
	SDP_CLSID_ServiceDiscoveryServerServiceClassID	defgroup sdp SDP This module describe functions and data structures used to start the SDP server and perform SDP queries.
	SDP_DATA_TYPE_ALTERNATIVE	This is macro SDP_DATA_TYPE_ALTERNATIVE.
	SDP_DATA_TYPE_BOOL	This is macro SDP_DATA_TYPE_BOOL.
	SDP_DATA_TYPE_INT	This is macro SDP_DATA_TYPE_INT.
	SDP_DATA_TYPE_INT128	This is macro SDP_DATA_TYPE_INT128.
	SDP_DATA_TYPE_INT16	This is macro SDP_DATA_TYPE_INT16.
	SDP_DATA_TYPE_INT32	This is macro SDP_DATA_TYPE_INT32.
	SDP_DATA_TYPE_INT64	This is macro SDP_DATA_TYPE_INT64.
	SDP_DATA_TYPE_INT8	This is macro SDP_DATA_TYPE_INT8.
	SDP_DATA_TYPE_NIL	This is macro SDP_DATA_TYPE_NIL.
	SDP_DATA_TYPE_SEQUENCE	This is macro SDP_DATA_TYPE_SEQUENCE.
	SDP_DATA_TYPE_STRING	This is macro SDP_DATA_TYPE_STRING.
	SDP_DATA_TYPE_UINT	This is macro SDP_DATA_TYPE_UINT.
	SDP_DATA_TYPE_UINT128	This is macro SDP_DATA_TYPE_UINT128.
	SDP_DATA_TYPE_UINT16	This is macro SDP_DATA_TYPE_UINT16.
	SDP_DATA_TYPE_UINT32	This is macro SDP_DATA_TYPE_UINT32.
	SDP_DATA_TYPE_UINT64	This is macro SDP_DATA_TYPE_UINT64.
	SDP_DATA_TYPE_UINT8	This is macro SDP_DATA_TYPE_UINT8.
	SDP_DATA_TYPE_URL	This is macro SDP_DATA_TYPE_URL.
	SDP_DATA_TYPE_UUID	This is macro SDP_DATA_TYPE_UUID.
	SDP_DATA_TYPE_UUID128	This is macro SDP_DATA_TYPE_UUID128.
	SDP_DATA_TYPE_UUID16	This is macro SDP_DATA_TYPE_UUID16.
	SDP_DATA_TYPE_UUID32	This is macro SDP_DATA_TYPE_UUID32.
	SDP_ERROR_INSUFFICIENT_RESOURCE	This is macro SDP_ERROR_INSUFFICIENT_RESOURCE.
	SDP_ERROR_INVALID_CONTINUATION_STATE	This is macro SDP_ERROR_INVALID_CONTINUATION_STATE.
	SDP_ERROR_INVALID_PDU_SIZE	This is macro SDP_ERROR_INVALID_PDU_SIZE.
	SDP_ERROR_INVALID_REQUEST_SYNTAX	This is macro SDP_ERROR_INVALID_REQUEST_SYNTAX.
	SDP_ERROR_INVALID_SDP_VERSION	This is macro SDP_ERROR_INVALID_SDP_VERSION.
	SDP_ERROR_INVALID_SR_HANDLE	This is macro SDP_ERROR_INVALID_SR_HANDLE.
	SDP_ERROR_RESERVED	This is macro SDP_ERROR_RESERVED.
	SDP_ErrorResponse	This is macro SDP_ErrorResponse.
	SDP_FTP_SERVICE_ID	This is macro SDP_FTP_SERVICE_ID.
	SDP_HID_SERVICE_ID	This is macro SDP_HID_SERVICE_ID.
	SDP_HSP_AG_SERVICE_ID	This is macro SDP_HSP_AG_SERVICE_ID.
	SDP_HSP_HS_SERVICE_ID	This is macro SDP_HSP_HS_SERVICE_ID.
	SDP_MAX_ATTRIBUTE_PATTERN_LEN	This is macro SDP_MAX_ATTRIBUTE_PATTERN_LEN.
	SDP_MAX_DATA_ELEMENT_LEN	This is macro SDP_MAX_DATA_ELEMENT_LEN.
	SDP_MAX_DATA_ELEMENTS	This is macro SDP_MAX_DATA_ELEMENTS.
	SDP_MAX_SEARCH_PATTERN_LEN	This is macro SDP_MAX_SEARCH_PATTERN_LEN.
	SDP_MAX_TRANSACTIONS	typedef struct _sdp_db { bt_int count; bt_sdp_service_record_p *records; } bt_sdp_db, *bt_sdp_db_p;
	SDP_PDU_HEADER_LEN	This is macro SDP_PDU_HEADER_LEN.
	SDP_RFCOMM_SERVICE_ID	This is macro SDP_RFCOMM_SERVICE_ID.
	SDP_ServiceAttributeRequest	This is macro SDP_ServiceAttributeRequest.
	SDP_ServiceAttributeResponse	This is macro SDP_ServiceAttributeResponse.
	SDP_ServiceSearchAttributeRequest	This is macro SDP_ServiceSearchAttributeRequest.
	SDP_ServiceSearchAttributeResponse	This is macro SDP_ServiceSearchAttributeResponse.

	SDP_ServiceSearchRequest	This is macro SDP_ServiceSearchRequest.
	SDP_ServiceSearchResponse	This is macro SDP_ServiceSearchResponse.
	SDP_SR_HANDLE_HDP_SINK	This is macro SDP_SR_HANDLE_HDP_SINK.
	SDP_SR_HANDLE_HDP_SOURCE	This is macro SDP_SR_HANDLE_HDP_SOURCE.
	SDP_SR_HANDLE_HFP_HF	This is macro SDP_SR_HANDLE_HFP_HF.
	SDP_SR_HANDLE_HID	This is macro SDP_SR_HANDLE_HID.
	SDP_SR_HANDLE_HID_KEYBOARD	This is macro SDP_SR_HANDLE_HID_KEYBOARD.
	SDP_SR_HANDLE_HSP_HS	This is macro SDP_SR_HANDLE_HSP_HS.
	SDP_SR_HANDLE_OBEXFileTransfer	This is macro SDP_SR_HANDLE_OBEXFileTransfer.
	SDP_SR_HANDLE_OBEXObjectPush	This is macro SDP_SR_HANDLE_OBEXObjectPush.
	SDP_SR_HANDLE_PNPINFORMATION	This is macro SDP_SR_HANDLE_PNPINFORMATION.
	SDP_SR_HANDLE_RFCOMM	This is macro SDP_SR_HANDLE_RFCOMM.
	SDP_SR_HANDLE_SERVER	This is macro SDP_SR_HANDLE_SERVER.
	SDP_SR_HANDLE_TEST	for simulating service search using continuation state
	bt_sdp_client_evt_connected_t	This is record bt_sdp_client_evt_connected_t.
	bt_sdp_client_evt_disconnected_t	This is record bt_sdp_client_evt_disconnected_t.
	_bt_sdp_found_attr_list_t	This is type bt_sdp_found_attr_list_t.
	_bt_sdp_packet_t	This is type bt_sdp_packet_t.
	_bt_sdp_sequence_t	This is type bt_sdp_sequence_t.
	_bt_sdp_serialization_state_t	This is type bt_sdp_serialization_state_t.
	_bt_sdp_server_attribute_t	This is type bt_sdp_server_attribute_t.
	_bt_sdp_server_data_element_t	Private types
	_bt_sdp_server_record_t	This is type bt_sdp_server_record_t.
	_bt_sdp_service_transaction_t	This is type bt_sdp_service_transaction_t.
	_bt_sdp_transaction_t	This is type bt_sdp_transaction_t.
	bt_sdp_client_callback_fp	This is type bt_sdp_client_callback_fp.
	bt_sdp_data_element_cp	This is type bt_sdp_data_element_cp.
	bt_sdp_data_element_p	This is type bt_sdp_data_element_p.
	bt_sdp_data_element_t	This is type bt_sdp_data_element_t.
	bt_sdp_found_attr_list_t	This is type bt_sdp_found_attr_list_t.
	bt_sdp_packet_t	This is type bt_sdp_packet_t.
	bt_sdp_read_de_callback_ftp	This is type bt_sdp_read_de_callback_ftp.
	bt_sdp_sequence_cp	This is type bt_sdp_sequence_cp.
	bt_sdp_sequence_p	This is type bt_sdp_sequence_p.
	bt_sdp_sequence_t	This is type bt_sdp_sequence_t.
	bt_sdp_serialization_state_p	This is type bt_sdp_serialization_state_p.
	bt_sdp_serialization_state_t	This is type bt_sdp_serialization_state_t.
	bt_sdp_server_attribute_t	This is type bt_sdp_server_attribute_t.
	bt_sdp_server_data_element_t	Private types
	bt_sdp_server_record_t	This is type bt_sdp_server_record_t.
	bt_sdp_service_attribute_callback_fp	This is type bt_sdp_service_attribute_callback_fp.
	bt_sdp_service_search_callback_fp	This is type bt_sdp_service_search_callback_fp.
	bt_sdp_service_transaction_p	This is type bt_sdp_service_transaction_p.
	bt_sdp_service_transaction_t	This is type bt_sdp_service_transaction_t.
	bt_sdp_transaction_t	This is type bt_sdp_transaction_t.
	bt_sr_handle_p	This is type bt_sr_handle_p.
	bt_sr_handle_t	This is type bt_sr_handle_t.
	SDP_CLIENT_EVT_CONNECTION_FAILED	This is macro SDP_CLIENT_EVT_CONNECTION_FAILED.
	SDP_ATTRID_HCRP_1284ID	This is macro SDP_ATTRID_HCRP_1284ID.
	SDP_ATTRID_HCRP_DeviceLocation	This is macro SDP_ATTRID_HCRP_DeviceLocation.
	SDP_ATTRID_HCRP_DeviceName	This is macro SDP_ATTRID_HCRP_DeviceName.
	SDP_ATTRID_HCRP_FriendlyName	This is macro SDP_ATTRID_HCRP_FriendlyName.
	SDP_CLSID_HARD_COPY_CABLE_REPLACEMENT	This is macro SDP_CLSID_HARD_COPY_CABLE_REPLACEMENT.
	SDP_CLSID_HARD_COPY_CONTROL_CHANNEL	This is macro SDP_CLSID_HARD_COPY_CONTROL_CHANNEL.
	SDP_CLSID_HARD_COPY_DATA_CHANNEL	This is macro SDP_CLSID_HARD_COPY_DATA_CHANNEL.

	SDP_CLSID_HARD_COPY_NOTIFICATION	This is macro SDP_CLSID_HARD_COPY_NOTIFICATION.
	SDP_CLSID_HCR_PRINT	This is macro SDP_CLSID_HCR_PRINT.
	SDP_CLSID_HCR_SCAN	This is macro SDP_CLSID_HCR_SCAN.

SDP Functions

	Name	Description
	bt_sdp_de_to_uuid	This is function bt_sdp_de_to_uuid.
	bt_sdp_packet_assembler	This is function bt_sdp_packet_assembler.
	bt_sdp_read_attribute	From sdp_db_utils.c
	bt_sdp_start	brief Start SDP server ingroup sdp details This function starts the SDP server. param l2cap_mgr The L2CAP manager on which the SDP server is to be started. param sdp_db A pointer to the SDP database define with BEGIN_SDP_DB and END_SDP_DB macros. return li c TRUE if the function succeeds. li c FALSE otherwise.
	sdp_compare_uuid_de	This is function sdp_compare_uuid_de.
	sdp_find_attributes	This is function sdp_find_attributes.
	sdp_find_service_records2	From sdp_utils.c
	_sdp_alloc_svc_tran_buffer	This is function _sdp_alloc_svc_tran_buffer.
	_sdp_alloc_tran_buffer	This is function _sdp_alloc_tran_buffer.
	_sdp_find_svc_transaction	This is function _sdp_find_svc_transaction.
	_sdp_find_transaction	This is function _sdp_find_transaction.
	_sdp_free_svc_tran_buffer	This is function _sdp_free_svc_tran_buffer.
	_sdp_free_tran_buffer	This is function _sdp_free_tran_buffer.
	_sdp_get_de_data_len	This is function _sdp_get_de_data_len.
	_sdp_get_de_hdr_len	This is function _sdp_get_de_hdr_len.
	_sdp_init_tran_buffers	From sdp_tran_buffer.c
	_sdp_read_de_header	This is function _sdp_read_de_header.
	_sdp_write_data_element	This is function _sdp_write_data_element.
	bt_sdp_request_service_attribute	brief Search attributes ingroup sdp details This function retrieves attribute values from a service record. param channel The L2CAP channel used to communicate to the remote SDP server. param sr The service record handle specifies the service record from which attribute values are to be retrieved. param pattern The attribute search pattern is a data element sequence where each element in the list is either an attribute ID or a range of attribute IDs. The pattern buffer must be valid for the duration of the search operation, i.e. until c callback is called for the first time. To define a... more
	bt_sdp_request_service_search	brief Search service records ingroup sdp details This function locates service records on a remote SDP server that match the given service search pattern. param channel The L2CAP channel used to communicate to the remote SDP server. param pattern The service search pattern is a data element sequence where each element in the sequence is a UUID. The sequence must contain at least one UUID. The maximum number of UUIDs in the sequence is 12. The pattern buffer must be valid for the duration of the search operation, i.e. until c callback is called. To define a data element sequence... more
	_bt_sdp_client_init	This is function _bt_sdp_client_init.

SPP Data Types and Constants

	Name	Description
	SPP_PORT_OPTION_ENCRYPTED	This is macro SPP_PORT_OPTION_ENCRYPTED.
	SPP_PORT_OPTION_MASTER	This is macro SPP_PORT_OPTION_MASTER.
	SPP_PORT_OPTION_SECURE	This is macro SPP_PORT_OPTION_SECURE.
	SPP_PORT_TYPE_INCOMING	This is macro SPP_PORT_TYPE_INCOMING.
	SPP_PORT_TYPE_OUTGOING	This is macro SPP_PORT_TYPE_OUTGOING.
	SPP_RS232_CTS	This is macro SPP_RS232_CTS.
	SPP_RS232_DCD	This is macro SPP_RS232_DCD.

	SPP_RS232_DSR	• @defgroup spp Serial Port Profile (SPP) The DotStack SPP API is a simple API for communicating over a Bluetooth link using the Bluetooth Serial Port Profile. Here are the steps for using this API: • Call the bt_spp_init() function. • Allocate a serial port structure with bt_spp_allocate(). One of the parameters to this function is a pointer to a callback function. That callback function will be called by the stack whenever the state of the serial port changes. • To connect to a remote device call bt_spp_connect(). The stack will notify when the connection is established by calling the state... more
	SPP_RS232_DTR	This is macro SPP_RS232_DTR.
	SPP_RS232_RI	This is macro SPP_RS232_RI.
	SPP_RS232_RTS	This is macro SPP_RS232_RTS.
	bt_spp_find_server_callback_fp	This is type bt_spp_find_server_callback_fp.
	bt_spp_port_event_e	• Serial port event. Values of this enumeration represent serial port events. Events are reported to the application through a callback function. The callback function is specified when the port is allocated using bt_spp_allocate().
	bt_spp_port_state_e	• Serial port state. Values of this enumeration represent states of the serial port.
	bt_spp_port_t	This is type bt_spp_port_t.
	bt_spp_receive_callback_fp	• Serial port receive callback. This callback function is called when a receive operation initiated by bt_spp_receive() completes. @param port Serial port on which the receive operation completed. @param bytes_received Number of received bytes. @param param Callback parameter that was specified when bt_spp_allocate() was called.
	bt_spp_send_callback_fp	• Serial port send callback. This callback function is called when a send operation initiated by bt_spp_send() completes. @param port Serial port on which the send operation completed. @param bytes_sent Number of bytes sent. This parameter is just a convenience as it always specifies the same number of bytes that was passed to bt_spp_send(); @param status Completion status. It is one of the values defined in the ::bt_spp_send_status_e enumeration. @param param Callback parameter that was specified when bt_spp_allocate() was called.
	bt_spp_send_status_e	Send operation status.
	bt_spp_state_callback_fp	• Serial port state callback. This callback function is called whenever the state of a serial port is changed. @param port Serial port which state has changed. @param evt Event describing the nature of state change. It is one of the values defined in the ::bt_spp_port_event_e enumeration. @param param Callback parameter that was specified when bt_spp_allocate() was called.

SPP Functions

	Name	Description
	bt_spp_allocate	• Allocate a serial port. The returned serial port is initially in the ::SPP_PORT_STATE_DISCONNECTED state. To establish a connection with a remote device, call bt_spp_connect(). To listen for incoming connections from other devices, call bt_spp_listen(). The p callback parameter must specify a callback function that will be used to notify about serial port events and state changes. When the port is not needed any more it must be deallocated by bt_spp_deallocate(). The maximum number of serial ports that can be allocated simultaneously is specified by the ::SPP_MAX_PORTS configuration parameter. @param l2cap_mgr L2CAP manager. @param callback Pointer to... more

	bt_spp_cancel_receive	• Cancel receive data. If a receive operation is currently in progress this function will cancel it. After calling this function the receive callback specified in bt_spp_receive() will not be called. If there is no receive operation in progress calling this function has no effect. @param port Serial port.
	bt_spp_cancel_send	• Cancel send data. If a send operation is currently in progress this function will try to cancel it. When the operation is canceled the send callback function will be called with the ::SPP_SEND_STATUS_CANCELED status. If this function is called but the active send operation completes successfuly before the stack can actually cancel it the call back function will still be called with the ::SPP_SEND_STATUS_CANCELED status. If there is no send operation in progess calling this function has no effect. @param port Serial port.
	bt_spp_clr_port_options	This is function bt_spp_clr_port_options.
	bt_spp_connect	• Connect to a remote device. This function initiates a connection to a remote device. When the connection is successfully established the port's callback is called with the ::SPP_PORT_EVENT_CONNECTED event. If connection fails the callback is called with the ::SPP_PORT_EVENT_CONNECTION_FAILED event. The port must be in ::SPP_PORT_STATE_DISCONNECTED state. Otherwise, the function will fail. @param port Serial port. @param remote_addr Bluetooth address of the remote device. @param channel RFCOMM server channel on which the connection is to be established. @return c TRUE if successful, c FALSE otherwise.
	bt_spp_deallocate	• Deallocate serial port. This function deallocates the specified port structure and other resources associated with it. The port must be in ::SPP_PORT_STATE_DISCONNECTED state. Otherwise, the function will fail. If the function completes successfully the application must not try to access any fields in the structure and must not use it with any other SPP functions. Also, it becomes available for subsequent allocation by bt_spp_port_allocate(). @param port Serial port structure to deallocate. @return c TRUE if successful, c FALSE otherwise.
	bt_spp_disconnect	• Disconnect from the remote device. This function initiates the disconnection process. When it is complete the the port's callback is called with the SPP_PORT_EVENT_DISCONNECTED event. If the port is already in the disconnected state the function does nothing and the callback is not called. @param port Serial port.
	bt_spp_find_server	This is function bt_spp_find_server.
	bt_spp_get_frame_length	• Get frame length. This function returns the RFCOMM frame length used by the RFCOMM protocol. The frame length depends on configuration of DotStack and configuration of the Bluetooth stack running on the remote device. In order to achieve maximum throughput over the serial port connection the application should send and receive data in chunks that are multiple of this frame length. @return RFCOMM frame length in bytes.
	bt_spp_get_hci_connection	• Get SPP port's ACL connection. This function returns a pinter to the structure that describes the ACL connection this port is on. @return Pointer to ACL connection description if the port is connected, NULL otherwise.
	bt_spp_get_local_modem_status	• Get local device's TS 07.10 control signals. This function returns current state of the local device's TS 07.10 controls signals. The signals are defined as a mask of the following constants: SPP_RS232_DSR SPP_RS232_RTS SPP_RS232_RI SPP_RS232_DCD @param port Serial port. @return local device's TS 07.10 control signals.

	bt_spp_get_remote_address	brief Get the address of the remote device this device is connected to. ingroup spp param port Serial port. return li c A pointer to bt_bdaddr structure that contains the address of the remote device.
	bt_spp_get_remote_modem_status	• Get remote device's TS 07.10 control signals. This function returns current state of the remote device's V.24 controls signals. The signals are defined as a mask of the following constants: SPP_RS232_DTR SPP_RS232_CTS SPP_RS232_RI SPP_RS232_DCD @param port Serial port. @return remote device's TS 07.10 control signals.
	bt_spp_init	• Initialize the SPP module. This function initializes all internal variables of the SPP module. It must be called prior to using any other functions in this module.
	bt_spp_listen	• Listen for incoming connections. This function registers the port to accept incoming connections from remote devices on a particular RFCOMM server channel. The specified server channel should be listed in the SDP database. Otherwise, remote devices will not be able to find out which server channel to use. When a remote device successfully establishes a connection on the specified port the port's callback is called with the ::SPP_PORT_EVENT_CONNECTED event. The port must be in ::SPP_PORT_STATE_DISCONNECTED state. Otherwise, the function will fail. @param port Serial port. @param channel The RFCOMM server channel on which to listen for connections. @return c TRUE... more
	bt_spp_receive	• Receive data. This function receives data from the serial port connection. The caller must provide a buffer and a callback function. Whenever the port receives data they are copied to the provided buffer and the callback function is called. The callback function is passed the length of received data and the same callback parameter that was specified when the port was allocated with bt_spp_allocate(). This function does not wait until the buffer is filled out completely. Any amount of received data will complete the operation. The port must be in ::SPP_PORT_STATE_CONNECTED state. Otherwise, the function will fail. Also, the... more
	bt_spp_send	• Send data. This function starts sending data over the serial port connection. Along with the data the caller must provide a callback function that is called when all data has been sent. Also, during execution of this operation the port's state callback function is called periodically with the ::SPP_PORT_EVENT_SEND_PROGRESS event. The port must be in ::SPP_PORT_STATE_CONNECTED state. Otherwise, the function will fail. Also, the function will fail if a previously started send operation is still in progress. The callback function is passed the same callback parameter that was specified when the port was allocated with bt_spp_allocate(). @param port Serial... more
	bt_spp_set_dtr	• Set local device's RS-232 DTR signal. Changes the state of local device's RS-232 DTR signal and notifies the remote device of the change. This signal corresponds to the TS 07.10's RFCOMM_MODEM_STATUS_RTC signal. If there are resources to send a command to the remote device this command will return TRUE. If the remote party has been successfully notified SPP_PORT_EVENT_LOCAL_MODEM_STATUS_CHANGED event will be reported. If the command could not be send to the remote party for any reason other than lack of resources SPP_PORT_EVENT_LOCAL_MODEM_STATUS_CHANGE_FAILED will be reported. If the status cannot be changed because there is no resources to send a... more

	bt_spp_set_local_modem_status	• Set local device's TS 07.10 control signals. Changes the state of local device's TS 07.10 control signals and notifies the remote device of the change. If there are resources to send a command to the remote device this command will return TRUE. If the remote party has been successfully notified SPP_PORT_EVENT_LOCAL_MODEM_STATUS_CHANGED event will be reported. If the command could not be send to the remote party for any reason other than lack of resources SPP_PORT_EVENT_LOCAL_MODEM_STATUS_CHANGE_FAILED will be reported. If the status cannot be changed because there is no resources to send a command to the remote device this function... more
	bt_spp_set_port_options	This is function bt_spp_set_port_options.
	bt_spp_set_rts	• Set local device's RS-232 RTS signal. Changes the state of local device's RS-232 DTR signal and notifies the remote device of the change. This signal corresponds to the TS 07.10's RFCOMM_MODEM_STATUS_RTR signal. If there are resources to send a command to the remote device this command will return TRUE. If the remote party has been successfully notified SPP_PORT_EVENT_LOCAL_MODEM_STATUS_CHANGED event will be reported. If the command could not be send to the remote party for any reason other than lack of resources SPP_PORT_EVENT_LOCAL_MODEM_STATUS_CHANGE_FAILED will be reported. If the status cannot be changed because there is no resources to send a... more
	_bt_spp_find_port	This is function _bt_spp_find_port.
	_bt_spp_handle_rx	This is function _bt_spp_handle_rx.
	_bt_spp_handle_tx	This is function _bt_spp_handle_tx.
	_bt_spp_rfcomm_read_data_callback	This is function _bt_spp_rfcomm_read_data_callback.
	_bt_spp_rfcomm_send_data_callback	This is function _bt_spp_rfcomm_send_data_callback.

SSP Data Types and Constants

	Name	Description
	OOB_DATA_HASH_LENGTH	Remote OOB Data Request Event
	OOB_DATA_RANDOMIZER_LENGTH	This is macro OOB_DATA_RANDOMIZER_LENGTH.
	SSP_MAX_MANAGERS	This is macro SSP_MAX_MANAGERS.
	_bt_ssp_io_capability	This is type bt_ssp_io_capability.
	_bt_ssp_keypress_notification	Keypress Notification Event
	_bt_ssp_oob_data	This is type bt_ssp_oob_data.
	_bt_spp_port_event_e	• Serial port event. Values of this enumeration represent serial port events. Events are reported to the application through a callback function. The callback function is specified when the port is allocated using bt_spp_allocate().
	_bt_ssp_simple_pairing_complete	Simple Pairing Complete Event
	_bt_spp_port_state_e	• Serial port state. Values of this enumeration represent states of the serial port.
	_bt_ssp_user_confirmation_request	User Confirmation Request Event
	_bt_spp_port_t	• Serial port structure. This structure represents a Bluetooth serial port. Application code may only use those fields that are documented. The rest of the fields are private to the SPP implementation.
	_bt_ssp_user_passkey_notification	User Passkey Notification Event
	_bt_spp_send_status_e	Send operation status.
	_bt_ssp_user_passkey_request	User Passkey Request Event
	_SSP_AUTHENTICATION_REQUIREMENTS	This is type SSP_AUTHENTICATION_REQUIREMENTS.
	_SSP_IO_CAPABILITY	This is type SSP_IO_CAPABILITY.
	_SSP_KEYPRESS_NOTIFICATION_TYPE	This is type SSP_KEYPRESS_NOTIFICATION_TYPE.
	_SSP_MODE	This is type SSP_MODE.
	_SSP_OOB_DATA_PRESENT	This is type SSP_OOB_DATA_PRESENT.

	bt_spp_read_local_oob_data_callback_fp	This is type bt_spp_read_local_oob_data_callback_fp.
	bt_ssp_io_capability	This is type bt_ssp_io_capability.
	bt_ssp_keypress_notification	Keypress Notification Event
	bt_ssp_oob_data	This is type bt_ssp_oob_data.
	bt_ssp_simple_pairing_complete	Simple Pairing Complete Event
	bt_ssp_user_confirmation_request	User Confirmation Request Event
	bt_ssp_user_passkey_notification	User Passkey Notification Event
	bt_ssp_user_passkey_request	User Passkey Request Event
	SSP_AUTHENTICATION_REQUIREMENTS	This is type SSP_AUTHENTICATION_REQUIREMENTS.
	SSP_IO_CAPABILITY	This is type SSP_IO_CAPABILITY.
	SSP_KEYPRESS_NOTIFICATION_TYPE	This is type SSP_KEYPRESS_NOTIFICATION_TYPE.
	SSP_MODE	This is type SSP_MODE.
	SSP_OOB_DATA_PRESENT	This is type SSP_OOB_DATA_PRESENT.
	_SSP_EVENT	This is type SSP_EVENT.
	SSP_EVENT	This is type SSP_EVENT.

SSP Functions

	Name	Description
	bt_ssp_evt_handler	This is function bt_ssp_evt_handler.
	bt_ssp_init	This is function bt_ssp_init.
	bt_ssp_read_local_oob_data	This is function bt_ssp_read_local_oob_data.
	bt_ssp_send_keypress_notification	This is function bt_ssp_send_keypress_notification.
	bt_ssp_send_user_confirmation	This is function bt_ssp_send_user_confirmation.
	bt_ssp_send_user_passkey	This is function bt_ssp_send_user_passkey.
	bt_ssp_set_io_capabilities	This is function bt_ssp_set_io_capabilities.
	bt_ssp_set_mode	This is function bt_ssp_set_mode.
	bt_ssp_set_oob_data	This is function bt_ssp_set_oob_data.
	ssp_evt_io_capability_request	This is function ssp_evt_io_capability_request.
	ssp_evt_io_capability_response	This is function ssp_evt_io_capability_response.
	ssp_evt_keypress_notification	This is function ssp_evt_keypress_notification.
	ssp_evt_oob_data_request	This is function ssp_evt_oob_data_request.
	ssp_evt_ssp_complete	This is function ssp_evt_ssp_complete.
	ssp_evt_user_confirmation_request	This is function ssp_evt_user_confirmation_request.
	ssp_evt_user_passkey_notification	This is function ssp_evt_user_passkey_notification.
	ssp_evt_user_passkey_request	This is function ssp_evt_user_passkey_request.
	_bt_spp_client_init	This is function _bt_spp_client_init.
	bt_spp_cancel_listen	• Stop listening for incoming connections. This function stops the port to accept incoming connections on a given server channel. @param port Serial port. @param channel The RFCOMM server channel to accept connections on. @return c TRUE if successful, c FALSE otherwise.
	bt_spp_find_server_ex	This is function bt_spp_find_server_ex.

Utility Data Types and Constants

	Name	Description
	AT_EVT_CMD_CODE	This is macro AT_EVT_CMD_CODE.
	AT_EVT_CMD_COMPLETED	This is macro AT_EVT_CMD_COMPLETED.
	AT_EVT_CMD_PARAM	This is macro AT_EVT_CMD_PARAM.
	AT_EVT_CMD_READ_CODE	This is macro AT_EVT_CMD_READ_CODE.
	AT_EVT_ERROR	This is macro AT_EVT_ERROR.
	AT_EVT_OK	This is macro AT_EVT_OK.
	AT_EVT_RING	This is macro AT_EVT_RING.
	ATCMD_BUFFER_LEN	This is macro ATCMD_BUFFER_LEN.
	bt_max	This is macro bt_max.
	bt_min	This is macro bt_min.

	BUFFER_HDR_LEN	This is macro BUFFER_HDR_LEN.
	BUFFER_STATE_FREE	for offsetof
	BUFFER_STATE_USED	This is macro BUFFER_STATE_USED.
	max	This is macro max.
	min	This is macro min.
	VCARD_CALL_TYPE_DIALED	This is macro VCARD_CALL_TYPE_DIALED.
	VCARD_CALL_TYPE_MISSED	This is macro VCARD_CALL_TYPE_MISSED.
	VCARD_CALL_TYPE_RECEIVED	This is macro VCARD_CALL_TYPE_RECEIVED.
	VCARD_EVT_PROPERTY_PARAM	This is macro VCARD_EVT_PROPERTY_PARAM.
	VCARD_EVT_PROPERTY_STARTED	This is macro VCARD_EVT_PROPERTY_STARTED.
	VCARD_EVT_PROPERTY_VALUE	This is macro VCARD_EVT_PROPERTY_VALUE.
	VCARD_EVT_VCARD_ENDED	This is macro VCARD_EVT_VCARD_ENDED.
	VCARD_EVT_VCARD_STARTED	This is macro VCARD_EVT_VCARD_STARTED.
	VCARD_PARAM_CALL_TYPE	This is macro VCARD_PARAM_CALL_TYPE.
	VCARD_PARAM_CELL	This is macro VCARD_PARAM_CELL.
	VCARD_PARAM_DIALED	This is macro VCARD_PARAM_DIALED.
	VCARD_PARAM_ENCODING	This is macro VCARD_PARAM_ENCODING.
	VCARD_PARAM_HOME	This is macro VCARD_PARAM_HOME.
	VCARD_PARAM_LANGUAGE	This is macro VCARD_PARAM_LANGUAGE.
	VCARD_PARAM_MISSED	This is macro VCARD_PARAM_MISSED.
	VCARD_PARAM_RECEIVED	This is macro VCARD_PARAM_RECEIVED.
	VCARD_PARAM_TYPE	This is macro VCARD_PARAM_TYPE.
	VCARD_PARAM_VALUE	This is macro VCARD_PARAM_VALUE.
	VCARD_PARAM_WORK	This is macro VCARD_PARAM_WORK.
	VCARD_PARSER_STATE_READ_PARAM_NAME	This is macro VCARD_PARSER_STATE_READ_PARAM_NAME.
	VCARD_PARSER_STATE_READ_PARAM_VALUE	This is macro VCARD_PARSER_STATE_READ_PARAM_VALUE.
	VCARD_PARSER_STATE_READ_TYPE_NAME	This is macro VCARD_PARSER_STATE_READ_TYPE_NAME.
	VCARD_PARSER_STATE_READ_TYPE_VALUE	This is macro VCARD_PARSER_STATE_READ_TYPE_VALUE.
	VCARD_PARSER_STATE_SKIP_PARAM	This is macro VCARD_PARSER_STATE_SKIP_PARAM.
	VCARD_PARSER_STATE_SKIP_TYPE	This is macro VCARD_PARSER_STATE_SKIP_TYPE.
	VCARD_PROPERTY_ADR	Delivery Address
	VCARD_PROPERTY_AGENT	vCard of Person Representing
	VCARD_PROPERTY_BDAY	Birthday
	VCARD_PROPERTY_BEGIN	This is macro VCARD_PROPERTY_BEGIN.
	VCARD_PROPERTY_CATEGORIES	Categories
	VCARD_PROPERTY_CLASS	Class information
	VCARD_PROPERTY_EMAIL	Electronic Mail Address
	VCARD_PROPERTY_END	This is macro VCARD_PROPERTY_END.
	VCARD_PROPERTY_FN	Formatted Name
	VCARD_PROPERTY_GEO	Geographic Position
	VCARD_PROPERTY_KEY	Public Encryption Key
	VCARD_PROPERTY_LABEL	Delivery
	VCARD_PROPERTY_LOGO	Organization Logo
	VCARD_PROPERTY_MAILER	Electronic Mail
	VCARD_PROPERTY_N	Structured Presentation of Name
	VCARD_PROPERTY_NICKNAME	Nickname
	VCARD_PROPERTY_NOTE	Comments
	VCARD_PROPERTY_ORG	Name of Organization
	VCARD_PROPERTY_PHOTO	Associated Image or Photo
	VCARD_PROPERTY_PROID	Product ID
	VCARD_PROPERTY_REV	Revision
	VCARD_PROPERTY_ROLE	Role within the Organization
	VCARD_PROPERTY_SORT_STRING	String used for sorting operations
	VCARD_PROPERTY_SOUND	Pronunciation of Name
	VCARD_PROPERTY_TEL	Telephone Number

	VCARD_PROPERTY_TITLE	Job
	VCARD_PROPERTY_TZ	Time Zone
	VCARD_PROPERTY_UID	Unique ID
	VCARD_PROPERTY_URL	Uniform Resource Locator
	VCARD_PROPERTY_VERSION	vCard Version
	VCARD_PROPERTY_X_IRMC_CALL_DATETIME	Time stamp
	XML_EVT_ATTR_NAME	This is macro XML_EVT_ATTR_NAME.
	XML_EVT_ATTR_VALUE	This is macro XML_EVT_ATTR_VALUE.
	XML_EVT_TAG_ENDED	This is macro XML_EVT_TAG_ENDED.
	XML_EVT_TAG_STARTED	This is macro XML_EVT_TAG_STARTED.
	XML_PARSER_STATE_READ_ATTR_NAME	This is macro XML_PARSER_STATE_READ_ATTR_NAME.
	XML_PARSER_STATE_READ_ATTR_VALUE	This is macro XML_PARSER_STATE_READ_ATTR_VALUE.
	XML_PARSER_STATE_READ_TAG_NAME	This is macro XML_PARSER_STATE_READ_TAG_NAME.
	XML_PARSER_STATE_READ_TAG_VALUE	This is macro XML_PARSER_STATE_READ_TAG_VALUE.
	XML_PARSER_STATE_SKIP_ATTR	This is macro XML_PARSER_STATE_SKIP_ATTR.
	XML_PARSER_STATE_SKIP_COMMENTS	This is macro XML_PARSER_STATE_SKIP_COMMENTS.
	XML_PARSER_STATE_SKIP_PROC_INST	This is macro XML_PARSER_STATE_SKIP_PROC_INST.
	XML_PARSER_STATE_SKIP_TAG	This is macro XML_PARSER_STATE_SKIP_TAG.
	XML_PARSER_STATE_WAIT_ATTR_NAME	This is macro XML_PARSER_STATE_WAIT_ATTR_NAME.
	XML_PARSER_STATE_WAIT_ATTR_VALUE	This is macro XML_PARSER_STATE_WAIT_ATTR_VALUE.
	XML_PARSER_STATE_WAIT_TAG_CLOSE	This is macro XML_PARSER_STATE_WAIT_TAG_CLOSE.
	XML_PARSER_STATE_WAIT_TAG_NAME	This is macro XML_PARSER_STATE_WAIT_TAG_NAME.
	_readbn	This is macro _readbn.
	_readui	This is macro _readui.
	_readuin	This is macro _readuin.
	_readul	This is macro _readul.
	_readuln	This is macro _readuln.
	_writebn	This is macro _writebn.
	_bt_at_parser_t	This is type bt_at_parser_t.
	_bt_buffer_header_t	This is type bt_buffer_header_t.
	_bt_vcard_evt_prop_param_t	This is type bt_vcard_evt_prop_param_t.
	_bt_vcard_evt_prop_t	This is type bt_vcard_evt_prop_t.
	_bt_vcard_parser_t	This is type bt_vcard_parser_t.
	_bt_xml_evt_attribute_t	This is type bt_xml_evt_attribute_t.
	_bt_xml_evt_attribute_value_t	This is type bt_xml_evt_attribute_value_t.
	_bt_xml_evt_tag_started_t	This is type bt_xml_evt_tag_started_t.
	_bt_xml_parser_t	This is type bt_xml_parser_t.
	_bt_buffer_mgr_t	This is type bt_buffer_mgr_t.
	_bt_packet_t	This is type bt_packet_t.
	_bt_queue_element_t	This is type bt_queue_element_t.
	_cmd_disconnection_res	This is type pcmd_disconnection_res.
	bt_at_parser_callback_pf	This is type bt_at_parser_callback_pf.
	bt_at_parser_t	This is type bt_at_parser_t.
	bt_bt_buffer_header_p	This is type bt_bt_buffer_header_p.
	bt_buffer_header_t	This is type bt_buffer_header_t.
	bt_buffer_mgr_p	This is type bt_buffer_mgr_p.
	bt_buffer_mgr_t	This is type bt_buffer_mgr_t.
	bt_packet_assembler_fp	This is type bt_packet_assembler_fp.
	bt_packet_t	This is type bt_packet_t.
	bt_queue_element_t	This is type bt_queue_element_t.
	bt_vcard_evt_prop_param_t	This is type bt_vcard_evt_prop_param_t.
	bt_vcard_evt_prop_t	This is type bt_vcard_evt_prop_t.
	bt_vcard_parser_callback_fp	This is type bt_vcard_parser_callback_fp.
	bt_vcard_parser_t	This is type bt_vcard_parser_t.
	bt_xml_evt_attribute_t	This is type bt_xml_evt_attribute_t.

	bt_xml_evt_attribute_value_t	This is type bt_xml_evt_attribute_value_t.
	bt_xml_evt_tag_started_t	This is type bt_xml_evt_tag_started_t.
	bt_xml_parser_callback_pf	This is type bt_xml_parser_callback_pf.
	bt_xml_parser_t	This is type bt_xml_parser_t.

Utility Functions

	Name	Description
	bt_alloc_buffer	This is function bt_alloc_buffer.
	bt_at_parse_fragment	This is function bt_at_parse_fragment.
	bt_at_parser_reset	This is function bt_at_parser_reset.
	bt_free_buffer	This is function bt_free_buffer.
	bt_get_buffer	This is function bt_get_buffer.
	bt_get_buffer_header	This is function bt_get_buffer_header.
	bt_get_buffer_index	This is function bt_get_buffer_index.
	bt_init_buffer_mgr	This is function bt_init_buffer_mgr.
	bt_q_add	This is function bt_q_add.
	bt_q_get	This is function bt_q_get.
	bt_q_get_head	This is function bt_q_get_head.
	bt_q_get_length	This is function bt_q_get_length.
	bt_q_get_next	This is function bt_q_get_next.
	bt_q_push	This is function bt_q_push.
	bt_q_remove	This is function bt_q_remove.
	bt_q_remove_by_idx	This is function bt_q_remove_by_idx.
	bt_vcard_parse_fragment	This is function bt_vcard_parse_fragment.
	bt_vcard_parser_reset	This is function bt_vcard_parser_reset.
	bt_xml_parse_fragment	This is function bt_xml_parse_fragment.
	bt_xml_parser_reset	This is function bt_xml_parser_reset.
	_compact_buffer	This is function _compact_buffer.
	_expand_buffer	This is function _expand_buffer.
	_read_bdaddr	This is function _read_bdaddr.
	_readb	This is function _readb.
	_readi	This is function _readi.
	_readin	This is function _readin.
	_readl	This is function _readl.
	_readln	This is function _readln.
	_readui	This is function _readui.
	_readuin	This is function _readuin.
	_readul	This is function _readul.
	_readuln	This is function _readuln.
	_str2ulong_dec	This is function _str2ulong_dec.
	_to_lower_case	This is function _to_lower_case.
	_ulong2str	This is function _ulong2str.
	_ulong2str_dec	This is function _ulong2str_dec.
	_write_bdaddr	This is function _write_bdaddr.
	_writeb	This is function _writeb.
	_writei	This is function _writei.
	_writein	This is function _writein.
	_writel	This is function _writel.
	_writeln	This is function _writeln.
	_writes	This is function _writes.
	_writesx	This is function _writesx.
	_zero_memory	This is function _zero_memory.
	_bt_memcpy	This is function _bt_memcpy.
	_is_empty_str	This is function _is_empty_str.
	_str2ulong	This is function _str2ulong.

Description

This section describes the Application Programming Interface (API) functions of the PIC32 Bluetooth Stack Library.

Bluetooth Support Functions

bt_bdaddr_is_null Function

File

bt_bdaddr.h

C

```
bt_bool bt_bdaddr_is_null(const bt_bdaddr_t* bda);
```

Description

This is function bt_bdaddr_is_null.

bt_bdaddrs_are_equal Function

File

bt_bdaddr.h

C

```
bt_bool bt_bdaddrs_are_equal(const bt_bdaddr_t* bda1, const bt_bdaddr_t* bda2);
```

Description

This is function bt_bdaddrs_are_equal.

bt_log_bdaddr Function

File

bt_log.h

C

```
void bt_log_bdaddr(const char* msg, const bt_bdaddr_t* addr);
```

Description

This is function bt_log_bdaddr.

bt_log_int Function

File

bt_log.h

C

```
void bt_log_int(const char* msg, int i);
```

Description

This is function bt_log_int.

bt_log_linkkey Function

File

bt_log.h

C

```
void bt_log_linkkey(const char* msg, const bt_linkkey_t* key);
```

Description

This is function bt_log_linkkey.

bt_log_memory Function

File

bt_log.h

C

```
void bt_log_memory(const char* msg, const bt_byte* ptr, bt_ulong count);
```

Description

This is function bt_log_memory.

bt_log_msg Function

File

bt_log.h

C

```
void bt_log_msg(const char* msg);
```

Description

This is function bt_log_msg.

bt_oem_assert Function

File

bt_oem.h

C

```
void bt_oem_assert(const char* file, int line);
```

Description

This is function bt_oem_assert.

bt_oem_get_device_class Function

File

bt_oem.h

C

```
bt_long bt_oem_get_device_class();
```

Description

This is function bt_oem_get_device_class.

bt_oem_get_device_name Function

File

bt_oem.h

C

```
const char* bt_oem_get_device_name();
```

Description

This is function bt_oem_get_device_name.

bt_oem_get_pin_code Function

File

bt_oem.h

C

```
void bt_oem_get_pin_code(bt_bdaddr_t* bdaddr_remote);
```

Description

This is function bt_oem_get_pin_code.

bt_oem_linkkey_notification Function

File

bt_oem.h

C

```
void bt_oem_linkkey_notification(bt_linkkey_notification_t* lkn);
```

Description

This is function bt_oem_linkkey_notification.

bt_oem_linkkey_request Function

File

bt_oem.h

C

```
void bt_oem_linkkey_request(bt_linkkey_request_t* lkr);
```

Description

This is function bt_oem_linkkey_request.

bt_oem_log_write Function

File

bt_log.h

C

```
void bt_oem_log_write(const char* msg);
```

Description

brief Output log message. ingroup log

details DotStack calls this function to output its debug information. Implementation should output or store the specified message to whatever device or medium where it can be examined and analyzed.

bt_oem_recv Function

File

bt_hcitr.h

C

```
void bt_oem_recv(bt_byte* buffer, bt_uint len, bt_oem_recv_callback_fp callback);
```

Description

- Receive data.

This function is called by the HCI layer when it needs more data from the HCI controller. Implementation of this function must receive the specified number of bytes from the HCI controller and call the provided callback function.

@param buffer Pointer to a buffer for the received data. The buffer must be long enough to accommodate the number of bytes specified by the par len parameter.

@param len Number of bytes to receive.

@param callback A callback function that must be called when the requested number of bytes have been received.

bt_oem_schedule_signals Function

File

bt_oem.h

C

```
void bt_oem_schedule_signals();
```

Description

This is function bt_oem_schedule_signals.

bt_oem_send Function

File

bt_hcitr.h

C

```
void bt_oem_send(const bt_byte* buffer, bt_uint len, bt_oem_send_callback_fp callback);
```

Description

- Send data.

This function is called by the HCI layer when it needs to send data to the HCI controller. Implementation of this function must send the specified number of bytes to the HCI controller and call the provided callback function.

@param buffer Pointer to the data to be sent .

@param len Number of bytes to send.

@param callback A callback function that must be called when all data have been sent.

bt_oem_storage_get_capacity Function

File

bt_storage.h

C

```
bt_uint bt_oem_storage_get_capacity();
```

Description

brief Get non-volatile storage capacity. ingroup stg

details Implementation of this function must return the capacity of its non-volatile storage.

bt_oem_storage_read Function

File

bt_storage.h

C

```
void bt_oem_storage_read(bt_int addr, bt_byte* buffer, bt_int len, bt_storage_callback_fp callback);
```

Description

brief Read from the non-volatile storage. ingroup stg

details This function is called by the stack to read from the non-volatile storage. This function must be implemented by the application. When this function is called the application must start a read operation. When the number of bytes specified by the c len parameter is read, the application must call the callback function specified by the c callback parameter. The application does not have to read the whole number of bytes during the call to this function. It may complete reading later and then call the completion callback. The stack guarantees that the destination data buffer will be available until the application calls the completion callback.

param addr The non-volatile storage address where to read data from. param buffer The receiving buffer. param len The number of bytes to read. param callback The completion callback function.

bt_oem_storage_start Function

File

bt_storage.h

C

```
void bt_oem_storage_start();
```

Description

brief Begin a sequence of non-volatile storage operations. ingroup stg

details DotStack calls this function when it starts a sequence of non-volatile storage operations. When the sequence is finished, DotStack will call bt_oem_storage_stop().

bt_oem_storage_stop Function

File

bt_storage.h

C

```
void bt_oem_storage_stop();
```

Description

brief End a sequence of non-volatile storage operations. ingroup stg

details DotStack calls this function when it finishes executing a sequence of non-volatile storage operations.

bt_oem_storage_write Function

File

bt_storage.h

C

```
void bt_oem_storage_write(bt_int addr, const bt_byte* data, bt_int len, bt_storage_callback_fp callback);
```

Description

brief Write to non-volatile storage. ingroup stg

details This function is called by the stack to write data to the non-volatile storage. This function must be implemented by the application. When this function is called, the application must start writing specified data to the non-volatile storage. When all data has been written, the application must call the callback function passed in the c callback parameter. The application does not have to complete the write operation during the call to this function. It may complete the operation later and then call the callback function. In this case, the application does not have to store the data in an internal buffer. The stack guarantees that the passed data will be present until the completion callback is called by the application.

param addr The persitent storage address where to write data to. param data Pointer to data. param len Data length. param callback The completion callback function.

bt_oem_timer_clear Function

File

bt_timer.h

C

```
void bt_oem_timer_clear(bt_uint timerId);
```

Description

brief Clear timer.

details This function must be implemented by the application. When this function is called the application must clear the specified timer. If it is already expired and a callback is currently pending, the application should also take measures to cancel the callback.

param timerId ID of the timer to clear.

bt_oem_timer_set Function

File

bt_timer.h

C

```
void bt_oem_timer_set(bt_uint timerId, bt_ulong milliseconds, bt_timer_callback_fp callback);
```

Description

brief Set timer.

details This function must be implemented by the application. When it is called, the application must set the specified timer. When the timer expires, the application must call the passed callback function. The function must not wait until the timer expires. It must set the timer and exit immediately.

param timerId ID of the timer to set. param milliseconds Timer interval in milliseconds param callback Timer expiration callback function.

bt_signal_process_pending Function

File

bt_signal.h

C

```
void bt_signal_process_pending();
```

Description

This is function bt_signal_process_pending.

bt_signal_register Function

File

bt_signal.h

C

```
bt_bool bt_signal_register(bt_signal_t* signal, bt_signal_handler_fp handler, void* handler_param);
```

Description

This is function bt_signal_register.

bt_signal_set Function

File

bt_signal.h

C

```
void bt_signal_set(bt_signal_t* signal);
```

Description

This is function bt_signal_set.

bt_signal_unregister Function

File

bt_signal.h

C

```
void bt_signal_unregister(bt_signal_t* signal);
```

Description

This is function bt_signal_unregister.

bt_sys_get_connectable Function

File

bt_system.h

C

```
bt_bool bt_sys_get_connectable();
```

Description

This is function bt_sys_get_connectable.

bt_sys_get_discoverable Function

File

bt_system.h

C

```
bt_bool bt_sys_get_discoverable();
```

Description

This is function bt_sys_get_discoverable.

bt_sys_get_l2cap_manager Function

File

bt_system.h

C

```
bt_l2cap_mgr_t* bt_sys_get_l2cap_manager();
```

Description

- Get the L2CAP manager.

This function returns the L2CAP manager. The L2CAP manager is created as part of the start up sequence.

@return The L2CAP manager.

bt_sys_init Function

File

bt_system.h

C

```
void bt_sys_init();
```

Description

- Initialize the Bluetooth system.

This function initializes all internal variables of HCI, L2CAP and SDP modules. It must be called by the application before it can access any functionality provided by the library. In addition to this initialization function the application must call initialization functions of all other profile modules the application is intended to use. E.g., if the application is using the SPP module the bt_spp_init() must be called right after calling bt_sys_init().

This function essentially calls bt_sys_init_ex(HCI_LINK_POLICY_ENABLE_ALL) so all link policy setting are enabled.

bt_sys_set_modes Function

File

bt_system.h

C

```
void bt_sys_set_modes(bt_bool discoverable, bt_bool connectable, bt_sys_callback_fp callback, void*
```

```
callback_param);
```

Description

This is function bt_sys_set_modes.

bt_sys_start Function

File

bt_system.h

C

```
void bt_sys_start(bt_bool discoverable, bt_bool connectable, const bt_byte* sdp_db, bt_uint sdp_db_len,
bt_sys_callback_fp callback, void* callback_param);
```

Description

- Start the Bluetooth system.

After all modules used by the application have been initialized this function should be called to start the Bluetooth system operation. During the start up sequence it will reset and initialize the HCI controller and then create the L2CAP manager. The application will be notified when the start up sequence completes by calling the provided callback function.

Also, the caller must provide an SDP database.

@param discoverable defines whether the device is discoverable after reset. @param connectable defines whether the device is connectable after reset. @param sdp_db SDP database data. @param sdp_db_len Length of SDP database data. @param callback A callback function that will be called when the start up sequence is complete. @param callback_param An arbitrary pointer that will be passed to the callback function.

HCITR_BCSP_ALLOCATE_BUFFERS Function

File

bt_oem_config.h

C

```
HCITR_BCSP_ALLOCATE_BUFFERS();
```

Description

This is function HCITR_BCSP_ALLOCATE_BUFFERS.

IAPEA_ALLOCATE_BUFFERS Function

File

bt_oem_config.h

C

```
IAPEA_ALLOCATE_BUFFERS();
```

Description

This is function IAPEA_ALLOCATE_BUFFERS.

bt_oem_ssp_callback Function

File

bt_oem.h

C

```
void bt_oem_ssp_callback(SSP_EVENT spp_event, void* event_param, void* init_param);
```

Description

This is function bt_oem_ssp_callback.

bt_signal_init Function

File

bt_signal.h

C

```
void bt_signal_init();
```

Description

This is function bt_signal_init.

bt_sys_init_ex Function

File

bt_system.h

C

```
void bt_sys_init_ex(bt_byte default_link_policy);
```

Description

- Initialize the Bluetooth system.

This function initializes all internal variables of HCI, L2CAP and SDP modules. It must be called by the application before it can access any functionality provided by the library. In addition to this initialization function the application must call initialization functions of all other profile modules the application is intended to use. E.g., if the application is using the SPP module the bt_spp_init() must be called right after calling bt_sys_init().

Also, the caller must provide an SDP database.

@param default_link_policy default link policy settings. This is a bitmask that defines the initial value of the link policy settings for all new BR/EDR connections. This value can be a combination of the following values: li HCI_LINK_POLICY_ENABLE_ROLE_SWITCH li HCI_LINK_POLICY_ENABLE_HOLD_MODE li HCI_LINK_POLICY_ENABLE_SNIFF_MODE li HCI_LINK_POLICY_ENABLE_PARK_STATE

To enable all settings pass HCI_LINK_POLICY_ENABLE_ALL.

HCRP_ALLOCATE_BUFFERS Function

File

bt_oem_config.h

C

```
HCRP_ALLOCATE_BUFFERS();
```

Description

This is function HCRP_ALLOCATE_BUFFERS.

MAP_ALLOCATE_BUFFERS Function

File

bt_oem_config.h

C

```
MAP_ALLOCATE_BUFFERS();
```

Description

This is function MAP_ALLOCATE_BUFFERS.

bt_hcitr_3wire_cancel_recv_packet Function

File

hcitr_3wire.h

C

```
void bt_hcitr_3wire_cancel_recv_packet();
```

Description

This is function bt_hcitr_3wire_cancel_recv_packet.

bt_q_contains Function

File

queue.h

C

```
bt_bool bt_q_contains(bt_queue_element_t* phead, void* element);
```

Description

This is function bt_q_contains.

Bluetooth Support Data Types and Constants

ARG_NOT_USED Macro

File

bt_private.h

C

```
#define ARG_NOT_USED(arg) ((void)(arg))
```

Description

This is macro ARG_NOT_USED.

BT_ASSERT Macro

File

bt_private.h

C

```
#define BT_ASSERT(test) ((test) ? (void)0 : bt_oem_assert(__FILE__,__LINE__))
```

Description

This is macro BT_ASSERT.

BT_FALSE Macro

File

bt_types.h

C

```
#define BT_FALSE 0
```

Description

This is macro BT_FALSE.

BT_LINKKEY_LENGTH Macro

File

bt_types.h

C

```
#define BT_LINKKEY_LENGTH 16
```

Description

This is macro BT_LINKKEY_LENGTH.

BT_LOG Macro

File

bt_log.h

C

```
#define BT_LOG(msg)
```

Description

This is macro BT_LOG.

BT_LOG_EX Macro

File

bt_log.h

C

```
#define BT_LOG_EX(msg, level)
```

Description

This is macro BT_LOG_EX.

BT_LOG_LEVEL_ALL Macro

File

bt_log.h

C

```
#define BT_LOG_LEVEL_ALL 0
```

Description

This is macro BT_LOG_LEVEL_ALL.

BT_LOG_LEVEL_DEBUG Macro

File

bt_log.h

C

```
#define BT_LOG_LEVEL_DEBUG 80
```

Description

This is macro BT_LOG_LEVEL_DEBUG.

BT_LOG_LEVEL_ERROR Macro

File

bt_log.h

C

```
#define BT_LOG_LEVEL_ERROR 240
```

Description

This is macro BT_LOG_LEVEL_ERROR.

BT_LOG_LEVEL_INFO Macro

File

bt_log.h

C

```
#define BT_LOG_LEVEL_INFO 160
```

Description

This is macro BT_LOG_LEVEL_INFO.

BT_LOG_LEVEL_OFF Macro

File

bt_log.h

C

```
#define BT_LOG_LEVEL_OFF 0xFF
```

Description

This is macro BT_LOG_LEVEL_OFF.

BT_LOGADDR Macro

File

bt_log.h

C

```
#define BT_LOGADDR(msg, a)
```

Description

This is macro BT_LOGADDR.

BT_LOGADDR_EX Macro

File

bt_log.h

C

```
#define BT_LOGADDR_EX(msg, a, level)
```

Description

This is macro BT_LOGADDR_EX.

BT_LOGINT Macro

File

bt_log.h

C

```
#define BT_LOGINT(msg, i)
```

Description

This is macro BT_LOGINT.

BT_LOGINT_EX Macro

File

bt_log.h

C

```
#define BT_LOGINT_EX(msg, i, level)
```

Description

This is macro BT_LOGINT_EX.

BT_LOGLINKKEY Macro

File

bt_log.h

C

```
#define BT_LOGLINKKEY(msg, key)
```

Description

This is macro BT_LOGLINKKEY.

BT_LOGLINKKEY_EX Macro

File

bt_log.h

C

```
#define BT_LOGLINKKEY_EX(msg, key, level)
```

Description

This is macro BT_LOGLINKKEY_EX.

BT_LOGMEMORY Macro

File

bt_log.h

C

```
#define BT_LOGMEMORY(msg, ptr, len)
```

Description

This is macro BT_LOGMEMORY.

BT_LOGMEMORY_EX Macro

File

bt_log.h

C

```
#define BT_LOGMEMORY_EX(msg, ptr, count, level)
```

Description

This is macro BT_LOGMEMORY_EX.

BT_LOGWRITE Macro

File

bt_log.h

C

```
#define BT_LOGWRITE(msg)
```

Description

This is macro BT_LOGWRITE.

BT_MAKE_BDADDR Macro

File

bt_types.h

C

```
#define BT_MAKE_BDADDR(bdaddr, maddr, laddr) bdaddr.bd_addr_m = maddr, bdaddr.bd_addr_l = laddr
```

Description

This is macro BT_MAKE_BDADDR.

BT_MAKE_BDADDR_LE Macro

File

bt_types.h

C

```
#define BT_MAKE_BDADDR_LE(bdaddr, addr_type, maddr, laddr) bdaddr.bd_addr_m = maddr | 0x80000000 |
((bt_ulong)addr_type << 30), bdaddr.bd_addr_l = laddr
```

Description

This is macro BT_MAKE_BDADDR_LE.

BT_NO Macro

File

bt_types.h

C

```
#define BT_NO 0
```

Description

This is macro BT_NO.

BT_TRUE Macro

File

bt_types.h

C

```
#define BT_TRUE 1
```

Description

This is macro BT_TRUE.

BT_YES Macro

File

bt_types.h

C

```
#define BT_YES 1
```

Description

This is macro BT_YES.

CDS_LAST_DEVICE_ADDR Macro

File

bt_storage.h

C

```
#define CDS_LAST_DEVICE_ADDR HIDS_LAST_DEVICE_ADDR
```

Description

This is macro CDS_LAST_DEVICE_ADDR.

CDS_SIGNATURE Macro

File

bt_storage.h

C

```
#define CDS_SIGNATURE HIDS_SIGNATURE
```

Description

This is macro CDS_SIGNATURE.

CDS_SIGNATURE_ADDR Macro

File

bt_storage.h

C

```
#define CDS_SIGNATURE_ADDR HIDS_SIGNATURE_ADDR
```

Description

This is macro CDS_SIGNATURE_ADDR.

FALSE Macro

File

bt_private.h

C

```
#define FALSE 0
```

Description

This is macro FALSE.

HCI_BDADDR_LEN Macro

File

bt_storage.h

C

```
#define HCI_BDADDR_LEN 6
```

Description

This is macro HCI_BDADDR_LEN.

HIDS_LAST_DEVICE_ADDR Macro

File

bt_storage.h

C

```
#define HIDS_LAST_DEVICE_ADDR HIDS_SIGNATURE_ADDR + 1
```

Description

This is macro HIDS_LAST_DEVICE_ADDR.

HIDS_SIGNATURE Macro

File

bt_storage.h

C

```
#define HIDS_SIGNATURE 0x8E
```

Description

This is macro HIDS_SIGNATURE.

HIDS_SIGNATURE_ADDR Macro

File

bt_storage.h

C

```
#define HIDS_SIGNATURE_ADDR LKS_SIGNATURE_ADDR + 2 + (HCI_BDADDR_LEN + HCI_LINK_KEY_LEN) * LKS_MAX_LINK_KEYS
```

Description

This is macro HIDS_SIGNATURE_ADDR.

LKS_FIRST_KEY_ADDR Macro

File

bt_storage.h

C

```
#define LKS_FIRST_KEY_ADDR 2
```

Description

This is macro LKS_FIRST_KEY_ADDR.

LKS_MAX_LINK_KEYS Macro

File

bt_storage.h

C

```
#define LKS_MAX_LINK_KEYS 5
```

Description

This is macro LKS_MAX_LINK_KEYS.

LKS_SIGNATURE Macro

File

bt_storage.h

C

```
#define LKS_SIGNATURE 0x9D
```

Description

This is macro LKS_SIGNATURE.

LKS_SIGNATURE_ADDR Macro

File

bt_storage.h

C

```
#define LKS_SIGNATURE_ADDR 0
```

Description

This is macro LKS_SIGNATURE_ADDR.

LOG Macro

File

bt_private.h

C

```
#define LOG(msg) BT_LOG(msg)
```

Description

This is macro LOG.

LOG_EX Macro

File

bt_private.h

C

```
#define LOG_EX(msg, level) BT_LOG_EX(msg, level)
```

Description

This is macro LOG_EX.

LOGADDR Macro

File

bt_private.h

C

```
#define LOGADDR(msg, a) BT_LOGADDR(msg, a)
```

Description

This is macro LOGADDR.

LOGADDR_EX Macro

File

bt_private.h

C

```
#define LOGADDR_EX(msg, a, level) BT_LOGADDR_EX(msg, a, level)
```

Description

This is macro LOGADDR_EX.

LOGCLEAR Macro

File

bt_private.h

C

```
#define LOGCLEAR BT_LOGCLEAR()
```

Description

This is macro LOGCLEAR.

LOGINT Macro

File

bt_private.h

C

```
#define LOGINT(msg, i) BT_LOGINT(msg, i)
```

Description

This is macro LOGINT.

LOGINT_EX Macro

File

bt_private.h

C

```
#define LOGINT_EX(msg, i, level) BT_LOGINT_EX(msg, i, level)
```

Description

This is macro LOGINT_EX.

LOGMEMORY Macro

File

bt_private.h

C

```
#define LOGMEMORY(msg, ptr, count) BT_LOGMEMORY(msg, ptr, count)
```

Description

This is macro LOGMEMORY.

LOGMEMORY_EX Macro

File

bt_private.h

C

```
#define LOGMEMORY_EX(msg, ptr, count, level) BT_LOGMEMORY_EX(msg, ptr, count, level)
```

Description

This is macro LOGMEMORY_EX.

LOGWRITE Macro

File

bt_private.h

C

```
#define LOGWRITE(msg) BT_LOGWRITE(msg)
```

Description

This is macro LOGWRITE.

NULL Macro

File

bt_private.h

C

```
#define NULL 0
```

Description

This is macro NULL.

TRUE Macro

File

bt_private.h

C

```
#define TRUE 1
```

Description

This is macro TRUE.

_bt_signal_t Structure

File

bt_signal.h

C

```
struct _bt_signal_t {
  bt_signal_t* next_signal;
  bt_byte signaled;
  bt_signal_handler_fp handler;
  void* handler_param;
};
```

Description

This is record _bt_signal_t.

bt_bdaddr_cp Type

File

bt_types.h

C

```
typedef const bt_bdaddr_t* bt_bdaddr_cp;
```

Description

This is type bt_bdaddr_cp.

bt_bdaddr_p Type

File

bt_types.h

C

```
typedef bt_bdaddr_t* bt_bdaddr_p;
```

Description

This is type bt_bdaddr_p.

bt_bdaddr_t Structure

File

bt_types.h

C

```
typedef struct _bt_bdaddr_s {
  bt_ulong bd_addr_l;
  bt_ulong bd_addr_m;
} bt_bdaddr_t;
```

Description

This is type bt_bdaddr_t.

bt_bool Type

File

bt_types.h

C

```
typedef bt_int bt_bool;
```

Description

This is type bt_bool.

bt_byte_cp Type

File

bt_types.h

C

```
typedef const bt_byte* bt_byte_cp;
```

Description

This is type bt_byte_cp.

bt_byte_p Type

File

bt_types.h

C

```
typedef bt_byte* bt_byte_p;
```

Description

This is type bt_byte_p.

bt_char Type

File

bt_types.h

C

```
typedef char bt_char;
```

Description

This is type bt_char.

bt_char_cp Type

File

bt_types.h

C

```
typedef const bt_char* bt_char_cp;
```

Description

This is type bt_char_cp.

bt_char_p Type

File

bt_types.h

C

```
typedef bt_char* bt_char_p;
```

Description

This is type bt_char_p.

bt_id Type

File

bt_types.h

C

```
typedef bt_int bt_id;
```

Description

This is type bt_id.

bt_int_cp Type

File

bt_types.h

C

```
typedef const bt_int* bt_int_cp;
```

Description

This is type bt_int_cp.

bt_int_p Type

File

bt_types.h

C

```
typedef bt_int* bt_int_p;
```

Description

This is type bt_int_p.

bt_linkkey_cp Type

File

bt_types.h

C

```
typedef const bt_linkkey_t* bt_linkkey_cp;
```

Description

This is type bt_linkkey_cp.

bt_linkkey_notification_t Structure

File

bt_oem.h

C

```
typedef struct _bt_linkkey_notification_t {
  bt_bdaddr_t bdaddr_remote;
  bt_linkkey_t key;
  bt_byte key_type;
} bt_linkkey_notification_t;
```

Description

This is type bt_linkkey_notification_t.

bt_linkkey_p Type

File

bt_types.h

C

```
typedef bt_linkkey_t* bt_linkkey_p;
```

Description

This is type bt_linkkey_p.

bt_linkkey_request_t Structure

File

bt_oem.h

C

```
typedef struct _bt_linkkey_request_t {
  bt_bdaddr_t bdaddr_remote;
} bt_linkkey_request_t;
```

Description

This is type bt_linkkey_request_t.

bt_linkkey_t Structure

File

bt_types.h

C

```
typedef struct _bt_linkkey_t {
  bt_byte data[BT_LINKKEY_LENGTH];
} bt_linkkey_t;
```

Description

This is type bt_linkkey_t.

bt_long_cp Type

File

bt_types.h

C

```
typedef const bt_long* bt_long_cp;
```

Description

This is type bt_long_cp.

bt_long_p Type

File

bt_types.h

C

```
typedef bt_long* bt_long_p;
```

Description

This is type bt_long_p.

bt_oem_recv_callback_fp Type

File

bt_hcitr.h

C

```
typedef void (* bt_oem_recv_callback_fp)(bt_uint len);
```

Description

- Receive callback.

This callback function is called when a receive operation initiated by bt_oem_recv() has completed.

@param len Number of received bytes. The value of this parameter should always be the same as the number of bytes requested in a call to bt_oem_recv().

bt_oem_send_callback_fp Type

File

bt_hcitr.h

C

```
typedef void (* bt_oem_send_callback_fp)(void);
```

Description

- Send callback.

This callback function is called when a send operation initiated by bt_oem_send() has completed.

bt_signal_handler_fp Type

File

bt_signal.h

C

```
typedef void (* bt_signal_handler_fp)(bt_signal_t* signal, void* param);
```

Description

This is type bt_signal_handler_fp.

bt_signal_t Type

File

bt_signal.h

C

```
typedef struct _bt_signal_t bt_signal_t;
```

Description

This is type bt_signal_t.

bt_storage_callback_fp Type

File

bt_storage.h

C

```
typedef void (* bt_storage_callback_fp)(void);
```

Description

brief Storage callback.

details This callback is called when a non-volatile storage operation completes.

bt_sys_callback_fp Type

File

bt_system.h

C

```
typedef void (* bt_sys_callback_fp)(bt_bool success, void* param);
```

Description

- System start callback.

This callback function is called when system start initiated by bt_sys_start() has completed.

@param success Success of the operation: c BT_TRUE if successfull, c BT_FALSE otherwise.

@param param Callback parameter that was specified when bt_sys_start() was called.

bt_timer_callback_fp Type

File

bt_timer.h

C

```
typedef void (* bt_timer_callback_fp)(void);
```

Description

brief Timer callback.

details This callback is called when a timer expires.

bt_timer_id Enumeration

File

bt_timer.h

C

```
typedef enum _bt_timer_id_enum {
  BT_TIMER_L2CAP,
  BT_TIMER_RFCOMM,
  BT_TIMER_WAKEUP_ACK,
  BT_TIMER_TEST,
  BT_TIMER_HCI,
  BT_TIMER_IAP,
  BT_TIMER_HSP_AG,
  BT_TIMER_HFP_AG,
  BT_TIMER_ATT,
  BT_TIMER_ATT_CLIENT,
  BT_TIMER_AVRCP,
  BT_TIMER_SMP,
  BT_TIMER_IAP2,
  BT_TIMER_HCRP,
  BT_TIMER_3WIRE_T0,
  BT_TIMER_3WIRE_T1,
  BT_TIMER_MAX
} bt_timer_id;
```

Description

This is type bt_timer_id.

bt_uint_cp Type

File

bt_types.h

C

```
typedef const bt_uint* bt_uint_cp;
```

Description

This is type bt_uint_cp.

bt_uint_p Type

File

bt_types.h

C

```
typedef bt_uint* bt_uint_p;
```

Description

This is type bt_uint_p.

bt_ulong_cp Type

File

bt_types.h

C

```
typedef const bt_ulong* bt_ulong_cp;
```

Description

This is type bt_ulong_cp.

bt_ulong_p Type

File

bt_types.h

C

```
typedef bt_ulong* bt_ulong_p;
```

Description

This is type bt_ulong_p.

bt_uuid_cp Type

File

bt_types.h

C

```
typedef const bt_uuid_t* bt_uuid_cp;
```

Description

This is type bt_uuid_cp.

bt_uuid_p Type

File

bt_types.h

C

```
typedef bt_uuid_t* bt_uuid_p;
```

Description

This is type bt_uuid_p.

bt_uuid_t Structure

File

bt_types.h

C

```
typedef struct _bt_uuid_s {
  bt_ulong uuid0;
  bt_ulong uuid1;
  bt_ulong uuid2;
  bt_ulong uuid3;
} bt_uuid_t;
```

Description

This is type bt_uuid_t.

bt_uuid16 Type

File

bt_types.h

C

```
typedef bt_uint bt_uuid16;
```

Description

This is type bt_uuid16.

bt_uuid32 Type

File

bt_types.h

C

```
typedef bt_ulong bt_uuid32;
```

Description

This is type bt_uuid32.

A2DP Functions

_bt_a2dp_avdtp_mgr_callback Function

File

a2dp_private.h

C

```
void _bt_a2dp_avdtp_mgr_callback(bt_avdtp_mgr_t* mgr, bt_byte evt, bt_avdtp_event_t* evt_param, void*
callback_param);
```

Description

This is function _bt_a2dp_avdtp_mgr_callback.

bt_a2dp_aac_codec_handler Function

File

a2dp_codec_aac.h

C

```
bt_byte bt_a2dp_aac_codec_handler(bt_avdtp_codec_t* codec, bt_byte opcode, bt_avdtp_codec_op_param_t*
op_param, bt_avdtp_mgr_t* mgr);
```

Description

This is function bt_a2dp_aac_codec_handler.

bt_a2dp_find_server Function

File

a2dp.h

C

```
bt_bool bt_a2dp_find_server(bt_bdaddr_t* deviceAddress, const bt_uuid_t* service_class_id,
bt_a2dp_find_server_callback_fp callback, bt_sdp_client_callback_fp client_callback, void* callback_param);
```

Description

This is function bt_a2dp_find_server.

bt_a2dp_find_sink Function

File

a2dp.h

C

```
bt_bool bt_a2dp_find_sink(bt_bdaddr_t* deviceAddress, bt_a2dp_find_server_callback_fp callback,
bt_sdp_client_callback_fp client_callback, void* callback_param);
```

Description

brief Find sink ingroup a2dp

details This function looks for a sink on a remote device specified by c deviceAddress and, if found, returns features supported by the sink.

param deviceAddress The address of a remote device. param callback The callback function that will be called when search has completed. param client_callback The optional callback function that an application can set if it wants to be notified of state changes of the SDP client. The c evt parameter of the callback can be one of the following values: li SDP_CLIENT_STATE_IDLE li SDP_CLIENT_STATE_CONNECTING li SDP_CLIENT_STATE_DISCONNECTING li SDP_CLIENT_STATE_CONNECTED

param callback_param A pointer to arbitrary data to be passed to the c callback and c client_callback callbacks.

return li c TRUE if the function succeeds. li c FALSE otherwise.

bt_a2dp_find_source Function

File

a2dp.h

C

```
bt_bool bt_a2dp_find_source(bt_bdaddr_t* deviceAddress, bt_a2dp_find_server_callback_fp callback,
bt_sdp_client_callback_fp client_callback, void* callback_param);
```

Description

brief Find source ingroup a2dp

details This function looks for a source on a remote device specified by c deviceAddress and, if found, returns features supported by the source.

param deviceAddress The address of a remote device. param callback The callback function that will be called when search has completed. param client_callback The optional callback function that an application can set if it wants to be notified of state changes of the SDP client. The c evt parameter of the callback can be one of the following values: li SDP_CLIENT_STATE_IDLE li SDP_CLIENT_STATE_CONNECTING li SDP_CLIENT_STATE_DISCONNECTING li SDP_CLIENT_STATE_CONNECTED

param callback_param A pointer to arbitrary data to be passed to the c callback and c client_callback callbacks.

return li c TRUE if the function succeeds. li c FALSE otherwise.

bt_a2dp_get_mgr Function

File

a2dp.h

C

```
bt_a2dp_mgr_t* bt_a2dp_get_mgr();
```

Description

brief Return a pointer to an instance of the A2DP manager. ingroup a2dp

details This function returns a pointer to an instance of the A@DP manager. There is only one instance of the manager allocated by the stack. The pointer is passed as the first parameter to all A2DP functions.

bt_a2dp_init Function

File

a2dp.h

C

```
void bt_a2dp_init();
```

Description

brief Initialize the A2DP layer. ingroup a2dp

details This function initializes the A2DP layer of the stack. It must be called prior to any other A2DP function can be called.

bt_a2dp_mpeg_codec_handler Function

File

a2dp_codec_mpeg.h

C

```
bt_byte bt_a2dp_mpeg_codec_handler(bt_avdtp_codec_t* codec, bt_byte opcode, bt_avdtp_codec_op_param_t*
op_param, bt_avdtp_mgr_t* mgr);
```

Description

This is function bt_a2dp_mpeg_codec_handler.

bt_a2dp_open_and_start_stream Function

File

a2dp.h

C

```
bt_bool bt_a2dp_open_and_start_stream(bt_a2dp_mgr_t* mgr, bt_byte strm_handle, bt_bdaddr_t* remote_addr,
bt_byte seid_int, bt_byte seid_acp, const bt_avdtp_sep_capabilities_t* caps);
```

Description

brief Open & start a stream ingroup a2dp

details Opening a stream involves sending 3 requests to a remote device - "set configuration", "open stream" and "start stream". Each event generates its own event which must be handled and acted accordingly by the application. To make the use of API easier dotstack combines all these requests in one request called "open & start stream". dotstack sends necessary requests in a proper sequence, handles responses and generates only one event (A2DP_EVT_OPEN_AND_START_STREAM_COMPLETED) at the end. If any of the individual requests has failed the event's parameter bt_a2dp_event_t::open_and_start_stream_completed is populated with the error code and request id which caused it.

param mgr A2DP manager. param strm_handle Stream handle. param remote_addr The address of a remote device. param seid_int Local SEP ID. param seid_acp Remote SEP ID. param caps Stream configuration.

return li c TRUE if the function succeeds. li c FALSE otherwise. No events will be generated.

bt_a2dp_register_aac_codec Function

File

a2dp.h

C

```
void bt_a2dp_register_aac_codec(bt_a2dp_mgr_t* mgr);
```

Description

brief Register default AAC codec ingroup a2dp

details This function adds AAC codec implemented by dotstack to the list of known codecs. For more information about codecs see description of ::bt_avdtp_register_codec. The only codec A2DP is mandatory to support is SBC. All other codecs are optional. If an application wants to use AAC codec it must call this function when it is initializing.

note dotstack codecs do not do actual encoding/decoding. their function is to parse and serialize codec's configuration.

param mgr A2DP manager.

bt_a2dp_register_callback Function

File

a2dp.h

C

```
void bt_a2dp_register_callback(bt_a2dp_mgr_t* mgr, bt_a2dp_mgr_callback_fp callback, void* callback_param);
```

Description

brief Register a A2DP application callback. ingroup a2dp

details In order to be notified of various events a consumer of the A2DP layer has to register a callback function. The stack will call this function whenever a new event has been generated passing the code of the event as the second parameter. The event can be one of the following values:

@arg A2DP_EVT_CTRL_CHANNEL_CONNECTED: Control channel connected. @arg A2DP_EVT_CTRL_CHANNEL_DISCONNECTED: Control channel disconnected. @arg A2DP_EVT_CTRL_CONNECTION_FAILED: Control channel connection failed (generated only if control connection has been initiated by the local device). @arg A2DP_EVT_DISCOVER_COMPLETED: Local device completed discovering remote SEPs. @arg A2DP_EVT_GET_SEP_CAPABILITIES_COMPLETED: Local device received a response to Get SEP capabilities operation. @arg A2DP_EVT_SET_STREAM_CONFIGURATION_COMPLETED: Local device received a response to Set stream configuration operation. @arg A2DP_EVT_GET_STREAM_CONFIGURATION_COMPLETED: Local device received a response to Get stream configuration operation. @arg A2DP_EVT_STREAM_RECONFIGURE_COMPLETED: Local device received a response to Reconfigure stream operation. @arg A2DP_EVT_OPEN_STREAM_COMPLETED: Local device received a response to Open stream operation. @arg A2DP_EVT_START_STREAM_COMPLETED: Local device received a response to Start stream operation. @arg A2DP_EVT_CLOSE_STREAM_COMPLETED: Local device received a response to Close stream operation. @arg A2DP_EVT_SUSPEND_STREAM_COMPLETED: Local device received a response to Suspend stream operation. @arg A2DP_EVT_STREAM_SECURITY_CONTROL_COMPLETED: Local device received a response to Stream security control operation. @arg A2DP_EVT_ABORT_STREAM_COMPLETED: Local device received a response to Abort stream operation. @arg A2DP_EVT_SEP_INFO_RECEIVED: SEP information received. @arg A2DP_EVT_SEP_CAPABILITIES_RECEIVED: SEP capabilities received. @arg A2DP_EVT_STREAM_CONFIGURATION_RECEIVED: Stream configuration received.

@arg A2DP_EVT_SET_STREAM_CONFIGURATION_REQUESTED: Remote device requested stream configuration. @arg A2DP_EVT_OPEN_STREAM_REQUESTED: Remote device requested to open a stream. @arg A2DP_EVT_START_STREAM_REQUESTED: Remote device requested to start a stream. @arg A2DP_EVT_CLOSE_STREAM_REQUESTED: Remote device requested to close a stream. @arg A2DP_EVT_SUSPEND_STREAM_REQUESTED: Remote device requested to suspend a stream. @arg A2DP_EVT_ABORT_STREAM_REQUESTED: Remote device requested to abort a stream. @arg A2DP_EVT_RECONFIGURE_STREAM_REQUESTED: Remote device requested to reconfigure a stream. @arg A2DP_EVT_MEDIA_PACKET_RECEIVED: Remote device sent a media packet. @arg A2DP_EVT_STREAM_CONFIGURED: A stream has been configured (This event is generated right after A2DP_EVT_SET_STREAM_CONFIGURATION_REQUESTED if the local devices accepted the request). @arg A2DP_EVT_STREAM_RECONFIGURED: A stream has been re-configured (This event is generated right after A2DP_EVT_RECONFIGURE_STREAM_REQUESTED if the local devices accepted the request). @arg A2DP_EVT_STREAM_OPENED: A stream has been opened (This event is generated as a result of local or remote stream opening request). @arg A2DP_EVT_STREAM_STARTED:

A stream has been started (This event is generated right after A2DP_EVT_START_STREAM_REQUESTED if the local devices accepted the request). @arg A2DP_EVT_STREAM_CLOSED: A stream has been close (This event is generated right after A2DP_EVT_CLOSE_STREAM_REQUESTED if the local devices accepted the request). @arg A2DP_EVT_STREAM_SUSPENDED: A stream has been suspended (This event is generated right after A2DP_EVT_SUSPEND_STREAM_REQUESTED if the local devices accepted the request). @arg A2DP_EVT_STREAM_ABORTED: A stream has been aborted (This event is generated right after A2DP_EVT_SUSPEND_STREAM_REQUESTED if the local devices accepted the request. It is also generated if connection between devices has been terminated by means other than A2DP signaling, e.g. devices going out of range). @arg A2DP_EVT_MEDIA_PACKET_SENT: The local device has successfully sent a media packet to the remote device. @arg A2DP_EVT_MEDIA_PACKET_SEND_FAILED: The local device was not able to send a media packet to the remote device.

@arg A2DP_EVT_OPEN_AND_START_STREAM_COMPLETED This event is generated when a local device completed "open and start" request.

param mgr AVDTP manager. param callback The callback function that will be called when the AVDTP generates an event. param callback_param A pointer to arbitrary data to be passed to the c callback callback.

bt_a2dp_register_mpeg_codec Function

File

a2dp.h

C

```
void bt_a2dp_register_mpeg_codec(bt_a2dp_mgr_t* mgr);
```

Description

brief Register default MPEG codec ingroup a2dp

details This function adds MPEG codec implemented by dotstack to the list of known codecs. For more information about codecs see description of ::bt_avdtp_register_codec. The only codec A2DP is mandatory to support is SBC. All other codecs are optional. If an application wants to use MPEG-1,2 codec it must call this function when it is initializing.

note dotstack codecs do not do actual encoding/decoding. their function is to parse and serialize codec's configuration.

param mgr A2DP manager.

bt_a2dp_sbc_codec_handler Function

File

a2dp_codec_sbc.h

C

```
bt_byte bt_a2dp_sbc_codec_handler(bt_avdtp_codec_t* codec, bt_byte opcode, bt_avdtp_codec_op_param_t*
op_param, bt_avdtp_mgr_t* mgr);
```

Description

This is function bt_a2dp_sbc_codec_handler.

bt_a2dp_start Function

File

a2dp.h

C

```
bt_bool bt_a2dp_start(bt_a2dp_mgr_t* mgr);
```

Description

brief Start the A2DP layer. ingroup a2dp

details This function makes the A2DP layer ready to accept connection requests from remote device. To make an outgoing connection calling this function is not required.

param mgr AVDTP manager.

return li c TRUE if the function succeeds. li c FALSE otherwise.

bt_a2dp_set_configuration Function

File

a2dp.h

C

```
bt_bool bt_a2dp_set_configuration(bt_a2dp_mgr_t* mgr, bt_byte strm_handle, bt_bdaddr_t* remote_addr,
bt_byte seid_int, bt_byte seid_acp, const bt_avdtp_sep_capabilities_t* caps);
```

Description

This is function bt_a2dp_set_configuration.

A2DP Data Types and Constants

_bt_a2dp_event_u Union

File

a2dp.h

C

```
union _bt_a2dp_event_u {
  bt_a2dp_evt_open_and_start_stream_completed_t open_and_start_stream_completed;
  bt_avdtp_evt_ctrl_channel_connected_t ctrl_channel_connected;
  bt_avdtp_evt_ctrl_connection_failed_t ctrl_connection_failed;
  bt_avdtp_evt_ctrl_channel_disconnected_t ctrl_channel_disconnected;
  bt_avdtp_evt_discover_completed_t discover_completed;
  bt_avdtp_evt_sep_info_received_t sep_info_received;
  bt_avdtp_evt_get_sep_capabilities_completed_t get_sep_capabilities_completed;
  bt_avdtp_evt_sep_capabilities_received_t sep_capabilities_received;
  bt_avdtp_evt_set_stream_configuration_completed_t set_stream_configuration_completed;
  bt_avdtp_evt_get_stream_configuration_completed_t get_stream_configuration_completed;
  bt_avdtp_evt_stream_reconfigure_completed_t stream_reconfigure_completed;
  bt_avdtp_evt_open_stream_completed_t open_stream_completed;
  bt_avdtp_evt_start_stream_completed_t start_stream_completed;
  bt_avdtp_evt_close_stream_completed_t close_stream_completed;
  bt_avdtp_evt_suspend_stream_completed_t suspend_stream_completed;
  bt_avdtp_evt_stream_security_control_completed_t security_control_completed;
  bt_avdtp_evt_set_stream_configuration_requested_t set_stream_configuration_requested;
  bt_avdtp_evt_reconfigure_stream_requested_t reconfigure_stream_requested;
  bt_avdtp_evt_open_stream_requested_t open_stream_requested;
  bt_avdtp_evt_start_stream_requested_t start_stream_requested;
  bt_avdtp_evt_suspend_stream_requested_t suspend_stream_requested;
  bt_avdtp_evt_close_stream_requested_t close_stream_requested;
  bt_avdtp_evt_abort_stream_requested_t abort_stream_requested;
  bt_avdtp_evt_set_stream_configuration_t set_stream_configuration;
  bt_avdtp_evt_stream_configured_t stream_configured;
  bt_avdtp_evt_stream_reconfigured_t stream_reconfigured;
  bt_avdtp_evt_stream_opened_t stream_opened;
  bt_avdtp_evt_stream_started_t stream_started;
  bt_avdtp_evt_stream_suspended_t stream_suspended;
  bt_avdtp_evt_stream_closed_t stream_closed;
  bt_avdtp_evt_stream_aborted_t stream_aborted;
  bt_avdtp_evt_media_packet_received_t media_packet_received;
  bt_avdtp_evt_media_packet_sent_t media_packet_sent;
  bt_avdtp_evt_media_packet_send_failed_t media_packet_send_failed;
};
```

Members

Members	Description
bt_a2dp_evt_open_and_start_stream_completed_t open_and_start_stream_completed;	< Valid if event is A2DP_EVT_OPEN_AND_START_STREAM_COMPLETED
bt_avdtp_evt_ctrl_channel_connected_t ctrl_channel_connected;	< Valid if event is A2DP_EVT_CTRL_CHANNEL_CONNECTED

bt_avdtp_evt_ctrl_connection_failed_t ctrl_connection_failed;	< Valid if event is A2DP_EVT_CTRL_CONNECTION_FAILED
bt_avdtp_evt_ctrl_channel_disconnected_t ctrl_channel_disconnected;	< Valid if event is A2DP_EVT_CTRL_CHANNEL_DISCONNECTED
bt_avdtp_evt_discover_completed_t discover_completed;	< Valid if event is A2DP_EVT_DISCOVER_COMPLETED
bt_avdtp_evt_sep_info_received_t sep_info_received;	< Valid if event is A2DP_EVT_SEP_INFO_RECEIVED
bt_avdtp_evt_get_sep_capabilities_completed_t get_sep_capabilities_completed;	< Valid if event is A2DP_EVT_GET_SEP_CAPABILITIES_COMPLETED
bt_avdtp_evt_sep_capabilities_received_t sep_capabilities_received;	< Valid if event is A2DP_EVT_SEP_CAPABILITIES_RECEIVED
bt_avdtp_evt_set_stream_configuration_completed_t set_stream_configuration_completed;	< Valid if event is A2DP_EVT_SET_STREAM_CONFIGURATION_COMPLETED
bt_avdtp_evt_get_stream_configuration_completed_t get_stream_configuration_completed;	< Valid if event is A2DP_EVT_GET_STREAM_CONFIGURATION_COMPLETED
bt_avdtp_evt_stream_reconfigure_completed_t stream_reconfigure_completed;	< Valid if event is A2DP_EVT_STREAM_RECONFIGURE_COMPLETED
bt_avdtp_evt_open_stream_completed_t open_stream_completed;	< Valid if event is A2DP_EVT_OPEN_STREAM_COMPLETED
bt_avdtp_evt_start_stream_completed_t start_stream_completed;	< Valid if event is A2DP_EVT_START_STREAM_COMPLETED
bt_avdtp_evt_close_stream_completed_t close_stream_completed;	< Valid if event is A2DP_EVT_CLOSE_STREAM_COMPLETED
bt_avdtp_evt_suspend_stream_completed_t suspend_stream_completed;	< Valid if event is A2DP_EVT_SUSPEND_STREAM_COMPLETED
bt_avdtp_evt_stream_security_control_completed_t security_control_completed;	< Valid if event is A2DP_EVT_STREAM_SECURITY_CONTROL_COMPLETED
bt_avdtp_evt_set_stream_configuration_requested_t set_stream_configuration_requested;	< Valid if event is A2DP_EVT_SET_STREAM_CONFIGURATION_REQUESTED
bt_avdtp_evt_reconfigure_stream_requested_t reconfigure_stream_requested;	< Valid if event is A2DP_EVT_RECONFIGURE_STREAM_REQUESTED
bt_avdtp_evt_open_stream_requested_t open_stream_requested;	< Valid if event is A2DP_EVT_OPEN_STREAM_REQUESTED
bt_avdtp_evt_start_stream_requested_t start_stream_requested;	< Valid if event is A2DP_EVT_START_STREAM_REQUESTED
bt_avdtp_evt_suspend_stream_requested_t suspend_stream_requested;	< Valid if event is A2DP_EVT_SUSPEND_STREAM_REQUESTED
bt_avdtp_evt_close_stream_requested_t close_stream_requested;	< Valid if event is A2DP_EVT_CLOSE_STREAM_REQUESTED
bt_avdtp_evt_abort_stream_requested_t abort_stream_requested;	< Valid if event is A2DP_EVT_ABORT_STREAM_REQUESTED
bt_avdtp_evt_stream_configured_t stream_configured;	< Valid if event is A2DP_EVT_STREAM_CONFIGURED
bt_avdtp_evt_stream_reconfigured_t stream_reconfigured;	< Valid if event is A2DP_EVT_STREAM_RECONFIGURED
bt_avdtp_evt_stream_opened_t stream_opened;	< Valid if event is A2DP_EVT_STREAM_OPENED
bt_avdtp_evt_stream_started_t stream_started;	< Valid if event is A2DP_EVT_STREAM_STARTED
bt_avdtp_evt_stream_suspended_t stream_suspended;	< Valid if event is A2DP_EVT_STREAM_SUSPENDED
bt_avdtp_evt_stream_closed_t stream_closed;	< Valid if event is A2DP_EVT_STREAM_CLOSED
bt_avdtp_evt_stream_aborted_t stream_aborted;	< Valid if event is A2DP_EVT_STREAM_ABORTED
bt_avdtp_evt_media_packet_received_t media_packet_received;	< Valid if event is A2DP_EVT_MEDIA_PACKET_RECEIVED
bt_avdtp_evt_media_packet_sent_t media_packet_sent;	< Valid if event is A2DP_EVT_MEDIA_PACKET_SENT
bt_avdtp_evt_media_packet_send_failed_t media_packet_send_failed;	< Valid if event is A2DP_EVT_MEDIA_PACKET_SEND_FAILED

Description

brief Parameter to an application callback. ingroup a2dp

details This union is used to pass event specific data to the A2DP consumer. Which member of the union points to a valid structure depends on the event reported to the consumer. In general, each event has a corresponding member in the union.

_bt_a2dp_mgr_t Structure

File

a2dp.h

C

```
struct _bt_a2dp_mgr_t {
  bt_byte state;
  bt_avdtp_mgr_t* avdtp_mgr;
  bt_avdtp_stream_t* connection_stream;
  bt_a2dp_mgr_callback_fp callback;
  void* callback_param;
};
```

Members

Members	Description
bt_byte state;	< Manager state. This value can be one of the following values: < li A2DP_MANAGER_STATE_IDLE < li A2DP_MANAGER_STATE_CONNECTING
bt_avdtp_mgr_t* avdtp_mgr;	< AVDTP manager.
bt_avdtp_stream_t* connection_stream;	< Pointer to a stream being open with ::bt_a2dp_open_and_start_stream.
bt_a2dp_mgr_callback_fp callback;	< Pointer to a function which a A2DP consumer must register in order to be notified of various events.
void* callback_param;	< Pointer to arbitrary data to be passed to the c callback.

Description

brief A2DP manager. ingroup a2dp

details A structure that glues all pieces together. There is only one instance of this structure allocated by dotstack. A pointer to the instance can be get with ::bt_a2dp_get_mgr.

bt_a2dp_aac_config_t Structure

File

a2dp_codec_aac.h

C

```
typedef struct _bt_a2dp_aac_config_t {
  bt_byte object_type;
  bt_uint sampling_frequency;
  bt_byte channels;
  bt_byte rfa;
  bt_byte vbr;
  bt_ulong bit_rate;
} bt_a2dp_aac_config_t;
```

Description

This is type bt_a2dp_aac_config_t.

bt_a2dp_event_t Type

File

a2dp.h

C

```
typedef union _bt_a2dp_event_u bt_a2dp_event_t;
```

Description

This is type bt_a2dp_event_t.

bt_a2dp_evt_open_and_start_stream_completed_t Structure

File

a2dp.h

C

```
typedef struct _bt_a2dp_evt_open_and_start_stream_completed_s {
  bt_byte failed_cmd;
  bt_byte err_code;
  bt_byte strm_handle;
  union {
    bt_avdtp_evt_set_stream_configuration_completed_t set_config;
    bt_avdtp_evt_open_stream_completed_t open;
    bt_avdtp_evt_start_stream_completed_t start;
  } evt_params;
} bt_a2dp_evt_open_and_start_stream_completed_t;
```

Members

Members	Description
bt_byte failed_cmd;	< ID of the failed command. This can be one of the following values: < li AVDTP_CMD_SET_CONFIGURATION < li AVDTP_CMD_OPEN < li AVDTP_CMD_START
bt_byte err_code;	< The result of the request. < li If the remote accepted all 3 requests sent during execution of the "open & start" request c err_code == AVDTP_ERROR_SUCCESS. < li Otherwise c err_code == the error code returned by the remote party.
bt_byte strm_handle;	< Stream handle.
bt_avdtp_evt_set_stream_configuration_completed_t set_config;	< If "set configuration" request failed this member contains description of an error returned by the remote party.
bt_avdtp_evt_open_stream_completed_t open;	< If open stream" request failed this member contains description of an error returned by the remote party.
bt_avdtp_evt_start_stream_completed_t start;	< If "start stream" request failed this member contains description of an error returned by the remote party.

Description

brief Parameter to A2DP_EVT_OPEN_AND_START_STREAM_COMPLETED event ingroup a2dp

details A pointer to this structure is passed to the A2DP application callback as a valid member of the bt_a2dp_event_t union - bt_a2dp_event_t::open_and_start_stream_completed - when A2DP completed a "open & start stream" request.

bt_a2dp_find_server_callback_fp Type

File

a2dp.h

C

```
typedef void (* bt_a2dp_find_server_callback_fp)(bt_uint supported_features, bt_bool found, void* param);
```

Description

brief Notify the application of the result of searching for a remote A2DP entity (sourse or sink) ingroup a2dp

details This function is called by the A2DP layer when searching for an A2DP entity on a remote device has completed.

param supported_features Features supported by a remote A2DP entity. param found c TRUE if an A2DP entity has been found on the remote device. c FALSE otherwise. param param pointer to arbitrary data passed to the bt_a2dp_find_source() or bt_a2dp_find_sink function through its c callback_param parameter.

bt_a2dp_mgr_callback_fp Type

File

a2dp.h

C

```
typedef void (* bt_a2dp_mgr_callback_fp)(bt_a2dp_mgr_t* mgr, bt_byte evt, bt_a2dp_event_t* evt_param, void*
callback_param);
```

Description

brief A2DP application callback. ingroup a2dp

details In order to be notified of various events a consumer of the A2DP layer has to register a callback function. The stack will call that function whenever a new event has been generated.

param mgr A2DP manager.

param evt A2DP event. The event can be one of the following values: @arg A2DP_EVT_CTRL_CHANNEL_CONNECTED: Control channel connected. @arg A2DP_EVT_CTRL_CHANNEL_DISCONNECTED: Control channel disconnected. @arg A2DP_EVT_CTRL_CONNECTION_FAILED: Control channel connection failed (generated only if control connection has been initiated by the local device). @arg A2DP_EVT_DISCOVER_COMPLETED: Local device completed discovering remote SEPs. @arg A2DP_EVT_GET_SEP_CAPABILITIES_COMPLETED: Local device received a response to Get SEP capabilities operation. @arg A2DP_EVT_SET_STREAM_CONFIGURATION_COMPLETED: Local device received a response to Set stream configuration operation. @arg A2DP_EVT_GET_STREAM_CONFIGURATION_COMPLETED: Local device received a response to Get stream configuration operation. @arg A2DP_EVT_STREAM_RECONFIGURE_COMPLETED: Local device received a response to Reconfigure stream operation. @arg A2DP_EVT_OPEN_STREAM_COMPLETED: Local device received a response to Open stream operation. @arg A2DP_EVT_START_STREAM_COMPLETED: Local device received a response to Start stream operation. @arg A2DP_EVT_CLOSE_STREAM_COMPLETED: Local device received a response to Close stream operation. @arg A2DP_EVT_SUSPEND_STREAM_COMPLETED: Local device received a response to Suspend stream operation. @arg A2DP_EVT_STREAM_SECURITY_CONTROL_COMPLETED: Local device received a response to Stream security control operation. @arg A2DP_EVT_ABORT_STREAM_COMPLETED: Local device received a response to Abort stream operation. @arg A2DP_EVT_SEP_INFO_RECEIVED: SEP information received. @arg A2DP_EVT_SEP_CAPABILITIES_RECEIVED: SEP capabilities received. @arg A2DP_EVT_STREAM_CONFIGURATION_RECEIVED: Stream configuration received.

@arg A2DP_EVT_SET_STREAM_CONFIGURATION_REQUESTED: Remote device requested stream configuration. @arg A2DP_EVT_OPEN_STREAM_REQUESTED: Remote device requested to open a stream. @arg A2DP_EVT_START_STREAM_REQUESTED: Remote device requested to start a stream. @arg A2DP_EVT_CLOSE_STREAM_REQUESTED: Remote device requested to close a stream. @arg A2DP_EVT_SUSPEND_STREAM_REQUESTED: Remote device requested to suspend a stream. @arg A2DP_EVT_ABORT_STREAM_REQUESTED: Remote device requested to abort a stream. @arg A2DP_EVT_RECONFIGURE_STREAM_REQUESTED: Remote device requested to reconfigure a stream. @arg A2DP_EVT_MEDIA_PACKET_RECEIVED: Remote device sent a media packet. @arg A2DP_EVT_STREAM_CONFIGURED: A stream has been configured (This event is generated right after A2DP_EVT_SET_STREAM_CONFIGURATION_REQUESTED if the local devices accepted the request). @arg A2DP_EVT_STREAM_RECONFIGURED: A stream has been re-configured (This event is generated right after A2DP_EVT_RECONFIGURE_STREAM_REQUESTED if the local devices accepted the request). @arg A2DP_EVT_STREAM_OPENED: A stream has been opened (This event is generated as a result of local or remote stream opening request). @arg A2DP_EVT_STREAM_STARTED: A stream has been started (This event is generated right after A2DP_EVT_START_STREAM_REQUESTED if the local devices accepted the request). @arg A2DP_EVT_STREAM_CLOSED: A stream has been close (This event is generated right after A2DP_EVT_CLOSE_STREAM_REQUESTED if the local devices accepted the request). @arg A2DP_EVT_STREAM_SUSPENDED: A stream has been suspended (This event is generated right after A2DP_EVT_SUSPEND_STREAM_REQUESTED if the local devices accepted the request). @arg A2DP_EVT_STREAM_ABORTED: A stream has been aborted (This event is generated right after A2DP_EVT_SUSPEND_STREAM_REQUESTED if the local devices accepted the request. It is also generated if connection between devices has been terminated by means other than AVDTP signaling, e.g. devices going out of rage). @arg A2DP_EVT_MEDIA_PACKET_SENT: The local device has successfully sent a media packet to the remote device. @arg A2DP_EVT_MEDIA_PACKET_SEND_FAILED: The local device was not able to send a media packet to the remote device.

@arg A2DP_EVT_OPEN_AND_START_STREAM_COMPLETED Local device completed "open and start" request.

param evt_param Event parameter. Which member of the bt_a2dp_event_t union is valid depends on the event: @arg A2DP_EVT_CTRL_CHANNEL_CONNECTED: c bt_avdtp_evt_ctrl_channel_connected_t ctrl_channel_connected @arg A2DP_EVT_CTRL_CHANNEL_DISCONNECTED: c bt_avdtp_evt_ctrl_channel_disconnected_t ctrl_channel_disconnected @arg A2DP_EVT_CTRL_CONNECTION_FAILED: c NULL @arg A2DP_EVT_DISCOVER_COMPLETED: c bt_avdtp_evt_discover_completed_t discover_completed; @arg A2DP_EVT_GET_SEP_CAPABILITIES_COMPLETED: c bt_avdtp_evt_get_sep_capabilities_completed_t get_sep_capabilities_completed @arg A2DP_EVT_SET_STREAM_CONFIGURATION_COMPLETED: c bt_avdtp_evt_set_stream_configuration_completed_t set_stream_configuration_completed @arg A2DP_EVT_GET_STREAM_CONFIGURATION_COMPLETED: c bt_avdtp_evt_get_stream_configuration_completed_t get_stream_configuration_completed @arg A2DP_EVT_STREAM_RECONFIGURE_COMPLETED: c bt_avdtp_evt_stream_reconfigure_completed_t stream_reconfigure_completed @arg A2DP_EVT_OPEN_STREAM_COMPLETED: c bt_avdtp_evt_open_stream_completed_t open_stream_completed @arg A2DP_EVT_START_STREAM_COMPLETED: c bt_avdtp_evt_start_stream_completed_t start_stream_completed @arg A2DP_EVT_CLOSE_STREAM_COMPLETED: c bt_avdtp_evt_close_stream_completed_t close_stream_completed @arg A2DP_EVT_SUSPEND_STREAM_COMPLETED: c bt_avdtp_evt_suspend_stream_completed_t suspend_stream_completed @arg A2DP_EVT_STREAM_SECURITY_CONTROL_COMPLETED: c bt_avdtp_evt_stream_security_control_completed_t security_control_completed @arg A2DP_EVT_ABORT_STREAM_COMPLETED: c bt_avdtp_evt_abort_stream_requested_t abort_stream_requested @arg A2DP_EVT_SEP_INFO_RECEIVED: c bt_avdtp_evt_sep_info_received_t sep_info_received @arg A2DP_EVT_SEP_CAPABILITIES_RECEIVED: c bt_avdtp_evt_sep_capabilities_received_t sep_capabilities_received @arg A2DP_EVT_STREAM_CONFIGURATION_RECEIVED: c bt_avdtp_evt_sep_capabilities_received_t sep_capabilities_received

@arg A2DP_EVT_SET_STREAM_CONFIGURATION_REQUESTED: c bt_avdtp_evt_set_stream_configuration_requested_t set_stream_configuration_requested @arg A2DP_EVT_OPEN_STREAM_REQUESTED: c bt_avdtp_evt_open_stream_requested_t open_stream_requested @arg A2DP_EVT_START_STREAM_REQUESTED: c bt_avdtp_evt_start_stream_requested_t start_stream_requested @arg A2DP_EVT_CLOSE_STREAM_REQUESTED: c bt_avdtp_evt_close_stream_requested_t close_stream_requested @arg

A2DP_EVT_SUSPEND_STREAM_REQUESTED: c bt_avdtp_evt_suspend_stream_requested_t suspend_stream_requested @arg A2DP_EVT_ABORT_STREAM_REQUESTED: c bt_avdtp_evt_abort_stream_requested_t abort_stream_requested @arg A2DP_EVT_RECONFIGURE_STREAM_REQUESTED: c bt_avdtp_evt_reconfigure_stream_requested_t reconfigure_stream_requested @arg A2DP_EVT_MEDIA_PACKET_RECEIVED: c bt_avdtp_evt_media_packet_received_t media_packet_received @arg A2DP_EVT_STREAM_CONFIGURED: c bt_avdtp_evt_stream_configured_t stream_configured @arg A2DP_EVT_STREAM_RECONFIGURED: c bt_avdtp_evt_stream_reconfigured_t stream_reconfigured @arg A2DP_EVT_STREAM_OPENED: c bt_avdtp_evt_stream_opened_t stream_opened @arg A2DP_EVT_STREAM_STARTED: c bt_avdtp_evt_stream_started_t stream_started @arg A2DP_EVT_STREAM_CLOSED: c bt_avdtp_evt_stream_closed_t stream_closed @arg A2DP_EVT_STREAM_SUSPENDED: c bt_avdtp_evt_stream_suspended_t stream_suspended @arg A2DP_EVT_STREAM_ABORTED: c bt_avdtp_evt_stream_aborted_t stream_aborted @arg A2DP_EVT_MEDIA_PACKET_SENT: c bt_avdtp_evt_media_packet_sent_t media_packet_sent @arg A2DP_EVT_MEDIA_PACKET_SEND_FAILED: c bt_avdtp_evt_media_packet_send_failed_t media_packet_send_failed

@arg A2DP_EVT_OPEN_AND_START_STREAM_COMPLETED c bt_a2dp_evt_open_and_start_stream_completed_t open_and_start_stream_completed

param callback_param A pointer to an arbitrary data set by a call to bt_avdtp_register_callback.

bt_a2dp_mgr_t Type

File

a2dp.h

C

```
typedef struct _bt_a2dp_mgr_t bt_a2dp_mgr_t;
```

Description

This is type bt_a2dp_mgr_t.

bt_a2dp_mpeg_config_t Structure

File

a2dp_codec_mpeg.h

C

```
typedef struct _bt_a2dp_mpeg_config_t {
  bt_byte layer;
  bt_byte crc;
  bt_byte channel_mode;
  bt_byte rfa;
  bt_byte mpf;
  bt_byte sampling_frequency;
  bt_byte vbr;
  bt_uint bit_rate;
} bt_a2dp_mpeg_config_t;
```

Description

This is type bt_a2dp_mpeg_config_t.

bt_a2dp_sbc_config_t Structure

File

a2dp_codec_sbc.h

C

```
typedef struct _bt_a2dp_sbc_config_t {
  bt_byte sampling_frequency;
  bt_byte channel_mode;
  bt_byte block_length;
  bt_byte subbands;
  bt_byte allocation_method;
  bt_byte min_bitpool_val;
  bt_byte max_bitpool_val;
} bt_a2dp_sbc_config_t;
```

Description

This is type bt_a2dp_sbc_config_t.

bt_a2dp_sbc_packet_info_t Structure

File

a2dp_codec_sbc.h

C

```
typedef struct _bt_a2dp_sbc_packet_info_t {
  bt_byte frames;
  bt_byte l;
  bt_byte s;
  bt_byte f;
  bt_uint frame_size;
} bt_a2dp_sbc_packet_info_t;
```

Description

This is type bt_a2dp_sbc_packet_info_t.

__A2DP_CODEC_AAC_H Macro

File

a2dp_codec_aac.h

C

```
#define __A2DP_CODEC_AAC_H
```

Description

This is macro __A2DP_CODEC_AAC_H.

__A2DP_CODEC_MPEG_H Macro

File

a2dp_codec_mpeg.h

C

```
#define __A2DP_CODEC_MPEG_H
```

Description

This is macro __A2DP_CODEC_MPEG_H.

__A2DP_CODEC_SBC_H Macro

File

a2dp_codec_sbc.h

C

```
#define __A2DP_CODEC_SBC_H
```

Description

This is macro __A2DP_CODEC_SBC_H.

__A2DP_H Macro

File

a2dp.h

C

```
#define __A2DP_H
```

Description

This is macro __A2DP_H.

__A2DP_PRIVATE_H Macro

File

a2dp_private.h

C

```
#define __A2DP_PRIVATE_H
```

Description

This is macro __A2DP_PRIVATE_H.

A2DP_EVT_ABORT_STREAM_COMPLETED Macro

File

a2dp.h

C

```
#define A2DP_EVT_ABORT_STREAM_COMPLETED AVDTP_EVT_ABORT_STREAM_COMPLETED ///< This event is
generated when a local device received a response (either positive or negative) to a "abort stream" request.
```

Description

< This event is generated when a local device received a response (either positive or negative) to a "abort stream" request.

A2DP_EVT_ABORT_STREAM_REQUESTED Macro

File

a2dp.h

C

```
#define A2DP_EVT_ABORT_STREAM_REQUESTED AVDTP_EVT_ABORT_STREAM_REQUESTED ///< This event is
generated when a local device received "abort stream" request.
```

Description

< This event is generated when a local device received "abort stream" request.

A2DP_EVT_CLOSE_STREAM_COMPLETED Macro

File

a2dp.h

C

```
#define A2DP_EVT_CLOSE_STREAM_COMPLETED AVDTP_EVT_CLOSE_STREAM_COMPLETED ///< This event is
generated when a local device received a response (either positive or negative) to a "close stream" request.
```

Description

< This event is generated when a local device received a response (either positive or negative) to a "close stream" request.

A2DP_EVT_CLOSE_STREAM_REQUESTED Macro

File

a2dp.h

C

```
#define A2DP_EVT_CLOSE_STREAM_REQUESTED AVDTP_EVT_CLOSE_STREAM_REQUESTED ///< This event is
generated when a local device received "close stream" request.
```

Description

< This event is generated when a local device received "close stream" request.

A2DP_EVT_CTRL_CHANNEL_CONNECTED Macro

File

a2dp.h

C

```
#define A2DP_EVT_CTRL_CHANNEL_CONNECTED AVDTP_EVT_CTRL_CHANNEL_CONNECTED ///< This event is generated when a control channel between two AVDTP entities has been established.
```

Description

< This event is generated when a control channel between two AVDTP entities has been established.

A2DP_EVT_CTRL_CHANNEL_DISCONNECTED Macro

File

a2dp.h

C

```
#define A2DP_EVT_CTRL_CHANNEL_DISCONNECTED AVDTP_EVT_CTRL_CHANNEL_DISCONNECTED ///< This event is generated when a control channel between two AVDTP entities has been terminated.
```

Description

< This event is generated when a control channel between two AVDTP entities has been terminated.

A2DP_EVT_CTRL_CONNECTION_FAILED Macro

File

a2dp.h

C

```
#define A2DP_EVT_CTRL_CONNECTION_FAILED AVDTP_EVT_CTRL_CONNECTION_FAILED ///< This event is generated when a local device failed to create a control channel between two AVDTP entities.
```

Description

< This event is generated when a local device failed to create a control channel between two AVDTP entities.

A2DP_EVT_DISCOVER_SEP_COMPLETED Macro

File

a2dp.h

C

```
#define A2DP_EVT_DISCOVER_SEP_COMPLETED AVDTP_EVT_DISCOVER_COMPLETED ///< This event is generated when a local device received a response (either positive or negative) to a "discover" request.
```

Description

< This event is generated when a local device received a response (either positive or negative) to a "discover" request.

A2DP_EVT_GET_SEP_CAPABILITIES_COMPLETED Macro

File

a2dp.h

C

```
#define A2DP_EVT_GET_SEP_CAPABILITIES_COMPLETED AVDTP_EVT_GET_SEP_CAPABILITIES_COMPLETED ///< This event is generated when a local device received a response (either positive or negative) to a "get SEP
```

```
capabilities" request.
```

Description

< This event is generated when a local device received a response (either positive or negative) to a "get SEP capabilities" request.

A2DP_EVT_GET_STREAM_CONFIGURATION_COMPLETED Macro

File

a2dp.h

C

```
#define A2DP_EVT_GET_STREAM_CONFIGURATION_COMPLETED AVDTP_EVT_GET_STREAM_CONFIGURATION_COMPLETED    ///<
This event is generated when a local device received a response (either positive or negative) to a "get
stream configuration" request.
```

Description

< This event is generated when a local device received a response (either positive or negative) to a "get stream configuration" request.

A2DP_EVT_MEDIA_PACKET_RECEIVED Macro

File

a2dp.h

C

```
#define A2DP_EVT_MEDIA_PACKET_RECEIVED AVDTP_EVT_MEDIA_PACKET_RECEIVED                ///< This event is
generated when a local device received a media packet.
```

Description

< This event is generated when a local device received a media packet.

A2DP_EVT_MEDIA_PACKET_SEND_FAILED Macro

File

a2dp.h

C

```
#define A2DP_EVT_MEDIA_PACKET_SEND_FAILED AVDTP_EVT_MEDIA_PACKET_SEND_FAILED            ///< This event is
generated when a local device failed to send a media packet.
```

Description

< This event is generated when a local device failed to send a media packet.

A2DP_EVT_MEDIA_PACKET_SENT Macro

File

a2dp.h

C

```
#define A2DP_EVT_MEDIA_PACKET_SENT AVDTP_EVT_MEDIA_PACKET_SENT                    ///< This event is
generated when a local device sent a media packet.
```

Description

< This event is generated when a local device sent a media packet.

A2DP_EVT_NOTHING Macro

File

a2dp.h

C

```
#define A2DP_EVT_NOTHING AVDTP_EVT_NULL
```

Description

This is macro A2DP_EVT_NOTHING.

A2DP_EVT_OPEN_AND_START_STREAM_COMPLETED Macro

File

a2dp.h

C

```
#define A2DP_EVT_OPEN_AND_START_STREAM_COMPLETED (AVDTP_EVT_LAST + 1)                    ///< This
event is generated when a local device completed "open and start" request.
```

Description

< This event is generated when a local device completed "open and start" request.

A2DP_EVT_OPEN_STREAM_COMPLETED Macro

File

a2dp.h

C

```
#define A2DP_EVT_OPEN_STREAM_COMPLETED AVDTP_EVT_OPEN_STREAM_COMPLETED          ///< This event is
generated when a local device received a response (either positive or negative) to a "open stream" request.
```

Description

< This event is generated when a local device received a response (either positive or negative) to a "open stream" request.

A2DP_EVT_OPEN_STREAM_REQUESTED Macro

File

a2dp.h

C

```
#define A2DP_EVT_OPEN_STREAM_REQUESTED AVDTP_EVT_OPEN_STREAM_REQUESTED          ///< This event is
generated when a local device received "open stream" request.
```

Description

< This event is generated when a local device received "open stream" request.

A2DP_EVT_RECONFIGURE_STREAM_COMPLETED Macro

File

a2dp.h

C

```
#define A2DP_EVT_RECONFIGURE_STREAM_COMPLETED AVDTP_EVT_STREAM_RECONFIGURE_COMPLETED            ///< This
event is generated when a local device received a response (either positive or negative) to a "change
stream configuration" request.
```

Description

< This event is generated when a local device received a response (either positive or negative) to a "change stream configuration" request.

A2DP_EVT_RECONFIGURE_STREAM_REQUESTED Macro

File

a2dp.h

C

```
#define A2DP_EVT_RECONFIGURE_STREAM_REQUESTED AVDTP_EVT_RECONFIGURE_STREAM_REQUESTED            ///< This event is generated when a local device received "change stream configuration" request.
```

Description

< This event is generated when a local device received "change stream configuration" request.

A2DP_EVT_SEP_CAPABILITIES_RECEIVED Macro

File

a2dp.h

C

```
#define A2DP_EVT_SEP_CAPABILITIES_RECEIVED AVDTP_EVT_SEP_CAPABILITIES_RECEIVED            ///< This event is generated when a local device received a positive response to a "get SEP capabilities" request.
```

Description

< This event is generated when a local device received a positive response to a "get SEP capabilities" request.

A2DP_EVT_SEP_INFO_RECEIVED Macro

File

a2dp.h

C

```
#define A2DP_EVT_SEP_INFO_RECEIVED AVDTP_EVT_SEP_INFO_RECEIVED                    ///< This event is generated for each SEP contained in a positive response to a "discover" request.
```

Description

< This event is generated for each SEP contained in a positive response to a "discover" request.

A2DP_EVT_SET_STREAM_CONFIGURATION_COMPLETED Macro

File

a2dp.h

C

```
#define A2DP_EVT_SET_STREAM_CONFIGURATION_COMPLETED AVDTP_EVT_SET_STREAM_CONFIGURATION_COMPLETED    ///< This event is generated when a local device received a response (either positive or negative) to a "set stream configuration" request.
```

Description

< This event is generated when a local device received a response (either positive or negative) to a "set stream configuration" request.

A2DP_EVT_SET_STREAM_CONFIGURATION_REQUESTED Macro

File

a2dp.h

C

```
#define A2DP_EVT_SET_STREAM_CONFIGURATION_REQUESTED AVDTP_EVT_SET_STREAM_CONFIGURATION_REQUESTED   ///< This event is generated when a local device received "set stream configuration" request.
```

Description

< This event is generated when a local device received "set stream configuration" request.

A2DP_EVT_START_STREAM_COMPLETED Macro

File

a2dp.h

C

```
#define A2DP_EVT_START_STREAM_COMPLETED AVDTP_EVT_START_STREAM_COMPLETED ///< This event is generated when a local device received a response (either positive or negative) to a "start stream" request.
```

Description

< This event is generated when a local device received a response (either positive or negative) to a "start stream" request.

A2DP_EVT_START_STREAM_REQUESTED Macro

File

a2dp.h

C

```
#define A2DP_EVT_START_STREAM_REQUESTED AVDTP_EVT_START_STREAM_REQUESTED ///< This event is generated when a local device received "start stream" request.
```

Description

< This event is generated when a local device received "start stream" request.

A2DP_EVT_STREAM_ABORTED Macro

File

a2dp.h

C

```
#define A2DP_EVT_STREAM_ABORTED AVDTP_EVT_STREAM_ABORTED ///< This event is generated when a local device has successfully aborted a stream.
```

Description

< This event is generated when a local device has successfully aborted a stream. < This event follows the A2DP_EVT_ABORT_STREAM_REQUESTED if the upper layer has accepted it. < This event is not generated if stream abortion was initiated by the local device.

A2DP_EVT_STREAM_CLOSED Macro

File

a2dp.h

C

```
#define A2DP_EVT_STREAM_CLOSED AVDTP_EVT_STREAM_CLOSED ///< This event is generated when a local device has successfully closed a stream.
```

Description

< This event is generated when a local device has successfully closed a stream. < This event follows the A2DP_EVT_CLOSE_STREAM_REQUESTED if the upper layer has accepted it. < This event is not generated if stream closing was initiated by the local device.

A2DP_EVT_STREAM_CONFIGURATION_RECEIVED Macro

File

a2dp.h

C

```
#define A2DP_EVT_STREAM_CONFIGURATION_RECEIVED AVDTP_EVT_STREAM_CONFIGURATION_RECEIVED ///< This
```

```
event is generated when a local device received a positive response to a "get stream configuration" request.
```

Description

< This event is generated when a local device received a positive response to a "get stream configuration" request.

A2DP_EVT_STREAM_CONFIGURED Macro

File

a2dp.h

C

```
#define A2DP_EVT_STREAM_CONFIGURED AVDTP_EVT_STREAM_CONFIGURED                    ///< This event is
generated when a local device has successfully configured a stream.
```

Description

< This event is generated when a local device has successfully configured a stream. < This event follows the A2DP_EVT_SET_STREAM_CONFIGURATION_REQUESTED if the upper layer has accepted it. < This event is not generated if stream configuration was initiated by the local device.

A2DP_EVT_STREAM_OPENED Macro

File

a2dp.h

C

```
#define A2DP_EVT_STREAM_OPENED AVDTP_EVT_STREAM_OPENED                        ///< This event is generated
when a local device has successfully opened a stream.
```

Description

< This event is generated when a local device has successfully opened a stream. < This event follows the A2DP_EVT_OPEN_STREAM_REQUESTED if the upper layer has accepted it. < This event is not generated if stream opening was initiated by the local device.

A2DP_EVT_STREAM_RECONFIGURED Macro

File

a2dp.h

C

```
#define A2DP_EVT_STREAM_RECONFIGURED AVDTP_EVT_STREAM_RECONFIGURED                  ///< This event is
generated when a local device has successfully reconfigured a stream.
```

Description

< This event is generated when a local device has successfully reconfigured a stream. < This event follows the A2DP_EVT_RECONFIGURE_STREAM_REQUESTED if the upper layer has accepted it. < This event is not generated if stream reconfiguration was initiated by the local device.

A2DP_EVT_STREAM_SECURITY_CONTROL_COMPLETED Macro

File

a2dp.h

C

```
#define A2DP_EVT_STREAM_SECURITY_CONTROL_COMPLETED AVDTP_EVT_STREAM_SECURITY_CONTROL_COMPLETED    ///< This
event is generated when a local device received a response (either positive or negative) to a "exchange
content protection control data" request.
```

Description

< This event is generated when a local device received a response (either positive or negative) to a "exchange content protection control data" request.

A2DP_EVT_STREAM_STARTED Macro

File

a2dp.h

C

```
#define A2DP_EVT_STREAM_STARTED AVDTP_EVT_STREAM_STARTED ///< This event is generated when a local device has successfully started a stream.
```

Description

< This event is generated when a local device has successfully started a stream. < This event follows the A2DP_EVT_START_STREAM_REQUESTED if the upper layer has accepted it. < This event is not generated if stream starting was initiated by the local device.

A2DP_EVT_STREAM_SUSPENDED Macro

File

a2dp.h

C

```
#define A2DP_EVT_STREAM_SUSPENDED AVDTP_EVT_STREAM_SUSPENDED ///< This event is generated when a local device has successfully suspended a stream.
```

Description

< This event is generated when a local device has successfully suspended a stream. < This event follows the A2DP_EVT_SUSPEND_STREAM_REQUESTED if the upper layer has accepted it. < This event is not generated if stream suspension was initiated by the local device.

A2DP_EVT_SUSPEND_STREAM_COMPLETED Macro

File

a2dp.h

C

```
#define A2DP_EVT_SUSPEND_STREAM_COMPLETED AVDTP_EVT_SUSPEND_STREAM_COMPLETED ///< This event is generated when a local device received a response (either positive or negative) to a "suspend stream" request.
```

Description

< This event is generated when a local device received a response (either positive or negative) to a "suspend stream" request.

A2DP_EVT_SUSPEND_STREAM_REQUESTED Macro

File

a2dp.h

C

```
#define A2DP_EVT_SUSPEND_STREAM_REQUESTED AVDTP_EVT_SUSPEND_STREAM_REQUESTED ///< This event is generated when a local device received "suspend stream" request.
```

Description

< This event is generated when a local device received "suspend stream" request.

A2DP_MANAGER_STATE_CONNECTING Macro

File

a2dp.h

C

```
#define A2DP_MANAGER_STATE_CONNECTING 1
```

Description

This is macro A2DP_MANAGER_STATE_CONNECTING.

A2DP_MANAGER_STATE_IDLE Macro

File

a2dp.h

C

```
#define A2DP_MANAGER_STATE_IDLE 0
```

Description

This is macro A2DP_MANAGER_STATE_IDLE.

A2DP_SINK_FEATURE_AMPLIFIER Macro

File

a2dp.h

C

```
#define A2DP_SINK_FEATURE_AMPLIFIER 8    ///< Amplifier
```

Description

< Amplifier

A2DP_SINK_FEATURE_HEADPHONE Macro

File

a2dp.h

C

```
#define A2DP_SINK_FEATURE_HEADPHONE 1    ///< Headphone
```

Description

< Headphone

A2DP_SINK_FEATURE_RECORDER Macro

File

a2dp.h

C

```
#define A2DP_SINK_FEATURE_RECORDER 4    ///< Recorder
```

Description

< Recorder

A2DP_SINK_FEATURE_SPEAKER Macro

File

a2dp.h

C

```
#define A2DP_SINK_FEATURE_SPEAKER 2    ///< Speaker
```

Description

< Speaker

A2DP_SOURCE_FEATURE_MICROPHONE Macro

File

a2dp.h

C

```
#define A2DP_SOURCE_FEATURE_MICROPHONE 2    ///< Mic
```

Description

< Mic

A2DP_SOURCE_FEATURE_MIXER Macro

File

a2dp.h

C

```
#define A2DP_SOURCE_FEATURE_MIXER 8    ///< Mixer
```

Description

< Mixer

A2DP_SOURCE_FEATURE_PLAYER Macro

File

a2dp.h

C

```
#define A2DP_SOURCE_FEATURE_PLAYER 1    ///< Player
```

Description

< Player

A2DP_SOURCE_FEATURE_TUNER Macro

File

a2dp.h

C

```
#define A2DP_SOURCE_FEATURE_TUNER 4    ///< Tuner
```

Description

< Tuner

AAC_CHANNELS_1 Macro

File

a2dp_codec_aac.h

C

```
#define AAC_CHANNELS_1 0x2
```

Description

This is macro AAC_CHANNELS_1.

AAC_CHANNELS_2 Macro

File

a2dp_codec_aac.h

C

```
#define AAC_CHANNELS_2 0x1
```

Description

This is macro AAC_CHANNELS_2.

AAC_CHANNELS_ALL Macro

File

a2dp_codec_aac.h

C

```
#define AAC_CHANNELS_ALL 0x3
```

Description

This is macro AAC_CHANNELS_ALL.

AAC_OBJECT_TYPE_MPEG_2_LC Macro

File

a2dp_codec_aac.h

C

```
#define AAC_OBJECT_TYPE_MPEG_2_LC 0x80
```

Description

This is macro AAC_OBJECT_TYPE_MPEG_2_LC.

AAC_OBJECT_TYPE_MPEG_4_LC Macro

File

a2dp_codec_aac.h

C

```
#define AAC_OBJECT_TYPE_MPEG_4_LC 0x40
```

Description

This is macro AAC_OBJECT_TYPE_MPEG_4_LC.

AAC_OBJECT_TYPE_MPEG_4_LTP Macro

File

a2dp_codec_aac.h

C

```
#define AAC_OBJECT_TYPE_MPEG_4_LTP 0x20
```

Description

This is macro AAC_OBJECT_TYPE_MPEG_4_LTP.

AAC_OBJECT_TYPE_MPEG_4_SCALABLE Macro

File

a2dp_codec_aac.h

C

```
#define AAC_OBJECT_TYPE_MPEG_4_SCALABLE 0x10
```

Description

This is macro AAC_OBJECT_TYPE_MPEG_4_SCALABLE.

AAC_SAMPLING_FREQUENCY_11025 Macro

File

a2dp_codec_aac.h

C

```
#define AAC_SAMPLING_FREQUENCY_11025 0x040
```

Description

This is macro AAC_SAMPLING_FREQUENCY_11025.

AAC_SAMPLING_FREQUENCY_12000 Macro

File

a2dp_codec_aac.h

C

```
#define AAC_SAMPLING_FREQUENCY_12000 0x020
```

Description

This is macro AAC_SAMPLING_FREQUENCY_12000.

AAC_SAMPLING_FREQUENCY_16000 Macro

File

a2dp_codec_aac.h

C

```
#define AAC_SAMPLING_FREQUENCY_16000 0x010
```

Description

This is macro AAC_SAMPLING_FREQUENCY_16000.

AAC_SAMPLING_FREQUENCY_22050 Macro

File

a2dp_codec_aac.h

C

```
#define AAC_SAMPLING_FREQUENCY_22050 0x008
```

Description

This is macro AAC_SAMPLING_FREQUENCY_22050.

AAC_SAMPLING_FREQUENCY_24000 Macro

File

a2dp_codec_aac.h

C

```
#define AAC_SAMPLING_FREQUENCY_24000 0x004
```

Description

This is macro AAC_SAMPLING_FREQUENCY_24000.

AAC_SAMPLING_FREQUENCY_32000 Macro

File

a2dp_codec_aac.h

C

```
#define AAC_SAMPLING_FREQUENCY_32000 0x002
```

Description

This is macro AAC_SAMPLING_FREQUENCY_32000.

AAC_SAMPLING_FREQUENCY_44100 Macro

File

a2dp_codec_aac.h

C

```
#define AAC_SAMPLING_FREQUENCY_44100 0x001
```

Description

This is macro AAC_SAMPLING_FREQUENCY_44100.

AAC_SAMPLING_FREQUENCY_48000 Macro

File

a2dp_codec_aac.h

C

```
#define AAC_SAMPLING_FREQUENCY_48000 0x800
```

Description

This is macro AAC_SAMPLING_FREQUENCY_48000.

AAC_SAMPLING_FREQUENCY_64000 Macro

File

a2dp_codec_aac.h

C

```
#define AAC_SAMPLING_FREQUENCY_64000 0x400
```

Description

This is macro AAC_SAMPLING_FREQUENCY_64000.

AAC_SAMPLING_FREQUENCY_8000 Macro

File

a2dp_codec_aac.h

C

```
#define AAC_SAMPLING_FREQUENCY_8000 0x080
```

Description

This is macro AAC_SAMPLING_FREQUENCY_8000.

AAC_SAMPLING_FREQUENCY_88200 Macro

File

a2dp_codec_aac.h

C

```
#define AAC_SAMPLING_FREQUENCY_88200 0x200
```

Description

This is macro AAC_SAMPLING_FREQUENCY_88200.

AAC_SAMPLING_FREQUENCY_96000 Macro

File

a2dp_codec_aac.h

C

```
#define AAC_SAMPLING_FREQUENCY_96000 0x100
```

Description

This is macro AAC_SAMPLING_FREQUENCY_96000.

AAC_SAMPLING_FREQUENCY_ALL Macro

File

a2dp_codec_aac.h

C

```
#define AAC_SAMPLING_FREQUENCY_ALL 0xFFF
```

Description

This is macro AAC_SAMPLING_FREQUENCY_ALL.

AAC_VBR_NOT_SUPPORTED Macro

File

a2dp_codec_aac.h

C

```
#define AAC_VBR_NOT_SUPPORTED 0
```

Description

This is macro AAC_VBR_NOT_SUPPORTED.

AAC_VBR_SUPPORTED Macro

File

a2dp_codec_aac.h

C

```
#define AAC_VBR_SUPPORTED 1
```

Description

This is macro AAC_VBR_SUPPORTED.

bt_a2dp_abort_stream Macro

File

a2dp.h

C

```
#define bt_a2dp_abort_stream(mgr, strm_handle) bt_avdtp_abort_stream(mgr->avdtp_mgr, strm_handle)
```

Description

brief Suspend a stream. ingroup a2dp

details This function tries to suspend a stream by sending a request to the remote party. The stream can be in any state state except AVDTP_STREAM_STATE_IDLE. As a result of this operation the A2DP_EVT_ABORT_STREAM_COMPLETED event will be generated. This operation cannot be rejected. The p evt_param.abort_stream_requested.err_code is always == AVDTP_ERROR_SUCCESS.

param mgr A2DP manager. param strm_handle Stream handle.

return li c TRUE if the function succeeds, i.e. the actual request has been sent to the remote party. li c FALSE otherwise. No events will be generated.

bt_a2dp_add_media_rx_buffer Macro

File

a2dp.h

C

```
#define bt_a2dp_add_media_rx_buffer(mgr, strm_handle, buffer) bt_avdtp_add_media_rx_buffer(mgr->avdtp_mgr,
strm_handle, buffer)
```

Description

brief Add a media packet buffer to a receive queue ingroup a2dp

details The consumer of A2DP is responsible for allocating and supplying A2DP with buffers used to store received packets. A2DP itself only has a queue for storing pointers to buffers supplied by the consumer. When a packet comes in A2DP finds the first buffer large enough to hold the received packet, copies the packet to the buffer and generates a A2DP_EVT_MEDIA_PACKET_RECEIVED event. The consumer then has to process the data in the buffer and return it back to the queue. If there is no buffers in the queue or none of the buffers is large enough the received packets is dropped. Each buffer has a field (data_len) that holds the length of the received buffer. This field is never 0 if the buffer contains a packet. If a channel closed regardless of what has caused that and there are still buffers in the queue A2DP generates a A2DP_EVT_MEDIA_PACKET_RECEIVED event for each buffer and sets the data_len to 0. This is to inform the A2DP consumer that the buffer has not been used and can be, for example, deallocated. This function adds a buffer to the receive queue.

param mgr A2DP manager. param strm_handle Stream handle. param buffer Pointer to a structure that holds the buffer and its parameters.

return li c TRUE if the function succeeds. li c FALSE otherwise. The function fails only if a stream specified by the p strm_handle parameter li does not exist. The stream can be in any state to call this function.

bt_a2dp_add_media_tx_buffer Macro

File

a2dp.h

C

```
#define bt_a2dp_add_media_tx_buffer(mgr, strm_handle, buffer) bt_avdtp_add_media_tx_buffer(mgr->avdtp_mgr,
strm_handle, buffer)
```

Description

brief Add a media packet buffer to a send queue ingroup a2dp

details When the consumer of A2DP wants to send a packet to a remote device it calls this function. The function adds the packet to a queue and tells A2DP that it has something to send. The packet will be send as soon as the stream goes to A2DP_STREAM_STATE_STREAMING state. When the packet has been successfully sent a A2DP_EVT_MEDIA_PACKET_SENT is generated. Otherwise a A2DP_EVT_MEDIA_PACKET_SEND_FAILED is generated. Regardless of the event generated the consumer can re-use the buffer as A2DP has removed it from the queue and gave up control over it. As in the case of received buffers, if a channel closed regardless of what has caused that and there are still buffers in the queue A2DP generates a A2DP_EVT_MEDIA_PACKET_SENT event for each buffer and sets the data_len field to 0. This is to inform the AVDTP consumer that the buffer has not been used and can be, for example, deallocated.

param mgr A2DP manager. param strm_handle Stream handle. param buffer Pointer to a structure that holds the buffer and its parameters.

return li c TRUE if the function succeeds. li c FALSE otherwise. The function fails only if a stream specified by the p strm_handle parameter li does not exist. The stream can be in any state to call this function.

bt_a2dp_call_codec Macro

File

a2dp.h

C

```
#define bt_a2dp_call_codec(codec, op, op_param, mgr) codec->codec_handler(codec, op, op_param,
mgr->avdtp_mgr)
```

Description

This is macro bt_a2dp_call_codec.

bt_a2dp_cancel_listen Macro

File

a2dp.h

C

```
#define bt_a2dp_cancel_listen(mgr, strm_handle, sep_id) bt_avdtp_cancel_listen(mgr->avdtp_mgr, strm_handle,
sep_id)
```

Description

brief Cancel listening for incoming connections. ingroup a2dp

details This function removes a SEP from a list of SEPS which a stream can use for incoming requests.

param mgr A2DP manager. param strm_handle Stream handle. param sep_id Local SEP ID.

return li c TRUE if the function succeeds. li c FALSE otherwise.

bt_a2dp_close_stream Macro

File

a2dp.h

C

```
#define bt_a2dp_close_stream(mgr, strm_handle) bt_avdtp_close_stream(mgr->avdtp_mgr, strm_handle)
```

Description

brief Close a stream. ingroup a2dp

details This function tries to close a stream by sending a request to the remote party. The stream has to be in AVDTP_STREAM_STATE_OPEN or AVDTP_STREAM_STATE_STREAMING state. As a result of this operation the A2DP_EVT_CLOSE_STREAM_COMPLETED event will be generated. If the stream has been closed the p evt_param.bt_avdtp_evt_close_stream_completed_t.err_code == AVDTP_ERROR_SUCCESS. Otherwise, if the remote device for any reason cannot or does not wish to close the stream, the p evt_param.bt_avdtp_evt_close_stream_completed_t.err_code == the error code sent by the remote.

param mgr A2DP manager. param strm_handle Stream handle.

return li c TRUE if the function succeeds, i.e. the actual request has been sent to the remote party. li c FALSE otherwise. No events will be generated.

bt_a2dp_connect Macro

File

a2dp.h

C

```
#define bt_a2dp_connect(mgr, remote_addr) bt_avdtp_connect(mgr->avdtp_mgr, remote_addr)
```

Description

brief Connect to a remote device. ingroup a2dp

details This function opens a control channel connection to a remote device specified by the p remote_addr. If connection cannot be initiated for some reason, for example, there is not enough resources, it returns FALSE and not events are generated. Otherwise the result of an attempt to connect to the remote device is reported via the AVDTP callback. The events generated will either be A2DP_EVT_CTRL_CHANNEL_CONNECTED or A2DP_EVT_CTRL_CHANNEL_CONNECTION_FAILED.

param mgr A2DP manager. param remote_addr The address of a remote device.

return li c TRUE if connection establishment has been started. li c FALSE otherwise.

bt_a2dp_connect_ex Macro

File

a2dp.h

C

```
#define bt_a2dp_connect_ex(mgr, remote_addr, acl_config) bt_avdtp_connect_ex(mgr->avdtp_mgr, remote_addr,
acl_config)
```

Description

brief Connect to a remote device. ingroup a2dp

details This function opens a control channel connection to a remote device specified by the p remote_addr. If connection cannot be initiated for some reason, for example, there is not enough resources, it returns FALSE and not events are generated. Otherwise the result of an attempt to connect to the remote device is reported via the AVDTP callback. The events generated will either be A2DP_EVT_CTRL_CHANNEL_CONNECTED or A2DP_EVT_CTRL_CHANNEL_CONNECTION_FAILED.

param mgr A2DP manager. param remote_addr The address of a remote device. param acl_config ACL link configuration. This can be a combination of the following values: li HCI_CONFIG_ENABLE_AUTHENTICATION li HCI_CONFIG_ENABLE_ENCRYPTION li HCI_CONFIG_BECOME_MASTER

return li c TRUE if connection establishment has been started. li c FALSE otherwise.

bt_a2dp_create_stream Macro

File

a2dp.h

C

```
#define bt_a2dp_create_stream(mgr) bt_avdtp_create_stream(mgr->avdtp_mgr)
```

Description

brief Create a stream. ingroup a2dp

details This function allocates memory for storing stream's data and assigns a stream handle. The stream handle is used to manipulate the stream - open, close, configure, suspend, abort.

param mgr A2DP manager.

return li c Stream handle if the function succeeds. li c 0 otherwise.

bt_a2dp_destroy_stream Macro

File

a2dp.h

C

```
#define bt_a2dp_destroy_stream(mgr, strm_handle) bt_avdtp_destroy_stream(mgr->avdtp_mgr, strm_handle)
```

Description

brief Destroy a stream. ingroup a2dp

details This function frees memory used by the stream. The stream has to exist and be in the "idle" state for this function to succeed. I.e. the stream has to be closed or aborted before this function can be called.

param mgr A2DP manager. param strm_handle Stream handle.

return li c TRUE if the function succeeds. li c FALSE otherwise.

bt_a2dp_disconnect Macro

File

a2dp.h

C

```
#define bt_a2dp_disconnect(mgr, remote_addr) bt_avdtp_disconnect(mgr->avdtp_mgr, remote_addr)
```

Description

brief Disconnect from a remote device. ingroup a2dp

details This function closes a control and transport channels on all streams associated with the remote device specified by the p remote_addr. As a result of this operation the following events will be generated: @arg A2DP_EVT_MEDIA_PACKET_RECEIVED: if a stream's receive queue is not empty this event is generated for each buffer with bt_media_packet_t::data_len set to 0 @arg A2DP_EVT_MEDIA_PACKET_SENT: if a stream's send queue is not empty this event is generated for each buffer with bt_media_packet_t::data_len set to 0 @arg A2DP_EVT_STREAM_CLOSED: this event is generate if a stream is in "closing" state as a result of a request from the remote device or ::bt_a2dp_close_stream call before ::bt_a2dp_disconnect call. @arg A2DP_EVT_STREAM_ABORTED: this event is generated if a stream is in "active" state at the time of bt_avdtp_disconnect call.

param mgr A2DP manager. param remote_addr The address of a remote device.

return li c TRUE if disconnection has been started. li c FALSE otherwise. No events will be generated.

bt_a2dp_discover Macro

File

a2dp.h

C

```
#define bt_a2dp_discover(mgr, remote_addr) bt_avdtp_discover(mgr->avdtp_mgr, remote_addr)
```

Description

brief Discover SEPs on a remote device. ingroup a2dp

details This function asks the remote device to send a list of all available SEPs. As a result of this operation the following events will be generated: @arg A2DP_EVT_SEP_INFO_RECEIVED: this event is generated for every SEP received from the remote device. the p evt_param.sep_info_received contains SEP information. @arg A2DP_EVT_DISCOVER_COMPLETED: this event is generated after the last A2DP_EVT_SEP_INFO_RECEIVED if the remote accepted the request and the p evt_param.discover_completed.err_code == AVDTP_ERROR_SUCCESS. if the remote rejected the request the p evt_param.discover_completed.err_code == the error code sent by the remote.

param mgr A2DP manager. param remote_addr The address of a remote device.

return li c TRUE if discover request has been sent. li c FALSE otherwise. No events will be generated.

bt_a2dp_find_codec Macro

File

a2dp.h

C

```
#define bt_a2dp_find_codec(mgr, codec_type) bt_avdtp_find_codec(mgr->avdtp_mgr, codec_type)
```

Description

brief Find a codec ingroup a2dp

details A2DP in theory can support any type of codec. Each codec uses its own format for exchanging capabilities and configuration information. In order to make our implementation do not care about these formats we use a simple way of telling AVDTP how to parse and serialize codec's configuration. The consumer of A2DP has to register a callback function (one per codec type) for each codec it wishes to support. That callback has to perform two function. The first one is to read the configuration received from the remote device and store it in a structure defined by the consumer. The second one is to serialize the data from a structure to a format (in case of standard A2DP codecs the format is defined in A2DP specification, vendor specific codecs can define their own formats) suitable for sending as a part of a AVDTP request. This function returns a pointer to a structure that holds a pointer to a codec's callback function.

param mgr A2DP manager. param codec_type Codec type. The codec_type can be one of the following values: @arg AVDTP_CODEC_TYPE_SBC: SBC @arg AVDTP_CODEC_TYPE_MPEG1_2_AUDIO: MPEG-1,2 (used in MP3 files) @arg AVDTP_CODEC_TYPE_MPEG2_4_AAC: MPEG-2,4 AAC (used in Apple products) @arg AVDTP_CODEC_TYPE_ATRAC: ATRAC (used in Sony products) @arg AVDTP_CODEC_TYPE_NON_A2DP: Non-A2DP Codec

return li c TRUE if the function succeeds. li c FALSE otherwise. The function fails if a callback for a codec type specified in the p codec parameter li has not been previously registered with ::bt_avdtp_register_codec.

bt_a2dp_get_capabilities Macro

File

a2dp.h

C

```
#define bt_a2dp_get_capabilities(mgr, remote_addr, seid_acp) bt_avdtp_get_capabilities(mgr->avdtp_mgr,
remote_addr, seid_acp)
```

Description

brief Get remote SEP capabilities. ingroup a2dp

details This function asks the remote device to send capabilities of a SEP specified by the p seid_acp. As a result of this operation the following events will be generated: @arg A2DP_EVT_SEP_CAPABILITIES_RECEIVED: this event is generated if the remote device accepted the request. the p evt_param.sep_capabilities_received contains SEP capabilities. @arg A2DP_EVT_GET_SEP_CAPABILITIES_COMPLETED: this event is generated right after A2DP_EVT_SEP_CAPABILITIES_RECEIVED if the remote accepted the request the p evt_param.get_sep_capabilities_completed.err_code == AVDTP_ERROR_SUCCESS. if the remote rejected the request the p evt_param.get_sep_capabilities_completed.err_code == the error code sent by the remote.

param mgr A2DP manager. param remote_addr The address of a remote device. param seid_acp The ID of a remote SEP.

return li c TRUE if discover request has been sent. li c FALSE otherwise. No events will be generated.

bt_a2dp_get_hci_connection Macro

File

a2dp.h

C

```
#define bt_a2dp_get_hci_connection(mgr, strm_handle) bt_avdtp_get_hci_connection(mgr->avdtp_mgr,
strm_handle)
```

Description

brief Get HCI connection for a stream ingroup a2dp

details This function returns a pointer to a structure that describes an HCI connection a stream is open on. The return value can be used to call various function from the HCI layer. For example, if an app wants to force disconnection from a remote device it can call ::bt_hci_disconnect.

param mgr A2DP manager. param strm_handle Stream handle.

return li c Pointer to a structure that describes an HCI connection if the function succeeds. li c NULL otherwise. The function fails only if a stream specified by the p strm_handle parameter li does not exist or there is no HCI connection between local and remote devices associated with the stream.

note This function has not been implemented.

bt_a2dp_get_stream_codec_config Macro

File

a2dp.h

C

```
#define bt_a2dp_get_stream_codec_config(mgr, strm_handle) bt_avdtp_get_stream_codec_config(mgr->avdtp_mgr,
strm_handle)
```

Description

brief Get the configuration of the codec currently used with the stream. ingroup a2dp

details This function returns a pointer to a structure that contains configuration of the codec currently used with the stream. The structure returned depends on the codec. The dotstack defines structures only for SBC, MPEG-1,2 and MPEG-2,4 AAC codecs: @arg SBC: bt_a2dp_sbc_config_t (defined in a2dp_sbc_codec.h) @arg MPEG-1,2: bt_a2dp_mpeg_config_t (defined in a2dp_mpeg_codec.h) @arg MPEG-2,4 AAC: bt_a2dp_aac_config_t (defined in a2dp_aac_codec.h)

param mgr A2DP manager. param strm_handle Stream handle.

return li The codec's configuration if strm_handle specifies a valid stream and the stream is in one of the following state:

AVDTP_STREAM_STATE_CONFIGURED AVDTP_STREAM_STATE_OPEN AVDTP_STREAM_STATE_STREAMING

li NULL otherwise.

bt_a2dp_get_stream_codec_type Macro

File

a2dp.h

C

```
#define bt_a2dp_get_stream_codec_type(mgr, strm_handle) bt_avdtp_get_stream_codec_type(mgr->avdtp_mgr,
strm_handle)
```

Description

brief Get the type of the codec currently used with the stream. ingroup a2dp

details This function returns the type of the codec currently used with the stream.

param mgr A2DP manager. param strm_handle Stream handle.

return @arg The type of the codec if strm_handle specifies a valid stream and the stream is in one of the following states:

AVDTP_STREAM_STATE_CONFIGURED AVDTP_STREAM_STATE_OPEN AVDTP_STREAM_STATE_STREAMING

@arg The result will be one of the following values:

AVDTP_CODEC_TYPE_SBC: SBC AVDTP_CODEC_TYPE_MPEG1_2_AUDIO: MPEG-1,2 (used in MP3 files) AVDTP_CODEC_TYPE_MPEG2_4_AAC: MPEG-2,4 AAC (used in Apple products) AVDTP_CODEC_TYPE_ATRAC: ATRAC (used in Sony products) AVDTP_CODEC_TYPE_NON_A2DP: Non-A2DP Codec

@arg 0xFF otherwise.

bt_a2dp_get_stream_local_sep_id Macro

File

a2dp.h

C

```
#define bt_a2dp_get_stream_local_sep_id(mgr, strm_handle) bt_avdtp_get_stream_local_sep_id(mgr->avdtp_mgr,
strm_handle)
```

Description

brief Get stream's local SEP ID. ingroup a2dp

details This function returns the ID of the local SEP associated with the stream.

param mgr A2DP manager. param strm_handle Stream handle.

return li The ID of the local SEP if strm_handle specifies a valid stream. li 0 otherwise.

bt_a2dp_get_stream_remote_address Macro

File

a2dp.h

C

```
#define bt_a2dp_get_stream_remote_address(mgr, strm_handle)
bt_avdtp_get_stream_remote_address(mgr->avdtp_mgr, strm_handle)
```

Description

brief Get stream's remote BT address. ingroup a2dp

details This function returns the address of the remote device associated with the stream.

param mgr A2DP manager. param strm_handle Stream handle.

return li The address of the remote device if strm_handle specifies a valid stream. li NULL otherwise.

bt_a2dp_get_stream_remote_sep_id Macro

File

a2dp.h

C

```
#define bt_a2dp_get_stream_remote_sep_id(mgr, strm_handle)
bt_avdtp_get_stream_remote_sep_id(mgr->avdtp_mgr, strm_handle)
```

Description

brief Get stream's remote SEP ID. ingroup a2dp

details This function returns the ID of the remote SEP associated with the stream.

param mgr A2DP manager. param strm_handle Stream handle.

return li The ID of the remote SEP if strm_handle specifies a valid stream. li 0 otherwise.

bt_a2dp_get_stream_state Macro

File

a2dp.h

C

```
#define bt_a2dp_get_stream_state(mgr, strm_handle) bt_avdtp_get_stream_state(mgr->avdtp_mgr, strm_handle)
```

Description

brief Get local stream state. ingroup a2dp

details This function returns local state of a stream specified by the p strm_handle. No request is sent to the remote party.

param mgr A2DP manager. param strm_handle Stream handle.

return The state of the stream. The result will be one of the following values: @arg AVDTP_STREAM_STATE_IDLE: The stream is idle. This can mean two things. The stream specified by p strm_handle does not exist or the stream is closed. @arg AVDTP_STREAM_OPENING_TRANSPORT_CHANNELS: The stream is opening transport channels. @arg AVDTP_STREAM_CLOSING_TRANSPORT_CHANNELS: The stream is closing transport channels. @arg AVDTP_STREAM_STATE_CONFIGURED: The stream has been configured. @arg AVDTP_STREAM_STATE_OPEN: The stream has been opened. @arg AVDTP_STREAM_STATE_STREAMING: The stream has been started. Depending on the local SEP type (source or sink) it means that the stream is can send or receive media packets. @arg AVDTP_STREAM_STATE_CLOSING: The stream is closing. This means that all transport channels associated with the stream are being closed. After they have been closed the stream goes to AVDTP_STREAM_STATE_IDLE state. @arg AVDTP_STREAM_STATE_ABORTING: The stream is aborting. This means that all transport channels associated with the stream are being closed. After they have been closed the stream goes to AVDTP_STREAM_STATE_IDLE state.

bt_a2dp_listen Macro

File

a2dp.h

C

```
#define bt_a2dp_listen(mgr, strm_handle, sep_id) bt_avdtp_listen(mgr->avdtp_mgr, strm_handle, sep_id)
```

Description

brief Listen for incoming connections. ingroup a2dp

details This function tells a stream that it can use a particular SEP to accept incoming requests to open it. The SEP can be associated with multiple streams but used with only one. The stream has to be closed before the SEP can be used with another stream. For outgoing connections this is not needed. Any SEP can be used with any stream given that the SEP is not already in use by another stream.

param mgr A2DP manager. param strm_handle Stream handle. param sep_id Local SEP ID.

return li c TRUE if the function succeeds. li c FALSE otherwise.

bt_a2dp_reconfigure_stream Macro

File

a2dp.h

C

```
#define bt_a2dp_reconfigure_stream(mgr, strm_handle, caps) bt_avdtp_reconfigure_stream(mgr->avdtp_mgr,
strm_handle, caps)
```

Description

brief Reconfigure stream. ingroup a2dp

details This function tries to change the stream's configuration. For this function to succeed the stream has to be open. As a result of this operation the A2DP_EVT_STREAM_RECONFIGURE_COMPLETED event will be generated. If reconfiguration was a success the p evt_param.stream_reconfigure_completed.err_code == AVDTP_ERROR_SUCCESS. Otherwise the p evt_param.stream_reconfigure_completed.err_code == the error code sent by the remote.

param mgr A2DP manager. param strm_handle Stream handle. param caps New stream configuration.

return li c TRUE if the function succeeds, i.e. the actual request has been sent to the remote party. li c FALSE otherwise. No events will be generated.

bt_a2dp_register_sink Macro

File

a2dp.h

C

```
#define bt_a2dp_register_sink(mgr, caps) bt_avdtp_register_sep(mgr->avdtp_mgr, AVDTP_SEP_TYPE_SINK, caps)
```

Description

brief Register a Sink SEP with the local A2DP manager. ingroup a2dp

details This function is used to add a sink SEP to a list of SEPs supported by the local A2DP entity.

param mgr A2DP manager. param caps The capabilities of a SEP.

return li c ID of a SEP if the function succeeds. li c FALSE otherwise.

bt_a2dp_register_source Macro

File

a2dp.h

C

```
#define bt_a2dp_register_source(mgr, caps) bt_avdtp_register_sep(mgr->avdtp_mgr, AVDTP_SEP_TYPE_SOURCE,
caps)
```

Description

brief Register a Source SEP with the local A2DP manager. ingroup a2dp

details This function is used to add a source SEP to a list of SEPs supported by the local A2DP entity.

param mgr A2DP manager. param caps The capabilities of a SEP.

return li c ID of a SEP if the function succeeds. li c FALSE otherwise.

bt_a2dp_remove_media_rx_buffer Macro

File

a2dp.h

C

```
#define bt_a2dp_remove_media_rx_buffer(mgr, strm_handle, buffer)
bt_avdtp_remove_media_rx_buffer(mgr->avdtp_mgr, strm_handle, buffer)
```

Description

brief Remove a media packet buffer from a receive queue ingroup a2dp

details The consumer of A2DP is responsible for allocating and supplying A2DP with buffers used to store received packets. A2DP itself only has a queue for storing pointers to buffers supplied by the consumer. When a packet comes in A2DP finds the first buffer large enough to hold the received packet, copies the packet to the buffer and generates a A2DP_EVT_MEDIA_PACKET_RECEIVED event. The consumer then has to process the data in the buffer and return it back to the queue. If there is no buffers in the queue or none of the buffers is large enough the received packets is dropped. Each buffer has a field (data_len) that holds the length of the received buffer. This field is never 0 if the buffer contains a packet. If a channel closed regardless of what has caused that and there are still buffers in the queue A2DP generates a A2DP_EVT_MEDIA_PACKET_RECEIVED event for each buffer and sets the data_len to 0. This is to inform the A2DP consumer that the buffer has not been used and can be, for example, deallocated. This function removes a buffer from the receive queue.

param mgr A2DP manager. param strm_handle Stream handle. param buffer Pointer to a structure that holds the buffer and its parameters.

return li c TRUE if the function succeeds. li c FALSE otherwise. The function fails only if a stream specified by the p strm_handle parameter li does not exist. The stream can be in any state to call this function.

bt_a2dp_remove_media_tx_buffer Macro

File

a2dp.h

C

```
#define bt_a2dp_remove_media_tx_buffer(mgr, strm_handle, buffer)
bt_avdtp_remove_media_tx_buffer(mgr->avdtp_mgr, strm_handle, buffer)
```

Description

brief Remove a media packet buffer from a send queue ingroup a2dp

details When the consumer of A2DP wants to send a packet to a remote device it calls bt_avdtp_add_media_tx_buffer function. The function adds the packet to a queue and tells A2DP that it has something to send. The packet will be send as soon as the stream goes to A2DP_STREAM_STATE_STREAMING state. The consumer has a chance to remove a packet from the queue before it has been sent to a remote device by calling ::bt_a2dp_remove_media_tx_buffer.

param mgr A2DP manager. param strm_handle Stream handle. param buffer Pointer to a structure that holds the buffer and its parameters.

return li c TRUE if the function succeeds. li c FALSE otherwise. The function fails only if a stream specified by the p strm_handle parameter li does not exist. The stream can be in any state to call this function.

note This function has not been implemented.

bt_a2dp_start_stream Macro

File

a2dp.h

C

```
#define bt_a2dp_start_stream(mgr, strm_handle) bt_avdtp_start_stream(mgr->avdtp_mgr, strm_handle)
```

Description

brief Start a stream. ingroup a2dp

details This function tries to start a stream by sending a request to the remote party. The stream has to be in AVDTP_STREAM_STATE_OPEN state. The stream goes to this state as a result of successful configuration or suspension (both can be initiated by either party). As a result of this operation the A2DP_EVT_START_STREAM_COMPLETED event will be generated. If the stream has been open the p evt_param.start_stream_requested.err_code == AVDTP_ERROR_SUCCESS. Otherwise, if the remote device for any reason cannot or does not wish to start the stream, the p evt_param.start_stream_requested.err_code == the error code sent by the remote.

param mgr A2DP manager. param strm_handle Stream handle.

return li c TRUE if the function succeeds, i.e. the actual request has been sent to the remote party. li c FALSE otherwise. No events will be generated.

bt_a2dp_suspend_stream Macro

File

a2dp.h

C

```
#define bt_a2dp_suspend_stream(mgr, strm_handle) bt_avdtp_suspend_stream(mgr->avdtp_mgr, strm_handle)
```

Description

brief Suspend a stream. ingroup a2dp

details This function tries to suspend a stream by sending a request to the remote party. The stream has to be in AVDTP_STREAM_STATE_STREAMING state. As a result of this operation the A2DP_EVT_SUSPEND_STREAM_COMPLETED event will be generated. If the stream has been suspended the p evt_param.bt_avdtp_evt_suspend_stream_requested_t.err_code == AVDTP_ERROR_SUCCESS. Otherwise, if the remote device for any reason cannot or does not wish to suspend the stream, the p evt_param.bt_avdtp_evt_suspend_stream_requested_t.err_code == the error code sent by the remote.

param mgr A2DP manager. param strm_handle Stream handle.

return li c TRUE if the function succeeds, i.e. the actual request has been sent to the remote party. li c FALSE otherwise. No events will be generated.

MPEG_BITRATE_0000 Macro

File

a2dp_codec_mpeg.h

C

```
#define MPEG_BITRATE_0000 0x0080
```

Description

This is macro MPEG_BITRATE_0000.

MPEG_BITRATE_0001 Macro

File

a2dp_codec_mpeg.h

C

```
#define MPEG_BITRATE_0001 0x0100
```

Description

This is macro MPEG_BITRATE_0001.

MPEG_BITRATE_0010 Macro

File

a2dp_codec_mpeg.h

C

```
#define MPEG_BITRATE_0010 0x0200
```

Description

This is macro MPEG_BITRATE_0010.

MPEG_BITRATE_0011 Macro

File

a2dp_codec_mpeg.h

C

```
#define MPEG_BITRATE_0011 0x0400
```

Description

This is macro MPEG_BITRATE_0011.

MPEG_BITRATE_0100 Macro

File

a2dp_codec_mpeg.h

C

```
#define MPEG_BITRATE_0100 0x0800
```

Description

This is macro MPEG_BITRATE_0100.

MPEG_BITRATE_0101 Macro

File

a2dp_codec_mpeg.h

C

```
#define MPEG_BITRATE_0101 0x1000
```

Description

This is macro MPEG_BITRATE_0101.

MPEG_BITRATE_0110 Macro

File

a2dp_codec_mpeg.h

C

```
#define MPEG_BITRATE_0110 0x2000
```

Description

This is macro MPEG_BITRATE_0110.

MPEG_BITRATE_0111 Macro

File

a2dp_codec_mpeg.h

C

```
#define MPEG_BITRATE_0111 0x4000
```

Description

This is macro MPEG_BITRATE_0111.

MPEG_BITRATE_1000 Macro

File

a2dp_codec_mpeg.h

C

```
#define MPEG_BITRATE_1000 0x0001
```

Description

This is macro MPEG_BITRATE_1000.

MPEG_BITRATE_1001 Macro

File

a2dp_codec_mpeg.h

C

```
#define MPEG_BITRATE_1001 0x0002
```

Description

This is macro MPEG_BITRATE_1001.

MPEG_BITRATE_1010 Macro

File

a2dp_codec_mpeg.h

C

```
#define MPEG_BITRATE_1010 0x0004
```

Description

This is macro MPEG_BITRATE_1010.

MPEG_BITRATE_1011 Macro

File

a2dp_codec_mpeg.h

C

```
#define MPEG_BITRATE_1011 0x0008
```

Description

This is macro MPEG_BITRATE_1011.

MPEG_BITRATE_1100 Macro

File

a2dp_codec_mpeg.h

C

```
#define MPEG_BITRATE_1100 0x0010
```

Description

This is macro MPEG_BITRATE_1100.

MPEG_BITRATE_1101 Macro

File

a2dp_codec_mpeg.h

C

```
#define MPEG_BITRATE_1101 0x0020
```

Description

This is macro MPEG_BITRATE_1101.

MPEG_BITRATE_1110 Macro

File

a2dp_codec_mpeg.h

C

```
#define MPEG_BITRATE_1110 0x0040
```

Description

This is macro MPEG_BITRATE_1110.

MPEG_BITRATE_ALL Macro

File

a2dp_codec_mpeg.h

C

```
#define MPEG_BITRATE_ALL 0x7FFF
```

Description

This is macro MPEG_BITRATE_ALL.

MPEG_CHANNEL_MODE_ALL Macro

File

a2dp_codec_mpeg.h

C

```
#define MPEG_CHANNEL_MODE_ALL 15
```

Description

This is macro MPEG_CHANNEL_MODE_ALL.

MPEG_CHANNEL_MODE_DUAL_CHANNEL Macro

File

a2dp_codec_mpeg.h

C

```
#define MPEG_CHANNEL_MODE_DUAL_CHANNEL 4
```

Description

This is macro MPEG_CHANNEL_MODE_DUAL_CHANNEL.

MPEG_CHANNEL_MODE_JOINT_STEREO Macro

File

a2dp_codec_mpeg.h

C

```
#define MPEG_CHANNEL_MODE_JOINT_STEREO 1
```

Description

This is macro MPEG_CHANNEL_MODE_JOINT_STEREO.

MPEG_CHANNEL_MODE_MONO Macro

File

a2dp_codec_mpeg.h

C

```
#define MPEG_CHANNEL_MODE_MONO 8
```

Description

This is macro MPEG_CHANNEL_MODE_MONO.

MPEG_CHANNEL_MODE_STEREO Macro

File

a2dp_codec_mpeg.h

C

```
#define MPEG_CHANNEL_MODE_STEREO 2
```

Description

This is macro MPEG_CHANNEL_MODE_STEREO.

MPEG_CRC_PROTECTION_NOT_SUPPORTED Macro

File

a2dp_codec_mpeg.h

C

```
#define MPEG_CRC_PROTECTION_NOT_SUPPORTED 0
```

Description

This is macro MPEG_CRC_PROTECTION_NOT_SUPPORTED.

MPEG_CRC_PROTECTION_SUPPORTED Macro

File

a2dp_codec_mpeg.h

C

```
#define MPEG_CRC_PROTECTION_SUPPORTED 1
```

Description

This is macro MPEG_CRC_PROTECTION_SUPPORTED.

MPEG_LAYER_1 Macro

File

a2dp_codec_mpeg.h

C

```
#define MPEG_LAYER_1 0x04
```

Description

This is macro MPEG_LAYER_1.

MPEG_LAYER_2 Macro

File

a2dp_codec_mpeg.h

C

```
#define MPEG_LAYER_2 0x02
```

Description

This is macro MPEG_LAYER_2.

MPEG_LAYER_3 Macro

File

a2dp_codec_mpeg.h

C

```
#define MPEG_LAYER_3 0x01
```

Description

This is macro MPEG_LAYER_3.

MPEG_LAYER_ALL Macro

File

a2dp_codec_mpeg.h

C

```
#define MPEG_LAYER_ALL 0x07
```

Description

This is macro MPEG_LAYER_ALL.

MPEG_MPF_1 Macro

File

a2dp_codec_mpeg.h

C

```
#define MPEG_MPF_1 0
```

Description

This is macro MPEG_MPF_1.

MPEG_MPF_2 Macro

File

a2dp_codec_mpeg.h

C

```
#define MPEG_MPF_2 1
```

Description

This is macro MPEG_MPF_2.

MPEG_SAMPLING_FREQUENCY_16000 Macro

File

a2dp_codec_mpeg.h

C

```
#define MPEG_SAMPLING_FREQUENCY_16000 0x20
```

Description

This is macro MPEG_SAMPLING_FREQUENCY_16000.

MPEG_SAMPLING_FREQUENCY_22050 Macro

File

a2dp_codec_mpeg.h

C

```
#define MPEG_SAMPLING_FREQUENCY_22050 0x10
```

Description

This is macro MPEG_SAMPLING_FREQUENCY_22050.

MPEG_SAMPLING_FREQUENCY_24000 Macro

File

a2dp_codec_mpeg.h

C

```
#define MPEG_SAMPLING_FREQUENCY_24000 0x08
```

Description

This is macro MPEG_SAMPLING_FREQUENCY_24000.

MPEG_SAMPLING_FREQUENCY_32000 Macro

File

a2dp_codec_mpeg.h

C

```
#define MPEG_SAMPLING_FREQUENCY_32000 0x04
```

Description

This is macro MPEG_SAMPLING_FREQUENCY_32000.

MPEG_SAMPLING_FREQUENCY_44100 Macro

File

a2dp_codec_mpeg.h

C

```
#define MPEG_SAMPLING_FREQUENCY_44100 0x02
```

Description

This is macro MPEG_SAMPLING_FREQUENCY_44100.

MPEG_SAMPLING_FREQUENCY_48000 Macro

File

a2dp_codec_mpeg.h

C

```
#define MPEG_SAMPLING_FREQUENCY_48000 0x01
```

Description

This is macro MPEG_SAMPLING_FREQUENCY_48000.

MPEG_SAMPLING_FREQUENCY_ALL Macro

File

a2dp_codec_mpeg.h

C

```
#define MPEG_SAMPLING_FREQUENCY_ALL 0x3F
```

Description

This is macro MPEG_SAMPLING_FREQUENCY_ALL.

MPEG_VBR_NOT_SUPPORTED Macro

File

a2dp_codec_mpeg.h

C

```
#define MPEG_VBR_NOT_SUPPORTED 0
```

Description

This is macro MPEG_VBR_NOT_SUPPORTED.

MPEG_VBR_SUPPORTED Macro

File

a2dp_codec_mpeg.h

C

```
#define MPEG_VBR_SUPPORTED 1
```

Description

This is macro MPEG_VBR_SUPPORTED.

SBC_ALLOCATION_METHOD_ALL Macro

File

a2dp_codec_sbc.h

C

```
#define SBC_ALLOCATION_METHOD_ALL 3
```

Description

This is macro SBC_ALLOCATION_METHOD_ALL.

SBC_ALLOCATION_METHOD_LOUDNESS Macro

File

a2dp_codec_sbc.h

C

```
#define SBC_ALLOCATION_METHOD_LOUDNESS 1
```

Description

This is macro SBC_ALLOCATION_METHOD_LOUDNESS.

SBC_ALLOCATION_METHOD_SNR Macro

File

a2dp_codec_sbc.h

C

```
#define SBC_ALLOCATION_METHOD_SNR 2
```

Description

This is macro SBC_ALLOCATION_METHOD_SNR.

SBC_BLOCK_LENGTH_12 Macro

File

a2dp_codec_sbc.h

C

```
#define SBC_BLOCK_LENGTH_12 2
```

Description

This is macro SBC_BLOCK_LENGTH_12.

SBC_BLOCK_LENGTH_16 Macro

File

a2dp_codec_sbc.h

C

```
#define SBC_BLOCK_LENGTH_16 1
```

Description

This is macro SBC_BLOCK_LENGTH_16.

SBC_BLOCK_LENGTH_4 Macro

File

a2dp_codec_sbc.h

C

```
#define SBC_BLOCK_LENGTH_4 8
```

Description

This is macro SBC_BLOCK_LENGTH_4.

SBC_BLOCK_LENGTH_8 Macro

File

a2dp_codec_sbc.h

C

```
#define SBC_BLOCK_LENGTH_8 4
```

Description

This is macro SBC_BLOCK_LENGTH_8.

SBC_BLOCK_LENGTH_ALL Macro

File

a2dp_codec_sbc.h

C

```
#define SBC_BLOCK_LENGTH_ALL 15
```

Description

This is macro SBC_BLOCK_LENGTH_ALL.

SBC_CHANNEL_MODE_ALL Macro

File

a2dp_codec_sbc.h

C

```
#define SBC_CHANNEL_MODE_ALL 15
```

Description

This is macro SBC_CHANNEL_MODE_ALL.

SBC_CHANNEL_MODE_DUAL_CHANNEL Macro

File

a2dp_codec_sbc.h

C

```
#define SBC_CHANNEL_MODE_DUAL_CHANNEL 4
```

Description

This is macro SBC_CHANNEL_MODE_DUAL_CHANNEL.

SBC_CHANNEL_MODE_JOINT_STEREO Macro

File

a2dp_codec_sbc.h

C

```
#define SBC_CHANNEL_MODE_JOINT_STEREO 1
```

Description

This is macro SBC_CHANNEL_MODE_JOINT_STEREO.

SBC_CHANNEL_MODE_MONO Macro

File

a2dp_codec_sbc.h

C

```
#define SBC_CHANNEL_MODE_MONO 8
```

Description

This is macro SBC_CHANNEL_MODE_MONO.

SBC_CHANNEL_MODE_STEREO Macro

File

a2dp_codec_sbc.h

C

```
#define SBC_CHANNEL_MODE_STEREO 2
```

Description

This is macro SBC_CHANNEL_MODE_STEREO.

SBC_SAMPLING_FREQUENCY_16000 Macro

File

a2dp_codec_sbc.h

C

```
#define SBC_SAMPLING_FREQUENCY_16000 8
```

Description

This is macro SBC_SAMPLING_FREQUENCY_16000.

SBC_SAMPLING_FREQUENCY_32000 Macro

File

a2dp_codec_sbc.h

C

```
#define SBC_SAMPLING_FREQUENCY_32000 4
```

Description

This is macro SBC_SAMPLING_FREQUENCY_32000.

SBC_SAMPLING_FREQUENCY_44100 Macro

File

a2dp_codec_sbc.h

C

```
#define SBC_SAMPLING_FREQUENCY_44100 2
```

Description

This is macro SBC_SAMPLING_FREQUENCY_44100.

SBC_SAMPLING_FREQUENCY_48000 Macro

File

a2dp_codec_sbc.h

C

```
#define SBC_SAMPLING_FREQUENCY_48000 1
```

Description

This is macro SBC_SAMPLING_FREQUENCY_48000.

SBC_SAMPLING_FREQUENCY_ALL Macro

File

a2dp_codec_sbc.h

C

```
#define SBC_SAMPLING_FREQUENCY_ALL 15
```

Description

This is macro SBC_SAMPLING_FREQUENCY_ALL.

SBC_SUBBANDS_4 Macro

File

a2dp_codec_sbc.h

C

```
#define SBC_SUBBANDS_4 2
```

Description

This is macro SBC_SUBBANDS_4.

SBC_SUBBANDS_8 Macro

File

a2dp_codec_sbc.h

C

```
#define SBC_SUBBANDS_8 1
```

Description

This is macro SBC_SUBBANDS_8.

SBC_SUBBANDS_ALL Macro

File

a2dp_codec_sbc.h

C

```
#define SBC_SUBBANDS_ALL 3
```

Description

This is macro SBC_SUBBANDS_ALL.

bt_a2dp_get_all_capabilities Macro

File

a2dp.h

C

```
#define bt_a2dp_get_all_capabilities(mgr, remote_addr, seid_acp)
bt_avdtp_get_all_capabilities(mgr->avdtp_mgr, remote_addr, seid_acp)
```

Description

brief Get remote SEP capabilities. ingroup a2dp

details This function asks the remote device to send capabilities of a SEP specified by the p seid_acp. As a result of this operation the following events will be generated: @arg A2DP_EVT_SEP_CAPABILITIES_RECEIVED: this event is generated if the remote device accepted the request. the p evt_param.sep_capabilities_received contains SEP capabilities. @arg A2DP_EVT_GET_SEP_CAPABILITIES_COMPLETED: this event is generated right after A2DP_EVT_SEP_CAPABILITIES_RECEIVED if the remote accepted the request the p evt_param.get_sep_capabilities_completed.err_code == AVDTP_ERROR_SUCCESS. if the remote rejected the request the p evt_param.get_sep_capabilities_completed.err_code == the error code sent by the remote.

param mgr A2DP manager. param remote_addr The address of a remote device. param seid_acp The ID of a remote SEP.

return li c TRUE if discover request has been sent. li c FALSE otherwise. No events will be generated.

bt_a2dp_get_configuration Macro

File

a2dp.h

C

```
#define bt_a2dp_get_configuration(mgr, strm_handle) bt_avdtp_get_configuration(mgr->avdtp_mgr, strm_handle)
```

Description

This is macro bt_a2dp_get_configuration.

bt_a2dp_open_stream Macro

File

a2dp.h

C

```
#define bt_a2dp_open_stream(mgr, strm_handle) bt_avdtp_open_stream(mgr->avdtp_mgr, strm_handle)
```

Description

This is macro bt_a2dp_open_stream.

bt_a2dp_clear_media_tx_queue Macro

File

a2dp.h

C

```
#define bt_a2dp_clear_media_tx_queue(mgr, strm_handle) bt_avdtp_clear_media_tx_queue(mgr->avdtp_mgr,
strm_handle)
```

Description

brief Clear send queue ingroup a2dp

details When the consumer of A2DP wants to send a packet to a remote device it calls bt_avdtp_add_media_tx_buffer function. The function adds the packet to a queue and tells A2DP that it has something to send. The packet will be send as soon as the stream goes to A2DP_STREAM_STATE_STREAMING state. The consumer can remove all packets from the queue before they have been sent to a remote device by calling ::bt_a2dp_clear_media_tx_queue.

param mgr A2DP manager. param strm_handle Stream handle.

return li c TRUE if the function succeeds. li c FALSE otherwise. The function fails only if a stream specified by the p strm_handle parameter li does

not exist. The stream can be in any state to call this function.

bt_a2dp_get_stream_config Macro

File

a2dp.h

C

```
#define bt_a2dp_get_stream_config(mgr, strm_handle) bt_avdtp_get_stream_config(mgr->avdtp_mgr, strm_handle)
```

Description

brief Get stream's configuration. ingroup a2dp

details This function returns a pointer to a structure holding current configuration of the stream.

param mgr A2DP manager. param strm_handle Stream handle.

return li The stream's configuration if strm_handle specifies a valid stream and the stream is in one of the following state:

AVDTP_STREAM_STATE_CONFIGURED AVDTP_STREAM_STATE_OPEN AVDTP_STREAM_STATE_STREAMING

li NULL otherwise.

bt_a2dp_get_stream_direction Macro

File

a2dp.h

C

```
#define bt_a2dp_get_stream_direction(mgr, strm_handle) bt_avdtp_get_stream_direction(mgr->avdtp_mgr,
strm_handle)
```

Description

This is macro bt_a2dp_get_stream_direction.

bt_a2dp_report_delay Macro

File

a2dp.h

C

```
#define bt_a2dp_report_delay(mgr, strm_handle, delay) bt_avdtp_report_delay(mgr->avdtp_mgr, strm_handle,
delay)
```

Description

This is macro bt_a2dp_report_delay.

bt_a2dp_set_media_tx_queue_limit Macro

File

a2dp.h

C

```
#define bt_a2dp_set_media_tx_queue_limit(mgr, strm_handle, limit)
bt_avdtp_set_media_tx_queue_limit(mgr->avdtp_mgr, strm_handle, limit)
```

Description

brief Set limit on the send queue ingroup a2dp

details When the consumer of A2DP wants to send a packet to a remote device it calls bt_avdtp_add_media_tx_buffer function. The function adds the packet to a queue and tells A2DP that it has something to send. The packet will be send as soon as the stream goes to A2DP_STREAM_STATE_STREAMING state. By default the send queue can contain unlimited number of packets. The consumer can set a limit on how many packets are held in the queue. In this case when new packet is added to the queue and the length of the queue exceeds the set limit the first packet is removed from the queue. The removed packet is not send to the remote device.

param mgr A2DP manager. param strm_handle Stream handle. param limit Queue limit.

return li c TRUE if the function succeeds. li c FALSE otherwise. The function fails only if a stream specified by the p strm_handle parameter li does not exist. The stream can be in any state to call this function.

A2DP_EVT_SET_STREAM_CONFIGURATION Macro

File

a2dp.h

C

```
#define A2DP_EVT_SET_STREAM_CONFIGURATION AVDTP_EVT_SET_STREAM_CONFIGURATION
```

Description

This is macro A2DP_EVT_SET_STREAM_CONFIGURATION.

AVCTP Functions

_bt_avctp_allocate_channel Function

File

avctp_private.h

C

```
bt_avctp_channel_t* _bt_avctp_allocate_channel(bt_avctp_mgr_t* mgr);
```

Description

This is function _bt_avctp_allocate_channel.

_bt_avctp_allocate_message Function

File

avctp_private.h

C

```
bt_avctp_message_t* _bt_avctp_allocate_message(bt_byte msg_type);
```

Description

This is function _bt_avctp_allocate_message.

_bt_avctp_allocate_transport Function

File

avctp_private.h

C

```
bt_avctp_transport_t* _bt_avctp_allocate_transport(bt_avctp_mgr_t* mgr);
```

Description

This is function _bt_avctp_allocate_transport.

_bt_avctp_find_channel Function

File

avctp_private.h

C

```
bt_avctp_channel_t* _bt_avctp_find_channel(bt_avctp_mgr_t* mgr, bt_bdaddr_t* remote_addr, bt_uint
profile_id, bt_int psm);
```

Description

This is function _bt_avctp_find_channel.

_bt_avctp_find_transport Function

File

avctp_private.h

C

```
bt_avctp_transport_t* _bt_avctp_find_transport(bt_avctp_mgr_t* mgr, bt_l2cap_channel_t* l2cap_channel);
```

Description

This is function _bt_avctp_find_transport.

_bt_avctp_free_channel Function

File

avctp_private.h

C

```
void _bt_avctp_free_channel(bt_avctp_channel_t* channel);
```

Description

This is function _bt_avctp_free_channel.

_bt_avctp_free_message Function

File

avctp_private.h

C

```
void _bt_avctp_free_message(bt_avctp_message_t* message);
```

Description

This is function _bt_avctp_free_message.

_bt_avctp_free_transport Function

File

avctp_private.h

C

```
void _bt_avctp_free_transport(bt_avctp_transport_t* transport);
```

Description

This is function _bt_avctp_free_transport.

_bt_avctp_init_message_buffers Function

File

avctp_private.h

C

```
void _bt_avctp_init_message_buffers();
```

Description

This is function _bt_avctp_init_message_buffers.

_bt_avctp_init_signal Function

File

avctp_private.h

C

```
void _bt_avctp_init_signal();
```

Description

This is function _bt_avctp_init_signal.

_bt_avctp_l2cap_read_data_callback Function

File

avctp_private.h

C

```
void _bt_avctp_l2cap_read_data_callback(struct _bt_l2cap_channel_t * pch, bt_byte* pdata, bt_int len);
```

Description

This is function _bt_avctp_l2cap_read_data_callback.

_bt_avctp_packet_assembler Function

File

avctp_packet.h

C

```
bt_int _bt_avctp_packet_assembler(bt_packet_t* packet, bt_byte* buffer, bt_int buffer_len);
```

Description

This is function _bt_avctp_packet_assembler.

_bt_avctp_send_ipid Function

File

avctp_private.h

C

```
bt_bool _bt_avctp_send_ipid(bt_avctp_transport_t* transport, bt_byte tran_id);
```

Description

This is function _bt_avctp_send_ipid.

_bt_avctp_set_signal Function

File

avctp_private.h

C

```
void _bt_avctp_set_signal();
```

Description

This is function _bt_avctp_set_signal.

_bt_avrcp_init_cmd_buffers Function

File

avctp_private.h

C

```
void _bt_avrcp_init_cmd_buffers();
```

Description

This is function _bt_avrcp_init_cmd_buffers.

bt_avctp_cancel_command Function

File

avctp.h

C

```
void bt_avctp_cancel_command(bt_avctp_channel_t* channel, bt_byte tran_id);
```

Description

brief Cancel a command message. ingroup avctp

details If a message has not yet been sent to the remote device, it can be canceled (i.e. removed from send queue) by calling this function.

param channel AVCTP channel. param tran_id Transaction Id. This value is obtained by calling bt_avctp_send_command.

return li c TRUE if the function succeeds. li c FALSE otherwise. No events will be generated.

bt_avctp_cancel_listen Function

File

avctp.h

C

```
void bt_avctp_cancel_listen(bt_avctp_channel_t* channel);
```

Description

brief Cancel listening for incoming connections. ingroup avctp

details This function stops listening for incoming connections on the specified channel.

param channel AVCTP channel.

return li c TRUE if the function succeeds. li c FALSE otherwise.

bt_avctp_cancel_response Function

File

avctp.h

C

```
void bt_avctp_cancel_response(bt_avctp_channel_t* channel, bt_byte tran_id);
```

Description

brief Cancel a response message. ingroup avctp

details If a message has not yet been sent to the remote device, it can be canceled (i.e. removed from send queue) by calling this function.

param channel AVCTP channel. param tran_id Transaction Id. This value is obtained by calling bt_avctp_send_command.

return li c TRUE if the function succeeds. li c FALSE otherwise. No events will be generated.

bt_avctp_connect Function

File

avctp.h

C

```
bt_bool bt_avctp_connect(bt_avctp_channel_t* channel, bt_bdaddr_t* remote_address, bt_uint acl_config);
```

Description

brief Connect to a remote device. ingroup avctp

details This function establishes a connection to a remote device specified by the p remote_address. If connection cannot be initiated for some reason, for example, there is not enough resources, it returns FALSE and no events are generated. Otherwise the result of an attempt to connect to the remote device is reported via the AVCTP callback. The events generated will either be AVCTP_EVT_CHANNEL_CONNECTED or AVCTP_EVT_CONNECTION_FAILED.

param channel AVCTP channel. param remote_address The address of a remote device. param acl_config ACL link configuration. This can be a combination of the following values: li HCI_CONFIG_ENABLE_AUTHENTICATION li HCI_CONFIG_ENABLE_ENCRYPTION li HCI_CONFIG_BECOME_MASTER

return li c TRUE if connection establishment has been started. li c FALSE otherwise.

bt_avctp_create_channel Function

File

avctp.h

C

```
bt_avctp_channel_t* bt_avctp_create_channel(bt_avctp_mgr_t* mgr, bt_uint profile_id, bt_int psm, bt_byte
l2cap_mode);
```

Description

brief Allocate AVCTP channel ingroup avctp

details This function allocates a new incoming AVCTP channel. The channel is intended to be used to accept a connection from a remote device. There can be only one channel for each combination of c profile_id and c psm.

param mgr AVCTP manager. param profile_id Profile Id param psm The PSM on which the underlying L2CAP channel will listen and accept incoming connections. param l2cap_mode Underlying L2CAP channel mode. This currently can only be CMODE_BASIC.

return li A pointer to the new AVCTP channel if the function succeeds. li c NULL otherwise.

bt_avctp_create_outgoing_channel Function

File

avctp.h

C

```
bt_avctp_channel_t* bt_avctp_create_outgoing_channel(bt_avctp_mgr_t* mgr, bt_uint profile_id, bt_int psm,
bt_byte l2cap_mode);
```

Description

brief Allocate AVCTP channel ingroup avctp

details This function allocates a new outgoing AVCTP channel. The channel is intended to be used to create a connection to a remote device. There can be multiple channels with the same c profile_id and c psm.

param mgr AVCTP manager. param profile_id Profile Id param psm The PSM on which the underlying L2CAP channel will listen and accept incoming connections. param l2cap_mode Underlying L2CAP channel mode. This currently can only be CMODE_BASIC.

return li A pointer to the new AVCTP channel if the function succeeds. li c NULL otherwise.

bt_avctp_destroy_channel Function

File

avctp.h

C

```
bt_bool bt_avctp_destroy_channel(bt_avctp_channel_t* channel);
```

Description

brief Destroy AVCTP channel. ingroup avctp

details This function frees memory used by the channel. The channel has to exist and be in the "idle" state for this function to succeed. I.e. the channel has to be disconnected before this function can be called.

param channel AVCTP channel.

return li c TRUE if the function succeeds. li c FALSE otherwise.

bt_avctp_disconnect Function

File

avctp.h

C

```
bt_bool bt_avctp_disconnect(bt_avctp_channel_t* channel);
```

Description

brief Disconnect from a remote device. ingroup avctp

details This function closes a connection to a remote device.

param channel AVCTP channel.

return li c TRUE if disconnection has been started. li c FALSE otherwise. No events will be generated.

bt_avctp_get_channel_remote_address Function

File

avctp.h

C

```
bt_bdaddr_t* bt_avctp_get_channel_remote_address(bt_avctp_channel_t* channel);
```

Description

brief Get channel's remote BT address. ingroup avctp

details This function returns the address of the remote device associated with the channel.

param channel AVCTP channel.

return li The address of the remote device if channel is connected. li NULL otherwise.

bt_avctp_get_channel_state Function

File

avctp.h

C

```
bt_byte bt_avctp_get_channel_state(bt_avctp_channel_t* channel);
```

Description

brief Get channel state ingroup avctp

details This function return current state of the specified channel

param channel AVCTP channel.

return li AVCTP_CHANNEL_STATE_FREE li AVCTP_CHANNEL_STATE_IDLE li AVCTP_CHANNEL_STATE_CONNECTING li AVCTP_CHANNEL_STATE_CONNECTED li AVCTP_CHANNEL_STATE_DISCONNECTING

bt_avctp_get_hci_connection Function

File

avctp.h

C

```
bt_hci_conn_state_t* bt_avctp_get_hci_connection(bt_avctp_channel_t* channel);
```

Description

brief Get HCI connection for a channel ingroup avctp

details This function returns a pointer to a structure that describes an HCI connection a channel is open on. The return value can be used to call various function from the HCI layer. For example, if an app wants to force disconnection from a remote device it can call ::bt_hci_disconnect.

param channel AVCTP channel.

return li c Pointer to a structure that describes an HCI connection if the function succeeds. li c NULL otherwise. The function fails only if a channel specified by the p channel parameter li does not exist or there is no HCI connection between local and remote devices associated with the channel.

bt_avctp_get_mgr Function

File

avctp.h

C

```
bt_avctp_mgr_t* bt_avctp_get_mgr();
```

Description

brief Return a pointer to an instance of the AVCTP manager. ingroup avctp

details This function returns a pointer to an instance of the AVCTP manager. There is only one instance of the manager allocated by the stack.

bt_avctp_init Function

File

avctp.h

C

```
void bt_avctp_init();
```

Description

brief Initialize the AVCTP layer. ingroup avctp

details This function initializes the AVCTP layer of the stack. It must be called prior to any other AVCTP function can be called.

bt_avctp_listen Function

File

avctp.h

C

```
bt_bool bt_avctp_listen(bt_avctp_channel_t* channel);
```

Description

brief Listen for incoming connections. ingroup avctp

details This function enables incoming connections on the specified AVCTP channel.

param channel AVCTP channel.

return li c TRUE if the function succeeds. li c FALSE otherwise.

bt_avctp_send_command Function

File

avctp.h

C

```
bt_bool bt_avctp_send_command(bt_avctp_channel_t* channel, bt_byte* data, bt_uint data_len, bt_byte*
tran_id);
```

Description

brief Send a command message to a remote device. ingroup avctp

details This function sends a command message to a remote device.

param channel AVCTP channel. param data Message body. param data_len Message body length. param tran_id Pointer to a bt_byte where AVRCP will write transaction id assigned to the message.

return li c TRUE if the function succeeds. li c FALSE otherwise. No events will be generated.

bt_avctp_send_response Function

File

avctp.h

C

```
bt_bool bt_avctp_send_response(bt_avctp_channel_t* channel, bt_byte tran_id, bt_byte* data, bt_uint data_len);
```

Description

brief Send a response message to a remote device. ingroup avctp

details This function sends a response message to a remote device.

param channel AVCTP channel. param tran_id Transaction Id. This value is obtained by calling bt_avctp_send_command. param data Message body. param data_len Message body length.

return li c TRUE if the function succeeds. li c FALSE otherwise. No events will be generated.

bt_avctp_set_callback Function

File

avctp.h

C

```
void bt_avctp_set_callback(bt_avctp_mgr_t* mgr, bt_avctp_mgr_callback_fp callback, void* callback_param);
```

Description

brief Register a AVCTP application callback. ingroup avctp

details In order to be notified of various events a consumer of the AVCTP layer has to register a callback function. The stack will call this function whenever a new event has been generated passing the code of the event as the second parameter. The event can be one of the following values: @arg AVCTP_EVT_CHANNEL_CONNECTED Channel connected. @arg AVCTP_EVT_CHANNEL_DISCONNECTED Channel disconnected. @arg AVCTP_EVT_CONNECTION_FAILED Channel connection failed (generated only if connection has been initiated by the local device).

@arg AVCTP_EVT_COMMAND_RECEIVED Command received. @arg AVCTP_EVT_RESPONE_RECEIVED Response received. @arg AVCTP_EVT_COMMAND_SENT Command sent. @arg AVCTP_EVT_RESPONSE_SENT Response sent. @arg AVCTP_EVT_COMMAND_CANCELLED Command canceled. @arg AVCTP_EVT_RESPONSE_CANCELLED Response canceled.

param mgr AVCTP manager. param callback The callback function that will be called when the AVCTP generates an event. param callback_param A pointer to arbitrary data to be passed to the c callback callback.

bt_avctp_start Function

File

avctp.h

C

```
bt_bool bt_avctp_start(bt_avctp_mgr_t* mgr);
```

Description

This is function bt_avctp_start.

AVCTP Data Types and Constants

_bt_avctp_channel_t Structure

File

avctp.h

C

```
struct _bt_avctp_channel_t {
  bt_byte state;
  bt_byte flags;
  bt_avctp_transport_t* transport;
  bt_uint profile_id;
  bt_int psm;
  bt_byte l2cap_mode;
  bt_avctp_mgr_t* avctp_mgr;
};
```

Description

brief AVCTP channel description ingroup avctp

details This structure is used to hold information about an AVCTP channel.

_bt_avctp_message_t Structure

File

avctp.h

C

```
struct _bt_avctp_message_t {
  bt_avctp_message_t* next_message;
  bt_byte tran_id;
  bt_byte packet_type;
  bt_byte num_packets;
  bt_byte message_type;
  bt_byte ipid;
  bt_uint profile_id;
  bt_byte* data;
  bt_uint data_len;
};
```

Description

brief AVCTP message description ingroup avctp

details This structure is used to hold information about an AVCTP message.

_bt_avctp_mgr_t Structure

File

avctp.h

C

```
struct _bt_avctp_mgr_t {
  bt_byte state;
  bt_byte flags;
  bt_avctp_channel_t* channels;
  bt_avctp_transport_t* transports;
  bt_avctp_mgr_callback_fp callback;
  void* callback_param;
  bt_avctp_channel_t* opening_channel;
};
```

Members

Members	Description
bt_byte state;	< Manager state. This value can be one of the following values: < li AVCTP_MANAGER_STATE_IDLE < li AVCTP_MANAGER_STATE_CONNECTING

bt_byte flags;	< Additional manager state. This value can be a combination of the following values: < li AVDTP_MANAGER_FLAG_SENDING_MEDIA_PACKET
bt_avctp_channel_t* channels;	< List of available AVCTP channels.
bt_avctp_transport_t* transports;	< List of available AVCTP transports.
bt_avctp_mgr_callback_fp callback;	< Pointer to a function used to notify the AVCTP consumer about various events.
void* callback_param;	< Pointer to arbitrary data to be passed to the c callback.
bt_avctp_channel_t* opening_channel;	< Pointer to a channle being open.

Description

brief AVCTP manager. ingroup avctp

details A structure that glues all pieces together. There is only one instance of this structure allocated by dotstack. A pointer to the instance can be get with c bt_avctp_get_mgr().

_bt_avctp_transport_t Structure

File

avctp.h

C

```
struct _bt_avctp_transport_t {
  bt_byte state;
  bt_byte flags;
  bt_bdaddr_t remote_addr;
  bt_l2cap_channel_t* l2cap_channel;
  bt_avctp_packet_t packet;
  bt_byte* tx_buffer;
  bt_byte* rx_buffer;
  bt_uint rx_message_len;
  bt_byte ref_count;
  bt_queue_element_t* send_cq_head;
  bt_queue_element_t* ack_cq_head;
  bt_byte next_transaction_id;
  bt_avctp_message_t* tx_message;
  bt_uint tx_message_remaining_data_len;
  bt_avctp_message_t rx_message;
  bt_avctp_mgr_t* avctp_mgr;
};
```

Description

brief AVCTP transport description ingroup avctp

details This structure is used to hold information about an AVCTP transport.

bt_avctp_channel_t Type

File

avctp.h

C

```
typedef struct _bt_avctp_channel_t bt_avctp_channel_t;
```

Description

This is type bt_avctp_channel_t.

bt_avctp_event_t Union

File

avctp.h

C

```
typedef union _bt_avctp_event_u {
  bt_avctp_evt_channel_connected_t channel_connected;
  bt_avctp_evt_channel_disconnected_t channel_disconnected;
```

```
  bt_avctp_evt_connection_failed_t connection_failed;
  bt_avctp_evt_command_received_t command_received;
  bt_avctp_evt_response_received_t response_received;
  bt_avctp_evt_command_sent_t command_sent;
  bt_avctp_evt_response_sent_t response_sent;
  bt_avctp_evt_command_cancelled_t command_cancelled;
  bt_avctp_evt_response_cancelled_t response_cancelled;
} bt_avctp_event_t;
```

Members

Members	Description
bt_avctp_evt_channel_connected_t channel_connected;	< Valid if event is AVCTP_EVT_CHANNEL_CONNECTED
bt_avctp_evt_channel_disconnected_t channel_disconnected;	< Valid if event is AVCTP_EVT_CHANNEL_DISCONNECTED
bt_avctp_evt_connection_failed_t connection_failed;	< Valid if event is AVCTP_EVT_CONNECTION_FAILED
bt_avctp_evt_command_received_t command_received;	< Valid if event is AVCTP_EVT_COMMAND_RECEIVED
bt_avctp_evt_response_received_t response_received;	< Valid if event is AVCTP_EVT_RESPONE_RECEIVED
bt_avctp_evt_command_sent_t command_sent;	< Valid if event is AVCTP_EVT_COMMAND_SENT
bt_avctp_evt_response_sent_t response_sent;	< Valid if event is AVCTP_EVT_RESPONSE_SENT
bt_avctp_evt_command_cancelled_t command_cancelled;	< Valid if event is AVCTP_EVT_COMMAND_CANCELLED
bt_avctp_evt_response_cancelled_t response_cancelled;	< Valid if event is AVCTP_EVT_RESPONSE_CANCELLED

Description

brief Parameter to an application callback. ingroup avctp

details This union is used to pass event specific data to the AVCTP consumer. Which member of the union points to a valid structure depends on the event reported to the consumer. In general, each event has a corresponding member in the union.

bt_avctp_evt_channel_connected_t Structure

File

avctp.h

C

```
typedef struct _bt_avctp_evt_channel_connected_t {
  bt_avctp_channel_t* channel;
} bt_avctp_evt_channel_connected_t;
```

Members

Members	Description
bt_avctp_channel_t* channel;	< AVCTP channel

Description

brief Parameter to AVCTP_EVT_CHANNEL_CONNECTED event ingroup avctp

details A pointer to this structure is passed to the AVCTP application callback as a valid member of the bt_avctp_event_t union - bt_avctp_event_t::channel_connected - when a channel between two devices has been established.

bt_avctp_evt_channel_disconnected_t Structure

File

avctp.h

C

```
typedef struct _bt_avctp_evt_channel_disconnected_t {
  bt_avctp_channel_t* channel;
} bt_avctp_evt_channel_disconnected_t;
```

Members

Members	Description
bt_avctp_channel_t* channel;	< AVCTP channel

Description

brief Parameter to AVCTP_EVT_CHANNEL_DISCONNECTED event ingroup avctp

details A pointer to this structure is passed to the AVCTP application callback as a valid member of the bt_avctp_event_t union - bt_avctp_event_t::channel_disconnected - when a channel between two devices has been terminated.

bt_avctp_evt_command_cancelled_t Structure

File

avctp.h

C

```
typedef struct _bt_avctp_evt_command_cancelled_t {
  bt_avctp_channel_t* channel;
  bt_avctp_message_t* command;
} bt_avctp_evt_command_cancelled_t;
```

Members

Members	Description
bt_avctp_channel_t* channel;	< AVCTP channel
bt_avctp_message_t* command;	< AVCTP command message

Description

brief Parameter to AVCTP_EVT_COMMAND_CANCELLED event ingroup avctp

details A pointer to this structure is passed to the AVCTP application callback as a valid member of the bt_avctp_event_t union - bt_avctp_event_t::command_cancelled - when sending a command message has been canceled.

bt_avctp_evt_command_received_t Structure

File

avctp.h

C

```
typedef struct _bt_avctp_evt_command_received_t {
  bt_avctp_channel_t* channel;
  bt_avctp_message_t* command;
} bt_avctp_evt_command_received_t;
```

Members

Members	Description
bt_avctp_channel_t* channel;	< AVCTP channel
bt_avctp_message_t* command;	< AVCTP command message

Description

brief Parameter to AVCTP_EVT_COMMAND_RECEIVED event ingroup avctp

details A pointer to this structure is passed to the AVCTP application callback as a valid member of the bt_avctp_event_t union - bt_avctp_event_t::command_received - when a local device received a command message.

bt_avctp_evt_command_sent_t Structure

File

avctp.h

C

```
typedef struct _bt_avctp_evt_command_sent_t {
  bt_avctp_channel_t* channel;
```

```
  bt_avctp_message_t* command;
} bt_avctp_evt_command_sent_t;
```

Members

Members	Description
bt_avctp_channel_t* channel;	< AVCTP channel
bt_avctp_message_t* command;	< AVCTP command message

Description

brief Parameter to AVCTP_EVT_COMMAND_SENT event ingroup avctp

details A pointer to this structure is passed to the AVCTP application callback as a valid member of the bt_avctp_event_t union - bt_avctp_event_t::command_sent - when a local device finished sending a command message.

bt_avctp_evt_connection_failed_t Structure

File

avctp.h

C

```
typedef struct _bt_avctp_evt_connection_failed_t {
  bt_avctp_channel_t* channel;
} bt_avctp_evt_connection_failed_t;
```

Members

Members	Description
bt_avctp_channel_t* channel;	< AVCTP channel

Description

brief Parameter to AVCTP_EVT_CONNECTION_FAILED event ingroup avctp

details A pointer to this structure is passed to the AVCTP application callback as a valid member of the bt_avctp_event_t union - bt_avctp_event_t::connection_failed - when a channel between two devices could not be established.

bt_avctp_evt_response_cancelled_t Structure

File

avctp.h

C

```
typedef struct _bt_avctp_evt_response_cancelled_t {
  bt_avctp_channel_t* channel;
  bt_avctp_message_t* response;
} bt_avctp_evt_response_cancelled_t;
```

Members

Members	Description
bt_avctp_channel_t* channel;	< AVCTP channel
bt_avctp_message_t* response;	< AVCTP response message

Description

brief Parameter to AVCTP_EVT_RESPONSE_CANCELLED event ingroup avctp

details A pointer to this structure is passed to the AVCTP application callback as a valid member of the bt_avctp_event_t union - bt_avctp_event_t::response_cancelled - when sending a response message has been canceled.

bt_avctp_evt_response_received_t Structure

File

avctp.h

C

```
typedef struct _bt_avctp_evt_response_received_t {
```

```
  bt_avctp_channel_t* channel;
  bt_avctp_message_t* response;
} bt_avctp_evt_response_received_t;
```

Members

Members	Description
bt_avctp_channel_t* channel;	< AVCTP channel
bt_avctp_message_t* response;	< AVCTP response message

Description

brief Parameter to AVCTP_EVT_RESPONSE_RECEIVED event ingroup avctp

details A pointer to this structure is passed to the AVCTP application callback as a valid member of the bt_avctp_event_t union - bt_avctp_event_t::response_received - when a local device received a response message.

bt_avctp_evt_response_sent_t Structure

File

avctp.h

C

```
typedef struct _bt_avctp_evt_response_sent_t {
  bt_avctp_channel_t* channel;
  bt_avctp_message_t* response;
} bt_avctp_evt_response_sent_t;
```

Members

Members	Description
bt_avctp_channel_t* channel;	< AVCTP channel
bt_avctp_message_t* response;	< AVCTP response message

Description

brief Parameter to AVCTP_EVT_RESPONSE_SENT event ingroup avctp

details A pointer to this structure is passed to the AVCTP application callback as a valid member of the bt_avctp_event_t union - bt_avctp_event_t::response_sent - when a local device finished sending a response message.

bt_avctp_message_t Type

File

avctp.h

C

```
typedef struct _bt_avctp_message_t bt_avctp_message_t;
```

Description

This is type bt_avctp_message_t.

bt_avctp_mgr_callback_fp Type

File

avctp.h

C

```
typedef void (* bt_avctp_mgr_callback_fp)(struct _bt_avctp_mgr_t* mgr, bt_byte evt, bt_avctp_event_t*
evt_param, void* callback_param);
```

Description

brief AVCTP application callback. ingroup avctp

details In order to be notified of various events a consumer of the AVCTP layer has to register a callback function (done with bt_avctp_set_callback()). The stack will call that function whenever a new event has been generated.

param mgr AVCTP manager.

param evt AVCTP event. The event can be one of the following values: @arg AVCTP_EVT_CHANNEL_CONNECTED Channel connected. @arg AVCTP_EVT_CHANNEL_DISCONNECTED Channel disconnected. @arg AVCTP_EVT_CONNECTION_FAILED Channel connection failed (generated only if connection has been initiated by the local device).

@arg AVCTP_EVT_COMMAND_RECEIVED Command received. @arg AVCTP_EVT_RESPONE_RECEIVED Response received. @arg AVCTP_EVT_COMMAND_SENT Command sent. @arg AVCTP_EVT_RESPONSE_SENT Response sent. @arg AVCTP_EVT_COMMAND_CANCELLED Command canceled. @arg AVCTP_EVT_RESPONSE_CANCELLED Response canceled.

param evt_param Event parameter. Which member of the bt_avctp_event_t union is valid depends on the event: @arg AVCTP_EVT_CHANNEL_CONNECTED c bt_avctp_evt_channel_connected_t channel_connected @arg AVCTP_EVT_CHANNEL_DISCONNECTED c bt_avctp_evt_channel_disconnected_t channel_disconnected @arg AVCTP_EVT_CONNECTION_FAILED c bt_avctp_evt_connection_failed_t connection_failed @arg AVCTP_EVT_COMMAND_RECEIVED c bt_avctp_evt_command_received_t command_received @arg AVCTP_EVT_RESPONE_RECEIVED c bt_avctp_evt_response_received_t response_received @arg AVCTP_EVT_COMMAND_SENT c bt_avctp_evt_command_sent_t command_sent @arg AVCTP_EVT_RESPONSE_SENT c bt_avctp_evt_response_sent_t response_sent @arg AVCTP_EVT_COMMAND_CANCELLED c bt_avctp_evt_command_cancelled_t command_cancelled @arg AVCTP_EVT_RESPONSE_CANCELLED c bt_avctp_evt_response_cancelled_t response_cancelled

param callback_param A pointer to an arbitrary data set by a call to bt_avctp_set_callback.

bt_avctp_mgr_t Type

File

avctp.h

C

```
typedef struct _bt_avctp_mgr_t bt_avctp_mgr_t;
```

Description

This is type bt_avctp_mgr_t.

bt_avctp_packet_t Structure

File

avctp_packet.h

C

```
typedef struct _bt_avctp_packet_t {
  bt_packet_t header;
  bt_byte tran_id;
  bt_byte packet_type;
  bt_byte num_packets;
  bt_byte message_type;
  bt_byte ipid;
  bt_uint profile_id;
  bt_byte* data;
  bt_uint data_len;
  bt_int data_pos;
} bt_avctp_packet_t;
```

Description

This is type bt_avctp_packet_t.

bt_avctp_transport_t Type

File

avctp.h

C

```
typedef struct _bt_avctp_transport_t bt_avctp_transport_t;
```

Description

This is type bt_avctp_transport_t.

__AVCTP_H Macro

File

avctp.h

C

```
#define __AVCTP_H
```

Description

This is macro __AVCTP_H.

__AVCTP_PACKET_H Macro

File

avctp_packet.h

C

```
#define __AVCTP_PACKET_H
```

Description

This is macro __AVCTP_PACKET_H.

__AVCTP_PRIVATE_H Macro

File

avctp_private.h

C

```
#define __AVCTP_PRIVATE_H
```

Description

This is macro __AVCTP_PRIVATE_H.

AVCTP_CHANNEL_FLAG_LISTENING Macro

File

avctp.h

C

```
#define AVCTP_CHANNEL_FLAG_LISTENING 1
```

Description

This is macro AVCTP_CHANNEL_FLAG_LISTENING.

AVCTP_CHANNEL_FLAG_SENDING Macro

File

avctp.h

C

```
#define AVCTP_CHANNEL_FLAG_SENDING 2
```

Description

This is macro AVCTP_CHANNEL_FLAG_SENDING.

AVCTP_CHANNEL_STATE_CONNECTED Macro

File

avctp.h

C

```
#define AVCTP_CHANNEL_STATE_CONNECTED 3
```

Description

This is macro AVCTP_CHANNEL_STATE_CONNECTED.

AVCTP_CHANNEL_STATE_CONNECTING Macro

File

avctp.h

C

```
#define AVCTP_CHANNEL_STATE_CONNECTING 2
```

Description

This is macro AVCTP_CHANNEL_STATE_CONNECTING.

AVCTP_CHANNEL_STATE_DISCONNECTING Macro

File

avctp.h

C

```
#define AVCTP_CHANNEL_STATE_DISCONNECTING 4
```

Description

This is macro AVCTP_CHANNEL_STATE_DISCONNECTING.

AVCTP_CHANNEL_STATE_FREE Macro

File

avctp.h

C

```
#define AVCTP_CHANNEL_STATE_FREE 0
```

Description

This is macro AVCTP_CHANNEL_STATE_FREE.

AVCTP_CHANNEL_STATE_IDLE Macro

File

avctp.h

C

```
#define AVCTP_CHANNEL_STATE_IDLE 1
```

Description

This is macro AVCTP_CHANNEL_STATE_IDLE.

AVCTP_ERROR_BAD_STATE Macro

File

avrcp.h

C

```
#define AVCTP_ERROR_BAD_STATE 0x31
```

Description

This is macro AVCTP_ERROR_BAD_STATE.

AVCTP_ERROR_SUCCESS Macro

File

avrcp.h

C

```
#define AVCTP_ERROR_SUCCESS 0
```

Description

This is macro AVCTP_ERROR_SUCCESS.

AVCTP_EVT_CHANNEL_CONNECTED Macro

File

avctp.h

C

```
#define AVCTP_EVT_CHANNEL_CONNECTED 1   ///< This event is generated when a channel between two AVCTP
entities has been established.
```

Description

< This event is generated when a channel between two AVCTP entities has been established.

AVCTP_EVT_CHANNEL_DISCONNECTED Macro

File

avctp.h

C

```
#define AVCTP_EVT_CHANNEL_DISCONNECTED 2   ///< This event is generated when a channel between two AVCTP
entities has been terminated.
```

Description

< This event is generated when a channel between two AVCTP entities has been terminated.

AVCTP_EVT_COMMAND_CANCELLED Macro

File

avctp.h

C

```
#define AVCTP_EVT_COMMAND_CANCELLED 54  ///< This event is generated when a command has been canceled.
```

Description

< This event is generated when a command has been canceled.

AVCTP_EVT_COMMAND_RECEIVED Macro

File

avctp.h

C

```
#define AVCTP_EVT_COMMAND_RECEIVED 50  ///< This event is generated when a local device received a command.
```

Description

< This event is generated when a local device received a command.

AVCTP_EVT_COMMAND_SENT Macro

File

avctp.h

C

```
#define AVCTP_EVT_COMMAND_SENT 52  ///< This event is generated when a local device finished sending a
command.
```

Description

< This event is generated when a local device finished sending a command.

AVCTP_EVT_CONNECTION_FAILED Macro

File

avctp.h

C

```
#define AVCTP_EVT_CONNECTION_FAILED 3   ///< This event is generated when a local device failed to create a
channel between two AVCTP entities.
```

Description

< This event is generated when a local device failed to create a channel between two AVCTP entities.

AVCTP_EVT_NOTHING Macro

File

avctp.h

C

```
#define AVCTP_EVT_NOTHING 0
```

Description

addtogroup avrcp @{

@name Events

details The following is a list of events AVRCP layer generates and can report to the upper layer when it completes executing an operation initiated by either local or remote device.

AVCTP_EVT_RESPONE_RECEIVED Macro

File

avctp.h

C

```
#define AVCTP_EVT_RESPONE_RECEIVED 51  ///< This event is generated when a local device received a response.
```

Description

< This event is generated when a local device received a response.

AVCTP_EVT_RESPONSE_CANCELLED Macro

File

avctp.h

C

```
#define AVCTP_EVT_RESPONSE_CANCELLED 55  ///< This event is generated when a response has been canceled.
```

Description

< This event is generated when a response has been canceled.

AVCTP_EVT_RESPONSE_SENT Macro

File

avctp.h

C

```
#define AVCTP_EVT_RESPONSE_SENT 53  ///< This event is generated when a local device finished sending a
response.
```

Description

< This event is generated when a local device finished sending a response.

AVCTP_MANAGER_STATE_IDLE Macro

File

avctp.h

C

```
#define AVCTP_MANAGER_STATE_IDLE 0
```

Description

This is macro AVCTP_MANAGER_STATE_IDLE.

AVCTP_MESSAGE_PACKET_TYPE_CONTINUE Macro

File

avrcp.h

C

```
#define AVCTP_MESSAGE_PACKET_TYPE_CONTINUE 2
```

Description

This is macro AVCTP_MESSAGE_PACKET_TYPE_CONTINUE.

AVCTP_MESSAGE_PACKET_TYPE_END Macro

File

avrcp.h

C

```
#define AVCTP_MESSAGE_PACKET_TYPE_END 3
```

Description

This is macro AVCTP_MESSAGE_PACKET_TYPE_END.

AVCTP_MESSAGE_PACKET_TYPE_SINGLE Macro

File

avrcp.h

C

```
#define AVCTP_MESSAGE_PACKET_TYPE_SINGLE 0
```

Description

This is macro AVCTP_MESSAGE_PACKET_TYPE_SINGLE.

AVCTP_MESSAGE_PACKET_TYPE_START Macro

File

avrcp.h

C

```
#define AVCTP_MESSAGE_PACKET_TYPE_START 1
```

Description

This is macro AVCTP_MESSAGE_PACKET_TYPE_START.

AVCTP_MESSAGE_TYPE_COMMAND Macro

File

avrcp.h

C

```
#define AVCTP_MESSAGE_TYPE_COMMAND 0
```

Description

This is macro AVCTP_MESSAGE_TYPE_COMMAND.

AVCTP_MESSAGE_TYPE_RESPONSE Macro

File

avrcp.h

C

```
#define AVCTP_MESSAGE_TYPE_RESPONSE 1
```

Description

This is macro AVCTP_MESSAGE_TYPE_RESPONSE.

AVCTP_TRANSPORT_FLAG_RX_MESSAGE_STARTED Macro

File

avctp.h

C

```
#define AVCTP_TRANSPORT_FLAG_RX_MESSAGE_STARTED 4
```

Description

This is macro AVCTP_TRANSPORT_FLAG_RX_MESSAGE_STARTED.

AVCTP_TRANSPORT_FLAG_SENDING Macro

File

avctp.h

C

```
#define AVCTP_TRANSPORT_FLAG_SENDING 2
```

Description

This is macro AVCTP_TRANSPORT_FLAG_SENDING.

AVCTP_TRANSPORT_STATE_CONNECTED Macro

File

avctp.h

C

```
#define AVCTP_TRANSPORT_STATE_CONNECTED 3
```

Description

This is macro AVCTP_TRANSPORT_STATE_CONNECTED.

AVCTP_TRANSPORT_STATE_CONNECTING Macro

File

avctp.h

C

```
#define AVCTP_TRANSPORT_STATE_CONNECTING 2
```

Description

This is macro AVCTP_TRANSPORT_STATE_CONNECTING.

AVCTP_TRANSPORT_STATE_DISCONNECTING Macro

File

avctp.h

C

```
#define AVCTP_TRANSPORT_STATE_DISCONNECTING 4
```

Description

This is macro AVCTP_TRANSPORT_STATE_DISCONNECTING.

AVCTP_TRANSPORT_STATE_FREE Macro

File

avctp.h

C

```
#define AVCTP_TRANSPORT_STATE_FREE 0
```

Description

This is macro AVCTP_TRANSPORT_STATE_FREE.

AVCTP_TRANSPORT_STATE_IDLE Macro

File

avctp.h

C

```
#define AVCTP_TRANSPORT_STATE_IDLE 1
```

Description

This is macro AVCTP_TRANSPORT_STATE_IDLE.

AVDTP_MANAGER_FLAG_SENDING_MEDIA_PACKET Macro

File

avdtp.h

C

```
#define AVDTP_MANAGER_FLAG_SENDING_MEDIA_PACKET 1
```

Description

This is macro AVDTP_MANAGER_FLAG_SENDING_MEDIA_PACKET.

AVDTP Functions

_bt_avdtp_add_param_byte Function

File

avdtp_control.h

C

```
bt_bool _bt_avdtp_add_param_byte(bt_avdtp_control_cmd_t* command, bt_byte value);
```

Description

This is function _bt_avdtp_add_param_byte.

_bt_avdtp_add_param_uint Function

File

avdtp_control.h

C

```
bt_bool _bt_avdtp_add_param_uint(bt_avdtp_control_cmd_t* command, bt_uint value);
```

Description

This is function _bt_avdtp_add_param_uint.

_bt_avdtp_allocate_buffers Function

File

avdtp_private.h

C

```
void _bt_avdtp_allocate_buffers();
```

Description

This is function _bt_avdtp_allocate_buffers.

_bt_avdtp_allocate_cmd Function

File

avdtp_private.h

C

```
bt_avdtp_control_cmd_t* _bt_avdtp_allocate_cmd(bt_byte msg_type);
```

Description

This is function _bt_avdtp_allocate_cmd.

_bt_avdtp_allocate_sep Function

File

avdtp_private.h

C

```
bt_avdtp_sep_t* _bt_avdtp_allocate_sep();
```

Description

This is function _bt_avdtp_allocate_sep.

_bt_avdtp_allocate_sep_config Function

File

avdtp_private.h

C

```
bt_avdtp_sep_capabilities_t* _bt_avdtp_allocate_sep_config();
```

Description

This is function _bt_avdtp_allocate_sep_config.

_bt_avdtp_allocate_stream Function

File

avdtp_private.h

C

```
bt_avdtp_stream_t* _bt_avdtp_allocate_stream(bt_avdtp_mgr_t* mgr);
```

Description

This is function _bt_avdtp_allocate_stream.

_bt_avdtp_allocate_transport_channel Function

File

avdtp_private.h

C

```
bt_avdtp_transport_channel_t* _bt_avdtp_allocate_transport_channel(bt_avdtp_mgr_t* mgr, bt_byte
strm_handle, bt_byte tc_id, bt_byte tc_type);
```

Description

This is function _bt_avdtp_allocate_transport_channel.

_bt_avdtp_allocate_transport_session_id Function

File

avdtp_private.h

C

```
bt_byte _bt_avdtp_allocate_transport_session_id(bt_avdtp_mgr_t* mgr, bt_byte ts_id);
```

Description

This is function _bt_avdtp_allocate_transport_session_id.

_bt_avdtp_begin_tc_channel_operation Function

File

avdtp_private.h

C

```
void _bt_avdtp_begin_tc_channel_operation(bt_byte opcode, bt_avdtp_transport_op_callback_fp callback, void*
callback_param);
```

Description

This is function _bt_avdtp_begin_tc_channel_operation.

_bt_avdtp_commit_tc_channel_operation Function

File

avdtp_private.h

C

```
void _bt_avdtp_commit_tc_channel_operation();
```

Description

This is function _bt_avdtp_commit_tc_channel_operation.

_bt_avdtp_control_channel_accept_handler Function

File

avdtp_private.h

C

```
void _bt_avdtp_control_channel_accept_handler(bt_avdtp_control_channel_t* channel, bt_byte cmd_code,
bt_byte trans_id, bt_byte* data, bt_int len, bt_int* offset);
```

Description

This is function _bt_avdtp_control_channel_accept_handler.

_bt_avdtp_control_channel_cmd_handler Function

File

avdtp_private.h

C

```
void _bt_avdtp_control_channel_cmd_handler(bt_avdtp_control_channel_t* channel, bt_byte cmd_code, bt_byte
trans_id, bt_byte* data, bt_int len, bt_int* offset);
```

Description

This is function _bt_avdtp_control_channel_cmd_handler.

_bt_avdtp_control_channel_event_handler Function

File

avdtp_control.h

C

```
void _bt_avdtp_control_channel_event_handler(bt_avdtp_control_channel_t* channel, bt_byte evt, void*
evt_param);
```

Description

This is function _bt_avdtp_control_channel_event_handler.

_bt_avdtp_control_channel_reject_handler Function

File

avdtp_private.h

C

```
void _bt_avdtp_control_channel_reject_handler(bt_avdtp_control_channel_t* channel, bt_byte cmd_code,
bt_byte trans_id, bt_byte* data, bt_int len, bt_int* offset);
```

Description

This is function _bt_avdtp_control_channel_reject_handler.

_bt_avdtp_execute_tc_channel_operation Function

File

avdtp_private.h

C

```
void _bt_avdtp_execute_tc_channel_operation();
```

Description

This is function _bt_avdtp_execute_tc_channel_operation.

_bt_avdtp_find_listening_stream Function

File

avdtp_private.h

C

```
bt_avdtp_stream_t* _bt_avdtp_find_listening_stream(bt_avdtp_mgr_t* mgr, bt_byte sep_id);
```

Description

This is function _bt_avdtp_find_listening_stream.

_bt_avdtp_find_stream Function

File

avdtp_private.h

C

```
bt_avdtp_stream_t* _bt_avdtp_find_stream(bt_avdtp_mgr_t* mgr, bt_byte strm_handle);
```

Description

This is function _bt_avdtp_find_stream.

_bt_avdtp_find_stream_by_remote_sep_id Function

File

avdtp_private.h

C

```
bt_avdtp_stream_t* _bt_avdtp_find_stream_by_remote_sep_id(bt_avdtp_mgr_t* mgr, bt_bdaddr_t* remote_addr,
bt_byte remote_seid);
```

Description

This is function _bt_avdtp_find_stream_by_remote_sep_id.

_bt_avdtp_find_stream_by_sep_id Function

File

avdtp_private.h

C

```
bt_avdtp_stream_t* _bt_avdtp_find_stream_by_sep_id(bt_avdtp_mgr_t* mgr, bt_bdaddr_t* remote_addr, bt_byte
sep_id);
```

Description

This is function _bt_avdtp_find_stream_by_sep_id.

_bt_avdtp_free_cmd Function

File

avdtp_private.h

C

```
void _bt_avdtp_free_cmd(bt_avdtp_control_cmd_t* command);
```

Description

This is function _bt_avdtp_free_cmd.

_bt_avdtp_free_sep_config Function

File

avdtp_private.h

C

```
void _bt_avdtp_free_sep_config(bt_avdtp_sep_capabilities_t* sep_config);
```

Description

This is function _bt_avdtp_free_sep_config.

_bt_avdtp_free_stream Function

File

avdtp_private.h

C

```
void _bt_avdtp_free_stream(bt_avdtp_stream_t* strm);
```

Description

This is function _bt_avdtp_free_stream.

_bt_avdtp_free_transport_channel Function

File

avdtp_private.h

C

```
void _bt_avdtp_free_transport_channel(bt_avdtp_transport_channel_t* channel);
```

Description

This is function _bt_avdtp_free_transport_channel.

_bt_avdtp_get_control_channel Function

File

avdtp_control.h

C

```
bt_avdtp_control_channel_t* _bt_avdtp_get_control_channel(struct _bt_avdtp_mgr_t* mgr, bt_bdaddr_t*
remote_addr);
```

Description

This is function _bt_avdtp_get_control_channel.

_bt_avdtp_init_cmd_buffers Function

File

avdtp_private.h

C

```
void _bt_avdtp_init_cmd_buffers();
```

Description

This is function _bt_avdtp_init_cmd_buffers.

_bt_avdtp_init_sep_config_buffers Function

File

avdtp_private.h

C

```
void _bt_avdtp_init_sep_config_buffers();
```

Description

This is function _bt_avdtp_init_sep_config_buffers.

_bt_avdtp_init_signal Function

File

avdtp_private.h

C

```
void _bt_avdtp_init_signal();
```

Description

This is function _bt_avdtp_init_signal.

_bt_avdtp_is_sep_inuse Function

File

avdtp_private.h

C

```
bt_bool _bt_avdtp_is_sep_inuse(bt_avdtp_mgr_t* mgr, bt_avdtp_sep_t* sep, bt_bdaddr_t* remote_addr);
```

Description

This is function _bt_avdtp_is_sep_inuse.

_bt_avdtp_open_control_channel_ex Function

File

avdtp_private.h

C

```
bt_bool _bt_avdtp_open_control_channel_ex(bt_avdtp_mgr_t* mgr, bt_bdaddr_t* remote_addr, bt_uint
acl_config);
```

Description

This is function _bt_avdtp_open_control_channel_ex.

_bt_avdtp_open_control_channel_ex Function

File

avdtp_control.h

C

```
bt_bool _bt_avdtp_open_control_channel_ex(struct _bt_avdtp_mgr_t* mgr, bt_bdaddr_t* remote_addr, bt_uint
acl_config);
```

Description

This is function _bt_avdtp_open_control_channel_ex.

_bt_avdtp_register_transport_channel_for_operation Function

File

avdtp_private.h

C

```
void _bt_avdtp_register_transport_channel_for_operation(bt_avdtp_transport_channel_t* channel);
```

Description

This is function _bt_avdtp_register_transport_channel_for_operation.

_bt_avdtp_send_command Function

File

avdtp_private.h

C

```
void _bt_avdtp_send_command(bt_avdtp_control_channel_t* channel, bt_avdtp_control_cmd_t* cmd);
```

Description

This is function _bt_avdtp_send_command.

_bt_avdtp_send_media_packet Function

File

avdtp_private.h

C

```
bt_bool _bt_avdtp_send_media_packet(bt_avdtp_stream_t* strm);
```

Description

This is function _bt_avdtp_send_media_packet.

_bt_avdtp_set_signal Function

File

avdtp_private.h

C

```
void _bt_avdtp_set_signal();
```

Description

This is function _bt_avdtp_set_signal.

_bt_avdtp_transport_l2cap_read_data_callback Function

File

avdtp_private.h

C

```
void _bt_avdtp_transport_l2cap_read_data_callback(struct _bt_l2cap_channel_t * pch, bt_byte_p pdata, bt_int
len);
```

Description

This is function _bt_avdtp_transport_l2cap_read_data_callback.

_bt_avdtp_transport_l2cap_state_changed_callback Function

File

avdtp_private.h

C

```
void _bt_avdtp_transport_l2cap_state_changed_callback(bt_l2cap_channel_t* pch, bt_int new_state, void*
param);
```

Description

This is function _bt_avdtp_transport_l2cap_state_changed_callback.

_bt_avdtp_write_caps Function

File

avdtp_private.h

C

```
bt_bool _bt_avdtp_write_caps(bt_avdtp_mgr_t* mgr, const bt_avdtp_sep_capabilities_t* caps,
bt_avdtp_stream_t* strm, bt_avdtp_control_cmd_t* command);
```

Description

This is function _bt_avdtp_write_caps.

bt_avdtp_abort_stream Function

File

avdtp.h

C

```
bt_bool bt_avdtp_abort_stream(bt_avdtp_mgr_t* mgr, bt_byte strm_handle);
```

Description

brief Suspend a stream. ingroup avdtp

details This function tries to suspend a stream by sending a request to the remote party. The stream can be in any state state except AVDTP_STREAM_STATE_IDLE. As a result of this operation the AVDTP_EVT_ABORT_STREAM_COMPLETED event will be generated. This operation cannot be rejected. The p evt_param.abort_stream_requested.err_code is always == AVDTP_ERROR_SUCCESS.

param mgr AVDTP manager. param strm_handle Stream handle.

return li c TRUE if the function succeeds, i.e. the actual request has been sent to the remote party. li c FALSE otherwise. No events will be generated.

bt_avdtp_add_media_rx_buffer Function

File

avdtp.h

C

```
bt_bool bt_avdtp_add_media_rx_buffer(bt_avdtp_mgr_t* mgr, bt_byte strm_handle, bt_media_packet_t* buffer);
```

Description

brief Add a media packet buffer to a receive queue ingroup avdtp

details The consumer of AVDTP is responsible for allocating and supplying AVDTP with buffers used to store received packets. AVDTP itself only has a queue for storing pointers to buffers supplied by the consumer. When a packet comes in AVDTP finds the first buffer large enough to hold the received packet, copies the packet to the buffer and generates a AVDTP_EVT_MEDIA_PACKET_RECEIVED event. The consumer then has to process the data in the buffer and return it back to the queue. If there is no buffers in the queue or none of the buffers is large enough the received packets is dropped. Each buffer has a field (data_len) that holds the length of the received buffer. This field is never 0 if the buffer contains a packet. If a channel closed regardless of what has caused that and there are still buffers in the queue AVDTP generates a AVDTP_EVT_MEDIA_PACKET_RECEIVED event for each buffer and sets the data_len to 0. This is to inform the AVDTP consumer that the buffer has not been used and can be, for example, deallocated. This function adds a buffer to the receive queue.

param mgr AVDTP manager. param strm_handle Stream handle. param buffer Pointer to a structure that holds the buffer and its parameters.

return li c TRUE if the function succeeds. li c FALSE otherwise. The function fails only if a stream specified by the p strm_handle parameter li does not exist. The stream can be in any state to call this function.

bt_avdtp_add_media_tx_buffer Function

File

avdtp.h

C

```
bt_bool bt_avdtp_add_media_tx_buffer(bt_avdtp_mgr_t* mgr, bt_byte strm_handle, bt_media_packet_t* buffer);
```

Description

brief Add a media packet buffer to a send queue ingroup avdtp

details When the consumer of AVDTP wants to send a packet to a remote device it calls this function. The function adds the packet to a queue and tells AVDTP that it has something to send. The packet will be send as soon as the stream goes to AVDTP_STREAM_STATE_STREAMING state. When the packet has been successfully sent a AVDTP_EVT_MEDIA_PACKET_SENT is generated. Otherwise a AVDTP_EVT_MEDIA_PACKET_SEND_FAILED is generated. Regardless of the event generated the consumer can re-use the buffer as AVDTP has removed it from the queue and gave up control over it. As in the case of received buffers, if a channel closed regardless of what has caused that and there are still buffers in the queue AVDTP generates a AVDTP_EVT_MEDIA_PACKET_SENT event for each buffer and sets the data_len field to 0. This is to inform the AVDTP consumer that the buffer has not been used and can be, for example, deallocated.

param mgr AVDTP manager. param strm_handle Stream handle. param buffer Pointer to a structure that holds the buffer and its parameters.

return li c TRUE if the function succeeds. li c FALSE otherwise. The function fails only if a stream specified by the p strm_handle parameter li does not exist. The stream can be in any state to call this function.

bt_avdtp_cancel_listen Function

File

avdtp.h

C

```
void bt_avdtp_cancel_listen(bt_avdtp_mgr_t* mgr, bt_byte strm_handle, bt_byte sep_id);
```

Description

brief Cancel listening for incoming connections. ingroup avdtp

details This function removes a SEP from a list of SEPS which a stream can use for incoming requests.

param mgr AVDTP manager. param strm_handle Stream handle. param sep_id Local SEP ID.

return li c TRUE if the function succeeds. li c FALSE otherwise.

bt_avdtp_close_stream Function

File

avdtp.h

C

```
bt_bool bt_avdtp_close_stream(bt_avdtp_mgr_t* mgr, bt_byte strm_handle);
```

Description

brief Close a stream. ingroup avdtp

details This function tries to close a stream by sending a request to the remote party. The stream has to be in AVDTP_STREAM_STATE_OPEN or AVDTP_STREAM_STATE_STREAMING state. As a result of this operation the AVDTP_EVT_CLOSE_STREAM_COMPLETED event will be generated. If the stream has been closed the p evt_param.bt_avdtp_evt_close_stream_completed_t.err_code == AVDTP_ERROR_SUCCESS. Otherwise, if the remote device for any reason cannot or does not wish to close the stream, the p evt_param.bt_avdtp_evt_close_stream_completed_t.err_code == the error code sent by the remote.

param mgr AVDTP manager. param strm_handle Stream handle.

return li c TRUE if the function succeeds, i.e. the actual request has been sent to the remote party. li c FALSE otherwise. No events will be generated.

bt_avdtp_create_stream Function

File

avdtp.h

C

```
bt_byte bt_avdtp_create_stream(bt_avdtp_mgr_t* mgr);
```

Description

brief Create a stream. ingroup avdtp

details This function allocates memory for storing stream's data and assigns a stream handle. The stream handle is used to manipulate the stream - open, close, configure, suspend, abort.

param mgr AVDTP manager.

return li c Stream handle if the function succeeds. li c 0 otherwise.

bt_avdtp_destroy_stream Function

File

avdtp.h

C

```
bt_bool bt_avdtp_destroy_stream(bt_avdtp_mgr_t* mgr, bt_byte strm_handle);
```

Description

brief Destroy a stream. ingroup avdtp

details This function frees memory used by the stream. The stream has to exist and be in the "idle" state for this function to succeed. I.e. the stream has to be closed or aborted before this function can be called.

param mgr AVDTP manager. param strm_handle Stream handle.

return li c TRUE if the function succeeds. li c FALSE otherwise.

bt_avdtp_disconnect Function

File

avdtp.h

C

```
bt_bool bt_avdtp_disconnect(bt_avdtp_mgr_t* mgr, bt_bdaddr_t* remote_addr);
```

Description

brief Disconnect from a remote device. ingroup avdtp

details This function closes a control and transport channels on all streams associated with the remote device specified by the p remote_addr. As a result of this operation the following events will be generated: @arg AVDTP_EVT_MEDIA_PACKET_RECEIVED: if a stream's receive queue is not empty this event is generated for each buffer with bt_media_packet_t::data_len set to 0 @arg AVDTP_EVT_MEDIA_PACKET_SENT: if a stream's send queue is not empty this event is generated for each buffer with bt_media_packet_t::data_len set to 0 @arg AVDTP_EVT_STREAM_CLOSED: this event is generate if a stream is in "closing" state as a result of a request from the remote device or bt_avdtp_close_stream call before bt_avdtp_disconnect call @arg AVDTP_EVT_STREAM_ABORTED: this event is generated if a stream is in "active" state at the time of bt_avdtp_disconnect call.

param mgr AVDTP manager. param remote_addr The address of a remote device.

return li c TRUE if disconnection has been started. li c FALSE otherwise. No events will be generated.

bt_avdtp_discover Function

File

avdtp.h

C

```
bt_bool bt_avdtp_discover(bt_avdtp_mgr_t* mgr, bt_bdaddr_t* remote_addr);
```

Description

brief Discover SEPs on a remote device. ingroup avdtp

details This function asks the remote device to send a list of all available SEPs. As a result of this operation the following events will be generated: @arg AVDTP_EVT_SEP_INFO_RECEIVED: this event is generated for every SEP received from the remote device. the p evt_param.sep_info_received contains SEP information. @arg AVDTP_EVT_DISCOVER_COMPLETED: this event is generated after last AVDTP_EVT_SEP_INFO_RECEIVED if the remote accepted the request and the p evt_param.discover_completed.err_code == AVDTP_ERROR_SUCCESS. if the remote rejected the request the p evt_param.discover_completed.err_code == the error code sent by the remote.

param mgr AVDTP manager. param remote_addr The address of a remote device.

return li c TRUE if discover request has been sent. li c FALSE otherwise. No events will be generated.

bt_avdtp_find_codec Function

File

avdtp.h

C

```
bt_avdtp_codec_t* bt_avdtp_find_codec(bt_avdtp_mgr_t* mgr, bt_byte codec_type);
```

Description

brief Find a codec ingroup avdtp

details AVDTP in theory can support any type of codec. Each codec uses its own format for exchanging capabilities and configuration information. In order to make out implementation do not care about these formats we use a simple way of telling AVDTP how to parse and serialize codec's configuration. The consumer of AVDTP (e.g. A2DP) for each codec it wishes to support has to register a callback function (one per codec type). That callback has to perform two function. The first one is to read the configuration received from the remote device and store it in a structure defined by the consumer. The second one is to serialize the data from a structure to a format (in case of standard A2DP codecs the format is defined in A2DP specification, vendor specific codecs can define their own formats) suitable for sending as a part of a AVDTP request. This

function returns a pointer to a structure that holds a pointer to a codec's callback function.

param mgr AVDTP manager. param codec_type Codec type. The codec_type can be one of the following values: @arg AVDTP_CODEC_TYPE_SBC: SBC @arg AVDTP_CODEC_TYPE_MPEG1_2_AUDIO: MPEG-1,2 (used in MP3 files) @arg AVDTP_CODEC_TYPE_MPEG2_4_AAC: MPEG-2,4 AAC (used in Apple products) @arg AVDTP_CODEC_TYPE_ATRAC: ATRAC (used in Sony products) @arg AVDTP_CODEC_TYPE_NON_A2DP: Non-A2DP Codec

return li c TRUE if the function succeeds. li c FALSE otherwise. The function fails if a callback for a codec type specified in the p codec parameter li has not been previously registered with bt_avdtp_register_codec.

bt_avdtp_get_capabilities Function

File

avdtp.h

C

```
bt_bool bt_avdtp_get_capabilities(bt_avdtp_mgr_t* mgr, bt_bdaddr_t* remote_addr, bt_byte seid_acp);
```

Description

brief Get remote SEP capabilities. ingroup avdtp

details This function asks the remote device to send capabilities of a SEP specified by the p seid_acp. As a result of this operation the following events will be generated: @arg AVDTP_EVT_SEP_CAPABILITIES_RECEIVED: this event is generated if the remote device accepted the request. the p evt_param.sep_capabilities_received contains SEP capabilities. @arg AVDTP_EVT_GET_SEP_CAPABILITIES_COMPLETED: this event is generated right after AVDTP_EVT_SEP_CAPABILITIES_RECEIVED if the remote accepted the request the p evt_param.get_sep_capabilities_completed.err_code == AVDTP_ERROR_SUCCESS. if the remote rejected the request the p evt_param.get_sep_capabilities_completed.err_code == the error code sent by the remote.

param mgr AVDTP manager. param remote_addr The address of a remote device. param seid_acp The ID of a remote SEP.

return li c TRUE if discover request has been sent. li c FALSE otherwise. No events will be generated.

bt_avdtp_get_configuration Function

File

avdtp.h

C

```
bt_bool bt_avdtp_get_configuration(bt_avdtp_mgr_t* mgr, bt_byte strm_handle);
```

Description

brief Get stream configuration. ingroup avdtp

details This function requests stream configuration from a remote device. As a result of this operation the following events will be generated: @arg AVDTP_EVT_STREAM_CONFIGURATION_RECEIVED: this event is generated if the remote accepted the request. the p ebt_para.sep_capabilities_received.caps will contain current stream configuration. @arg AVDTP_EVT_GET_STREAM_CONFIGURATION_COMPLETED: If the remote accepted the request the p evt_param.get_stream_configuration_completed.err_code == AVDTP_ERROR_SUCCESS. if the remote rejected the request the p evt_param.get_stream_configuration_completed.err_code == the error code sent by the remote.

param mgr AVDTP manager. param strm_handle Stream handle.

return li c TRUE if the function succeeds, i.e. the actual request has been sent to the remote party. li c FALSE otherwise. No events will be generated.

bt_avdtp_get_hci_connection Function

File

avdtp.h

C

```
bt_hci_conn_state_t* bt_avdtp_get_hci_connection(bt_avdtp_mgr_t* mgr, bt_byte strm_handle);
```

Description

brief Get HCI connection for a stream ingroup avdtp

details This function returns a pointer to a structure that describes an HCI connection a stream is open on. The return value can be used to call various function from the HCI layer. For example, if an app wants to force disconnection from a remote device it can call bt_hci_disconnect.

param mgr AVDTP manager. param strm_handle Stream handle.

return li c Pointer to a structure that describes an HCI connection if the function succeeds. li c NULL otherwise. The function fails only if a stream specified by the p strm_handle parameter li does not exist or there is no HCI connection between local and remote devices associated with the stream.

note This function has not been implemented.

bt_avdtp_get_mgr Function

File

avdtp.h

C

```
bt_avdtp_mgr_t* bt_avdtp_get_mgr();
```

Description

brief Return a pointer to an instance of the AVDTP manager. ingroup avdtp

details This function returns a pointer to an instance of the AVDTP manager. There is only one instance of the manager allocated by the stack. The pointer is passed as the first parameter to all AVDTP functions.

bt_avdtp_get_sep Function

File

avdtp.h

C

```
bt_avdtp_sep_t* bt_avdtp_get_sep(bt_avdtp_mgr_t* mgr, bt_byte sep_id);
```

Description

brief Get a SEP info by its ID. ingroup avdtp

details This function returns a pointer to bt_avdtp_sep_t structure that describes a SEP previously registered with bt_avdtp_register_sep.

param mgr AVDTP manager. param sep_id The ID of a SEP.

return li c Pointer to bt_avdtp_sep_t if the SEP is in the list of registered SEPs. li c NULL otherwise.

bt_avdtp_get_stream_codec_config Function

File

avdtp.h

C

```
void* bt_avdtp_get_stream_codec_config(bt_avdtp_mgr_t* mgr, bt_byte strm_handle);
```

Description

brief Get the configuration of the codec currently used with the stream. ingroup avdtp

details This function returns a pointer to a structure that contains configuration of the codec currently used with the stream. The structure returned depends on the codec. The dotstack defines structures only for SBC, MPEG-1,2 and MPEG-2,4 AAC codecs: @arg SBC: bt_a2dp_sbc_config_t (defined in a2dp_sbc_codec.h) @arg MPEG-1,2: bt_a2dp_mpeg_config_t (defined in a2dp_mpeg_codec.h) @arg MPEG-2,4 AAC: bt_a2dp_aac_config_t (defined in a2dp_aac_codec.h)

param mgr AVDTP manager. param strm_handle Stream handle.

return li The codec's configuration if strm_handle specifies a valid stream and the stream is in one of the following state:

AVDTP_STREAM_STATE_CONFIGURED AVDTP_STREAM_STATE_OPEN AVDTP_STREAM_STATE_STREAMING

li NULL otherwise.

bt_avdtp_get_stream_codec_type Function

File

avdtp.h

C

```
bt_byte bt_avdtp_get_stream_codec_type(bt_avdtp_mgr_t* mgr, bt_byte strm_handle);
```

Description

brief Get the type of the codec currently used with the stream. ingroup avdtp

details This function returns the type of the codec currently used with the stream.

param mgr AVDTP manager. param strm_handle Stream handle.

return @arg The type of the codec if strm_handle specifies a valid stream and the stream is in one of the following states:

AVDTP_STREAM_STATE_CONFIGURED AVDTP_STREAM_STATE_OPEN AVDTP_STREAM_STATE_STREAMING

@arg The result will be one of the following values:

AVDTP_CODEC_TYPE_SBC: SBC AVDTP_CODEC_TYPE_MPEG1_2_AUDIO: MPEG-1,2 (used in MP3 files) AVDTP_CODEC_TYPE_MPEG2_4_AAC: MPEG-2,4 AAC (used in Apple products) AVDTP_CODEC_TYPE_ATRAC: ATRAC (used in Sony products) AVDTP_CODEC_TYPE_NON_A2DP: Non-A2DP Codec

@arg 0xFF otherwise.

bt_avdtp_get_stream_config Function

File

avdtp.h

C

```
bt_avdtp_sep_capabilities_t* bt_avdtp_get_stream_config(bt_avdtp_mgr_t* mgr, bt_byte strm_handle);
```

Description

brief Get stream's configuration. ingroup avdtp

details This function returns a pointer to a structure holding the current configuration of stream.

param mgr AVDTP manager. param strm_handle Stream handle.

return li The stream's configuration if strm_handle specifies a valid stream and the stream is in one of the following state:

AVDTP_STREAM_STATE_CONFIGURED AVDTP_STREAM_STATE_OPEN AVDTP_STREAM_STATE_STREAMING

li NULL otherwise.

bt_avdtp_get_stream_local_sep_id Function

File

avdtp.h

C

```
bt_byte bt_avdtp_get_stream_local_sep_id(bt_avdtp_mgr_t* mgr, bt_byte strm_handle);
```

Description

brief Get stream's local SEP ID. ingroup avdtp

details This function returns the ID of the local SEP associated with the stream.

param mgr AVDTP manager. param strm_handle Stream handle.

return li The ID of the local SEP if strm_handle specifies a valid stream. li 0 otherwise.

bt_avdtp_get_stream_remote_address Function

File

avdtp.h

C

```
bt_bdaddr_t* bt_avdtp_get_stream_remote_address(bt_avdtp_mgr_t* mgr, bt_byte strm_handle);
```

Description

brief Get stream's remote BT address. ingroup avdtp

details This function returns the address of the remote device associated with the stream.

param mgr AVDTP manager. param strm_handle Stream handle.

return li The address of the remote device if strm_handle specifies a valid stream. li NULL otherwise.

bt_avdtp_get_stream_remote_sep_id Function

File

avdtp.h

C

```
bt_byte bt_avdtp_get_stream_remote_sep_id(bt_avdtp_mgr_t* mgr, bt_byte strm_handle);
```

Description

brief Get stream's remote SEP ID. ingroup avdtp

details This function returns the ID of the remote SEP associated with the stream.

param mgr AVDTP manager. param strm_handle Stream handle.

return li The ID of the remote SEP if strm_handle specifies a valid stream. li 0 otherwise.

bt_avdtp_get_stream_state Function

File

avdtp.h

C

```
bt_byte bt_avdtp_get_stream_state(bt_avdtp_mgr_t* mgr, bt_byte strm_handle);
```

Description

brief Get local stream state. ingroup avdtp

details This function returns local state of a stream specified by the p strm_handle. No request is sent to the remote party.

param mgr AVDTP manager. param strm_handle Stream handle.

return The state of the stream. The result will be one of the following values: @arg AVDTP_STREAM_STATE_IDLE: The stream is idle. This can mean two things. The stream specified by p strm_handle does not exist or the stream is closed. @arg AVDTP_STREAM_OPENING_TRANSPORT_CHANNELS: The stream is opening transport channels. @arg AVDTP_STREAM_CLOSING_TRANSPORT_CHANNELS: The stream is closing transport channels. @arg AVDTP_STREAM_STATE_CONFIGURED: The stream has been configured. @arg AVDTP_STREAM_STATE_OPEN: The stream has been opened. @arg AVDTP_STREAM_STATE_STREAMING: The stream has been started. Depending on the local SEP type (source or sink) it means that the stream is can send or receive media packets. @arg AVDTP_STREAM_STATE_CLOSING: The stream is closing. This means that all transport channels associated with the stream are being closed. After they have been closed the stream goes to AVDTP_STREAM_STATE_IDLE state. @arg AVDTP_STREAM_STATE_ABORTING: The stream is aborting. This means that all transport channels associated with the stream are being closed. After they have been closed the stream goes to AVDTP_STREAM_STATE_IDLE state.

bt_avdtp_init Function

File

avdtp.h

C

```
void bt_avdtp_init();
```

Description

brief Initialize the AVDTP layer. ingroup avdtp

details This function initializes the AVDTP layer of the stack. It must be called prior to any other AVDTP function can be called.

bt_avdtp_listen Function

File

avdtp.h

C

```
bt_bool bt_avdtp_listen(bt_avdtp_mgr_t* mgr, bt_byte strm_handle, bt_byte sep_id);
```

Description

brief Listen for incoming connections. ingroup avdtp

details This function tells a stream that it can use a particular SEP to accept incoming requests to open it. The SEP can be associated with multiple streams but used with only one. The stream has to be closed before the SEP can be used with another stream. For outgoing connections this is not needed. Any SEP can be used with any stream given that the SEP is not already in use by another stream.

param mgr AVDTP manager. param strm_handle Stream handle. param sep_id Local SEP ID.

return li c TRUE if the function succeeds. li c FALSE otherwise.

bt_avdtp_open_stream Function

File

avdtp.h

C

```
bt_bool bt_avdtp_open_stream(bt_avdtp_mgr_t* mgr, bt_byte strm_handle);
```

Description

brief Open a stream. ingroup avdtp

details This function tries to open a stream by sending a request to the remote party. The stream has to be already configured with a bt_avdtp_set_configuration call. As a result of this operation the AVDTP_EVT_OPEN_STREAM_COMPLETED event will be generated. If the stream has been open the p evt_param.open_stream_completed.err_code == AVDTP_ERROR_SUCCESS. Otherwise, if the remote device for any reason cannot or does not wish to open the stream, the p evt_param.open_stream_completed.err_code == the error code sent by the remote.

param mgr AVDTP manager. param strm_handle Stream handle.

return li c TRUE if the function succeeds, i.e. the actual request has been sent to the remote party. li c FALSE otherwise. No events will be generated.

bt_avdtp_reconfigure_stream Function

File

avdtp.h

C

```
bt_bool bt_avdtp_reconfigure_stream(bt_avdtp_mgr_t* mgr, bt_byte strm_handle, bt_avdtp_sep_capabilities_t*
caps);
```

Description

brief Reconfigure stream. ingroup avdtp

details This function tries to change the stream's configuration. For this function to succeed the stream has to be open. As a result of this operation the AVDTP_EVT_STREAM_RECONFIGURE_COMPLETED event will be generated. If reconfiguration was a success the p evt_param.stream_reconfigure_completed.err_code == AVDTP_ERROR_SUCCESS. Otherwise the p evt_param.stream_reconfigure_completed.err_code == the error code sent by the remote.

param mgr AVDTP manager. param strm_handle Stream handle. param caps New stream configuration.

return li c TRUE if the function succeeds, i.e. the actual request has been sent to the remote party. li c FALSE otherwise. No events will be generated.

bt_avdtp_register_callback Function

File

avdtp.h

C

```
void bt_avdtp_register_callback(bt_avdtp_mgr_t* mgr, bt_avdtp_mgr_callback_fp callback, void*
callback_param);
```

Description

brief Register a AVDTP application callback. ingroup avdtp

details In order to be notified of various events a consumer of the AVDTP layer has to register a callback function. The stack will call this function whenever a new event has been generated passing the code of the event as the second parameter. The event can be one of the following values:

@arg AVDTP_EVT_CTRL_CHANNEL_CONNECTED: Control channel connected. @arg AVDTP_EVT_CTRL_CHANNEL_DISCONNECTED:

Control channel disconnected. @arg AVDTP_EVT_CTRL_CONNECTION_FAILED: Control channel connection failed (generated only if control connection has been initiated by the local device). @arg AVDTP_EVT_DISCOVER_COMPLETED: Local device completed discovering remote SEPs. @arg AVDTP_EVT_GET_SEP_CAPABILITIES_COMPLETED: Local device received a response to Get SEP capabilities operation. @arg AVDTP_EVT_SET_STREAM_CONFIGURATION_COMPLETED: Local device received a response to Set stream configuration operation. @arg AVDTP_EVT_GET_STREAM_CONFIGURATION_COMPLETED: Local device received a response to Get stream configuration operation. @arg AVDTP_EVT_STREAM_RECONFIGURE_COMPLETED: Local device received a response to Reconfigure stream operation. @arg AVDTP_EVT_OPEN_STREAM_COMPLETED: Local device received a response to Open stream operation. @arg AVDTP_EVT_START_STREAM_COMPLETED: Local device received a response to Start stream operation. @arg AVDTP_EVT_CLOSE_STREAM_COMPLETED: Local device received a response to Close stream operation. @arg AVDTP_EVT_SUSPEND_STREAM_COMPLETED: Local device received a response to Suspend stream operation. @arg AVDTP_EVT_STREAM_SECURITY_CONTROL_COMPLETED: Local device received a response to Stream security control operation. @arg AVDTP_EVT_ABORT_STREAM_COMPLETED: Local device received a response to Abort stream operation. @arg AVDTP_EVT_SEP_INFO_RECEIVED: SEP information received. @arg AVDTP_EVT_SEP_CAPABILITIES_RECEIVED: SEP capabilities received. @arg AVDTP_EVT_STREAM_CONFIGURATION_RECEIVED: Stream configuration received.

@arg AVDTP_EVT_SET_STREAM_CONFIGURATION_REQUESTED: Remote device requested stream configuration. @arg AVDTP_EVT_OPEN_STREAM_REQUESTED: Remote device requested to open a stream. @arg AVDTP_EVT_START_STREAM_REQUESTED: Remote device requested to start a stream. @arg AVDTP_EVT_CLOSE_STREAM_REQUESTED: Remote device requested to close a stream. @arg AVDTP_EVT_SUSPEND_STREAM_REQUESTED: Remote device requested to suspend a stream. @arg AVDTP_EVT_ABORT_STREAM_REQUESTED: Remote device requested to abort a stream. @arg AVDTP_EVT_RECONFIGURE_STREAM_REQUESTED: Remote device requested to reconfigure a stream. @arg AVDTP_EVT_MEDIA_PACKET_RECEIVED: Remote device sent a media packet. @arg AVDTP_EVT_STREAM_CONFIGURED: A stream has been configured (This event is generated right after AVDTP_EVT_SET_STREAM_CONFIGURATION_REQUESTED if the local devices accepted the request). @arg AVDTP_EVT_STREAM_RECONFIGURED: A stream has been re-configured (This event is generated right after AVDTP_EVT_RECONFIGURE_STREAM_REQUESTED if the local devices accepted the request). @arg AVDTP_EVT_STREAM_OPENED: A stream has been opened (This event is generated as a result of local or remote stream opening request). @arg AVDTP_EVT_STREAM_STARTED: A stream has been started (This event is generated right after AVDTP_EVT_START_STREAM_REQUESTED if the local devices accepted the request). @arg AVDTP_EVT_STREAM_CLOSED: A stream has been close (This event is generated right after AVDTP_EVT_CLOSE_STREAM_REQUESTED if the local devices accepted the request). @arg AVDTP_EVT_STREAM_SUSPENDED: A stream has been suspended (This event is generated right after AVDTP_EVT_SUSPEND_STREAM_REQUESTED if the local devices accepted the request). @arg AVDTP_EVT_STREAM_ABORTED: A stream has been aborted (This event is generated right after AVDTP_EVT_SUSPEND_STREAM_REQUESTED if the local devices accepted the request. It is also generated if connection between devices has been terminated by means other than AVDTP signaling, e.g. devices going out of rage). @arg AVDTP_EVT_MEDIA_PACKET_SENT: The local device has successfully sent a media packet to the remote device. @arg AVDTP_EVT_MEDIA_PACKET_SEND_FAILED: The local device was not able to send a media packet to the remote device.

param mgr AVDTP manager. param callback The callback function that will be called when the AVDTP generates an event. param callback_param A pointer to arbitrary data to be passed to the c callback callback.

bt_avdtp_register_codec Function

File

avdtp.h

C

```
bt_bool bt_avdtp_register_codec(bt_avdtp_mgr_t* mgr, bt_avdtp_codec_t* codec);
```

Description

brief Register a codec ingroup avdtp

details AVDTP in theory can support any type of codec. Each codec uses its own format for exchanging capabilities and configuration information. In order to make out implementation do not care about these formats we use a simple way of telling AVDTP how to parse and serialize codec's configuration. The consumer of AVDTP (e.g. A2DP) for each codec it wishes to support has to register a callback function (one per codec type). That callback has to perform two function. The first one is to read the configuration received from the remote device and store it in a structure defined by the consumer. The second one is to serialize the data from a structure to a format (in case of standard A2DP codecs the format is defined in A2DP specification, vendor specific codecs can define their own formats) suitable for sending as a part of a AVDTP request. This function adds a codec's callback function to an internal list.

param mgr AVDTP manager. param codec Pointer to a structure specifying codec type and a callback.

return li c TRUE if the function succeeds. li c FALSE otherwise. The function fails if there already is a callback for a codec type specified in the p codec parameter.

bt_avdtp_register_sep Function

File

avdtp.h

C

```
bt_byte bt_avdtp_register_sep(bt_avdtp_mgr_t* mgr, bt_byte type, const bt_avdtp_sep_capabilities_t* caps);
```

Description

brief Register a SEP with the local AVDTP manager. ingroup avdtp

details This function is used to make a list of SEPs supported by the local ADVTP entity.

param mgr AVDTP manager. param type The type of a SEP. The type can be one of the following values: @arg AVDTP_SEP_TYPE_SOURCE: The SEP is a source. @arg AVDTP_SEP_TYPE_SINK: The SEP is a sink. param caps The capabilities of a SEP.

return li c ID of a SEP if the function succeeds. li c FALSE otherwise.

bt_avdtp_remove_media_rx_buffer Function

File

avdtp.h

C

```
bt_bool bt_avdtp_remove_media_rx_buffer(bt_avdtp_mgr_t* mgr, bt_byte strm_handle, bt_media_packet_t*
buffer);
```

Description

brief Remove a media packet buffer from a receive queue ingroup avdtp

details The consumer of AVDTP is responsible for allocating and supplying AVDTP with buffers used to store received packets. AVDTP itself only has a queue for storing pointers to buffers supplied by the consumer. When a packet comes in AVDTP finds the first buffer large enough to hold the received packet, copies the packet to the buffer and generates a AVDTP_EVT_MEDIA_PACKET_RECEIVED event. The consumer then has to process the data in the buffer and return it back to the queue. If there is no buffers in the queue or none of the buffers is large enough the received packets is dropped. Each buffer has a field (data_len) that holds the length of the received buffer. This field is never 0 if the buffer contains a packet. If a channel closed regardless of what has caused that and there are still buffers in the queue AVDTP generates a AVDTP_EVT_MEDIA_PACKET_RECEIVED event for each buffer and sets the data_len to 0. This is to inform the AVDTP consumer that the buffer has not been used and can be, for example, deallocated. This function removes a buffer from the receive queue.

param mgr AVDTP manager. param strm_handle Stream handle. param buffer Pointer to a structure that holds the buffer and its parameters.

return li c TRUE if the function succeeds. li c FALSE otherwise. The function fails only if a stream specified by the p strm_handle parameter li does not exist. The stream can be in any state to call this function.

bt_avdtp_remove_media_tx_buffer Function

File

avdtp.h

C

```
bt_bool bt_avdtp_remove_media_tx_buffer(bt_avdtp_mgr_t* mgr, bt_byte strm_handle, bt_media_packet_t*
buffer);
```

Description

brief Remove a media packet buffer from a send queue ingroup avdtp

details When the consumer of AVDTP wants to send a packet to a remote device it calls bt_avdtp_add_media_tx_buffer function. The function adds the packet to a queue and tells AVDTP that it has something to send. The packet will be send as soon as the stream goes to AVDTP_STREAM_STATE_STREAMING state. The consumer has a chance to remove a packet from the queue before it has been sent to a remote device by calling bt_avdtp_remove_media_tx_buffer.

param mgr AVDTP manager. param strm_handle Stream handle. param buffer Pointer to a structure that holds the buffer and its parameters.

return li c TRUE if the function succeeds. li c FALSE otherwise. The function fails only if a stream specified by the p strm_handle parameter li does not exist. The stream can be in any state to call this function.

note This function has not been implemented.

bt_avdtp_security_control Function

File

avdtp.h

C

```
bt_bool bt_avdtp_security_control(bt_avdtp_mgr_t* mgr, bt_byte strm_handle, bt_byte* sc_data, bt_byte sc_data_len);
```

Description

brief Exchange content protection control data. ingroup avdtp

details This function tries to establish content protection by sending a request to the remote party. The stream can be in any state state except AVDTP_STREAM_STATE_IDLE, AVDTP_STREAM_STATE_CLOSING, AVDTP_STREAM_STATE_ABORTING. As a result of this operation the AVDTP_EVT_STREAM_SECURITY_CONTROL_COMPLETED event will be generated. If the stream's content protection data has been accepted by the remote party the p evt_param.security_control_completed.err_code == AVDTP_ERROR_SUCCESS. Otherwise the p evt_param.security_control_completed.err_code == the error code sent by the remote.

note The dotstack does not support content protection. Although the request can be sent it will not affect the operation of the AVDTP in any way.

param mgr AVDTP manager. param strm_handle Stream handle.

return li c TRUE if the function succeeds, i.e. the actual request has been sent to the remote party. li c FALSE otherwise. No events will be generated.

bt_avdtp_set_configuration Function

File

avdtp.h

C

```
bt_bool bt_avdtp_set_configuration(bt_avdtp_mgr_t* mgr, bt_byte strm_handle, bt_bdaddr_t* remote_addr, bt_byte seid_int, bt_byte seid_acp, const bt_avdtp_sep_capabilities_t* caps);
```

Description

brief Set stream configuration. ingroup avdtp

details This function tries to configure a stream before opening it. As a result of this operation the AVDTP_EVT_SET_STREAM_CONFIGURATION_COMPLETED event will be generated. If configuration was a success the p evt_param.set_stream_configuration_completed.err_code == AVDTP_ERROR_SUCCESS. Otherwise the p evt_param.set_stream_configuration_completed.err_code == the error code sent by the remote and p evt_param.set_stream_configuration_completed.svc_category == the value of the first Service Category to fail.

param mgr AVDTP manager. param strm_handle Stream handle. param remote_addr The address of a remote device. param seid_int Local SEP ID. param seid_acp Remote SEP ID. param caps Stream configuration.

return li c TRUE if the function succeeds. li c FALSE otherwise. No events will be generated.

bt_avdtp_start Function

File

avdtp.h

C

```
bt_bool bt_avdtp_start(bt_avdtp_mgr_t* mgr);
```

Description

brief Start the AVDTP layer. ingroup avdtp

details This function makes the AVDTP layer ready to accept connection requests from remote device. To make an outgoing connection calling this function is not required.

param mgr AVDTP manager.

return li c TRUE if the function succeeds. li c FALSE otherwise.

bt_avdtp_start_stream Function

File

avdtp.h

C

```
bt_bool bt_avdtp_start_stream(bt_avdtp_mgr_t* mgr, bt_byte strm_handle);
```

Description

brief Start a stream. ingroup avdtp

details This function tries to start a stream by sending a request to the remote party. The stream has to be in AVDTP_STREAM_STATE_OPEN state. The stream goes to this state as a result of successful configuration or suspension (both can be initiated by either party). As a result of this operation the AVDTP_EVT_START_STREAM_COMPLETED event will be generated. If the stream has been open the p evt_param.start_stream_requested.err_code == AVDTP_ERROR_SUCCESS. Otherwise, if the remote device for any reason cannot or does not wish to start the stream, the p evt_param.start_stream_requested.err_code == the error code sent by the remote.

param mgr AVDTP manager. param strm_handle Stream handle.

return li c TRUE if the function succeeds, i.e. the actual request has been sent to the remote party. li c FALSE otherwise. No events will be generated.

bt_avdtp_suspend_stream Function

File

avdtp.h

C

```
bt_bool bt_avdtp_suspend_stream(bt_avdtp_mgr_t* mgr, bt_byte strm_handle);
```

Description

brief Suspend a stream. ingroup avdtp

details This function tries to suspend a stream by sending a request to the remote party. The stream has to be in AVDTP_STREAM_STATE_STREAMING state. As a result of this operation the AVDTP_EVT_SUSPEND_STREAM_COMPLETED event will be generated. If the stream has been suspended the p evt_param.bt_avdtp_evt_suspend_stream_requested_t.err_code == AVDTP_ERROR_SUCCESS. Otherwise, if the remote device for any reason cannot or does not wish to suspend the stream, the p evt_param.bt_avdtp_evt_suspend_stream_requested_t.err_code == the error code sent by the remote.

param mgr AVDTP manager. param strm_handle Stream handle.

return li c TRUE if the function succeeds, i.e. the actual request has been sent to the remote party. li c FALSE otherwise. No events will be generated.

bt_avdtp_unregister_codec Function

File

avdtp.h

C

```
bt_bool bt_avdtp_unregister_codec(bt_avdtp_mgr_t* mgr, bt_byte codec_type);
```

Description

brief Unregister a codec ingroup avdtp

details AVDTP in theory can support any type of codec. Each codec uses its own format for exchanging capabilities and configuration information. In order to make out implementation do not care about these formats we use a simple way of telling AVDTP how to parse and serialize codec's configuration. The consumer of AVDTP (e.g. A2DP) for each codec it wishes to support has to register a callback function (one per codec type). That callback has to perform two function. The first one is to read the configuration received from the remote device and store it in a structure defined by the consumer. The second one is to serialize the data from a structure to a format (in case of standard A2DP codecs the format is defined in A2DP specification, vendor specific codecs can define their own formats) suitable for sending as a part of a AVDTP request. This function removes a codec's callback function from an internal list.

param mgr AVDTP manager. param codec_type Codec type. The codec_type can be one of the following values: @arg AVDTP_CODEC_TYPE_SBC: SBC @arg AVDTP_CODEC_TYPE_MPEG1_2_AUDIO: MPEG-1,2 (used in MP3 files) @arg AVDTP_CODEC_TYPE_MPEG2_4_AAC: MPEG-2,4 AAC (used in Apple products) @arg AVDTP_CODEC_TYPE_ATRAC: ATRAC (used in Sony products) @arg AVDTP_CODEC_TYPE_NON_A2DP: Non-A2DP Codec

return li c TRUE if the function succeeds. li c FALSE otherwise. The function fails if a callback for a codec type specified in the p codec parameter li has not been previously registered with bt_avdtp_register_codec.

_bt_avdtp_add_param_uintn Function

File

avdtp_control.h

C

```
bt_bool _bt_avdtp_add_param_uintn(bt_avdtp_control_cmd_t* command, bt_uint value);
```

Description

This is function _bt_avdtp_add_param_uintn.

_bt_avdtp_read_caps Function

File

avdtp_private.h

C

```
bt_byte _bt_avdtp_read_caps(bt_avdtp_mgr_t* mgr, bt_avdtp_sep_capabilities_t* caps, bt_byte* data, bt_int
len, bt_int* offset, bt_byte* err_category);
```

Description

This is function _bt_avdtp_read_caps.

bt_avdtp_get_l2cap_channel Function

File

avdtp.h

C

```
bt_l2cap_channel_t* bt_avdtp_get_l2cap_channel(bt_avdtp_mgr_t* mgr, bt_byte strm_handle, bt_byte
transport_session);
```

Description

This is function bt_avdtp_get_l2cap_channel.

bt_avdtp_clear_media_tx_queue Function

File

avdtp.h

C

```
bt_bool bt_avdtp_clear_media_tx_queue(bt_avdtp_mgr_t* mgr, bt_byte strm_handle);
```

Description

brief Clear send queue ingroup avdtp

details When the consumer of AVDTP wants to send a packet to a remote device it calls bt_avdtp_add_media_tx_buffer function. The function adds the packet to a queue and tells AVDTP that it has something to send. The packet will be send as soon as the stream goes to AVDTP_STREAM_STATE_STREAMING state. The consumer can remove all packets from the queue before they have been sent to a remote device by calling ::bt_avdtp_clear_media_tx_queue.

param mgr AVDTP manager. param strm_handle Stream handle.

return li c TRUE if the function succeeds. li c FALSE otherwise. The function fails only if a stream specified by the p strm_handle parameter li does not exist. The stream can be in any state to call this function.

bt_avdtp_get_all_capabilities Function

File

avdtp.h

C

```
bt_bool bt_avdtp_get_all_capabilities(bt_avdtp_mgr_t* mgr, bt_bdaddr_t* remote_addr, bt_byte seid_acp);
```

Description

brief Get remote SEP capabilities. ingroup avdtp

details This function asks the remote device to send capabilities of a SEP specified by the p seid_acp. As a result of this operation the following

events will be generated: @arg AVDTP_EVT_SEP_CAPABILITIES_RECEIVED: this event is generated if the remote device accepted the request. the p evt_param.sep_capabilities_received contains SEP capabilities. @arg AVDTP_EVT_GET_SEP_CAPABILITIES_COMPLETED: this event is generated right after AVDTP_EVT_SEP_CAPABILITIES_RECEIVED if the remote accepted the request the p evt_param.get_sep_capabilities_completed.err_code == AVDTP_ERROR_SUCCESS. if the remote rejected the request the p evt_param.get_sep_capabilities_completed.err_code == the error code sent by the remote.

param mgr AVDTP manager. param remote_addr The address of a remote device. param seid_acp The ID of a remote SEP.

return li c TRUE if discover request has been sent. li c FALSE otherwise. No events will be generated.

bt_avdtp_get_stream_direction Function

File

avdtp.h

C

```
bt_byte bt_avdtp_get_stream_direction(bt_avdtp_mgr_t* mgr, bt_byte strm_handle);
```

Description

This is function bt_avdtp_get_stream_direction.

bt_avdtp_report_delay Function

File

avdtp.h

C

```
bt_bool bt_avdtp_report_delay(bt_avdtp_mgr_t* mgr, bt_byte strm_handle, bt_uint delay);
```

Description

brief Report delay value of a Sink to a Source. ingroup avdtp

details This function sends the delay value of a Sink to a Source. This enables synchronous playback of audio and video. Delay reports are always sent from the Sink to the Source. If the Sink's delay report has been accepted by the Source the p evt_param.delay_report_completed.err_code == AVDTP_ERROR_SUCCESS. Otherwise the p evt_param.delay_report_completed.err_code == the error code sent by the Source.

param mgr AVDTP manager. param strm_handle Stream handle. param delay The delay value in 1/10 milliseconds.

return li c TRUE if the function succeeds, i.e. the actual request has been sent to the remote party. li c FALSE otherwise. No events will be generated.

bt_avdtp_set_media_tx_queue_limit Function

File

avdtp.h

C

```
bt_bool bt_avdtp_set_media_tx_queue_limit(bt_avdtp_mgr_t* mgr, bt_byte strm_handle, bt_uint limit);
```

Description

brief Set limit on the send queue ingroup avdtp

details When the consumer of AVDTP wants to send a packet to a remote device it calls bt_avdtp_add_media_tx_buffer function. The function adds the packet to a queue and tells AVDTP that it has something to send. The packet will be send as soon as the stream goes to AVDTP_STREAM_STATE_STREAMING state. By default the send queue can contain unlimited number of packets. The consumer can set a limit on how many packets are held in the queue. In this case when new packet is added to the queue and the length of the queue exceeds the set limit the first packet is removed from the queue. The removed packet is not send to the remote device.

param mgr AVDTP manager. param strm_handle Stream handle. param limit Queue limit.

return li c TRUE if the function succeeds. li c FALSE otherwise. The function fails only if a stream specified by the p strm_handle parameter li does not exist. The stream can be in any state to call this function.

AVDTP Data Types and Constants

_bt_avdtp_codec_op_param_u Union

File

avdtp.h

C

```
union _bt_avdtp_codec_op_param_u {
  bt_avdtp_codec_op_parse_config_t parse;
  bt_avdtp_codec_op_serialize_config_t serialize;
  bt_avdtp_codec_op_parse_packet_t parse_packet;
};
```

Members

Members	Description
bt_avdtp_codec_op_parse_config_t parse;	< Valid if operation is AVDTP_CODEC_OPCODE_PARSE_CONFIG.
bt_avdtp_codec_op_serialize_config_t serialize;	< Valid if operation is AVDTP_CODEC_OPCODE_SERIALIZE_CONFIG.
bt_avdtp_codec_op_parse_packet_t parse_packet;	< This member is currently not used.

Description

brief Parameter to a codec handler. ingroup avdtp

details This union is used to pass operation specific data to a codec handler. Which member of the union points to a valid structure depends on the operation.

_bt_avdtp_sep_t Structure

File

avdtp.h

C

```
struct _bt_avdtp_sep_t {
  bt_avdtp_sep_t* next_sep;
  bt_byte id;
  bt_byte type;
  const bt_avdtp_sep_capabilities_t* caps;
  bt_byte state;
};
```

Members

Members	Description
bt_avdtp_sep_t* next_sep;	< Pointer to next SEP
bt_byte id;	< ID of the SEP
bt_byte type;	< Type of the SEP. This can be one of the following values: < li AVDTP_SEP_TYPE_SOURCE < li AVDTP_SEP_TYPE_SINK
const bt_avdtp_sep_capabilities_t* caps;	< SEP capabilities
bt_byte state;	< State of the SEP buffer. This can be one of the following values: < li AVDTP_SEP_STATE_FREE < li AVDTP_SEP_STATE_IDLE

Description

brief SEP description ingroup avdtp

details This structure is used to hold information about SEPs available on a local device.

_bt_media_packet_t Structure

File

avdtp.h

C

```
struct _bt_media_packet_t {
  bt_media_packet_t* next_packet;
```

```
  bt_byte version;
  bt_byte csrc_count;
  bt_bool marker;
  bt_byte payload_type;
  bt_uint seq_number;
  bt_ulong timestamp;
  bt_ulong ssrc;
  bt_ulong* csrc_list;
  bt_byte* data;
  bt_int data_len;
  bt_int max_data_len;
  bt_cp_header_t cp_header;
};
```

Members

Members	Description
bt_media_packet_t* next_packet;	< Pointer to next buffer
bt_byte version;	< Version of the RTP implementation.
bt_byte csrc_count;	< The CSRC count contains the number of CSRC identifiers that follow the fixed header.
bt_bool marker;	< The interpretation of the marker is defined by a profile. It is < intended to allow significant events such as frame boundaries to < be marked in the packet stream.
bt_byte payload_type;	< This field identifies the format of the RTP payload and determines its interpretation by the application. < A profile specifies default static mapping of payload type codes to payload formats.
bt_uint seq_number;	< The sequence number increments by one for each media packet sent, and may be used by the receiver to detect packet loss and to restore packet sequence.
bt_ulong timestamp;	< The Time Stamp reflects the sampling instant of the first octet in the media packet.
bt_ulong ssrc;	< The SSRC field identifies the synchronization source. This < identifier is chosen randomly, with the intent that no two synchronization < sources, within the same media transport < session, shall have the same SSRC identifier.
bt_ulong* csrc_list;	< The CSRC list identifies the contributing sources for the payload contained in this packet.
bt_byte* data;	< Pointer to a buffer to store media data. This pointer must be allocated by the AVDTP consumer < before adding this structure to receive (::bt_avdtp_add_media_rx_buffer) or send queue (::bt_avdtp_add_media_tx_buffer).
bt_int data_len;	< Lentgth of the media data.
bt_int max_data_len;	< Maximum length of the media data that can be stored in the buffer pointed by the c data member.
bt_cp_header_t cp_header;	< Content protection header.

Description

brief Media packet buffer ingroup avdtp

details This structure is used to receive and send media packet from/to the remote device. See more information about usage of this structure in descriptions of ::bt_avdtp_add_media_rx_buffer and ::bt_avdtp_add_media_tx_buffer.

bt_avdtp_codec_handler_fp Type

File

avdtp.h

C

```
typedef bt_byte (* bt_avdtp_codec_handler_fp)(struct _bt_avdtp_codec_t* codec, bt_byte opcode,
bt_avdtp_codec_op_param_t* op_param, struct _bt_avdtp_mgr_t* mgr);
```

Description

brief Codec handler. ingroup avdtp

details AVDTP in theory can support any type of codec. Each codec uses its own format for exchanging capabilities and configuration information. In order to make out implementation do not care about these formats we use a simple way of telling AVDTP how to parse and serialize codec's configuration. The consumer of AVDTP (e.g. A2DP) for each codec it wishes to support has to register a callback function (one per codec type). That callback has to perform two function. The first one is to read the configuration received from the remote device and store it in a structure defined by the consumer. The second one is to serialize the data from a structure to a format (in case of standard A2DP codecs the format is defined in A2DP specification, vendor specific codecs can define their own formats) suitable for sending as a part of a AVDTP request.n This typedef defines the interace for the callback function.

param codec A pointer to a structure that describes a codec. param opcode The code of an operation to execute. The c opcode can be one of the following values: @arg AVDTP_CODEC_OPCODE_PARSE_CONFIG: The handler has to parse configuration received from the remote device

and store it in a structure defined by the consumer. @arg AVDTP_CODEC_OPCODE_SERIALIZE_CONFIG: The handler has to serialize the data from a consumer defined structure to a format suitable for sending as a part of a AVDTP request. param op_param A pointer to the operation's specific parameters. Parameters are passed as a pointer to the bt_avdtp_codec_op_param_t union. Which member of the union points to a valid structure depends on the value of the c opcode: @arg bt_avdtp_codec_op_param_t::parse: Valid if c opcode == AVDTP_CODEC_OPCODE_PARSE_CONFIG. @arg bt_avdtp_codec_op_param_t::serialize: Valid if c opcode == AVDTP_CODEC_OPCODE_SERIALIZE_CONFIG. param mgr AVDTP manager.

bt_avdtp_codec_op_decode_t Structure

File

avdtp.h

C

```
typedef struct _bt_avdtp_codec_op_decode_t {
  void* codec_config;
  bt_byte* in_buffer;
  bt_int in_len;
  bt_byte* out_buffer;
  bt_int out_len;
} bt_avdtp_codec_op_decode_t;
```

Description

There is currently no use for this structure.

bt_avdtp_codec_op_encode_t Structure

File

avdtp.h

C

```
typedef struct _bt_avdtp_codec_op_encode_t {
  void* codec_config;
  bt_byte* in_buffer;
  bt_int in_len;
  bt_byte* out_buffer;
  bt_int out_len;
} bt_avdtp_codec_op_encode_t;
```

Description

There is currently no use for this structure.

bt_avdtp_codec_op_param_t Type

File

avdtp.h

C

```
typedef union _bt_avdtp_codec_op_param_u bt_avdtp_codec_op_param_t;
```

Description

This is type bt_avdtp_codec_op_param_t.

bt_avdtp_codec_op_parse_config_t Structure

File

avdtp.h

C

```
typedef struct _bt_avdtp_codec_op_parse_config_t {
  void* codec_config;
  bt_byte codec_config_max_size;
  bt_byte* buffer;
  bt_int buffer_len;
```

```
  bt_int offset;
} bt_avdtp_codec_op_parse_config_t;
```

Members

Members	Description
void* codec_config;	< A pointer to a structure defined by the AVDTP consumer where codec's configuration < will be stored by the handler. The format of the structure is totally up to < the AVDTP consumer. The dotstack defines such structures for SBC, MPEG-1,2 and MPEG-2,4 AAC: < @arg SBC: bt_a2dp_sbc_config_t (defined in a2dp_sbc_codec.h) < @arg MPEG-1,2: bt_a2dp_mpeg_config_t (defined in a2dp_mpeg_codec.h) < @arg MPEG-2,4 AAC: bt_a2dp_aac_config_t (defined in a2dp_aac_codec.h)
bt_byte codec_config_max_size;	< The maximum size of a buffer pointed to by c codec_config field.
bt_byte* buffer;	< A pointer to a buffer holding codec's configuration in OTA format.
bt_int buffer_len;	< The length of the c buffer.
bt_int offset;	< The c buffer points to a complete packet received from the remoted device. The c offset < points to a location in the c buffer where codec's configuration starts.

Description

brief Parameter to AVDTP_CODEC_OPCODE_PARSE_CONFIG operation. ingroup avdtp

details A pointer to this structure is passed to the codec handler as a valid member of the bt_avdtp_codec_op_param_t union - bt_avdtp_codec_op_param_t::parse - when ADVDTP needs to parse codec's capabilities/configuration received from the remote device.

bt_avdtp_codec_op_parse_packet_t Structure

File

avdtp.h

C

```
typedef struct _bt_avdtp_codec_op_parse_packet_t {
  void* packet_info;
  bt_media_packet_t* packet;
} bt_avdtp_codec_op_parse_packet_t;
```

Description

There is currently no use for this structure.

bt_avdtp_codec_op_serialize_config_t Structure

File

avdtp.h

C

```
typedef struct _bt_avdtp_codec_op_serialize_config_t {
  void* codec_config;
  bt_byte* buffer;
  bt_int buffer_len;
  bt_int offset;
} bt_avdtp_codec_op_serialize_config_t;
```

Members

Members	Description
void* codec_config;	< A pointer to a structure defined by the AVDTP consumer where codec's configuration < will be read from by the handler. The format of the structure is totally up to < the AVDTP consumer. The dotstack defines such structures for SBC, MPEG-1,2 and MPEG-2,4 AAC: < @arg SBC: bt_a2dp_sbc_config_t (defined in a2dp_sbc_codec.h) < @arg MPEG-1,2: bt_a2dp_mpeg_config_t (defined in a2dp_mpeg_codec.h) < @arg MPEG-2,4 AAC: bt_a2dp_aac_config_t (defined in a2dp_aac_codec.h)
bt_byte* buffer;	< A pointer to a buffer where the handler has to write codec's configuration in OTA format.
bt_int buffer_len;	< The length of the c buffer.
bt_int offset;	< The c buffer points to a complete packet that will be sent to the remote device. The c offset < points to a location in the c buffer where codec's configuration has to be written.

Description

brief Parameter to AVDTP_CODEC_OPCODE_SERIALIZE_CONFIG operation. ingroup avdtp

details A pointer to this structure is passed to the codec handler as a valid member of the bt_avdtp_codec_op_param_t union - bt_avdtp_codec_op_param_t::serialize - when ADVDTP needs to serialize codec's capabilities/configuration for sending to the remote device.

bt_avdtp_codec_t Structure

File

avdtp.h

C

```
typedef struct _bt_avdtp_codec_t {
  struct _bt_avdtp_codec_t* next_codec;
  bt_byte codec_type;
  bt_avdtp_codec_handler_fp codec_handler;
} bt_avdtp_codec_t;
```

Members

Members	Description
bt_byte codec_type;	< Codec type. < The c codec_type can be one of the following values: < @arg AVDTP_CODEC_TYPE_SBC: SBC < @arg AVDTP_CODEC_TYPE_MPEG1_2_AUDIO: MPEG-1,2 (used in MP3 files) < @arg AVDTP_CODEC_TYPE_MPEG2_4_AAC: MPEG-2,4 AAC (used in Apple products) < @arg AVDTP_CODEC_TYPE_ATRAC: ATRAC (used in Sony products) < @arg AVDTP_CODEC_TYPE_NON_A2DP: Non-A2DP Codec
bt_avdtp_codec_handler_fp codec_handler;	< A pointer to a codec handler.

Description

brief Codec handler description. ingroup avdtp

details This structure is used to register a codec handler for parsing/serializing codec capabilities and configuration. See description of the ::bt_avdtp_register_codec for more details.

bt_avdtp_control_channel_t Structure

File

avdtp_control.h

C

```
typedef struct _bt_avdtp_control_channel_t {
  bt_byte state;
  bt_bdaddr_t remote_addr;
  bt_l2cap_channel_t* l2cap_channel;
  bt_byte* tx_buffer;
  bt_queue_element_t* send_cq_head;
  bt_queue_element_t* ack_cq_head;
  bt_byte next_transaction_id;
  bt_byte next_ts_id;
  bt_byte next_tc_id;
  struct _bt_avdtp_mgr_t* avdtp_mgr;
  bt_byte remaining_connect_attemtps;
  bt_bdaddr_t connect_address;
  bt_uint connect_acl_config;
} bt_avdtp_control_channel_t;
```

Description

This is type bt_avdtp_control_channel_t.

bt_avdtp_control_cmd_t Structure

File

avdtp_control.h

C

```
typedef struct _bt_avdtp_control_cmd_t {
  struct _bt_avdtp_control_cmd_t* next_cmd;
  bt_byte code;
  bt_byte type;
  bt_byte transaction_id;
  bt_byte* params;
  bt_uint params_len;
} bt_avdtp_control_cmd_t;
```

Description

This is type bt_avdtp_control_cmd_t.

bt_avdtp_ctrl_evt_data_received_t Structure

File

avdtp_control.h

C

```
typedef struct _bt_avdtp_ctrl_evt_data_received_t {
  bt_byte* data;
  bt_int len;
} bt_avdtp_ctrl_evt_data_received_t;
```

Description

This is type bt_avdtp_ctrl_evt_data_received_t.

bt_avdtp_event_t Union

File

avdtp.h

C

```
typedef union _bt_avdtp_event_u {
  bt_avdtp_evt_ctrl_channel_connected_t ctrl_channel_connected;
  bt_avdtp_evt_ctrl_connection_failed_t ctrl_connection_failed;
  bt_avdtp_evt_ctrl_channel_disconnected_t ctrl_channel_disconnected;
  bt_avdtp_evt_discover_completed_t discover_completed;
  bt_avdtp_evt_sep_info_received_t sep_info_received;
  bt_avdtp_evt_get_sep_capabilities_completed_t get_sep_capabilities_completed;
  bt_avdtp_evt_sep_capabilities_received_t sep_capabilities_received;
  bt_avdtp_evt_set_stream_configuration_completed_t set_stream_configuration_completed;
  bt_avdtp_evt_get_stream_configuration_completed_t get_stream_configuration_completed;
  bt_avdtp_evt_stream_reconfigure_completed_t stream_reconfigure_completed;
  bt_avdtp_evt_open_stream_completed_t open_stream_completed;
  bt_avdtp_evt_start_stream_completed_t start_stream_completed;
  bt_avdtp_evt_close_stream_completed_t close_stream_completed;
  bt_avdtp_evt_suspend_stream_completed_t suspend_stream_completed;
  bt_avdtp_evt_stream_security_control_completed_t security_control_completed;
  bt_avdtp_evt_set_stream_configuration_requested_t set_stream_configuration_requested;
  bt_avdtp_evt_reconfigure_stream_requested_t reconfigure_stream_requested;
  bt_avdtp_evt_open_stream_requested_t open_stream_requested;
  bt_avdtp_evt_start_stream_requested_t start_stream_requested;
  bt_avdtp_evt_suspend_stream_requested_t suspend_stream_requested;
  bt_avdtp_evt_close_stream_requested_t close_stream_requested;
  bt_avdtp_evt_abort_stream_requested_t abort_stream_requested;
  bt_avdtp_evt_delay_report_completed_t delay_report_completed;
  bt_avdtp_evt_set_stream_configuration_t set_stream_configuration;
  bt_avdtp_evt_stream_configured_t stream_configured;
  bt_avdtp_evt_stream_reconfigured_t stream_reconfigured;
  bt_avdtp_evt_stream_opened_t stream_opened;
  bt_avdtp_evt_stream_started_t stream_started;
  bt_avdtp_evt_stream_suspended_t stream_suspended;
  bt_avdtp_evt_stream_closed_t stream_closed;
  bt_avdtp_evt_stream_aborted_t stream_aborted;
  bt_avdtp_evt_media_packet_received_t media_packet_received;
  bt_avdtp_evt_media_packet_sent_t media_packet_sent;
```

```
  bt_avdtp_evt_media_packet_send_failed_t media_packet_send_failed;
} bt_avdtp_event_t;
```

Members

Members	Description
bt_avdtp_evt_ctrl_channel_connected_t ctrl_channel_connected;	< Valid if event is AVDTP_EVT_CTRL_CHANNEL_CONNECTED
bt_avdtp_evt_ctrl_connection_failed_t ctrl_connection_failed;	< Valid if event is AVDTP_EVT_CTRL_CONNECTION_FAILED
bt_avdtp_evt_ctrl_channel_disconnected_t ctrl_channel_disconnected;	< Valid if event is AVDTP_EVT_CTRL_CHANNEL_DISCONNECTED
bt_avdtp_evt_discover_completed_t discover_completed;	< Valid if event is AVDTP_EVT_DISCOVER_COMPLETED
bt_avdtp_evt_sep_info_received_t sep_info_received;	< Valid if event is AVDTP_EVT_SEP_INFO_RECEIVED
bt_avdtp_evt_get_sep_capabilities_completed_t get_sep_capabilities_completed;	< Valid if event is AVDTP_EVT_GET_SEP_CAPABILITIES_COMPLETED
bt_avdtp_evt_sep_capabilities_received_t sep_capabilities_received;	< Valid if event is AVDTP_EVT_SEP_CAPABILITIES_RECEIVED
bt_avdtp_evt_set_stream_configuration_completed_t set_stream_configuration_completed;	< Valid if event is AVDTP_EVT_SET_STREAM_CONFIGURATION_COMPLETED
bt_avdtp_evt_get_stream_configuration_completed_t get_stream_configuration_completed;	< Valid if event is AVDTP_EVT_GET_STREAM_CONFIGURATION_COMPLETED
bt_avdtp_evt_stream_reconfigure_completed_t stream_reconfigure_completed;	< Valid if event is AVDTP_EVT_STREAM_RECONFIGURE_COMPLETED
bt_avdtp_evt_open_stream_completed_t open_stream_completed;	< Valid if event is AVDTP_EVT_OPEN_STREAM_COMPLETED
bt_avdtp_evt_start_stream_completed_t start_stream_completed;	< Valid if event is AVDTP_EVT_START_STREAM_COMPLETED
bt_avdtp_evt_close_stream_completed_t close_stream_completed;	< Valid if event is AVDTP_EVT_CLOSE_STREAM_COMPLETED
bt_avdtp_evt_suspend_stream_completed_t suspend_stream_completed;	< Valid if event is AVDTP_EVT_SUSPEND_STREAM_COMPLETED
bt_avdtp_evt_stream_security_control_completed_t security_control_completed;	< Valid if event is AVDTP_EVT_STREAM_SECURITY_CONTROL_COMPLETED
bt_avdtp_evt_set_stream_configuration_requested_t set_stream_configuration_requested;	< Valid if event is AVDTP_EVT_SET_STREAM_CONFIGURATION_REQUESTED
bt_avdtp_evt_reconfigure_stream_requested_t reconfigure_stream_requested;	< Valid if event is AVDTP_EVT_RECONFIGURE_STREAM_REQUESTED
bt_avdtp_evt_open_stream_requested_t open_stream_requested;	< Valid if event is AVDTP_EVT_OPEN_STREAM_REQUESTED
bt_avdtp_evt_start_stream_requested_t start_stream_requested;	< Valid if event is AVDTP_EVT_START_STREAM_REQUESTED
bt_avdtp_evt_suspend_stream_requested_t suspend_stream_requested;	< Valid if event is AVDTP_EVT_SUSPEND_STREAM_REQUESTED
bt_avdtp_evt_close_stream_requested_t close_stream_requested;	< Valid if event is AVDTP_EVT_CLOSE_STREAM_REQUESTED
bt_avdtp_evt_abort_stream_requested_t abort_stream_requested;	< Valid if event is AVDTP_EVT_ABORT_STREAM_REQUESTED
bt_avdtp_evt_delay_report_completed_t delay_report_completed;	< Valid if event is AVDTP_EVT_DELAYREPORT_COMPLETED
bt_avdtp_evt_stream_configured_t stream_configured;	< Valid if event is AVDTP_EVT_STREAM_CONFIGURED
bt_avdtp_evt_stream_reconfigured_t stream_reconfigured;	< Valid if event is AVDTP_EVT_STREAM_RECONFIGURED
bt_avdtp_evt_stream_opened_t stream_opened;	< Valid if event is AVDTP_EVT_STREAM_OPENED
bt_avdtp_evt_stream_started_t stream_started;	< Valid if event is AVDTP_EVT_STREAM_STARTED
bt_avdtp_evt_stream_suspended_t stream_suspended;	< Valid if event is AVDTP_EVT_STREAM_SUSPENDED
bt_avdtp_evt_stream_closed_t stream_closed;	< Valid if event is AVDTP_EVT_STREAM_CLOSED
bt_avdtp_evt_stream_aborted_t stream_aborted;	< Valid if event is AVDTP_EVT_STREAM_ABORTED
bt_avdtp_evt_media_packet_received_t media_packet_received;	< Valid if event is AVDTP_EVT_MEDIA_PACKET_RECEIVED

bt_avdtp_evt_media_packet_sent_t media_packet_sent;	< Valid if event is AVDTP_EVT_MEDIA_PACKET_SENT
bt_avdtp_evt_media_packet_send_failed_t media_packet_send_failed;	< Valid if event is AVDTP_EVT_MEDIA_PACKET_SEND_FAILED

Description

brief Parameter to an application callback. ingroup avdtp

details This union is used to pass event specific data to the AVDTP consumer. Which member of the union points to a valid structure depends on the event reported to the consumer. In general, each event has a corresponding member in the union.

bt_avdtp_evt_abort_stream_requested_t Structure

File

avdtp.h

C

```
typedef struct _bt_avdtp_evt_abort_stream_requested_t {
  bt_byte err_code;
  bt_byte strm_handle;
} bt_avdtp_evt_abort_stream_requested_t;
```

Members

Members	Description
bt_byte err_code;	< The result to be sent to the remote party. < li If local device accepts the configuration requested by the remote device < it should set c err_code to AVDTP_ERROR_SUCCESS. Otherwise it should set c err_code < to one of the AVDTP_ERROR_ constants.
bt_byte strm_handle;	< The handle of a stream to abort.

Description

brief Parameter to AVDTP_EVT_ABORT_STREAM_REQUESTED event ingroup avdtp

details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::abort_stream_requested - when AVDTP received a "abort stream" request.

bt_avdtp_evt_close_stream_completed_t Structure

File

avdtp.h

C

```
typedef struct _bt_avdtp_evt_close_stream_completed_t {
  bt_byte err_code;
  bt_byte strm_handle;
} bt_avdtp_evt_close_stream_completed_t;
```

Members

Members	Description
bt_byte err_code;	< The result of the request. < li If the remote accepted the request c err_code == AVDTP_ERROR_SUCCESS. < li Otherwise c err_code == the error code returned by the remote party.
bt_byte strm_handle;	< Stream handle.

Description

brief Parameter to AVDTP_EVT_CLOSE_STREAM_COMPLETED event ingroup avdtp

details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::close_stream_completed - when AVDTP received a response to a "close stream" request.

bt_avdtp_evt_close_stream_requested_t Structure

File

avdtp.h

C

```
typedef struct _bt_avdtp_evt_close_stream_requested_t {
  bt_byte err_code;
  bt_byte strm_handle;
} bt_avdtp_evt_close_stream_requested_t;
```

Members

Members	Description
bt_byte err_code;	< The result to be sent to the remote party. < li If local device accepts the configuration requested by the remote device < it should set c err_code to AVDTP_ERROR_SUCCESS. Otherwise it should set c err_code < to one of the AVDTP_ERROR_ constants.
bt_byte strm_handle;	< The handle of a stream to close.

Description

brief Parameter to AVDTP_EVT_CLOSE_STREAM_REQUESTED event ingroup avdtp

details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::close_stream_requested - when AVDTP received a "close stream" request.

bt_avdtp_evt_ctrl_channel_connected_t Structure

File

avdtp.h

C

```
typedef struct _bt_avdtp_evt_ctrl_channel_connected_s {
  bt_bdaddr_t* bdaddr;
} bt_avdtp_evt_ctrl_channel_connected_t;
```

Members

Members	Description
bt_bdaddr_t* bdaddr;	< the address of a remote device

Description

brief Parameter to AVDTP_EVT_CTRL_CHANNEL_CONNECTED event ingroup avdtp

details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::ctrl_channel_connected - when a control channel between two devices has been established.

bt_avdtp_evt_ctrl_channel_disconnected_t Structure

File

avdtp.h

C

```
typedef struct _bt_avdtp_evt_ctrl_channel_disconnected_s {
  bt_bdaddr_t* bdaddr;
} bt_avdtp_evt_ctrl_channel_disconnected_t;
```

Members

Members	Description
bt_bdaddr_t* bdaddr;	< the address of a remote device

Description

brief Parameter to AVDTP_EVT_CTRL_CHANNEL_DISCONNECTED event ingroup avdtp

details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::ctrl_channel_disconnected - when a control channel between two devices has been terminated.

bt_avdtp_evt_discover_completed_t Structure

File

avdtp.h

C

```
typedef struct _bt_avdtp_evt_discover_completed_t {
  bt_byte err_code;
  bt_bdaddr_t* bdaddr;
} bt_avdtp_evt_discover_completed_t;
```

Members

Members	Description
bt_byte err_code;	< The result of discovering. < li If the remote accepted the request c err_code == AVDTP_ERROR_SUCCESS. < li Otherwise c err_code == the error code returned by the remote party.
bt_bdaddr_t* bdaddr;	< the address of a remote device

Description

brief Parameter to AVDTP_EVT_DISCOVER_COMPLETED event ingroup avdtp

details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::discover_completed - when AVDTP completed discovering SEPs available on a remote device.

bt_avdtp_evt_get_sep_capabilities_completed_t Structure

File

avdtp.h

C

```
typedef struct _bt_avdtp_evt_get_sep_capabilities_completed_t {
  bt_byte err_code;
  bt_bdaddr_t* bdaddr;
} bt_avdtp_evt_get_sep_capabilities_completed_t;
```

Members

Members	Description
bt_byte err_code;	< The result of the request. < li If the remote accepted the request c err_code == AVDTP_ERROR_SUCCESS. < li Otherwise c err_code == the error code returned by the remote party.
bt_bdaddr_t* bdaddr;	< the address of a remote device

Description

brief Parameter to AVDTP_EVT_GET_SEP_CAPABILITIES_COMPLETED event ingroup avdtp

details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::get_sep_capabilities_completed - when AVDTP received a response to a "get SEP capabilities" request. AVDTP_EVT_GET_SEP_CAPABILITIES_COMPLETED only informs the status of the request - success or failure. In case of success another event - AVDTP_EVT_SEP_CAPABILITIES_RECEIVED - is generate with a pointer to a structure that holds actual SEP's capabilities.

bt_avdtp_evt_get_stream_configuration_completed_t Structure

File

avdtp.h

C

```
typedef struct _bt_avdtp_evt_get_stream_configuration_completed_t {
  bt_byte err_code;
  bt_byte strm_handle;
} bt_avdtp_evt_get_stream_configuration_completed_t;
```

Description

brief Parameter to AVDTP_EVT_SET_STREAM_CONFIGURATION_COMPLETED event ingroup avdtp

details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::get_stream_configuration_completed - when AVDTP received a response to a "get stream configuration" request. AVDTP_EVT_SET_STREAM_CONFIGURATION_COMPLETED only informs the status of the request - success or failure. In case of success another event - AVDTP_EVT_STREAM_CONFIGURATION_RECEIVED - is generate with a pointer to a structure that hold actual stream's configuration.

bt_avdtp_evt_media_packet_received_t Structure

File

avdtp.h

C

```
typedef struct _bt_avdtp_evt_media_packet_received_t {
  bt_byte strm_handle;
  bt_media_packet_t* packet;
} bt_avdtp_evt_media_packet_received_t;
```

Members

Members	Description
bt_byte strm_handle;	< The handle of a stream which received a packet.
bt_media_packet_t* packet;	< A pointer to a media packet buffer.

Description

brief Parameter to AVDTP_EVT_MEDIA_PACKET_RECEIVED event ingroup avdtp

details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::media_packet_received - when ADVDTP received a media packet from the remote device.

bt_avdtp_evt_media_packet_send_failed_t Structure

File

avdtp.h

C

```
typedef struct _bt_avdtp_evt_media_packet_send_failed_t {
  bt_byte strm_handle;
  bt_media_packet_t* packet;
} bt_avdtp_evt_media_packet_send_failed_t;
```

Members

Members	Description
bt_byte strm_handle;	< The handle of a stream which received a packet.
bt_media_packet_t* packet;	< A pointer to a media packet buffer.

Description

brief Parameter to AVDTP_EVT_MEDIA_PACKET_SEND_FAILED event ingroup avdtp

details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::media_packet_send_failed - when ADVDTP failed to send a media packet to the remote device.

bt_avdtp_evt_media_packet_sent_t Structure

File

avdtp.h

C

```
typedef struct _bt_avdtp_evt_media_packet_sent_t {
  bt_byte strm_handle;
  bt_media_packet_t* packet;
} bt_avdtp_evt_media_packet_sent_t;
```

Members

Members	Description
bt_byte strm_handle;	< The handle of a stream which received a packet.
bt_media_packet_t* packet;	< A pointer to a media packet buffer.

Description

brief Parameter to AVDTP_EVT_MEDIA_PACKET_SENT event ingroup avdtp

details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::media_packet_sent - when ADVDTP sent a media packet to the remote device.

bt_avdtp_evt_open_stream_completed_t Structure

File

avdtp.h

C

```
typedef struct _bt_avdtp_evt_open_stream_completed_t {
  bt_byte err_code;
  bt_byte strm_handle;
} bt_avdtp_evt_open_stream_completed_t;
```

Members

Members	Description
bt_byte err_code;	< The result of the request. < li If the remote accepted the request c err_code == AVDTP_ERROR_SUCCESS. < li Otherwise c err_code == the error code returned by the remote party.
bt_byte strm_handle;	< Stream handle.

Description

brief Parameter to AVDTP_EVT_OPEN_STREAM_COMPLETED event ingroup avdtp

details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::open_stream_completed - when AVDTP received a response to a "open stream" request.

bt_avdtp_evt_open_stream_requested_t Structure

File

avdtp.h

C

```
typedef struct _bt_avdtp_evt_open_stream_requested_t {
  bt_byte err_code;
  bt_byte strm_handle;
} bt_avdtp_evt_open_stream_requested_t;
```

Members

Members	Description
bt_byte err_code;	< The result to be sent to the remote party. < li If local device accepts the configuration requested by the remote device < it should set c err_code to AVDTP_ERROR_SUCCESS. Otherwise it should set c err_code < to one of the AVDTP_ERROR_ constants.
bt_byte strm_handle;	< The handle of a stream to open.

Description

brief Parameter to AVDTP_EVT_OPEN_STREAM_REQUESTED event ingroup avdtp

details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::open_stream_requested - when AVDTP received a "open stream" request.

bt_avdtp_evt_reconfigure_stream_requested_t Structure

File

avdtp.h

C

```
typedef struct _bt_avdtp_evt_reconfigure_stream_requested_t {
  bt_byte err_code;
  bt_byte err_category;
  bt_avdtp_sep_t* sep;
  bt_avdtp_sep_capabilities_t* config;
  bt_byte strm_handle;
} bt_avdtp_evt_reconfigure_stream_requested_t;
```

Members

Members	Description
bt_byte err_code;	< The result to be sent to the remote party. < li If local device accepts the configuration requested by the remote device < it should set c err_code to AVDTP_ERROR_SUCCESS. Otherwise it should set c err_code < to one of the AVDTP_ERROR_ constants.
bt_byte err_category;	< If local device cannot accept the request it should set c err_category to the value of the forst Service Category that failed.
bt_avdtp_sep_t* sep;	< Description of the local SEP.
bt_avdtp_sep_capabilities_t* config;	< Stream configuration requested by the remote party.
bt_byte strm_handle;	< Stream handle.

Description

brief Parameter to AVDTP_EVT_RECONFIGURE_STREAM_REQUESTED event ingroup avdtp

details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::reconfigure_stream_requested - when AVDTP received a "change stream configuration" request.

bt_avdtp_evt_sep_capabilities_received_t Structure

File

avdtp.h

C

```
typedef struct _bt_avdtp_evt_sep_capabilities_received_t {
  bt_avdtp_sep_capabilities_t* caps;
  bt_bdaddr_t* bdaddr;
} bt_avdtp_evt_sep_capabilities_received_t;
```

Members

Members	Description
bt_avdtp_sep_capabilities_t* caps;	< SEP capabilities or stream configuration.

Description

brief Parameter to AVDTP_EVT_SEP_CAPABILITIES_RECEIVED and AVDTP_EVT_STREAM_CONFIGURATION_RECEIVED events ingroup avdtp

details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::sep_capabilities_received - when AVDTP received a positive response to a "get SEP capabilities" or "get stream configuration" request.

bt_avdtp_evt_sep_info_received_t Structure

File

avdtp.h

C

```
typedef struct _bt_avdtp_evt_sep_info_received_t {
  bt_byte sep_id;
```

```
  bt_byte sep_type;
  bt_bool in_use;
  bt_byte media_type;
  bt_bdaddr_t* bdaddr;
} bt_avdtp_evt_sep_info_received_t;
```

Members

Members	Description
bt_byte sep_id;	< SEP ID.
bt_byte sep_type;	< SEP type. This can be either AVDTP_SEP_TYPE_SOURCE or AVDTP_SEP_TYPE_SINK.
bt_bool in_use;	< A flag indicating if a SEP is already being used.
bt_byte media_type;	< Type of media supported by this SEP. This can be on of the following values: < li AVDTP_MEDIA_TYPE_AUDIO < li AVDTP_MEDIA_TYPE_VIDEO < li AVDTP_MEDIA_TYPE_MULTIMEDIA

Description

brief Parameter to AVDTP_EVT_SEP_INFO_RECEIVED event ingroup avdtp

details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::sep_info_received - when AVDTP received positive result to a "discover" request. AVDTP_EVT_SEP_INFO_RECEIVED is generated for every SEP received from the remote device.

bt_avdtp_evt_set_stream_configuration_completed_t Structure

File

avdtp.h

C

```
typedef struct _bt_avdtp_evt_set_stream_configuration_completed_t {
  bt_byte err_code;
  bt_byte strm_handle;
  bt_byte svc_category;
} bt_avdtp_evt_set_stream_configuration_completed_t;
```

Members

Members	Description
bt_byte err_code;	< The result of the request. < li If the remote accepted the request c err_code == AVDTP_ERROR_SUCCESS. < li Otherwise c err_code == the error code returned by the remote party.
bt_byte strm_handle;	< Stream handle.
bt_byte svc_category;	< The value of the first Service Category to fail if the remote rejected the request.

Description

brief Parameter to AVDTP_EVT_SET_STREAM_CONFIGURATION_COMPLETED event ingroup avdtp

details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::set_stream_configuration_completed - when AVDTP received a response to a "set stream configuration" request.

bt_avdtp_evt_set_stream_configuration_requested_t Structure

File

avdtp.h

C

```
typedef struct _bt_avdtp_evt_set_stream_configuration_requested_t {
  bt_byte err_code;
  bt_byte err_category;
  bt_avdtp_sep_t* sep;
  bt_byte int_sep_id;
  bt_avdtp_sep_capabilities_t* config;
  bt_byte strm_handle;
} bt_avdtp_evt_set_stream_configuration_requested_t;
```

Members

Members	Description
bt_byte err_code;	< The result to be sent to the remote party. < li If local device accepts the configuration requested by the remote device < it should set c err_code to AVDTP_ERROR_SUCCESS. Otherwise it should set c err_code < to one of the AVDTP_ERROR_ constants.
bt_byte err_category;	< If local device cannot accept the request it should set c err_category to the value of the forst Service Category that failed.
bt_avdtp_sep_t* sep;	< Description of the local SEP.
bt_byte int_sep_id;	< The ID of the remote SEP.
bt_avdtp_sep_capabilities_t* config;	< Stream configuration requested by the remote party.
bt_byte strm_handle;	< Stream handle.

Description

brief Parameter to AVDTP_EVT_SET_STREAM_CONFIGURATION_REQUESTED event ingroup avdtp

details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::set_stream_configuration_requested - when AVDTP received a "set stream configuration" request.

bt_avdtp_evt_start_stream_completed_t Structure

File

avdtp.h

C

```
typedef struct _bt_avdtp_evt_start_stream_completed_t {
  bt_byte err_code;
  bt_byte strm_handle;
} bt_avdtp_evt_start_stream_completed_t;
```

Members

Members	Description
bt_byte err_code;	< The result of the request. < li If the remote accepted the request c err_code == AVDTP_ERROR_SUCCESS. < li Otherwise c err_code == the error code returned by the remote party.
bt_byte strm_handle;	< Stream handle.

Description

brief Parameter to AVDTP_EVT_START_STREAM_COMPLETED event ingroup avdtp

details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::start_stream_completed - when AVDTP received a response to a "start stream" request.

bt_avdtp_evt_start_stream_requested_t Structure

File

avdtp.h

C

```
typedef struct _bt_avdtp_evt_start_stream_requested_t {
  bt_byte err_code;
  bt_byte strm_handle;
} bt_avdtp_evt_start_stream_requested_t;
```

Members

Members	Description
bt_byte err_code;	< The result to be sent to the remote party. < li If local device accepts the configuration requested by the remote device < it should set c err_code to AVDTP_ERROR_SUCCESS. Otherwise it should set c err_code < to one of the AVDTP_ERROR_ constants.
bt_byte strm_handle;	< The handle of a stream to start.

Description

brief Parameter to AVDTP_EVT_START_STREAM_REQUESTED event ingroup avdtp

details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::start_stream_requested - when AVDTP received a "start stream" request.

bt_avdtp_evt_stream_aborted_t Structure

File

avdtp.h

C

```
typedef struct _bt_avdtp_evt_stream_aborted_t {
  bt_byte strm_handle;
} bt_avdtp_evt_stream_aborted_t;
```

Members

Members	Description
bt_byte strm_handle;	< The handle of a stream that has been aborted.

Description

brief Parameter to AVDTP_EVT_STREAM_ABORTED event ingroup avdtp

details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::stream_closed - to notify the AVDTP consumer that a stream has been successfully aborted.

bt_avdtp_evt_stream_closed_t Structure

File

avdtp.h

C

```
typedef struct _bt_avdtp_evt_stream_closed_t {
  bt_byte strm_handle;
} bt_avdtp_evt_stream_closed_t;
```

Members

Members	Description
bt_byte strm_handle;	< The handle of a stream that has been closed.

Description

brief Parameter to AVDTP_EVT_STREAM_CLOSED event ingroup avdtp

details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::stream_closed - to notify the AVDTP consumer that a stream has been successfully closed.

bt_avdtp_evt_stream_configured_t Structure

File

avdtp.h

C

```
typedef struct _bt_avdtp_evt_stream_configured_t {
  bt_byte strm_handle;
} bt_avdtp_evt_stream_configured_t;
```

Members

Members	Description
bt_byte strm_handle;	< The handle of a stream that has been configured.

Description

brief Parameter to AVDTP_EVT_STREAM_CONFIGURED event ingroup avdtp

details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::stream_configured - to notify the AVDTP consumer that a stream configuration has been successfully completed.

bt_avdtp_evt_stream_opened_t Structure

File

avdtp.h

C

```
typedef struct _bt_avdtp_evt_stream_opened_t {
  bt_byte strm_handle;
} bt_avdtp_evt_stream_opened_t;
```

Members

Members	Description
bt_byte strm_handle;	< The handle of a stream that has been opened.

Description

brief Parameter to AVDTP_EVT_STREAM_OPENED event ingroup avdtp

details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::stream_opened - to notify the AVDTP consumer that a stream has been successfully opened.

bt_avdtp_evt_stream_reconfigure_completed_t Structure

File

avdtp.h

C

```
typedef struct _bt_avdtp_evt_stream_reconfigure_completed_t {
  bt_byte err_code;
  bt_byte strm_handle;
  bt_byte svc_category;
} bt_avdtp_evt_stream_reconfigure_completed_t;
```

Members

Members	Description
bt_byte err_code;	< The result of the request. < li If the remote accepted the request c err_code == AVDTP_ERROR_SUCCESS. < li Otherwise c err_code == the error code returned by the remote party.
bt_byte strm_handle;	< Stream handle.
bt_byte svc_category;	< The value of the first Service Category to fail if the remote rejected the request.

Description

brief Parameter to AVDTP_EVT_STREAM_RECONFIGURE_COMPLETED event ingroup avdtp

details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::stream_reconfigure_completed - when AVDTP received a response to a "change stream configuration" request.

bt_avdtp_evt_stream_reconfigured_t Structure

File

avdtp.h

C

```
typedef struct _bt_avdtp_evt_stream_reconfigured_t {
  bt_byte strm_handle;
} bt_avdtp_evt_stream_reconfigured_t;
```

Members

Members	Description
bt_byte strm_handle;	< The handle of a stream that has been reconfigured.

Description

brief Parameter to AVDTP_EVT_STREAM_RECONFIGURED event ingroup avdtp

details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::stream_reconfigured - to notify the AVDTP consumer that a stream configuration has been successfully changed.

bt_avdtp_evt_stream_security_control_completed_t Structure

File

avdtp.h

C

```
typedef struct _bt_avdtp_evt_stream_security_control_completed_t {
  bt_byte err_code;
} bt_avdtp_evt_stream_security_control_completed_t;
```

Members

Members	Description
bt_byte err_code;	< The result of the request. < li If the remote accepted the request c err_code == AVDTP_ERROR_SUCCESS. < li Otherwise c err_code == the error code returned by the remote party.

Description

brief Parameter to AVDTP_EVT_STREAM_SECURITY_CONTROL_COMPLETED event ingroup avdtp

details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::security_control_completed - when AVDTP received a response to a "exchange content protection control data" request.

bt_avdtp_evt_stream_started_t Structure

File

avdtp.h

C

```
typedef struct _bt_avdtp_evt_stream_started_t {
  bt_byte strm_handle;
} bt_avdtp_evt_stream_started_t;
```

Members

Members	Description
bt_byte strm_handle;	< The handle of a stream that has been started.

Description

brief Parameter to AVDTP_EVT_STREAM_STARTED event ingroup avdtp

details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::stream_started - to notify the AVDTP consumer that a stream has been successfully started.

bt_avdtp_evt_stream_suspended_t Structure

File

avdtp.h

C

```
typedef struct _bt_avdtp_evt_stream_suspended_t {
  bt_byte strm_handle;
} bt_avdtp_evt_stream_suspended_t;
```

Members

Members	Description
bt_byte strm_handle;	< The handle of a stream that has been suspended.

Description

brief Parameter to AVDTP_EVT_STREAM_SUSPENDED event ingroup avdtp

details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::stream_suspended - to notify the AVDTP consumer that a stream has been successfully suspended.

bt_avdtp_evt_suspend_stream_completed_t Structure

File

avdtp.h

C

```
typedef struct _bt_avdtp_evt_suspend_stream_completed_t {
  bt_byte err_code;
  bt_byte strm_handle;
} bt_avdtp_evt_suspend_stream_completed_t;
```

Members

Members	Description
bt_byte err_code;	< The result of the request. < li If the remote accepted the request c err_code == AVDTP_ERROR_SUCCESS. < li Otherwise c err_code == the error code returned by the remote party.
bt_byte strm_handle;	< Stream handle.

Description

brief Parameter to AVDTP_EVT_SUSPEND_STREAM_COMPLETED event ingroup avdtp

details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::suspend_stream_completed - when AVDTP received a response to a "suspend stream" request.

bt_avdtp_evt_suspend_stream_requested_t Structure

File

avdtp.h

C

```
typedef struct _bt_avdtp_evt_suspend_stream_requested_t {
  bt_byte err_code;
  bt_byte strm_handle;
} bt_avdtp_evt_suspend_stream_requested_t;
```

Members

Members	Description
bt_byte err_code;	< The result to be sent to the remote party. < li If local device accepts the configuration requested by the remote device < it should set c err_code to AVDTP_ERROR_SUCCESS. Otherwise it should set c err_code < to one of the AVDTP_ERROR_ constants.
bt_byte strm_handle;	< The handle of a stream to suspend.

Description

brief Parameter to AVDTP_EVT_SUSPEND_STREAM_REQUESTED event ingroup avdtp

details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::suspend_stream_requested - when AVDTP received a "suspend stream" request.

bt_avdtp_mgr_callback_fp Type

File

avdtp.h

C

```
typedef void (* bt_avdtp_mgr_callback_fp)(struct _bt_avdtp_mgr_t* mgr, bt_byte evt, bt_avdtp_event_t*
evt_param, void* callback_param);
```

Description

brief AVDTP application callback. ingroup avdtp

details In order to be notified of various events a consumer of the AVDTP layer has to register a callback function. The stack will call that function whenever a new event has been generated.

param mgr AVDTP manager.

param evt AVDTP event. The event can be one of the following values: @arg AVDTP_EVT_CTRL_CHANNEL_CONNECTED: Control channel connected. @arg AVDTP_EVT_CTRL_CHANNEL_DISCONNECTED: Control channel disconnected. @arg AVDTP_EVT_CTRL_CONNECTION_FAILED: Control channel connection failed (generated only if control connection has been initiated by the local device). @arg AVDTP_EVT_DISCOVER_COMPLETED: Local device completed discovering remote SEPs. @arg AVDTP_EVT_GET_SEP_CAPABILITIES_COMPLETED: Local device received a response to Get SEP capabilities operation. @arg AVDTP_EVT_SET_STREAM_CONFIGURATION_COMPLETED: Local device received a response to Set stream configuration operation. @arg AVDTP_EVT_GET_STREAM_CONFIGURATION_COMPLETED: Local device received a response to Get stream configuration operation. @arg AVDTP_EVT_STREAM_RECONFIGURE_COMPLETED: Local device received a response to Reconfigure stream operation. @arg AVDTP_EVT_OPEN_STREAM_COMPLETED: Local device received a response to Open stream operation. @arg AVDTP_EVT_START_STREAM_COMPLETED: Local device received a response to Start stream operation. @arg AVDTP_EVT_CLOSE_STREAM_COMPLETED: Local device received a response to Close stream operation. @arg AVDTP_EVT_SUSPEND_STREAM_COMPLETED: Local device received a response to Suspend stream operation. @arg AVDTP_EVT_STREAM_SECURITY_CONTROL_COMPLETED: Local device received a response to Stream security control operation. @arg AVDTP_EVT_ABORT_STREAM_COMPLETED: Local device received a response to Abort stream operation. @arg AVDTP_EVT_SEP_INFO_RECEIVED: SEP information received. @arg AVDTP_EVT_SEP_CAPABILITIES_RECEIVED: SEP capabilities received. @arg AVDTP_EVT_STREAM_CONFIGURATION_RECEIVED: Stream configuration received.

@arg AVDTP_EVT_SET_STREAM_CONFIGURATION_REQUESTED: Remote device requested stream configuration. @arg AVDTP_EVT_OPEN_STREAM_REQUESTED: Remote device requested to open a stream. @arg AVDTP_EVT_START_STREAM_REQUESTED: Remote device requested to start a stream. @arg AVDTP_EVT_CLOSE_STREAM_REQUESTED: Remote device requested to close a stream. @arg AVDTP_EVT_SUSPEND_STREAM_REQUESTED: Remote device requested to suspend a stream. @arg AVDTP_EVT_ABORT_STREAM_REQUESTED: Remote device requested to abort a stream. @arg AVDTP_EVT_RECONFIGURE_STREAM_REQUESTED: Remote device requested to reconfigure a stream. @arg AVDTP_EVT_MEDIA_PACKET_RECEIVED: Remote device sent a media packet. @arg AVDTP_EVT_STREAM_CONFIGURED: A stream has been configured (This event is generated right after AVDTP_EVT_SET_STREAM_CONFIGURATION_REQUESTED if the local devices accepted the request). @arg AVDTP_EVT_STREAM_RECONFIGURED: A stream has been re-configured (This event is generated right after AVDTP_EVT_RECONFIGURE_STREAM_REQUESTED if the local devices accepted the request). @arg AVDTP_EVT_STREAM_OPENED: A stream has been opened (This event is generated as a result of local or remote stream opening request). @arg AVDTP_EVT_STREAM_STARTED: A stream has been started (This event is generated right after AVDTP_EVT_START_STREAM_REQUESTED if the local devices accepted the request). @arg AVDTP_EVT_STREAM_CLOSED: A stream has been close (This event is generated right after AVDTP_EVT_CLOSE_STREAM_REQUESTED if the local devices accepted the request). @arg AVDTP_EVT_STREAM_SUSPENDED: A stream has been suspended (This event is generated right after AVDTP_EVT_SUSPEND_STREAM_REQUESTED if the local devices accepted the request). @arg AVDTP_EVT_STREAM_ABORTED: A stream has been aborted (This event is generated right after AVDTP_EVT_SUSPEND_STREAM_REQUESTED if the local devices accepted the request. It is also generated if connection between devices has been terminated by means other than AVDTP signaling, e.g. devices going out of rage). @arg AVDTP_EVT_MEDIA_PACKET_SENT: The local device has successfully sent a media packet to the remote device. @arg AVDTP_EVT_MEDIA_PACKET_SEND_FAILED: The local device was not able to send a media packet to the remote device.

param evt_param Event parameter. Which member of the bt_avdtp_event_t union is valid depends on the event: @arg AVDTP_EVT_CTRL_CHANNEL_CONNECTED: c bt_avdtp_evt_ctrl_channel_connected_t ctrl_channel_connected @arg AVDTP_EVT_CTRL_CHANNEL_DISCONNECTED: c bt_avdtp_evt_ctrl_channel_disconnected_t ctrl_channel_disconnected @arg AVDTP_EVT_CTRL_CONNECTION_FAILED: c NULL @arg AVDTP_EVT_DISCOVER_COMPLETED: c bt_avdtp_evt_discover_completed_t discover_completed; @arg AVDTP_EVT_GET_SEP_CAPABILITIES_COMPLETED: c bt_avdtp_evt_get_sep_capabilities_completed_t get_sep_capabilities_completed @arg AVDTP_EVT_SET_STREAM_CONFIGURATION_COMPLETED: c bt_avdtp_evt_set_stream_configuration_completed_t set_stream_configuration_completed @arg AVDTP_EVT_GET_STREAM_CONFIGURATION_COMPLETED: c bt_avdtp_evt_get_stream_configuration_completed_t get_stream_configuration_completed @arg AVDTP_EVT_STREAM_RECONFIGURE_COMPLETED: c bt_avdtp_evt_stream_reconfigure_completed_t stream_reconfigure_completed @arg AVDTP_EVT_OPEN_STREAM_COMPLETED: c bt_avdtp_evt_open_stream_completed_t open_stream_completed @arg AVDTP_EVT_START_STREAM_COMPLETED: c bt_avdtp_evt_start_stream_completed_t start_stream_completed @arg AVDTP_EVT_CLOSE_STREAM_COMPLETED: c bt_avdtp_evt_close_stream_completed_t close_stream_completed @arg AVDTP_EVT_SUSPEND_STREAM_COMPLETED: c bt_avdtp_evt_suspend_stream_completed_t suspend_stream_completed @arg AVDTP_EVT_STREAM_SECURITY_CONTROL_COMPLETED: c bt_avdtp_evt_stream_security_control_completed_t security_control_completed @arg AVDTP_EVT_ABORT_STREAM_COMPLETED: c bt_avdtp_evt_abort_stream_requested_t abort_stream_requested @arg AVDTP_EVT_SEP_INFO_RECEIVED: c bt_avdtp_evt_sep_info_received_t sep_info_received @arg AVDTP_EVT_SEP_CAPABILITIES_RECEIVED: c bt_avdtp_evt_sep_capabilities_received_t sep_capabilities_received @arg AVDTP_EVT_STREAM_CONFIGURATION_RECEIVED: c bt_avdtp_evt_sep_capabilities_received_t sep_capabilities_received

@arg AVDTP_EVT_SET_STREAM_CONFIGURATION_REQUESTED: c bt_avdtp_evt_set_stream_configuration_requested_t set_stream_configuration_requested @arg AVDTP_EVT_OPEN_STREAM_REQUESTED: c bt_avdtp_evt_open_stream_requested_t open_stream_requested @arg AVDTP_EVT_START_STREAM_REQUESTED: c bt_avdtp_evt_start_stream_requested_t start_stream_requested @arg AVDTP_EVT_CLOSE_STREAM_REQUESTED: c bt_avdtp_evt_close_stream_requested_t close_stream_requested @arg AVDTP_EVT_SUSPEND_STREAM_REQUESTED: c bt_avdtp_evt_suspend_stream_requested_t suspend_stream_requested @arg

AVDTP_EVT_ABORT_STREAM_REQUESTED: c bt_avdtp_evt_abort_stream_requested_t abort_stream_requested @arg AVDTP_EVT_RECONFIGURE_STREAM_REQUESTED: c bt_avdtp_evt_reconfigure_stream_requested_t reconfigure_stream_requested @arg AVDTP_EVT_MEDIA_PACKET_RECEIVED: c bt_avdtp_evt_media_packet_received_t media_packet_received @arg AVDTP_EVT_STREAM_CONFIGURED: c bt_avdtp_evt_stream_configured_t stream_configured @arg AVDTP_EVT_STREAM_RECONFIGURED: c bt_avdtp_evt_stream_reconfigured_t stream_reconfigured @arg AVDTP_EVT_STREAM_OPENED: c bt_avdtp_evt_stream_opened_t stream_opened @arg AVDTP_EVT_STREAM_STARTED: c bt_avdtp_evt_stream_started_t stream_started @arg AVDTP_EVT_STREAM_CLOSED: c bt_avdtp_evt_stream_closed_t stream_closed @arg AVDTP_EVT_STREAM_SUSPENDED: c bt_avdtp_evt_stream_suspended_t stream_suspended @arg AVDTP_EVT_STREAM_ABORTED: c bt_avdtp_evt_stream_aborted_t stream_aborted @arg AVDTP_EVT_MEDIA_PACKET_SENT: c bt_avdtp_evt_media_packet_sent_t media_packet_sent @arg AVDTP_EVT_MEDIA_PACKET_SEND_FAILED: c bt_avdtp_evt_media_packet_send_failed_t media_packet_send_failed

param callback_param A pointer to an arbitrary data set by a call to bt_avdtp_register_callback.

bt_avdtp_mgr_t Structure

File

avdtp.h

C

```
typedef struct _bt_avdtp_mgr_t {
  bt_byte state;
  bt_byte flags;
  bt_avdtp_sep_t* seps;
  bt_byte next_sep_id;
  bt_avdtp_stream_t* streams;
  bt_byte next_stream_handle;
  bt_avdtp_control_channel_t* control_channels;
  bt_avdtp_transport_channel_t* transport_channels;
  bt_avdtp_codec_t* codecs;
  bt_avdtp_mgr_callback_fp callback;
  void* callback_param;
  bt_avdtp_control_cmd_t* pending_command;
  bt_avdtp_stream_t* opening_strm;
  bt_avdtp_stream_t* sending_strm;
} bt_avdtp_mgr_t;
```

Members

Members	Description
bt_byte state;	< Manager state. This value can be one of the following values: < li AVDTP_MANAGER_STATE_IDLE < li AVDTP_MANAGER_STATE_CONNECTING
bt_byte flags;	< Additional manager state. This value can be a combination of the following values: < li AVDTP_MANAGER_FLAG_SENDING_MEDIA_PACKET
bt_avdtp_sep_t* seps;	< Pointer to a buffer of SEPs available for allocating with c bt_avdtp_register_sep.
bt_byte next_sep_id;	< Holds ID of the next SEP to be allocated. Every time c bt_avdtp_register_sep is called this value is increased by 1.
bt_avdtp_stream_t* streams;	< Pointer to a buffer of streams available for allocating with c bt_avdtp_create_stream.
bt_byte next_stream_handle;	< Holds ID of the next stream to be allocated. Every time c bt_avdtp_create_stream is called this value is increased by 1.
bt_avdtp_control_channel_t* control_channels;	< Pointer to a buffer of available control channles.
bt_avdtp_transport_channel_t* transport_channels;	< Pointer to a buffer of available transport channles.
bt_avdtp_codec_t* codecs;	< A list of supported codecs.
bt_avdtp_mgr_callback_fp callback;	< Pointer to a function which a AVDTP consumer must register in order to be notified of various events.
void* callback_param;	< Pointer to arbitrary data to be passed to the c callback.
bt_avdtp_control_cmd_t* pending_command;	< If local device wants to send a command to a remote device but control channel does not exists this member holds a pointer to the command until the channel is created.
bt_avdtp_stream_t* opening_strm;	< Holds a pointer to a stream being opened by a remote device.
bt_avdtp_stream_t* sending_strm;	< Holds a pointer to a stream sending media packet.

Description

brief AVDTP manager. ingroup avdtp

details A structure that glues all pieces together. There is only one instance of this structure allocated by dotstack. A pointer to the instance can be get with c bt_avdtp_get_mgr().

bt_avdtp_sep_capabilities_t Structure

File

avdtp.h

C

```
typedef struct _bt_avdtp_sep_capabilities_t {
  bt_byte categories;
  bt_byte media_type;
  bt_byte codec_type;
  void* codec_config;
  bt_uint cp_type;
  bt_byte* cp_specific;
  bt_byte cp_specific_len;
  bt_byte recovery_type;
  bt_byte max_recovery_window;
  bt_byte max_parity_code_packets;
  bt_byte header_compression_options;
  bt_byte multiplexing_options;
  bt_byte tsid_media;
  bt_byte tcid_media;
  bt_byte tsid_reporting;
  bt_byte tcid_reporting;
  bt_byte tsid_recovery;
  bt_byte tcid_recovery;
} bt_avdtp_sep_capabilities_t;
```

Members

Members	Description
bt_byte categories;	< Defines service capabilities exposed by a SEP to a remote party. This can be a combination of the following values: < li AVDTP_SEP_CAPABILITY_FLAG_MEDIA_TRANSPORT < li AVDTP_SEP_CAPABILITY_FLAG_REPORTING < li AVDTP_SEP_CAPABILITY_FLAG_RECOVERY < li AVDTP_SEP_CAPABILITY_FLAG_CONTENT_PROTECTION < li AVDTP_SEP_CAPABILITY_FLAG_HEADER_COMPRESSION < li AVDTP_SEP_CAPABILITY_FLAG_MULTIPLEXING < li AVDTP_SEP_CAPABILITY_FLAG_MEDIA_CODEC
bt_byte media_type;	< Type of media supported by this SEP. This can be on of the following values: < li AVDTP_MEDIA_TYPE_AUDIO < li AVDTP_MEDIA_TYPE_VIDEO < li AVDTP_MEDIA_TYPE_MULTIMEDIA
bt_byte codec_type;	< Codec type supported by this SEP. This can be on of the following values: < li AVDTP_CODEC_TYPE_SBC < li AVDTP_CODEC_TYPE_MPEG1_2_AUDIO < li AVDTP_CODEC_TYPE_MPEG2_4_AAC < li AVDTP_CODEC_TYPE_ATRAC < li AVDTP_CODEC_TYPE_NON_A2DP
void* codec_config;	< Pointer to a buffer that holds codec specific capabilities
bt_uint cp_type;	< Type of content protection supported by this SEP. < note Content protection is currently not supported by dotstack.
bt_byte* cp_specific;	< Pointer to a buffer holding content protection specific data. < note Content protection is currently not supported by dotstack.
bt_byte cp_specific_len;	< Length of the content protection specific data. < note Content protection is currently not supported by dotstack.
bt_byte recovery_type;	< Type of recovery supported by this SEP. < note Recovery is currently not supported by dotstack.
bt_byte max_recovery_window;	< Recovery window size. < note Recovery is currently not supported by dotstack.
bt_byte max_parity_code_packets;	< Maximum number of parity codec packets. < note Recovery is currently not supported by dotstack.
bt_byte header_compression_options;	< Header compression configuration < note Header compression is currently not supported by dotstack.
bt_byte multiplexing_options;	< Multiplexing configuration < note Multiplexing is currently not supported by dotstack.
bt_byte tsid_media;	< ID of the media transport session
bt_byte tcid_media;	< ID of the media transport channel
bt_byte tsid_reporting;	< ID of the reporting transport session
bt_byte tcid_reporting;	< ID of the reporting transport channel
bt_byte tsid_recovery;	< ID of the recovery transport session

bt_byte tcid_recovery;	< ID of the recovery transport channel

Description

brief SEP capabilities ingroup avdtp

details This structure is used to hold SEP capabilities.

bt_avdtp_sep_t Type

File

avdtp.h

C

```
typedef struct _bt_avdtp_sep_t bt_avdtp_sep_t;
```

Description

This is type bt_avdtp_sep_t.

bt_avdtp_stream_t Structure

File

avdtp.h

C

```
typedef struct _bt_avdtp_stream_t {
  struct _bt_avdtp_stream_t* next_stream;
  bt_byte handle;
  bt_byte state;
  bt_byte flags;
  struct _bt_avdtp_sep_t* sep;
  bt_avdtp_sep_capabilities_t* config;
  bt_byte remote_seid;
  bt_bdaddr_t remote_addr;
  bt_avdtp_transport_session_t sessions[AVDTP_MAX_STREAM_TRANSPORT_SESSION];
  struct _bt_avdtp_mgr_t* mgr;
  bt_byte cur_channel_index;
  bt_byte* listen_sep_list;
  bt_byte listen_sep_count;
  bt_queue_element_t* media_rx_queue;
  bt_queue_element_t* media_tx_queue;
  bt_uint media_tx_queue_limit;
} bt_avdtp_stream_t;
```

Members

Members	Description
struct _bt_avdtp_stream_t* next_stream;	< Pointer to next stream.
bt_byte handle;	< Stream handle. This values is allocated by dotstack and is used to manipulate the stream by the AVDTP consumer.
bt_byte state;	< State of the stream. This value can be one of the following values: < li AVDTP_STREAM_STATE_IDLE < li AVDTP_STREAM_OPENING_TRANSPORT_CHANNELS < li AVDTP_STREAM_CLOSING_TRANSPORT_CHANNELS < li AVDTP_STREAM_STATE_CONFIGURED < li AVDTP_STREAM_STATE_OPEN < li AVDTP_STREAM_STATE_STREAMING < li AVDTP_STREAM_STATE_CLOSING < li AVDTP_STREAM_STATE_ABORTING
bt_byte flags;	< Additional stream state. This value can be one of the following values: < li AVDTP_STREAM_FLAG_LISTENING
struct _bt_avdtp_sep_t* sep;	< Local SEP this stream is connected to.
bt_avdtp_sep_capabilities_t* config;	< Current SEP configuration.
bt_byte remote_seid;	< ID of the remote SEP.
bt_bdaddr_t remote_addr;	< BT address of the remote device.
bt_avdtp_transport_session_t sessions[AVDTP_MAX_STREAM_TRANSPORT_SESSION];	< List of transport session available/active on this stream. < note There can be up to 3 (AVDTP_MAX_STREAM_TRANSPORT_SESSION == 3) transport sessions on a stream - media, reporting and recovery. < dotstack supports only media sessions so the other two are never used.

struct _bt_avdtp_mgr_t* mgr;	< AVDTP manager this stream belongs to.
bt_byte cur_channel_index;	< This value is currenlty not used.
bt_byte* listen_sep_list;	< A list of SEPs this channel this channel is listening on (i.e. can accept incoming connection requests).
bt_byte listen_sep_count;	< The number of SEPs in c listen_sep_list.
bt_queue_element_t* media_rx_queue;	< A list of media packet buffer for receiving incoming packets.
bt_queue_element_t* media_tx_queue;	< A list of media packet buffer to be sent to a remote device.
bt_uint media_tx_queue_limit;	< Maximum number of packets that TX queue can hold. 0 - unlimited.

Description

brief Stream description ingroup avdtp

details This structure is used to hold information about streams available on a local device.

bt_avdtp_transport_channel_t Structure

File

avdtp.h

C

```
typedef struct _bt_avdtp_transport_channel_t {
  struct _bt_avdtp_transport_channel_t* next_channel;
  bt_byte id;
  bt_byte type;
  bt_l2cap_channel_t* l2cap_channel;
  bt_byte ref_count;
  bt_byte connect_ref_count;
} bt_avdtp_transport_channel_t;
```

Members

Members	Description
struct _bt_avdtp_transport_channel_t* next_channel;	< Pointer to next channel.
bt_byte id;	< ID of the channel.
bt_byte type;	< Type of the channel. This can be one of the following values: < li AVDTP_TRANSPORT_CHANNEL_TYPE_DEDICATED < li AVDTP_TRANSPORT_CHANNEL_TYPE_SHARED < note Shared channles (i.e. multiplexing) is currently not supported by dotstack
bt_l2cap_channel_t* l2cap_channel;	< L2CAp channel used to transfer this AVDTP channel's data.
bt_byte ref_count;	< Channel's reference count. This value is intended for use with shared channels. < When ref count reaches 0 the channel is closed. < note Shared channles (i.e. multiplexing) is currently not supported by dotstack
bt_byte connect_ref_count;	< This value is currently not used.

Description

brief Transport channel description ingroup avdtp

details This structure is used to hold information about transport channels available on a local device.

bt_avdtp_transport_op_callback_fp Type

File

avdtp_private.h

C

```
typedef void (* bt_avdtp_transport_op_callback_fp)(bt_byte status, bt_byte opcode, void* param);
```

Description

This is type bt_avdtp_transport_op_callback_fp.

bt_avdtp_transport_session_t Structure

File

avdtp.h

C

```
typedef struct _bt_avdtp_transport_session_t {
  bt_byte id;
  bt_byte type;
  bt_avdtp_transport_channel_t* transport_channel;
} bt_avdtp_transport_session_t;
```

Members

Members	Description
bt_byte id;	< ID of the transport session.
bt_byte type;	< Type of the transport session. This can be one of the following values: < li AVDTP_TRANSPORT_SESSION_TYPE_MEDIA < li AVDTP_TRANSPORT_SESSION_TYPE_REPORTING < li AVDTP_TRANSPORT_SESSION_TYPE_RECOVERY
bt_avdtp_transport_channel_t* transport_channel;	< Transport channel used for this transport session.

Description

brief Transport session description ingroup avdtp

details This structure is used to hold information about transport sessions available on a local device.

bt_media_packet_t Type

File

avdtp.h

C

```
typedef struct _bt_media_packet_t bt_media_packet_t;
```

Description

This is type bt_media_packet_t.

__AVDTP_CONFIG_H Macro

File

avdtp_config.h

C

```
#define __AVDTP_CONFIG_H
```

Description

This is macro __AVDTP_CONFIG_H.

__AVDTP_CONTROL_H Macro

File

avdtp_control.h

C

```
#define __AVDTP_CONTROL_H
```

Description

This is macro __AVDTP_CONTROL_H.

__AVDTP_H Macro

File

avdtp.h

C

```
#define __AVDTP_H
```

Description

This is macro __AVDTP_H.

__AVDTP_PRIVATE_H Macro

File

avdtp_private.h

C

```
#define __AVDTP_PRIVATE_H
```

Description

This is macro __AVDTP_PRIVATE_H.

AVDTP_CODEC_OPCODE_PARSE_CONFIG Macro

File

avdtp.h

C

```
#define AVDTP_CODEC_OPCODE_PARSE_CONFIG 0          ///< Parse codec configuration.
```

Description

< Parse codec configuration.

AVDTP_CODEC_OPCODE_PARSE_PACKET Macro

File

avdtp.h

C

```
#define AVDTP_CODEC_OPCODE_PARSE_PACKET 2
```

Description

This is macro AVDTP_CODEC_OPCODE_PARSE_PACKET.

AVDTP_CODEC_OPCODE_SERIALIZE_CONFIG Macro

File

avdtp.h

C

```
#define AVDTP_CODEC_OPCODE_SERIALIZE_CONFIG 1          ///< Serialize codec configuration.
```

Description

< Serialize codec configuration.

AVDTP_CODEC_TYPE_ATRAC Macro

File

avdtp.h

C

```
#define AVDTP_CODEC_TYPE_ATRAC 0x04 ///< ATRAC (proprietary codec owned by Sony Corporation).
```

Description

< ATRAC (proprietary codec owned by Sony Corporation).

AVDTP_CODEC_TYPE_MPEG1_2_AUDIO Macro

File

avdtp.h

C

```
#define AVDTP_CODEC_TYPE_MPEG1_2_AUDIO 0x01 ///< MPEG-1,2 (optional).
```

Description

< MPEG-1,2 (optional).

AVDTP_CODEC_TYPE_MPEG2_4_AAC Macro

File

avdtp.h

C

```
#define AVDTP_CODEC_TYPE_MPEG2_4_AAC 0x02 ///< MPEG-2,4 AAC (optional, used in Apple's products).
```

Description

< MPEG-2,4 AAC (optional, used in Apple's products).

AVDTP_CODEC_TYPE_NON_A2DP Macro

File

avdtp.h

C

```
#define AVDTP_CODEC_TYPE_NON_A2DP 0xFF ///< Vendor specific.
```

Description

< Vendor specific.

AVDTP_CODEC_TYPE_SBC Macro

File

avdtp.h

C

```
#define AVDTP_CODEC_TYPE_SBC 0x00 ///< SBC (mandatory to support in A2DP profile).
```

Description

< SBC (mandatory to support in A2DP profile).

AVDTP_CTLR_CHANNEL_EVT_CONNECTED Macro

File

avdtp_control.h

C

```
#define AVDTP_CTLR_CHANNEL_EVT_CONNECTED 1
```

Description

This is macro AVDTP_CTLR_CHANNEL_EVT_CONNECTED.

AVDTP_CTLR_CHANNEL_EVT_CONNECTION_FAILED Macro

File

avdtp_control.h

C

```
#define AVDTP_CTLR_CHANNEL_EVT_CONNECTION_FAILED 3
```

Description

This is macro AVDTP_CTLR_CHANNEL_EVT_CONNECTION_FAILED.

AVDTP_CTLR_CHANNEL_EVT_DISCONNECTED Macro

File

avdtp_control.h

C

```
#define AVDTP_CTLR_CHANNEL_EVT_DISCONNECTED 2
```

Description

This is macro AVDTP_CTLR_CHANNEL_EVT_DISCONNECTED.

AVDTP_CTLR_CHANNEL_EVT_NOTHING Macro

File

avdtp_control.h

C

```
#define AVDTP_CTLR_CHANNEL_EVT_NOTHING 0
```

Description

This is macro AVDTP_CTLR_CHANNEL_EVT_NOTHING.

AVDTP_CTRL_CHANNEL_EVT_DATA_RECEIVED Macro

File

avdtp_control.h

C

```
#define AVDTP_CTRL_CHANNEL_EVT_DATA_RECEIVED 4
```

Description

This is macro AVDTP_CTRL_CHANNEL_EVT_DATA_RECEIVED.

AVDTP_CTRL_CHANNEL_STATE_CONNECTED Macro

File

avdtp_control.h

C

```
#define AVDTP_CTRL_CHANNEL_STATE_CONNECTED 1
```

Description

This is macro AVDTP_CTRL_CHANNEL_STATE_CONNECTED.

AVDTP_CTRL_CHANNEL_STATE_CONNECTING Macro

File

avdtp_control.h

C

```
#define AVDTP_CTRL_CHANNEL_STATE_CONNECTING 2
```

Description

This is macro AVDTP_CTRL_CHANNEL_STATE_CONNECTING.

AVDTP_CTRL_CHANNEL_STATE_DISCONNECTED Macro

File

avdtp_control.h

C

```
#define AVDTP_CTRL_CHANNEL_STATE_DISCONNECTED 0
```

Description

This is macro AVDTP_CTRL_CHANNEL_STATE_DISCONNECTED.

AVDTP_CTRL_MESSAGE_TYPE_ACCEPT Macro

File

avdtp_control.h

C

```
#define AVDTP_CTRL_MESSAGE_TYPE_ACCEPT 2
```

Description

This is macro AVDTP_CTRL_MESSAGE_TYPE_ACCEPT.

AVDTP_CTRL_MESSAGE_TYPE_COMMAND Macro

File

avdtp_control.h

C

```
#define AVDTP_CTRL_MESSAGE_TYPE_COMMAND 0
```

Description

This is macro AVDTP_CTRL_MESSAGE_TYPE_COMMAND.

AVDTP_CTRL_MESSAGE_TYPE_FLD Macro

File

avdtp_control.h

C

```
#define AVDTP_CTRL_MESSAGE_TYPE_FLD 3
```

Description

This is macro AVDTP_CTRL_MESSAGE_TYPE_FLD.

AVDTP_CTRL_MESSAGE_TYPE_REJECT Macro

File

avdtp_control.h

C

```
#define AVDTP_CTRL_MESSAGE_TYPE_REJECT 3
```

Description

This is macro AVDTP_CTRL_MESSAGE_TYPE_REJECT.

AVDTP_CTRL_PACKET_TYPE_CONTINUE Macro

File

avdtp_control.h

C

```
#define AVDTP_CTRL_PACKET_TYPE_CONTINUE 0x08
```

Description

This is macro AVDTP_CTRL_PACKET_TYPE_CONTINUE.

AVDTP_CTRL_PACKET_TYPE_END Macro

File

avdtp_control.h

C

```
#define AVDTP_CTRL_PACKET_TYPE_END 0x0c
```

Description

This is macro AVDTP_CTRL_PACKET_TYPE_END.

AVDTP_CTRL_PACKET_TYPE_FLD Macro

File

avdtp_control.h

C

```
#define AVDTP_CTRL_PACKET_TYPE_FLD 0x0c
```

Description

This is macro AVDTP_CTRL_PACKET_TYPE_FLD.

AVDTP_CTRL_PACKET_TYPE_SIGNLE Macro

File

avdtp_control.h

C

```
#define AVDTP_CTRL_PACKET_TYPE_SIGNLE 0
```

Description

This is macro AVDTP_CTRL_PACKET_TYPE_SIGNLE.

AVDTP_CTRL_PACKET_TYPE_START Macro

File

avdtp_control.h

C

```
#define AVDTP_CTRL_PACKET_TYPE_START 0x04
```

Description

This is macro AVDTP_CTRL_PACKET_TYPE_START.

AVDTP_ERROR_BAD_ACP_SEID Macro

File

avdtp.h

C

```
#define AVDTP_ERROR_BAD_ACP_SEID 0x12    ///< The requested command indicates an invalid ACP SEP ID (not
addressable)
```

Description

< The requested command indicates an invalid ACP SEP ID (not addressable)

AVDTP_ERROR_BAD_CP_FORMAT Macro

File

avdtp.h

C

```
#define AVDTP_ERROR_BAD_CP_FORMAT 0x27    ///< The format of Content Protection Service Capability is not
correct.
```

Description

< The format of Content Protection Service Capability is not correct.

AVDTP_ERROR_BAD_HEADER_FORMAT Macro

File

avdtp.h

C

```
#define AVDTP_ERROR_BAD_HEADER_FORMAT 0x01    ///< The request packet header format is invalid.
```

Description

< The request packet header format is invalid.

AVDTP_ERROR_BAD_LENGTH Macro

File

avdtp.h

C

```
#define AVDTP_ERROR_BAD_LENGTH 0x11    ///< The request packet length is not match the assumed length.
```

Description

< The request packet length is not match the assumed length.

AVDTP_ERROR_BAD_MEDIA_TRANSPORT_FORMAT Macro

File

avdtp.h

C

```
#define AVDTP_ERROR_BAD_MEDIA_TRANSPORT_FORMAT 0x23    ///< The format of Media Transport Capability is not
correct.
```

Description

< The format of Media Transport Capability is not correct.

AVDTP_ERROR_BAD_MULTIPLEXING_FORMAT Macro

File

avdtp.h

C

```
#define AVDTP_ERROR_BAD_MULTIPLEXING_FORMAT 0x28    ///< The format of Multiplexing Service Capability is
not correct.
```

Description

< The format of Multiplexing Service Capability is not correct.

AVDTP_ERROR_BAD_PAYLOAD_FORMAT Macro

File

avdtp.h

C

```
#define AVDTP_ERROR_BAD_PAYLOAD_FORMAT 0x18    ///< The requested command has an incorrect payload format.
```

Description

< The requested command has an incorrect payload format.

AVDTP_ERROR_BAD_RECOVERY_FORMAT Macro

File

avdtp.h

C

```
#define AVDTP_ERROR_BAD_RECOVERY_FORMAT 0x25    ///< The format of Recovery Service Capability is not
correct.
```

Description

< The format of Recovery Service Capability is not correct.

AVDTP_ERROR_BAD_RECOVERY_TYPE Macro

File

avdtp.h

C

```
#define AVDTP_ERROR_BAD_RECOVERY_TYPE 0x22    ///< The requested Recovery Type is not defined in AVDTP.
```

Description

< The requested Recovery Type is not defined in AVDTP.

AVDTP_ERROR_BAD_ROHC_FORMAT Macro

File

avdtp.h

C

```
#define AVDTP_ERROR_BAD_ROHC_FORMAT 0x26    ///< The format of Header Compression Service Capability is not
correct.
```

Description

< The format of Header Compression Service Capability is not correct.

AVDTP_ERROR_BAD_SERV_CATEGORY Macro

File

avdtp.h

C

```
#define AVDTP_ERROR_BAD_SERV_CATEGORY 0x17    ///< The value of Service Category in the request packet is
not defined in AVDTP.
```

Description

< The value of Service Category in the request packet is not defined in AVDTP.

AVDTP_ERROR_BAD_STATE Macro

File

avdtp.h

C

```
#define AVDTP_ERROR_BAD_STATE 0x31    ///< The stream is in state that does not permit executing commands.
```

Description

< The stream is in state that does not permit executing commands.

AVDTP_ERROR_FAILED_TO_CONNECT_CONTROL Macro

File

avdtp.h

C

```
#define AVDTP_ERROR_FAILED_TO_CONNECT_CONTROL 0x41    ///< An attempt to establish a control channel has
failed.
```

Description

< An attempt to establish a control channel has failed.

AVDTP_ERROR_FAILED_TO_CONNECT_TRANSPORT Macro

File

avdtp.h

C

```
#define AVDTP_ERROR_FAILED_TO_CONNECT_TRANSPORT 0x40    ///< An attempt to establish a transport channel
has failed.
```

Description

< An attempt to establish a transport channel has failed.

AVDTP_ERROR_INVALID_CAPABILITIES Macro

File

avdtp.h

C

```
#define AVDTP_ERROR_INVALID_CAPABILITIES 0x1a    ///< The reconfigure command is an attempt to reconfigure
a transport service capabilities of the SEP. Reconfigure is only permitted for application service
capabilities.
```

Description

< The reconfigure command is an attempt to reconfigure a transport service capabilities of the SEP. Reconfigure is only permitted for application service capabilities.

AVDTP_ERROR_NOT_SUPPORTED_COMMAND Macro

File

avdtp.h

C

```
#define AVDTP_ERROR_NOT_SUPPORTED_COMMAND 0x19    ///< The requested command is not supported by the device.
```

Description

< The requested command is not supported by the device.

AVDTP_ERROR_SEP_IN_USE Macro

File

avdtp.h

C

```
#define AVDTP_ERROR_SEP_IN_USE 0x13    ///< The SEP is in use.
```

Description

< The SEP is in use.

AVDTP_ERROR_SEP_NOT_IN_USE Macro

File

avdtp.h

C

```
#define AVDTP_ERROR_SEP_NOT_IN_USE 0x14    ///< The SEP is not in use.
```

Description

< The SEP is not in use.

AVDTP_ERROR_SUCCESS Macro

File

avdtp.h

C

```
#define AVDTP_ERROR_SUCCESS 0       ///< The operation completed with no errors.
```

Description

< The operation completed with no errors.

AVDTP_ERROR_UNSUPPORTED_CONFIGURAION Macro

File

avdtp.h

C

```
#define AVDTP_ERROR_UNSUPPORTED_CONFIGURAION 0x29    ///< Configuration not supported.
```

Description

< Configuration not supported.

AVDTP_EVT_ABORT_STREAM_COMPLETED Macro

File

avdtp.h

C

```
#define AVDTP_EVT_ABORT_STREAM_COMPLETED 14    ///< This event is generated when a local device received a
response (either positive or negative) to a "abort stream" request.
```

Description

< This event is generated when a local device received a response (either positive or negative) to a "abort stream" request.

AVDTP_EVT_ABORT_STREAM_REQUESTED Macro

File

avdtp.h

C

```
#define AVDTP_EVT_ABORT_STREAM_REQUESTED 55    ///< This event is generated when a local device received
"abort stream" request.
```

Description

< This event is generated when a local device received "abort stream" request.

AVDTP_EVT_CLOSE_STREAM_COMPLETED Macro

File

avdtp.h

C

```
#define AVDTP_EVT_CLOSE_STREAM_COMPLETED 11    ///< This event is generated when a local device received a
response (either positive or negative) to a "close stream" request.
```

Description

< This event is generated when a local device received a response (either positive or negative) to a "close stream" request.

AVDTP_EVT_CLOSE_STREAM_REQUESTED Macro

File

avdtp.h

C

```
#define AVDTP_EVT_CLOSE_STREAM_REQUESTED 53    ///< This event is generated when a local device received
"close stream" request.
```

Description

< This event is generated when a local device received "close stream" request.

AVDTP_EVT_CTRL_CHANNEL_CONNECTED Macro

File

avdtp.h

C

```
#define AVDTP_EVT_CTRL_CHANNEL_CONNECTED 1     ///< This event is generated when a control channel between
two AVDTP entities has been established.
```

Description

< This event is generated when a control channel between two AVDTP entities has been established.

AVDTP_EVT_CTRL_CHANNEL_DISCONNECTED Macro

File

avdtp.h

C

```
#define AVDTP_EVT_CTRL_CHANNEL_DISCONNECTED 2     ///< This event is generated when a control channel
between two AVDTP entities has been terminated.
```

Description

< This event is generated when a control channel between two AVDTP entities has been terminated.

AVDTP_EVT_CTRL_CONNECTION_FAILED Macro

File

avdtp.h

C

```
#define AVDTP_EVT_CTRL_CONNECTION_FAILED 3     ///< This event is generated when a local device failed to
create a control channel between two AVDTP entities.
```

Description

< This event is generated when a local device failed to create a control channel between two AVDTP entities.

AVDTP_EVT_DISCOVER_COMPLETED Macro

File

avdtp.h

C

```
#define AVDTP_EVT_DISCOVER_COMPLETED 4     ///< This event is generated when a local device received a
response (either positive or negative) to a "discover" request.
```

Description

< This event is generated when a local device received a response (either positive or negative) to a "discover" request.

AVDTP_EVT_GET_SEP_CAPABILITIES_COMPLETED Macro

File

avdtp.h

C

```
#define AVDTP_EVT_GET_SEP_CAPABILITIES_COMPLETED 5     ///< This event is generated when a local device
received a response (either positive or negative) to a "get SEP capabilities" request.
```

Description

< This event is generated when a local device received a response (either positive or negative) to a "get SEP capabilities" request.

AVDTP_EVT_GET_STREAM_CONFIGURATION_COMPLETED Macro

File

avdtp.h

C

```
#define AVDTP_EVT_GET_STREAM_CONFIGURATION_COMPLETED 7     ///< This event is generated when a local device
received a response (either positive or negative) to a "get stream configuration" request.
```

Description

< This event is generated when a local device received a response (either positive or negative) to a "get stream configuration" request.

AVDTP_EVT_LAST Macro

File

avdtp.h

C

```
#define AVDTP_EVT_LAST 99
```

Description

This is macro AVDTP_EVT_LAST.

AVDTP_EVT_MEDIA_PACKET_RECEIVED Macro

File

avdtp.h

C

```
#define AVDTP_EVT_MEDIA_PACKET_RECEIVED 57    ///< This event is generated when a local device received a
media packet.
```

Description

< This event is generated when a local device received a media packet.

AVDTP_EVT_MEDIA_PACKET_SEND_FAILED Macro

File

avdtp.h

C

```
#define AVDTP_EVT_MEDIA_PACKET_SEND_FAILED 66    ///< This event is generated when a local device failed to
send a media packet.
```

Description

< This event is generated when a local device failed to send a media packet.

AVDTP_EVT_MEDIA_PACKET_SENT Macro

File

avdtp.h

C

```
#define AVDTP_EVT_MEDIA_PACKET_SENT 65    ///< This event is generated when a local device sent a media packet.
```

Description

< This event is generated when a local device sent a media packet.

AVDTP_EVT_NULL Macro

File

avdtp.h

C

```
#define AVDTP_EVT_NULL 0
```

Description

This is macro AVDTP_EVT_NULL.

AVDTP_EVT_OPEN_STREAM_COMPLETED Macro

File

avdtp.h

C

```
#define AVDTP_EVT_OPEN_STREAM_COMPLETED 9     ///< This event is generated when a local device received a response (either positive or negative) to a "open stream" request.
```

Description

< This event is generated when a local device received a response (either positive or negative) to a "open stream" request.

AVDTP_EVT_OPEN_STREAM_REQUESTED Macro

File

avdtp.h

C

```
#define AVDTP_EVT_OPEN_STREAM_REQUESTED 51    ///< This event is generated when a local device received "open stream" request.
```

Description

< This event is generated when a local device received "open stream" request.

AVDTP_EVT_RECONFIGURE_STREAM_REQUESTED Macro

File

avdtp.h

C

```
#define AVDTP_EVT_RECONFIGURE_STREAM_REQUESTED 56    ///< This event is generated when a local device received "change stream configuration" request.
```

Description

< This event is generated when a local device received "change stream configuration" request.

AVDTP_EVT_SEP_CAPABILITIES_RECEIVED Macro

File

avdtp.h

C

```
#define AVDTP_EVT_SEP_CAPABILITIES_RECEIVED 16    ///< This event is generated when a local device received
a positive response to a "get SEP capabilities" request.
```

Description

< This event is generated when a local device received a positive response to a "get SEP capabilities" request.

AVDTP_EVT_SEP_INFO_RECEIVED Macro

File

avdtp.h

C

```
#define AVDTP_EVT_SEP_INFO_RECEIVED 15    ///< This event is generated for each SEP contained in a positive
response to a "discover" request.
```

Description

< This event is generated for each SEP contained in a positive response to a "discover" request.

AVDTP_EVT_SET_STREAM_CONFIGURATION_COMPLETED Macro

File

avdtp.h

C

```
#define AVDTP_EVT_SET_STREAM_CONFIGURATION_COMPLETED 6     ///< This event is generated when a local device
received a response (either positive or negative) to a "set stream configuration" request.
```

Description

< This event is generated when a local device received a response (either positive or negative) to a "set stream configuration" request.

AVDTP_EVT_SET_STREAM_CONFIGURATION_REQUESTED Macro

File

avdtp.h

C

```
#define AVDTP_EVT_SET_STREAM_CONFIGURATION_REQUESTED 50    ///< This event is generated when a local device
received "set stream configuration" request.
```

Description

< This event is generated when a local device received "set stream configuration" request.

AVDTP_EVT_START_STREAM_COMPLETED Macro

File

avdtp.h

C

```
#define AVDTP_EVT_START_STREAM_COMPLETED 10    ///< This event is generated when a local device received a
response (either positive or negative) to a "start stream" request.
```

Description

< This event is generated when a local device received a response (either positive or negative) to a "start stream" request.

AVDTP_EVT_START_STREAM_REQUESTED Macro

File

avdtp.h

C

```
#define AVDTP_EVT_START_STREAM_REQUESTED 52    ///< This event is generated when a local device received
"start stream" request.
```

Description

< This event is generated when a local device received "start stream" request.

AVDTP_EVT_STREAM_ABORTED Macro

File

avdtp.h

C

```
#define AVDTP_EVT_STREAM_ABORTED 64    ///< This event is generated when a local device has successfully
aborted a stream.
```

Description

< This event is generated when a local device has successfully aborted a stream. < This event follows the AVDTP_EVT_ABORT_STREAM_REQUESTED if the upper layer has accepted it. < This event is not generated if stream abortion was initiated by the local device.

AVDTP_EVT_STREAM_CLOSED Macro

File

avdtp.h

C

```
#define AVDTP_EVT_STREAM_CLOSED 62    ///< This event is generated when a local device has successfully
closed a stream.
```

Description

< This event is generated when a local device has successfully closed a stream. < This event follows the AVDTP_EVT_CLOSE_STREAM_REQUESTED if the upper layer has accepted it. < This event is not generated if stream closing was initiated by the local device.

AVDTP_EVT_STREAM_CONFIGURATION_RECEIVED Macro

File

avdtp.h

C

```
#define AVDTP_EVT_STREAM_CONFIGURATION_RECEIVED 17    ///< This event is generated when a local device
received a positive response to a "get stream configuration" request.
```

Description

< This event is generated when a local device received a positive response to a "get stream configuration" request.

AVDTP_EVT_STREAM_CONFIGURED Macro

File

avdtp.h

C

```
#define AVDTP_EVT_STREAM_CONFIGURED 58    ///< This event is generated when a local device has successfully
```

```
configured a stream.
```

Description

< This event is generated when a local device has successfully configured a stream. < This event follows the AVDTP_EVT_SET_STREAM_CONFIGURATION_REQUESTED if the upper layer has accepted it. < This event is not generated if stream configuration was initiated by the local device.

AVDTP_EVT_STREAM_OPENED Macro

File

avdtp.h

C

```
#define AVDTP_EVT_STREAM_OPENED 60    ///< This event is generated when a local device has successfully
opened a stream.
```

Description

< This event is generated when a local device has successfully opened a stream. < This event follows the AVDTP_EVT_OPEN_STREAM_REQUESTED if the upper layer has accepted it. < This event is not generated if stream opening was initiated by the local device.

AVDTP_EVT_STREAM_RECONFIGURE_COMPLETED Macro

File

avdtp.h

C

```
#define AVDTP_EVT_STREAM_RECONFIGURE_COMPLETED 8     ///< This event is generated when a local device
received a response (either positive or negative) to a "change stream configuration" request.
```

Description

< This event is generated when a local device received a response (either positive or negative) to a "change stream configuration" request.

AVDTP_EVT_STREAM_RECONFIGURED Macro

File

avdtp.h

C

```
#define AVDTP_EVT_STREAM_RECONFIGURED 59    ///< This event is generated when a local device has
successfully reconfigured a stream.
```

Description

< This event is generated when a local device has successfully reconfigured a stream. < This event follows the AVDTP_EVT_RECONFIGURE_STREAM_REQUESTED if the upper layer has accepted it. < This event is not generated if stream reconfiguration was initiated by the local device.

AVDTP_EVT_STREAM_SECURITY_CONTROL_COMPLETED Macro

File

avdtp.h

C

```
#define AVDTP_EVT_STREAM_SECURITY_CONTROL_COMPLETED 13    ///< This event is generated when a local device
received a response (either positive or negative) to a "exchange content protection control data" request.
```

Description

< This event is generated when a local device received a response (either positive or negative) to a "exchange content protection control data" request.

AVDTP_EVT_STREAM_STARTED Macro

File

avdtp.h

C

```
#define AVDTP_EVT_STREAM_STARTED 61    ///< This event is generated when a local device has successfully
started a stream.
```

Description

< This event is generated when a local device has successfully started a stream. < This event follows the AVDTP_EVT_START_STREAM_REQUESTED if the upper layer has accepted it. < This event is not generated if stream starting was initiated by the local device.

AVDTP_EVT_STREAM_SUSPENDED Macro

File

avdtp.h

C

```
#define AVDTP_EVT_STREAM_SUSPENDED 63    ///< This event is generated when a local device has successfully
suspended a stream.
```

Description

< This event is generated when a local device has successfully suspended a stream. < This event follows the AVDTP_EVT_SUSPEND_STREAM_REQUESTED if the upper layer has accepted it. < This event is not generated if stream suspension was initiated by the local device.

AVDTP_EVT_SUSPEND_STREAM_COMPLETED Macro

File

avdtp.h

C

```
#define AVDTP_EVT_SUSPEND_STREAM_COMPLETED 12    ///< This event is generated when a local device received
a response (either positive or negative) to a "suspend stream" request.
```

Description

< This event is generated when a local device received a response (either positive or negative) to a "suspend stream" request.

AVDTP_EVT_SUSPEND_STREAM_REQUESTED Macro

File

avdtp.h

C

```
#define AVDTP_EVT_SUSPEND_STREAM_REQUESTED 54    ///< This event is generated when a local device received
"suspend stream" request.
```

Description

< This event is generated when a local device received "suspend stream" request.

AVDTP_MANAGER_STATE_CONNECTING Macro

File

avdtp.h

C

```
#define AVDTP_MANAGER_STATE_CONNECTING 1
```

Description

This is macro AVDTP_MANAGER_STATE_CONNECTING.

AVDTP_MANAGER_STATE_IDLE Macro

File

avdtp.h

C

```
#define AVDTP_MANAGER_STATE_IDLE 0
```

Description

This is macro AVDTP_MANAGER_STATE_IDLE.

AVDTP_MAX_STREAM_TRANSPORT_SESSION Macro

File

avdtp.h

C

```
#define AVDTP_MAX_STREAM_TRANSPORT_SESSION 3
```

Description

This is macro AVDTP_MAX_STREAM_TRANSPORT_SESSION.

AVDTP_MEDIA_TYPE_AUDIO Macro

File

avdtp.h

C

```
#define AVDTP_MEDIA_TYPE_AUDIO 0 ///< Audio.
```

Description

< Audio.

AVDTP_MEDIA_TYPE_MULTIMEDIA Macro

File

avdtp.h

C

```
#define AVDTP_MEDIA_TYPE_MULTIMEDIA 2 ///< Both Audio & Video.
```

Description

< Both Audio & Video.

AVDTP_MEDIA_TYPE_VIDEO Macro

File

avdtp.h

C

```
#define AVDTP_MEDIA_TYPE_VIDEO 1 ///< Video.
```

Description

< Video.

AVDTP_SEP_CAPABILITY_FLAG_CONTENT_PROTECTION Macro

File

avdtp.h

C

```
#define AVDTP_SEP_CAPABILITY_FLAG_CONTENT_PROTECTION 0x08 ///< Content Prortection. A SEP is capable of
transferring content protection packets.
```

Description

< Content Prortection. A SEP is capable of transferring content protection packets.

AVDTP_SEP_CAPABILITY_FLAG_HEADER_COMPRESSION Macro

File

avdtp.h

C

```
#define AVDTP_SEP_CAPABILITY_FLAG_HEADER_COMPRESSION 0x10 ///< Header Compression. A SEP can use header
compression for transferring Media or Recovery packets.
```

Description

< Header Compression. A SEP can use header compression for transferring Media or Recovery packets.

AVDTP_SEP_CAPABILITY_FLAG_MEDIA_CODEC Macro

File

avdtp.h

C

```
#define AVDTP_SEP_CAPABILITY_FLAG_MEDIA_CODEC 0x40 ///< Media Codec. Defines which codec a SEP supports. A
SEP can support only one codec.
```

Description

< Media Codec. Defines which codec a SEP supports. A SEP can support only one codec.

AVDTP_SEP_CAPABILITY_FLAG_MEDIA_TRANSPORT Macro

File

avdtp.h

C

```
#define AVDTP_SEP_CAPABILITY_FLAG_MEDIA_TRANSPORT 0x01 ///< Media. A SEP is capable of transferring media
(audio, video or both) packets.
```

Description

< Media. A SEP is capable of transferring media (audio, video or both) packets.

AVDTP_SEP_CAPABILITY_FLAG_MULTIPLEXING Macro

File

avdtp.h

C

```
#define AVDTP_SEP_CAPABILITY_FLAG_MULTIPLEXING 0x20 ///< Multiplexing. Multiple transport sessions,
belonging to the same or to a different stream, can share a common transport (L2CAP) channel.
```

Description

< Multiplexing. Multiple transport sessions, belonging to the same or to a different stream, can share a common transport (L2CAP) channel.

AVDTP_SEP_CAPABILITY_FLAG_RECOVERY Macro

File

avdtp.h

C

```
#define AVDTP_SEP_CAPABILITY_FLAG_RECOVERY 0x04 ///< Recovery. A SEP is capable of transferring recovery
packets.
```

Description

< Recovery. A SEP is capable of transferring recovery packets.

AVDTP_SEP_CAPABILITY_FLAG_REPORTING Macro

File

avdtp.h

C

```
#define AVDTP_SEP_CAPABILITY_FLAG_REPORTING 0x02 ///< Reporting. A SEP is capable of transferring reporting
packets.
```

Description

< Reporting. A SEP is capable of transferring reporting packets.

AVDTP_SEP_SERVICE_CAPABILITY_CONTENT_PROTECTION Macro

File

avdtp.h

C

```
#define AVDTP_SEP_SERVICE_CAPABILITY_CONTENT_PROTECTION 4 ///< Content Prortection. A SEP is capable of
transferring content protection packets.
```

Description

< Content Prortection. A SEP is capable of transferring content protection packets.

AVDTP_SEP_SERVICE_CAPABILITY_HEADER_COMPRESSION Macro

File

avdtp.h

C

```
#define AVDTP_SEP_SERVICE_CAPABILITY_HEADER_COMPRESSION 5 ///< Header Compression. A SEP can use header
compression for transferring Media or Recovery packets.
```

Description

< Header Compression. A SEP can use header compression for transferring Media or Recovery packets.

AVDTP_SEP_SERVICE_CAPABILITY_MEDIA_CODEC Macro

File

avdtp.h

C

```
#define AVDTP_SEP_SERVICE_CAPABILITY_MEDIA_CODEC 7 ///< Media Codec. Defines which codec a SEP supports. A
SEP can support only one codec.
```

Description

< Media Codec. Defines which codec a SEP supports. A SEP can support only one codec.

AVDTP_SEP_SERVICE_CAPABILITY_MEDIA_TRANSPORT Macro

File

avdtp.h

C

```
#define AVDTP_SEP_SERVICE_CAPABILITY_MEDIA_TRANSPORT 1 ///< Media. A SEP is capable of transferring media
(audio, video or both) packets.
```

Description

< Media. A SEP is capable of transferring media (audio, video or both) packets.

AVDTP_SEP_SERVICE_CAPABILITY_MULTIPLEXING Macro

File

avdtp.h

C

```
#define AVDTP_SEP_SERVICE_CAPABILITY_MULTIPLEXING 6 ///< Multiplexing. Multiple transport sessions,
belonging to the same or to a different stream, can share a common transport (L2CAP) channel.
```

Description

< Multiplexing. Multiple transport sessions, belonging to the same or to a different stream, can share a common transport (L2CAP) channel.

AVDTP_SEP_SERVICE_CAPABILITY_RECOVERY Macro

File

avdtp.h

C

```
#define AVDTP_SEP_SERVICE_CAPABILITY_RECOVERY 3 ///< Recovery. A SEP is capable of transferring recovery
packets.
```

Description

< Recovery. A SEP is capable of transferring recovery packets.

AVDTP_SEP_SERVICE_CAPABILITY_REPORTING Macro

File

avdtp.h

C

```
#define AVDTP_SEP_SERVICE_CAPABILITY_REPORTING 2 ///< Reporting. A SEP is capable of transferring reporting
packets.
```

Description

< Reporting. A SEP is capable of transferring reporting packets.

AVDTP_SEP_STATE_FREE Macro

File

avdtp.h

C

```
#define AVDTP_SEP_STATE_FREE 0
```

Description

This is macro AVDTP_SEP_STATE_FREE.

AVDTP_SEP_STATE_IDLE Macro

File

avdtp.h

C

```
#define AVDTP_SEP_STATE_IDLE 1
```

Description

This is macro AVDTP_SEP_STATE_IDLE.

AVDTP_SEP_TYPE_SINK Macro

File

avdtp.h

C

```
#define AVDTP_SEP_TYPE_SINK 1 ///< Sink (usually a device like a headphones or BMW).
```

Description

< Sink (usually a device like a headphones or BMW).

AVDTP_SEP_TYPE_SOURCE Macro

File

avdtp.h

C

```
#define AVDTP_SEP_TYPE_SOURCE 0 ///< Source (usually a device like a phone, desktop or laptop).
```

Description

< Source (usually a device like a phone, desktop or laptop).

AVDTP_STREAM_CLOSING_TRANSPORT_CHANNELS Macro

File

avdtp.h

C

```
#define AVDTP_STREAM_CLOSING_TRANSPORT_CHANNELS 2      ///< The stream is closing transport channels.
```

Description

< The stream is closing transport channels.

AVDTP_STREAM_FLAG_LISTENING Macro

File

avdtp.h

C

```
#define AVDTP_STREAM_FLAG_LISTENING 1
```

Description

This is macro AVDTP_STREAM_FLAG_LISTENING.

AVDTP_STREAM_OPENING_TRANSPORT_CHANNELS Macro

File

avdtp.h

C

```
#define AVDTP_STREAM_OPENING_TRANSPORT_CHANNELS 1      ///< The stream is opening transport channels.
```

Description

< The stream is opening transport channels.

AVDTP_STREAM_STATE_ABORTING Macro

File

avdtp.h

C

```
#define AVDTP_STREAM_STATE_ABORTING 7      ///< The stream is aborting. This means that all transport
channels associated with the stream are being closed.
```

Description

< The stream is aborting. This means that all transport channels associated with the stream are being closed. < After they have been closed the stream goes to AVDTP_STREAM_STATE_IDLE state.

AVDTP_STREAM_STATE_CLOSING Macro

File

avdtp.h

C

```
#define AVDTP_STREAM_STATE_CLOSING 6      ///< The stream is closing. This means that all transport
channels associated with the stream are being closed.
```

Description

< The stream is closing. This means that all transport channels associated with the stream are being closed. < After they have been closed the stream goes to AVDTP_STREAM_STATE_IDLE state.

AVDTP_STREAM_STATE_CONFIGURED Macro

File

avdtp.h

C

```
#define AVDTP_STREAM_STATE_CONFIGURED 3      ///< The stream has been configured.
```

Description

< The stream has been configured.

AVDTP_STREAM_STATE_IDLE Macro

File

avdtp.h

C

```
#define AVDTP_STREAM_STATE_IDLE 0      ///< The stream is idle. This can mean two things. The stream
specified by strm_handle does not exist
```

Description

< The stream is idle. This can mean two things. The stream specified by strm_handle does not exist < or the stream is closed.

AVDTP_STREAM_STATE_OPEN Macro

File

avdtp.h

C

```
#define AVDTP_STREAM_STATE_OPEN 4      ///< The stream has been opened.
```

Description

< The stream has been opened.

AVDTP_STREAM_STATE_STREAMING Macro

File

avdtp.h

C

```
#define AVDTP_STREAM_STATE_STREAMING 5      ///< The stream has been started. Depending on the local SEP
type (source or sink) it means
```

Description

< The stream has been started. Depending on the local SEP type (source or sink) it means < that the stream can send or receive media packets.

AVDTP_TC_OPCODE_CONNECT Macro

File

avdtp_private.h

C

```
#define AVDTP_TC_OPCODE_CONNECT 0
```

Description

This is macro AVDTP_TC_OPCODE_CONNECT.

AVDTP_TC_OPCODE_DISCONNECT Macro

File

avdtp_private.h

C

```
#define AVDTP_TC_OPCODE_DISCONNECT 1
```

Description

This is macro AVDTP_TC_OPCODE_DISCONNECT.

AVDTP_TC_STATUS_ERROR Macro

File

avdtp_private.h

C

```
#define AVDTP_TC_STATUS_ERROR 0xFF
```

Description

This is macro AVDTP_TC_STATUS_ERROR.

AVDTP_TC_STATUS_SUCCESS Macro

File

avdtp_private.h

C

```
#define AVDTP_TC_STATUS_SUCCESS 0x00
```

Description

This is macro AVDTP_TC_STATUS_SUCCESS.

AVDTP_TRANSPORT_CHANNEL_TYPE_DEDICATED Macro

File

avdtp.h

C

```
#define AVDTP_TRANSPORT_CHANNEL_TYPE_DEDICATED 0
```

Description

This is macro AVDTP_TRANSPORT_CHANNEL_TYPE_DEDICATED.

AVDTP_TRANSPORT_CHANNEL_TYPE_SHARED Macro

File

avdtp.h

C

```
#define AVDTP_TRANSPORT_CHANNEL_TYPE_SHARED 1
```

Description

This is macro AVDTP_TRANSPORT_CHANNEL_TYPE_SHARED.

AVDTP_TRANSPORT_SESSION_TYPE_MEDIA Macro

File

avdtp.h

C

```
#define AVDTP_TRANSPORT_SESSION_TYPE_MEDIA 0 ///< Media (audio or video).
```

Description

< Media (audio or video).

AVDTP_TRANSPORT_SESSION_TYPE_RECOVERY Macro

File

avdtp.h

C

```
#define AVDTP_TRANSPORT_SESSION_TYPE_RECOVERY 2 ///< Recovery (currently not supported).
```

Description

< Recovery (currently not supported).

AVDTP_TRANSPORT_SESSION_TYPE_REPORTING Macro

File

avdtp.h

C

```
#define AVDTP_TRANSPORT_SESSION_TYPE_REPORTING 1 ///< Reporting (currently not supported).
```

Description

< Reporting (currently not supported).

bt_avdtp_connect Macro

File

avdtp.h

C

```
#define bt_avdtp_connect(mgr, remote_addr) _bt_avdtp_open_control_channel_ex(mgr, remote_addr,
HCI_CONFIG_ENABLE_AUTHENTICATION | HCI_CONFIG_ENABLE_ENCRYPTION)
```

Description

brief Connect to a remote device. ingroup avdtp

details This function opens a control channel connection to a remote device specified by the p remote_addr. If connection cannot be initiated for some reason, for example, there is not enough resources, it returns FALSE and not events are generated. Otherwise the result of an attempt to connect to the remote device is reported via the AVDTP callback. The events generated will either be AVDTP_EVT_CTRL_CHANNEL_CONNECTED or AVDTP_EVT_CTRL_CHANNEL_CONNECTION_FAILED.

param mgr AVDTP manager. param remote_addr The address of a remote device.

return li c TRUE if connection establishment has been started. li c FALSE otherwise.

bt_avdtp_connect_ex Macro

File

avdtp.h

C

```
#define bt_avdtp_connect_ex(mgr, remote_addr, acl_config) _bt_avdtp_open_control_channel_ex(mgr,
remote_addr, acl_config)
```

Description

brief Connect to a remote device. ingroup avdtp

details This function opens a control channel connection to a remote device specified by the p remote_addr. If connection cannot be initiated for some reason, for example, there is not enough resources, it returns FALSE and not events are generated. Otherwise the result of an attempt to connect to the remote device is reported via the AVDTP callback. The events generated will either be AVDTP_EVT_CTRL_CHANNEL_CONNECTED or AVDTP_EVT_CTRL_CHANNEL_CONNECTION_FAILED.

param mgr AVDTP manager. param remote_addr The address of a remote device. param acl_config ACL link configuration. This can be a combination of the following values: li HCI_CONFIG_ENABLE_AUTHENTICATION li HCI_CONFIG_ENABLE_ENCRYPTION li HCI_CONFIG_BECOME_MASTER

return li c TRUE if connection establishment has been started. li c FALSE otherwise.

AVDTP_CMD_DELAYREPORT Macro

File

avdtp_control.h

C

```
#define AVDTP_CMD_DELAYREPORT 13
```

Description

This is macro AVDTP_CMD_DELAYREPORT.

AVDTP_CMD_GET_ALL_CAPABILITIES Macro

File

avdtp_control.h

C

```
#define AVDTP_CMD_GET_ALL_CAPABILITIES 12
```

Description

This is macro AVDTP_CMD_GET_ALL_CAPABILITIES.

AVDTP_CONTENT_PROTECTION_METHOD_SCMS_T Macro

File

avdtp.h

C

```
#define AVDTP_CONTENT_PROTECTION_METHOD_SCMS_T 2
```

Description

This is macro AVDTP_CONTENT_PROTECTION_METHOD_SCMS_T.

AVDTP_CTRL_MESSAGE_TYPE_GENERAL_REJECT Macro

File

avdtp_control.h

C

```
#define AVDTP_CTRL_MESSAGE_TYPE_GENERAL_REJECT 1
```

Description

This is macro AVDTP_CTRL_MESSAGE_TYPE_GENERAL_REJECT.

AVDTP_SCMS_T_CP_BIT Macro

File

avdtp.h

C

```
#define AVDTP_SCMS_T_CP_BIT 2
```

Description

This is macro AVDTP_SCMS_T_CP_BIT.

AVDTP_SCMS_T_L_BIT Macro

File

avdtp.h

C

```
#define AVDTP_SCMS_T_L_BIT 1
```

Description

This is macro AVDTP_SCMS_T_L_BIT.

bt_avdtp_evt_delay_report_completed_t Structure

File

avdtp.h

C

```
typedef struct _bt_avdtp_evt_delay_report_completed_t {
  bt_byte err_code;
} bt_avdtp_evt_delay_report_completed_t;
```

Members

Members	Description
bt_byte err_code;	< The result of the request. < li If the remote accepted the request c err_code == AVDTP_ERROR_SUCCESS. < li Otherwise c err_code == the error code returned by the remote party.

Description

brief Parameter to AVDTP_EVT_DELAYREPORT_COMPLETED event ingroup avdtp

details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::delay_report_completed - when AVDTP received a response to a "delay report" request.

bt_cp_header_t Type

File

avdtp.h

C

```
typedef struct _bt_cp_header_s bt_cp_header_t;
```

Description

This is type bt_cp_header_t.

_bt_cp_header_s Structure

File

avdtp.h

C

```
struct _bt_cp_header_s {
  union {
    bt_byte scmst;
    struct {
      bt_byte* value;
      bt_uint length;
    } unknown;
  } header;
};
```

Description

This is record _bt_cp_header_s.

AVDTP_EVT_DELAYREPORT_COMPLETED Macro

File

avdtp.h

C

```
#define AVDTP_EVT_DELAYREPORT_COMPLETED 18    ///< This event is generated when a local device received a
response (either positive or negative) to a "delay report" request.
```

Description

< This event is generated when a local device received a response (either positive or negative) to a "delay report" request.

AVDTP_EVT_SET_STREAM_CONFIGURATION Macro

File

avdtp.h

C

```
#define AVDTP_EVT_SET_STREAM_CONFIGURATION 19
```

Description

This is macro AVDTP_EVT_SET_STREAM_CONFIGURATION.

AVDTP_SEP_CAPABILITY_FLAG_DELAY_REPORTING Macro

File

avdtp.h

C

```
#define AVDTP_SEP_CAPABILITY_FLAG_DELAY_REPORTING 0x80 ///< Delat reporitng.
```

Description

< Delat reporitng.

AVDTP_SEP_SERVICE_CAPABILITY_DELAY_REPORTING Macro

File

avdtp.h

C

```
#define AVDTP_SEP_SERVICE_CAPABILITY_DELAY_REPORTING 8 ///< Delay Reporting.
```

Description

< Delay Reporting.

bt_avdtp_evt_ctrl_connection_failed_t Structure

File

avdtp.h

C

```
typedef struct _bt_avdtp_evt_ctrl_connection_failed_s {
  bt_bdaddr_t* bdaddr;
} bt_avdtp_evt_ctrl_connection_failed_t;
```

Members

Members	Description
bt_bdaddr_t* bdaddr;	< the address of a remote device

Description

brief Parameter to AVDTP_EVT_CTRL_CONNECTION_FAILED event ingroup avdtp

details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::ctrl_connection_failed - when a control channel between two devices has been established.

bt_avdtp_evt_set_stream_configuration_t Structure

File

avdtp.h

C

```
typedef struct _bt_avdtp_evt_set_stream_configuration_t {
  bt_byte err_code;
  bt_byte err_category;
  bt_avdtp_sep_t* sep;
  bt_byte int_sep_id;
  bt_avdtp_sep_capabilities_t* config;
  bt_byte strm_handle;
  bt_byte src_evt_code;
} bt_avdtp_evt_set_stream_configuration_t;
```

Description

brief Parameter to AVDTP_EVT_SET_STREAM_CONFIGURATION event ingroup avdtp

details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::set_stream_configuration - when AVDTP received a "set stream configuration" request.

AVRCP Functions

_bt_avrcp_allocate_browsing_cmd Function

File

avrcp_private.h

C

```
bt_av_command_t* _bt_avrcp_allocate_browsing_cmd(bt_byte pduid);
```

Description

This is function _bt_avrcp_allocate_browsing_cmd.

_bt_avrcp_allocate_bt_specific_cmd Function

File

avrcp_private.h

C

```
bt_av_command_t* _bt_avrcp_allocate_bt_specific_cmd(bt_byte ctype, bt_byte pduid);
```

Description

This is function _bt_avrcp_allocate_bt_specific_cmd.

_bt_avrcp_allocate_bt_specific_response Function

File

avrcp_private.h

C

```
bt_av_command_t* _bt_avrcp_allocate_bt_specific_response(bt_byte ctype, bt_byte tran_id, bt_byte pduid);
```

Description

This is function _bt_avrcp_allocate_bt_specific_response.

_bt_avrcp_allocate_channel Function

File

avrcp_private.h

C

```
bt_avrcp_channel_t* _bt_avrcp_allocate_channel(bt_avrcp_mgr_t* mgr);
```

Description

void _bt_avctp_l2cap_read_data_callback(struct _channel *pch, bt_byte* pdata, bt_int len); bt_avctp_channel_t* _bt_avctp_find_channel(bt_avctp_mgr_t* mgr, bt_bdaddr_t* remote_addr, bt_uint profile_id);

_bt_avrcp_allocate_cmd Function

File

avrcp_private.h

C

```
bt_av_command_t* _bt_avrcp_allocate_cmd();
```

Description

This is function _bt_avrcp_allocate_cmd.

_bt_avrcp_allocate_response Function

File

avrcp_private.h

C

```
bt_av_command_t* _bt_avrcp_allocate_response(bt_byte tran_id);
```

Description

This is function _bt_avrcp_allocate_response.

_bt_avrcp_allocate_simple_panel_cmd Function

File

avrcp_private.h

C

```
bt_av_command_t* _bt_avrcp_allocate_simple_panel_cmd(bt_byte ctype, bt_byte opid, bt_byte button_state);
```

Description

This is function _bt_avrcp_allocate_simple_panel_cmd.

_bt_avrcp_allocate_simple_panel_response Function

File

avrcp_private.h

C

```
bt_av_command_t* _bt_avrcp_allocate_simple_panel_response(bt_byte ctype, bt_byte tran_id, bt_byte opid,
bt_byte button_state);
```

Description

This is function _bt_avrcp_allocate_simple_panel_response.

_bt_avrcp_find_channel Function

File

avrcp_private.h

C

```
bt_avrcp_channel_t* _bt_avrcp_find_channel(bt_avrcp_mgr_t* mgr, bt_avctp_channel_t* avctp_channel);
```

Description

This is function _bt_avrcp_find_channel.

_bt_avrcp_free_channel Function

File

avrcp_private.h

C

```
void _bt_avrcp_free_channel(bt_avrcp_channel_t* channel);
```

Description

This is function _bt_avrcp_free_channel.

_bt_avrcp_free_cmd Function

File

avrcp_private.h

C

```
void _bt_avrcp_free_cmd(bt_av_command_t* command);
```

Description

This is function _bt_avrcp_free_cmd.

_bt_avrcp_get_tick_count Function

File

avrcp_private.h

C

```
bt_long _bt_avrcp_get_tick_count();
```

Description

This is function _bt_avrcp_get_tick_count.

_bt_avrcp_handle_add_to_now_playing Function

File

avrcp_private.h

C

```
void _bt_avrcp_handle_add_to_now_playing(bt_avrcp_channel_t* channel, bt_byte ctype, bt_byte* data, bt_uint
len);
```

Description

This is function _bt_avrcp_handle_add_to_now_playing.

_bt_avrcp_handle_command Function

File

avrcp_private.h

C

```
void _bt_avrcp_handle_command(bt_avrcp_mgr_t* mgr, bt_avctp_evt_command_received_t* evt_param);
```

Description

This is function _bt_avrcp_handle_command.

_bt_avrcp_handle_command_sent Function

File

avrcp_private.h

C

```
void _bt_avrcp_handle_command_sent(bt_avrcp_mgr_t* mgr, bt_avctp_evt_command_sent_t* evt_param);
```

Description

This is function _bt_avrcp_handle_command_sent.

_bt_avrcp_handle_get_capabilities Function

File

avrcp_private.h

C

```
void _bt_avrcp_handle_get_capabilities(bt_avrcp_channel_t* channel, bt_byte ctype, bt_byte* data, bt_uint
len);
```

Description

This is function _bt_avrcp_handle_get_capabilities.

_bt_avrcp_handle_get_current_player_application_setting_value Function

File

avrcp_private.h

C

```
void _bt_avrcp_handle_get_current_player_application_setting_value(bt_avrcp_channel_t* channel, bt_byte
ctype, bt_byte* data, bt_uint len, void* response_buffer);
```

Description

This is function _bt_avrcp_handle_get_current_player_application_setting_value.

_bt_avrcp_handle_get_element_attributes Function

File

avrcp_private.h

C

```
void _bt_avrcp_handle_get_element_attributes(bt_avrcp_channel_t* channel, bt_byte ctype, bt_byte* data,
bt_uint len);
```

Description

This is function _bt_avrcp_handle_get_element_attributes.

_bt_avrcp_handle_get_play_status Function

File

avrcp_private.h

C

```
void _bt_avrcp_handle_get_play_status(bt_avrcp_channel_t* channel, bt_byte ctype, bt_byte* data, bt_uint
len);
```

Description

This is function _bt_avrcp_handle_get_play_status.

_bt_avrcp_handle_get_player_application_setting_attribute_text Function

File

avrcp_private.h

C

```
void _bt_avrcp_handle_get_player_application_setting_attribute_text(bt_avrcp_channel_t* channel, bt_byte
ctype, bt_byte* data, bt_uint len, void* response_buffer);
```

Description

This is function _bt_avrcp_handle_get_player_application_setting_attribute_text.

_bt_avrcp_handle_get_player_application_setting_value_text Function

File

avrcp_private.h

C

```
void _bt_avrcp_handle_get_player_application_setting_value_text(bt_avrcp_channel_t* channel, bt_byte ctype,
bt_byte* data, bt_uint len, void* response_buffer);
```

Description

This is function _bt_avrcp_handle_get_player_application_setting_value_text.

_bt_avrcp_handle_inform_battery_status_of_ct Function

File

avrcp_private.h

C

```
void _bt_avrcp_handle_inform_battery_status_of_ct(bt_avrcp_channel_t* channel, bt_byte ctype, bt_byte*
data, bt_uint len);
```

Description

This is function _bt_avrcp_handle_inform_battery_status_of_ct.

_bt_avrcp_handle_inform_displayable_character_set Function

File

avrcp_private.h

C

```
void _bt_avrcp_handle_inform_displayable_character_set(bt_avrcp_channel_t* channel, bt_byte ctype, bt_byte*
data, bt_uint len);
```

Description

This is function _bt_avrcp_handle_inform_displayable_character_set.

_bt_avrcp_handle_list_player_application_setting_attributes Function

File

avrcp_private.h

C

```
void _bt_avrcp_handle_list_player_application_setting_attributes(bt_avrcp_channel_t* channel, bt_byte
ctype, bt_byte* data, bt_uint len, void* response_buffer);
```

Description

This is function _bt_avrcp_handle_list_player_application_setting_attributes.

_bt_avrcp_handle_list_player_application_setting_values Function

File

avrcp_private.h

C

```
void _bt_avrcp_handle_list_player_application_setting_values(bt_avrcp_channel_t* channel, bt_byte ctype,
bt_byte* data, bt_uint len, void* response_buffer);
```

Description

This is function _bt_avrcp_handle_list_player_application_setting_values.

_bt_avrcp_handle_play_item Function

File

avrcp_private.h

C

```
void _bt_avrcp_handle_play_item(bt_avrcp_channel_t* channel, bt_byte ctype, bt_byte* data, bt_uint len);
```

Description

This is function _bt_avrcp_handle_play_item.

_bt_avrcp_handle_register_notification Function

File

avrcp_private.h

C

```
void _bt_avrcp_handle_register_notification(bt_avrcp_channel_t* channel, bt_byte ctype, bt_byte event_id,
bt_byte* data, bt_uint len);
```

Description

This is function _bt_avrcp_handle_register_notification.

_bt_avrcp_handle_request_continuing_response Function

File

avrcp_private.h

C

```
void _bt_avrcp_handle_request_continuing_response(bt_avrcp_channel_t* channel, bt_byte ctype, bt_byte*
data, bt_uint len);
```

Description

This is function _bt_avrcp_handle_request_continuing_response.

_bt_avrcp_handle_response Function

File

avrcp_private.h

C

```
void _bt_avrcp_handle_response(bt_avrcp_mgr_t* mgr, bt_avctp_evt_response_received_t* evt_param);
```

Description

This is function _bt_avrcp_handle_response.

_bt_avrcp_handle_response_sent Function

File

avrcp_private.h

C

```
void _bt_avrcp_handle_response_sent(bt_avrcp_mgr_t* mgr, bt_avctp_evt_response_sent_t* evt_param);
```

Description

This is function _bt_avrcp_handle_response_sent.

_bt_avrcp_handle_set_absolute_volume Function

File

avrcp_private.h

C

```
void _bt_avrcp_handle_set_absolute_volume(bt_avrcp_channel_t* channel, bt_byte ctype, bt_byte* data,
bt_uint len);
```

Description

This is function _bt_avrcp_handle_set_absolute_volume.

_bt_avrcp_handle_set_addressed_player Function

File

avrcp_private.h

C

```
void _bt_avrcp_handle_set_addressed_player(bt_avrcp_channel_t* channel, bt_byte ctype, bt_byte* data,
bt_uint len);
```

Description

This is function _bt_avrcp_handle_set_addressed_player.

_bt_avrcp_handle_set_player_application_setting_value Function

File

avrcp_private.h

C

```
void _bt_avrcp_handle_set_player_application_setting_value(bt_avrcp_channel_t* channel, bt_byte ctype,
bt_byte* data, bt_uint len, void* response_buffer);
```

Description

This is function _bt_avrcp_handle_set_player_application_setting_value.

_bt_avrcp_init_signal Function

File

avrcp_private.h

C

```
void _bt_avrcp_init_signal();
```

Description

void _bt_avctp_init_message_buffers();

_bt_avrcp_init_timer Function

File

avrcp_private.h

C

```
void _bt_avrcp_init_timer();
```

Description

This is function _bt_avrcp_init_timer.

_bt_avrcp_register_next_notification Function

File

avrcp_private.h

C

```
void _bt_avrcp_register_next_notification(bt_avrcp_channel_t* channel);
```

Description

This is function _bt_avrcp_register_next_notification.

_bt_avrcp_register_pending_notification Function

File

avrcp_private.h

C

```
void _bt_avrcp_register_pending_notification(bt_avrcp_channel_t* channel);
```

Description

This is function _bt_avrcp_register_pending_notification.

_bt_avrcp_send_notifications Function

File

avrcp_private.h

C

```
void _bt_avrcp_send_notifications(bt_avrcp_channel_t* channel);
```

Description

This is function _bt_avrcp_send_notifications.

_bt_avrcp_set_signal Function

File

avrcp_private.h

C

```
void _bt_avrcp_set_signal();
```

Description

This is function _bt_avrcp_set_signal.

_bt_avrcp_start_timer Function

File

avrcp_private.h

C

```
void _bt_avrcp_start_timer();
```

Description

This is function _bt_avrcp_start_timer.

_bt_avrcp_tg_handle_get_capabilities Function

File

avrcp_private.h

C

```
void _bt_avrcp_tg_handle_get_capabilities(bt_avrcp_channel_t* channel, bt_byte ctype, bt_byte tran_id,
bt_byte* data, bt_uint len);
```

Description

This is function _bt_avrcp_tg_handle_get_capabilities.

_bt_avrcp_tg_handle_get_element_attributes Function

File

avrcp_private.h

C

```
void _bt_avrcp_tg_handle_get_element_attributes(bt_avrcp_channel_t* channel, bt_byte ctype, bt_byte
tran_id, bt_byte* data, bt_uint len);
```

Description

This is function _bt_avrcp_tg_handle_get_element_attributes.

_bt_avrcp_tg_handle_get_play_status Function

File

avrcp_private.h

C

```
void _bt_avrcp_tg_handle_get_play_status(bt_avrcp_channel_t* channel, bt_byte ctype, bt_byte tran_id,
bt_byte* data, bt_uint len);
```

Description

This is function _bt_avrcp_tg_handle_get_play_status.

_bt_avrcp_tg_handle_inform_battery_status_of_ct Function

File

avrcp_private.h

C

```
void _bt_avrcp_tg_handle_inform_battery_status_of_ct(bt_avrcp_channel_t* channel, bt_byte ctype, bt_byte
tran_id, bt_byte* data, bt_uint len);
```

Description

This is function _bt_avrcp_tg_handle_inform_battery_status_of_ct.

_bt_avrcp_tg_handle_inform_displayable_character_set Function

File

avrcp_private.h

C

```
void _bt_avrcp_tg_handle_inform_displayable_character_set(bt_avrcp_channel_t* channel, bt_byte ctype,
bt_byte tran_id, bt_byte* data, bt_uint len);
```

Description

This is function _bt_avrcp_tg_handle_inform_displayable_character_set.

_bt_avrcp_tg_handle_register_notification Function

File

avrcp_private.h

C

```
void _bt_avrcp_tg_handle_register_notification(bt_avrcp_channel_t* channel, bt_byte ctype, bt_byte tran_id,
bt_byte* data, bt_uint len);
```

Description

This is function _bt_avrcp_tg_handle_register_notification.

_bt_avrcp_tg_handle_set_absolute_volume Function

File

avrcp_private.h

C

```
void _bt_avrcp_tg_handle_set_absolute_volume(bt_avrcp_channel_t* channel, bt_byte ctype, bt_byte tran_id,
bt_byte* data, bt_uint len);
```

Description

This is function _bt_avrcp_tg_handle_set_absolute_volume.

_bt_avrcp_write_command_header Function

File

avrcp_private.h

C

```
bt_int _bt_avrcp_write_command_header(bt_av_command_t* command);
```

Description

This is function _bt_avrcp_write_command_header.

bt_avrcp_abort_continuing_response Function

File

avrcp_private.h

C

```
bt_bool bt_avrcp_abort_continuing_response(bt_avrcp_channel_t* channel, bt_byte pduid);
```

Description

This is function bt_avrcp_abort_continuing_response.

bt_avrcp_add_to_now_playing Function

File

avrcp.h

C

```
bt_bool bt_avrcp_add_to_now_playing(bt_avrcp_channel_t* channel, bt_byte scope, bt_av_element_id_t*
element_id, bt_uint counter);
```

Description

brief Add to "now playing" list. ingroup avrcp

details This function adds a media element specified by p element_id to the "now playing" list on the target.

param channel AVRCP channel. param scope The scope in which the p element_id is valid. This value can be on the following values: li AVC_SCOPE_MEDIA_PLAYER_LIST li AVC_MEDIA_PLAYER_VIRTUAL_FILESYSTEM li AVC_SEARCH li AVC_NOW_PLAYING

param element_id UID of the media element to be added to the "now playing" list. param counter UID counter.

return li c TRUE if the function succeeds. li c FALSE otherwise.

bt_avrcp_avrcpt_request_continuing_response Function

File

avrcp_private.h

C

```
bt_bool bt_avrcp_avrcpt_request_continuing_response(bt_avrcp_channel_t* channel, bt_byte pduid);
```

Description

This is function bt_avrcp_avrcpt_request_continuing_response.

bt_avrcp_cancel_find Function

File

avrcp.h

C

```
bt_bool bt_avrcp_cancel_find();
```

Description

brief Cancel finding Targets ingroup avrcp

details This function stops AVRCP layer looking for targets on nearby devices. As a result of this operation the AVRCP_EVT_SEARCH_COMPLETED event will be generated.

return li c TRUE if the function succeeds. li c FALSE otherwise. the callback is not called in this case.

bt_avrcp_cancel_listen Function

File

avrcp.h

C

```
void bt_avrcp_cancel_listen(bt_avrcp_channel_t* channel);
```

Description

brief Cancel listening for incoming connections. ingroup avrcp

details This function stops listening for incoming connections on a specified channel.

param channel AVRCP channel.

return li c TRUE if the function succeeds. li c FALSE otherwise.

bt_avrcp_connect Function

File

avrcp.h

C

```
bt_bool bt_avrcp_connect(bt_avrcp_channel_t* channel, bt_bdaddr_t* remote_address);
```

Description

brief Connect to a remote device. ingroup avrcp

details This function establishes a connection to a remote device specified by the p remote_address. If connection cannot be initiated for some reason, for example, there is not enough resources, it returns FALSE and no events are generated. Otherwise the result of an attempt to connect to the remote device is reported via the AVRCP callback. The events generated will either be AVRCP_EVT_CONTROL_CHANNEL_CONNECTED or AVRCP_EVT_CONTROL_CONNECTION_FAILED.

param channel AVRCP channel. param remote_address The address of a remote device.

return li c TRUE if connection establishment has been started. li c FALSE otherwise.

bt_avrcp_create_channel Function

File

avrcp.h

C

```
bt_avrcp_channel_t* bt_avrcp_create_channel(bt_avrcp_mgr_t* mgr, bt_bool create_browsing_channel);
```

Description

brief Allocate AVRCP channel ingroup avrcp

details This function allocates a new incoming AVRCP channel. The channel is intended to be used to accept a connection from a remote device. There can be only one incoming channel.

param mgr AVRCP manager. param create_browsing_channel Defines weather a browsing channel will be created.

return li A pointer to the new AVRCP channel if the function succeeds. li c NULL otherwise.

bt_avrcp_create_outgoing_channel Function

File

avrcp.h

C

```
bt_avrcp_channel_t* bt_avrcp_create_outgoing_channel(bt_avrcp_mgr_t* mgr, bt_bool create_browsing_channel);
```

Description

brief Allocate AVRCP channel ingroup avrcp

details This function allocates a new outgoing AVRCP channel. The channel is intended to be used to create a connection to a remote device. There can be multiple outgoing channels.

param mgr AVRCP manager. param create_browsing_channel Defines weather a browsing channel will be created.

return li A pointer to the new AVRCP channel if the function succeeds. li c NULL otherwise.

bt_avrcp_destroy_channel Function

File

avrcp.h

C

```
bt_bool bt_avrcp_destroy_channel(bt_avrcp_channel_t* channel);
```

Description

brief Destroy AVRCP channel. ingroup avrcp

details This function frees memory used by the channel. The channel has to exist and be in the "idle" state for this function to succeed. I.e. the channel has to be disconnected before this function can be called.

param channel AVRCP channel.

return li c TRUE if the function succeeds. li c FALSE otherwise.

bt_avrcp_disconnect Function

File

avrcp.h

C

```
bt_bool bt_avrcp_disconnect(bt_avrcp_channel_t* channel);
```

Description

brief Disconnect from a remote device. ingroup avrcp

details This function closes a connection to a remote device.

param channel AVRCP channel.

return li c TRUE if disconnection has been started. li c FALSE otherwise. No events will be generated.

bt_avrcp_find_controller Function

File

avrcp.h

C

```
bt_bool bt_avrcp_find_controller(bt_bdaddr_t* deviceAddress, bt_avrcp_find_callback_fp callback,
bt_sdp_client_callback_fp client_callback, void* callback_param);
```

Description

brief Find Controller ingroup avrcp

details This function looks for a controller on a remote device specified by p deviceAddress.

param deviceAddress The address of a remote device. param callback The callback function that will be called when search has completed. param client_callback The optional callback function that an application can set if it wants to be notified of state changes of the SDP client. The c evt parameter of the callback can be one of the following values: li SDP_CLIENT_STATE_IDLE li SDP_CLIENT_STATE_CONNECTING li SDP_CLIENT_STATE_DISCONNECTING li SDP_CLIENT_STATE_CONNECTED param callback_param A pointer to arbitrary data to be passed to the p callback and p client_callback callbacks.

return li c TRUE if the function succeeds. li c FALSE otherwise. the callback is not called in this case.

bt_avrcp_find_target Function

File

avrcp.h

C

```
bt_bool bt_avrcp_find_target(bt_bdaddr_t* deviceAddress, bt_avrcp_find_callback_fp callback,
bt_sdp_client_callback_fp client_callback, void* callback_param);
```

Description

brief Find Target ingroup avrcp

details This function looks for a target on a remote device specified by p deviceAddress.

param deviceAddress The address of a remote device. param callback The callback function that will be called when search has completed. param client_callback The optional callback function that an application can set if it wants to be notified of state changes of the SDP client. The c evt parameter of the callback can be one of the following values: li SDP_CLIENT_STATE_IDLE li SDP_CLIENT_STATE_CONNECTING li SDP_CLIENT_STATE_DISCONNECTING li SDP_CLIENT_STATE_CONNECTED param callback_param A pointer to arbitrary data to be passed to the p callback and p client_callback callbacks.

return li c TRUE if the function succeeds. li c FALSE otherwise. the callback is not called in this case.

bt_avrcp_find_targets Function

File

avrcp.h

C

```
bt_bool bt_avrcp_find_targets(bt_byte search_length);
```

Description

brief Find Targets ingroup avrcp

details This function looks for targets on nearby devices. The AVRCP_EVT_SEARCH_COMPLETED event is generated when the search has completed.

param search_length The amount of time the search will be performed for.

return li c TRUE if the function succeeds. li c FALSE otherwise. the callback is not called in this case.

bt_avrcp_get_browsing_channel_state Function

File

avrcp.h

C

```
bt_byte bt_avrcp_get_browsing_channel_state(bt_avrcp_channel_t* channel);
```

Description

brief Get AVCTP browsing channel state. ingroup avrcp

details This function returns status of the AVCTP browsing channel.

param channel AVRCP channel.

return Returns of the following values: li AVCTP_CHANNEL_STATE_FREE li AVCTP_CHANNEL_STATE_IDLE li AVCTP_CHANNEL_STATE_CONNECTING li AVCTP_CHANNEL_STATE_CONNECTED li AVCTP_CHANNEL_STATE_DISCONNECTING

bt_avrcp_get_channel_remote_address Function

File

avrcp.h

C

```
bt_bdaddr_t* bt_avrcp_get_channel_remote_address(bt_avrcp_channel_t* channel);
```

Description

brief Get channel's remote BT address. ingroup avrcp

details This function returns the address of the remote device associated with the channel.

param channel AVRCP channel.

return li The address of the remote device if channel is connected. li NULL otherwise.

bt_avrcp_get_company_id_list Function

File

avrcp.h

C

```
bt_bool bt_avrcp_get_company_id_list(bt_avrcp_channel_t* channel);
```

Description

brief Get Company ID list. ingroup avrcp

details This function requests a list of company id's supported by the remote device

param channel AVRCP channel.

return li c TRUE if the function succeeds. li c FALSE otherwise.

bt_avrcp_get_control_channel_state Function

File

avrcp.h

C

```
bt_byte bt_avrcp_get_control_channel_state(bt_avrcp_channel_t* channel);
```

Description

brief Get AVCTP control channel state. ingroup avrcp

details This function returns status of the AVCTP control channel.

param channel AVRCP channel.

return Returns of the following values: li AVCTP_CHANNEL_STATE_FREE li AVCTP_CHANNEL_STATE_IDLE li AVCTP_CHANNEL_STATE_CONNECTING li AVCTP_CHANNEL_STATE_CONNECTED li AVCTP_CHANNEL_STATE_DISCONNECTING

bt_avrcp_get_current_player_application_setting_value Function

File

avrcp.h

C

```
bt_bool bt_avrcp_get_current_player_application_setting_value(bt_avrcp_channel_t* channel,
bt_av_player_setting_current_values_t* response_buffer);
```

Description

brief Get current player setting values. ingroup avrcp

details This function requests a list of current set values for the player application on the target. The list of attribute ids whose values have to be returned is passed via the p response_buffer parameter. The caller has to set bt_av_player_setting_current_values_t::setting_id_list to a list of player setting attribute ids, bt_av_player_setting_current_values_t::count to the number of entries in the list, bt_av_player_setting_current_values_t::setting_value_id_list to a buffer where returned values will be stored.

param channel AVRCP channel. param response_buffer Pointer to bt_av_player_setting_current_values_t structure initialized as stated above.

return li c TRUE if the function succeeds. li c FALSE otherwise.

bt_avrcp_get_element_attributes Function

File

avrcp.h

C

```
bt_bool bt_avrcp_get_element_attributes(bt_avrcp_channel_t* channel, bt_av_element_id_t* element_id,
bt_uint attr_mask);
```

Description

brief Get media element attributes. ingroup avrcp

details This function requests the attributes of the element specified with p element_id.

note Currently p element_id is ignored. The AVRCP specification defines that only UID 0 can be used to return the attributes of the current track.

param channel AVRCP channel. param element_id UID of the media element whose attributes are requested. param attr_mask Bitmask that defines which attributes are requested. This value can be a combination of the following values: li AVC_MEDIA_ATTR_FLAG_TITLE li AVC_MEDIA_ATTR_FLAG_ARTIST li AVC_MEDIA_ATTR_FLAG_ALBUM li AVC_MEDIA_ATTR_FLAG_NUMBER li AVC_MEDIA_ATTR_FLAG_TOTAL_NUMBER li AVC_MEDIA_ATTR_FLAG_GENRE li AVC_MEDIA_ATTR_FLAG_PLAYING_TIME li AVC_MEDIA_ATTR_FLAG_ALL

return li c TRUE if the function succeeds. li c FALSE otherwise.

bt_avrcp_get_hci_connection Function

File

avrcp.h

C

```
bt_hci_conn_state_p* bt_avrcp_get_hci_connection(bt_avrcp_channel_t* channel);
```

Description

brief Get HCI connection for a channel ingroup avrcp

details This function returns a pointer to a structure that describes an HCI connection a channel is open on. The return value can be used to call various function from the HCI layer. For example, if an app wants to force disconnection from a remote device it can call ::bt_hci_disconnect.

param channel AVRCP channel.

return li c Pointer to a structure that describes an HCI connection if the function succeeds. li c NULL otherwise. The function fails only if a channel specified by the p channel parameter li does not exist or there is no HCI connection between local and remote devices associated with the channel.

bt_avrcp_get_mgr Function

File

avrcp.h

C

```
bt_avrcp_mgr_t* bt_avrcp_get_mgr();
```

Description

brief Return a pointer to an instance of the AVRCP manager. ingroup avrcp

details This function returns a pointer to an instance of the AVRCP manager. There is only one instance of the manager allocated by the stack.

bt_avrcp_get_play_status Function

File

avrcp.h

C

```
bt_bool bt_avrcp_get_play_status(bt_avrcp_channel_t* channel, bt_uint repeat_interval);
```

Description

brief Get playback status. ingroup avrcp

details This function requests the status of the currently playing media at the target.

param channel AVRCP channel. param repeat_interval Interval in milliseconds at which AVRCP polls the target for playback status. If 0 is passed polling is stopped.

return li c TRUE if the function succeeds. li c FALSE otherwise.

bt_avrcp_get_player_application_setting_attr_text Function

File

avrcp.h

C

```
bt_bool bt_avrcp_get_player_application_setting_attr_text(bt_avrcp_channel_t* channel,
```

```
bt_av_player_settings_text_t* response_buffer);
```

Description

brief Get player setting attribute text. ingroup avrcp

details This function requests the target device to provide supported player application setting attribute displayable text for the provided player application setting attributes. The list of attribute ids whose displayable text have to be returned is passed via the p response_buffer parameter. The caller has to set bt_av_player_settings_text_t::setting_id_list to a list of player setting attribute ids, bt_av_player_settings_text_t::count to the number of entries in the list, bt_av_player_settings_text_t::setting_text_list to a buffer where returned values will be stored.

param channel AVRCP channel. param response_buffer Pointer to bt_av_player_settings_text_t structure initialized as stated above.

return li c TRUE if the function succeeds. li c FALSE otherwise.

bt_avrcp_get_player_application_setting_value_text Function

File

avrcp.h

C

```
bt_bool bt_avrcp_get_player_application_setting_value_text(bt_avrcp_channel_t* channel, bt_byte attr_id,
bt_av_player_setting_values_text_t* response_buffer);
```

Description

brief Get player setting value text. ingroup avrcp

details This function request the target device to provide target supported player application setting value displayable text for the provided player application setting attribute values. The list of attribute ids whose value displayable text have to be returned is passed via the p response_buffer parameter. The caller has to set bt_av_player_setting_values_text_t::setting_value_id_list to a list of player setting attribute value ids, bt_av_player_setting_values_text_t::count to the number of entries in the list, bt_av_player_setting_values_text_t::setting_value_text_list to a buffer where returned values will be stored.

param channel AVRCP channel. param response_buffer Pointer to bt_av_player_setting_values_text_t structure initialized as stated above.

return li c TRUE if the function succeeds. li c FALSE otherwise.

bt_avrcp_get_subuint_info Function

File

avrcp_command.h

C

```
bt_bool bt_avrcp_get_subuint_info(bt_avrcp_channel_t* channel);
```

Description

brief Get subunit info ingroup avrcp

details This function is used to request subunit info from the target.

param channel AVRCP channel.

return li c TRUE if the function succeeds. li c FALSE otherwise.

bt_avrcp_get_supported_event_id_list Function

File

avrcp.h

C

```
bt_bool bt_avrcp_get_supported_event_id_list(bt_avrcp_channel_t* channel);
```

Description

brief Get supported events. ingroup avrcp

details This function requests a list of events supported by the remote device

param channel AVRCP channel.

return li c TRUE if the function succeeds. li c FALSE otherwise.

bt_avrcp_get_unit_info Function

File

avrcp_command.h

C

```
bt_bool bt_avrcp_get_unit_info(bt_avrcp_channel_t* channel);
```

Description

brief Get unit info ingroup avrcp

details This function is used to request unit info from the target.

param channel AVRCP channel.

return li c TRUE if the function succeeds. li c FALSE otherwise.

bt_avrcp_inform_battery_status Function

File

avrcp.h

C

```
bt_bool bt_avrcp_inform_battery_status(bt_avrcp_channel_t* channel, bt_byte status);
```

Description

brief Inform controller's battery status. ingroup avrcp

details This function is used to inform the target about the controller's battery status.

param channel AVRCP channel. param status Battery status. This can be one of the following values: li AVC_BATTERY_STATUS_NORMAL li AVC_BATTERY_STATUS_WARNING li AVC_BATTERY_STATUS_CRITICAL li AVC_BATTERY_STATUS_EXTERNAL li AVC_BATTERY_STATUS_FULL_CHARGE

return li c TRUE if the function succeeds. li c FALSE otherwise.

bt_avrcp_inform_displayable_character_set Function

File

avrcp.h

C

```
bt_bool bt_avrcp_inform_displayable_character_set(bt_avrcp_channel_t* channel, bt_uint* charset_list,
bt_byte charset_count);
```

Description

brief Inform displayable character set. ingroup avrcp

details This function informs the list of character set supported by the controller to the target.

param channel AVRCP channel. param charset_list List of displayable character sets. param charset_count Number of entries in the list.

return li c TRUE if the function succeeds. li c FALSE otherwise.

bt_avrcp_init_controller Function

File

avrcp.h

C

```
void bt_avrcp_init_controller();
```

Description

brief Initialize AVRCP to be used in controller mode. ingroup avrcp

details This function initializes the AVRCP layer of the stack in controller mode. It must be called prior to any other AVRCP function can be called.

bt_avrcp_init_target Function

File

avrcp.h

C

```
void bt_avrcp_init_target(bt_ulong company_id, bt_uint supported_events);
```

Description

brief Initialize AVRCP to be used in target mode. ingroup avrcp

details This function initializes the AVRCP layer of the stack in target mode. It must be called prior to any other AVRCP function can be called.

param company_id The 24-bit unique ID obtained from the IEEE Registration Authority Committee. If the vendor of a TG device does not have the unique ID, the value 0xFFFFFF may be used. param supported_events Bitmask that specifies events supported by the target. This value can be a combination of the following values: li AVC_EVENT_FLAG_PLAYBACK_STATUS_CHANGED li AVC_EVENT_FLAG_TRACK_CHANGED li AVC_EVENT_FLAG_TRACK_REACHED_END li AVC_EVENT_FLAG_TRACK_REACHED_START li AVC_EVENT_FLAG_PLAYBACK_POS_CHANGED li AVC_EVENT_FLAG_BATT_STATUS_CHANGED li AVC_EVENT_FLAG_SYSTEM_STATUS_CHANGED li AVC_EVENT_FLAG_PLAYER_APPLICATION_SETTING_CHANGED li AVC_EVENT_FLAG_NOW_PLAYING_CONTENT_CHANGED li AVC_EVENT_FLAG_AVAILABLE_PLAYERS_CHANGED li AVC_EVENT_FLAG_ADDRESSED_PLAYER_CHANGED li AVC_EVENT_FLAG_UIDS_CHANGED li AVC_EVENT_FLAG_VOLUME_CHANGED

bt_avrcp_list_player_application_setting_attributes Function

File

avrcp.h

C

```
bt_bool bt_avrcp_list_player_application_setting_attributes(bt_avrcp_channel_t* channel,
bt_av_player_settings_t* response_buffer);
```

Description

brief Get supported player setting attributes. ingroup avrcp

details This function request the target device to provide target supported player application setting attributes. The list of attribute ids is stored in the setting_id_list member of the p response_buffer parameter. The caller has to set bt_av_player_settings_t::setting_id_list to a buffer where returned values will be stored and bt_av_player_settings_t::count to the number of entries in the list.

param channel AVRCP channel. param response_buffer Pointer to bt_av_player_settings_t structure initialized as stated above.

return li c TRUE if the function succeeds. li c FALSE otherwise.

bt_avrcp_list_player_application_setting_values Function

File

avrcp.h

C

```
bt_bool bt_avrcp_list_player_application_setting_values(bt_avrcp_channel_t* channel, bt_byte attr_id,
bt_av_player_setting_values_t* response_buffer);
```

Description

brief Get player setting attribute values. ingroup avrcp

details This function requests the target device to list the set of possible values for the requested player application setting attribute. The list of attribute value ids is stored in the setting_value_id_list member of the p response_buffer parameter. The caller has to set bt_av_player_setting_values_t::setting_value_id_list to a buffer where returned values will be stored and bt_av_player_setting_values_t::count to the number of entries in the list.

param channel AVRCP channel. param response_buffer Pointer to bt_av_player_setting_values_t structure initialized as stated above.

return li c TRUE if the function succeeds. li c FALSE otherwise.

bt_avrcp_listen Function

File

avrcp.h

C

```
bt_bool bt_avrcp_listen(bt_avrcp_channel_t* channel);
```

Description

brief Listen for incoming connections. ingroup avrcp

details This function enables incoming connections on the specified AVRCP channel.

param channel AVRCP channel.

return li c TRUE if the function succeeds. li c FALSE otherwise.

bt_avrcp_play_item Function

File

avrcp.h

C

```
bt_bool bt_avrcp_play_item(bt_avrcp_channel_t* channel, bt_byte scope, bt_av_element_id_t* element_id,
bt_uint counter);
```

Description

brief Play media item. ingroup avrcp

details This function starts playing an item indicated by the UID.

param channel AVRCP channel. param scope The scope in which the p element_id is valid. This value can be on the following values: li AVC_SCOPE_MEDIA_PLAYER_LIST li AVC_MEDIA_PLAYER_VIRTUAL_FILESYSTEM li AVC_SEARCH li AVC_NOW_PLAYING param element_id UID of the media element to be played. param counter UID counter.

return li c TRUE if the function succeeds. li c FALSE otherwise.

bt_avrcp_register_notification Function

File

avrcp.h

C

```
bt_bool bt_avrcp_register_notification(bt_avrcp_channel_t* channel, bt_byte event_id, bt_ulong
playback_interval);
```

Description

brief Register notification. ingroup avrcp

details This function registers with the target to receive notifications asynchronously based on specific events occurring.

param channel AVRCP channel. param event_id Event Id. This value can be one of the following values: li AVC_EVENT_PLAYBACK_STATUS_CHANGED li AVC_EVENT_TRACK_CHANGED li AVC_EVENT_TRACK_REACHED_END li AVC_EVENT_TRACK_REACHED_START li AVC_EVENT_PLAYBACK_POS_CHANGED li AVC_EVENT_BATT_STATUS_CHANGED li AVC_EVENT_SYSTEM_STATUS_CHANGED li AVC_EVENT_PLAYER_APPLICATION_SETTING_CHANGED li AVC_EVENT_NOW_PLAYING_CONTENT_CHANGED li AVC_EVENT_AVAILABLE_PLAYERS_CHANGED li AVC_EVENT_ADDRESSED_PLAYER_CHANGED li AVC_EVENT_UIDS_CHANGED li AVC_EVENT_VOLUME_CHANGED param playback_interval The time interval (in seconds) at which the change in playback position will be notified. Applicable for AVC_EVENT_PLAYBACK_POS_CHANGED event.

return li c TRUE if the function succeeds. li c FALSE otherwise.

bt_avrcp_register_notifications Function

File

avrcp.h

C

```
bt_bool bt_avrcp_register_notifications(bt_avrcp_channel_t* channel, bt_uint event_mask);
```

Description

brief Register notifications. ingroup avrcp

details This function registers with the target to receive notifications asynchronously based on specific events occurring. This function is used to register multiple notifications with one call.

note This function cannot be used to register for AVC_EVENT_PLAYBACK_POS_CHANGED event.

param channel AVRCP channel. param event_mask Bitmask that specifies which events to register for. This value can be a combination of the following values: li AVC_EVENT_FLAG_PLAYBACK_STATUS_CHANGED li AVC_EVENT_FLAG_TRACK_CHANGED li AVC_EVENT_FLAG_TRACK_REACHED_END li AVC_EVENT_FLAG_TRACK_REACHED_START li AVC_EVENT_FLAG_BATT_STATUS_CHANGED li AVC_EVENT_FLAG_SYSTEM_STATUS_CHANGED li AVC_EVENT_FLAG_PLAYER_APPLICATION_SETTING_CHANGED li AVC_EVENT_FLAG_NOW_PLAYING_CONTENT_CHANGED li AVC_EVENT_FLAG_AVAILABLE_PLAYERS_CHANGED li AVC_EVENT_FLAG_ADDRESSED_PLAYER_CHANGED li AVC_EVENT_FLAG_UIDS_CHANGED li AVC_EVENT_FLAG_VOLUME_CHANGED li AVC_EVENT_FLAG_ALL

return li c TRUE if the function succeeds. li c FALSE otherwise.

bt_avrcp_send_button_click Function

File

avrcp_command.h

C

```
bt_bool bt_avrcp_send_button_click(bt_avrcp_channel_t* channel, bt_byte button_id);
```

Description

brief Send AV/C Panel Subunit "click" PASS THROUGH command ingroup avrcp

details This function is used to send a button click. Two PATH THROUGTH commands are sent. The first command has button state set to AVC_PANEL_BUTTON_PRESSED. The second command gas button state set to AVC_PANEL_BUTTON_RELEASED

param channel AVRCP channel. param button_id Operation Id. This value can be on of the AVC_PANEL_OPID_... constants

return li c TRUE if the function succeeds. li c FALSE otherwise.

bt_avrcp_send_cmd Function

File

avrcp.h

C

```
bt_bool bt_avrcp_send_cmd(bt_avrcp_channel_t* channel, bt_av_command_t* command);
```

Description

brief Send AVRCP command ingroup avrcp

details This function sends a command to the remote device.

param channel AVRCP channel. param command Command to be sent.

return li c TRUE if the function succeeds. li c FALSE otherwise.

bt_avrcp_send_panel_control Function

File

avrcp_command.h

C

```
bt_bool bt_avrcp_send_panel_control(bt_avrcp_channel_t* channel, bt_byte opid, bt_byte button_state);
```

Description

brief Send AV/C Panel Subunit "control" PASS THROUGH command ingroup avrcp

details This function is used to send AV/C Panel Subunit PASS THROUGH command with command type set to AVC_CTYPE_CONTROL.

param channel AVRCP channel. param opid Operation Id. This value can be on of the AVC_PANEL_OPID_... constants param button_state

Button state. This can be on of the following values: li AVC_PANEL_BUTTON_PRESSED li AVC_PANEL_BUTTON_RELEASED

return li c TRUE if the function succeeds. li c FALSE otherwise.

bt_avrcp_send_press_panel_control Function

File

avrcp_command.h

C

```
bt_bool bt_avrcp_send_press_panel_control(bt_avrcp_channel_t* channel, bt_byte opid);
```

Description

brief Send AV/C Panel Subunit "pressed" PASS THROUGH command ingroup avrcp

details This function is used to send AV/C Panel Subunit PASS THROUGH command with button state set to AVC_PANEL_BUTTON_PRESSED.

param channel AVRCP channel. param opid Operation Id. This value can be on of the AVC_PANEL_OPID_... constants

return li c TRUE if the function succeeds. li c FALSE otherwise.

bt_avrcp_send_simple_panel_cmd Function

File

avrcp_command.h

C

```
bt_bool bt_avrcp_send_simple_panel_cmd(bt_avrcp_channel_t* channel, bt_byte ctype, bt_byte opid, bt_byte
button_state);
```

Description

brief Send AV/C Panel Subunit PASS THROUGH command ingroup avrcp

details This function is used to send AV/C Panel Subunit PASS THROUGH command to the target.

param channel AVRCP channel. param ctype Command type. This value can be on of the following values: li AVC_CTYPE_CONTROL 0 li AVC_CTYPE_STATUS 1 li AVC_CTYPE_SPECIFIC_IQUIRY 2 li AVC_CTYPE_NOTIFY 3 li AVC_CTYPE_GENERAL_INQUORY 4 param opid Operation Id. This value can be on of the AVC_PANEL_OPID_... constants param button_state Button state. This can be on of the following values: li AVC_PANEL_BUTTON_PRESSED li AVC_PANEL_BUTTON_RELEASED

return li c TRUE if the function succeeds. li c FALSE otherwise.

bt_avrcp_set_absolute_volume Function

File

avrcp.h

C

```
bt_bool bt_avrcp_set_absolute_volume(bt_avrcp_channel_t* channel, bt_byte volume);
```

Description

brief Set absolute volume. ingroup avrcp

details This function is used to set an absolute volume to be used by the rendering device.

param channel AVRCP channel. param volume Volume

return li c TRUE if the function succeeds. li c FALSE otherwise.

bt_avrcp_set_addressed_player Function

File

avrcp.h

C

```
bt_bool bt_avrcp_set_addressed_player(bt_avrcp_channel_t* channel, bt_uint player_id);
```

Description

brief Set addressed player. ingroup avrcp

details This function is used to inform the target of which media player the controller wishes to control.

param channel AVRCP channel. param player_id Player Id.

return li c TRUE if the function succeeds. li c FALSE otherwise.

bt_avrcp_set_player_application_setting_value Function

File

avrcp.h

C

```
bt_bool bt_avrcp_set_player_application_setting_value(bt_avrcp_channel_t* channel, bt_byte* attr_id_list,
bt_byte* attr_value_list, bt_byte attr_id_count);
```

Description

brief Set player setting attribute values. ingroup avrcp

details This function requests to set the player application setting list of player application setting values on the target device for the corresponding defined list of setting attributes. for the requested player application setting attribute. The list of attribute value ids is stored in the setting_value_id_list member of the p response_buffer parameter. The caller has to set bt_av_player_setting_values_t::setting_value_id_list to a buffer where returned values will be stored and bt_av_player_setting_values_t::count to the number of entries in the list.

param channel AVRCP channel. param attr_id_list List of setting attribute ids. param attr_value_list List of setting attribute value ids. param attr_id_count The number of entries in both lists.

return li c TRUE if the function succeeds. li c FALSE otherwise.

bt_avrcp_start Function

File

avrcp.h

C

```
void bt_avrcp_start(bt_avrcp_mgr_t* mgr, bt_avrcp_mgr_callback_fp callback, void* callback_param);
```

Description

brief Start the AVRCP layer. ingroup avrcp

details In order to be notified of various events a consumer of the AVRCP layer has to register a callback function. The stack will call the callback function whenever a new event has been generated passing the code of the event as the second parameter. bt_avrcp_start() stores pointers to the c callback and c callback_param in the bt_avrcp_mgr_t structure.

param mgr AVRCP manager. param callback The callback function that will be called when the AVRCP generates an event. param callback_param A pointer to arbitrary data to be passed to the c callback callback.

bt_avrcp_tg_send_element_attributes Function

File

avrcp.h

C

```
bt_bool bt_avrcp_tg_send_element_attributes(bt_avrcp_channel_t* channel, bt_byte tran_id,
bt_av_element_attribute_t* attrs, bt_byte attr_count);
```

Description

brief Send media element attributes ingroup avrcp

details This function is used to send the system status when AVRCP is running in target mode. If there are active connections with controllers, the ones that registered for AVC_EVENT_SYSTEM_STATUS_CHANGED event will be notified.

param status Battery status

bt_avrcp_tg_set_absolute_volume Function

File

avrcp.h

C

```
void bt_avrcp_tg_set_absolute_volume(bt_byte volume);
```

Description

brief Set absolute volume ingroup avrcp

details This function is used to set the absolute volume when AVRCP is running in target mode. If there are active connections with controllers, the ones that registered for AVC_EVENT_VOLUME_CHANGED event will be notified.

param track_id Track Id param song_length The length of the current track. param song_position The position of the current track.

bt_avrcp_tg_set_battery_status Function

File

avrcp.h

C

```
void bt_avrcp_tg_set_battery_status(bt_byte status);
```

Description

brief Set battery status ingroup avrcp

details This function is used to set the battery status when AVRCP is running in target mode. If there are active connections with controllers, the ones that registered for AVC_EVENT_BATT_STATUS_CHANGED event will be notified.

param status Battery status

bt_avrcp_tg_set_channel_absolute_volume Function

File

avrcp.h

C

```
void bt_avrcp_tg_set_channel_absolute_volume(bt_avrcp_channel_t* channel, bt_byte volume);
```

Description

This is function bt_avrcp_tg_set_channel_absolute_volume.

bt_avrcp_tg_set_current_track Function

File

avrcp.h

C

```
void bt_avrcp_tg_set_current_track(bt_av_element_id_t* track_id, bt_ulong song_length, bt_ulong
song_position);
```

Description

brief Set current track ingroup avrcp

details This function is used to set the current track when AVRCP is running in target mode. If there are active connections with controllers, the ones that registered for AVC_EVENT_TRACK_CHANGED event will be notified.

param track_id Track Id param song_length The length of the current track. param song_position The position of the current track.

bt_avrcp_tg_set_play_status Function

File

avrcp.h

C

```
void bt_avrcp_tg_set_play_status(bt_ulong song_length, bt_ulong song_position, bt_byte play_status);
```

Description

local target control *

- brief Set playback status
- ingroup avrcp

*

- details This function is used to set the playback status when AVRCP is running in target mode.
- If there are active connections with controllers, the ones that registered
- for AVC_EVENT_PLAYBACK_STATUS_CHANGED event will be notified.

*

- param song_length The length of the current track.
- param song_position The position of the current track.
- param play_status Playback status. This value can be one of the following values:
- li AVC_PLAY_STATUS_STOPPED
- li AVC_PLAY_STATUS_PLAYING
- li AVC_PLAY_STATUS_PAUSED
- li AVC_PLAY_STATUS_FW_SEEK
- li AVC_PLAY_STATUS_REV_SEEK

bt_avrcp_tg_set_system_status Function

File

avrcp.h

C

```
void bt_avrcp_tg_set_system_status(bt_byte status);
```

Description

brief Set system status ingroup avrcp

details This function is used to set the system status when AVRCP is running in target mode. If there are active connections with controllers, the ones that registered for AVC_EVENT_SYSTEM_STATUS_CHANGED event will be notified.

param status Battery status

_bt_avrcp_allocate_browsing_response Function

File

avrcp_private.h

C

```
bt_av_command_t* _bt_avrcp_allocate_browsing_response(bt_byte tran_id, bt_byte pduid);
```

Description

This is function _bt_avrcp_allocate_browsing_response.

_bt_avrcp_general_reject Function

File

avrcp_private.h

C

```
bt_bool _bt_avrcp_general_reject(bt_avrcp_channel_t* channel, bt_byte reason, bt_byte tran_id);
```

Description

This is function _bt_avrcp_general_reject.

_bt_avrcp_send_rejected_response Function

File

avrcp_private.h

C

```
void _bt_avrcp_send_rejected_response(bt_avrcp_channel_t* channel, bt_byte tran_id, bt_byte pdu_id);
```

Description

This is function _bt_avrcp_send_rejected_response.

bt_avrcp_send_release_panel_control Function

File

avrcp_command.h

C

```
bt_bool bt_avrcp_send_release_panel_control(bt_avrcp_channel_t* channel, bt_byte opid);
```

Description

brief Send AV/C Panel Subunit "released" PASS THROUGH command ingroup avrcp

details This function is used to send AV/C Panel Subunit PASS THROUGH command with button state set to AVC_PANEL_BUTTON_RELEASED.

param channel AVRCP channel. param opid Operation Id. This value can be on of the AVC_PANEL_OPID_... constants

return li c TRUE if the function succeeds. li c FALSE otherwise.

AVRCP Data Types and Constants

_bt_av_command_t Structure

File

avrcp_command.h

C

```
struct _bt_av_command_t {
  bt_av_command_t* next_command;
  bt_byte cmd_type;
  bt_byte flags;
  bt_byte ctype;
  bt_byte subunit_type;
  bt_byte subunit_id;
  bt_byte opcode;
  bt_byte pdu_id;
  bt_byte* parameters;
  bt_uint params_len;
  bt_av_response_t* response_buffer;
  bt_long send_time;
  bt_byte tran_id;
};
```

Description

This is record _bt_av_command_t.

_bt_avrcp_channel_t Structure

File

avrcp.h

C

```
struct _bt_avrcp_channel_t {
  bt_byte state;
  bt_byte flags;
  bt_uint cur_event_mask;
  bt_uint new_event_mask;
  bt_byte play_status;
  bt_uint req_play_status_interval;
  bt_ulong play_status_req_time;
  bt_avctp_channel_t* control_channel;
  bt_avctp_channel_t* browsing_channel;
  bt_avrcp_mgr_t* avrcp_mgr;
  bt_queue_element_t* send_cq_head;
  bt_queue_element_t* ack_cq_head;
  bt_uint tg_event_mask;
  bt_uint tg_interim_event_mask;
  bt_uint tg_signalled_event_mask;
  bt_ulong notify_song_position;
  bt_byte register_notification_tran_id[AVC_MAXEVENTS];
  bt_byte tg_next_event_id;
  bt_byte volume;
};
```

Members

Members	Description
bt_uint tg_event_mask;	target data

Description

brief AVRCP channel description ingroup avrcp

details This structure is used to hold information about an AVRCP channel.

_bt_avrcp_mgr_t Structure

File

avrcp.h

C

```
struct _bt_avrcp_mgr_t {
  bt_byte state;
  bt_byte flags;
  bt_ulong company_id;
  bt_uint supported_events;
  bt_ulong song_length;
  bt_ulong song_position;
  bt_byte play_status;
  bt_byte volume;
  bt_byte battery_status;
  bt_byte system_status;
  bt_av_element_id_t current_track_id;
  bt_uint uid_counter;
  bt_avrcp_channel_t* channels;
  bt_avrcp_mgr_callback_fp callback;
  void* callback_param;
};
```

Members

Members	Description
bt_byte state;	< Manager state. This value can be one of the following values: < li AVRCP_MANAGER_STATE_IDLE < li AVRCP_MANAGER_STATE_CONNECTING < li AVRCP_MANAGER_STATE_DISCONNECTING

bt_byte flags;	< Additional manager state. This value can be a combination of the following values: < li AVRCP_MANAGER_FLAG_SEARCHING
bt_ulong company_id;	< The 24-bit unique ID obtained from the IEEE Registration Authority Committee. < If the vendor of a TG device does not have the unique ID, the value 0xFFFFFF may be used.
bt_uint supported_events;	< Events supported by the target. This value can be a combination of the following values: < li AVC_EVENT_FLAG_PLAYBACK_STATUS_CHANGED < li AVC_EVENT_FLAG_TRACK_CHANGED < li AVC_EVENT_FLAG_TRACK_REACHED_END < li AVC_EVENT_FLAG_TRACK_REACHED_START < li AVC_EVENT_FLAG_PLAYBACK_POS_CHANGED < li AVC_EVENT_FLAG_BATT_STATUS_CHANGED < li AVC_EVENT_FLAG_SYSTEM_STATUS_CHANGED < li AVC_EVENT_FLAG_PLAYER_APPLICATION_SETTING_CHANGED < li AVC_EVENT_FLAG_NOW_PLAYING_CONTENT_CHANGED < li AVC_EVENT_FLAG_AVAILABLE_PLAYERS_CHANGED < li AVC_EVENT_FLAG_ADDRESSED_PLAYER_CHANGED < li AVC_EVENT_FLAG_UIDS_CHANGED < li AVC_EVENT_FLAG_VOLUME_CHANGED
bt_ulong song_length;	< Current song length
bt_ulong song_position;	< Current song poistion
bt_byte play_status;	< Current playback status
bt_byte volume;	< Current volumne
bt_byte battery_status;	< Current battery statusThis can be one of the following values: < li AVC_BATTERY_STATUS_NORMAL < li AVC_BATTERY_STATUS_WARNING < li AVC_BATTERY_STATUS_CRITICAL < li AVC_BATTERY_STATUS_EXTERNAL < li AVC_BATTERY_STATUS_FULL_CHARGE
bt_byte system_status;	< Current system status
bt_av_element_id_t current_track_id;	< Current track id
bt_uint uid_counter;	< The number of media items in the target.
bt_avrcp_channel_t* channels;	< List of available AVRCP channels.
bt_avrcp_mgr_callback_fp callback;	< Pointer to a function used to notify the AVRCP consumer about various events.
void* callback_param;	< Pointer to arbitrary data to be passed to the c callback.

Description

brief AVRCP manager. ingroup avrcp

details A structure that glues all pieces together. There is only one instance of this structure allocated by dotstack. A pointer to the instance can be get with c bt_avrcp_get_mgr().

bt_av_add_to_now_playing_t Structure

File

avrcp_command.h

C

```
typedef struct _bt_av_add_to_now_playing_s {
  bt_av_response_t header;
  bt_byte status;
} bt_av_add_to_now_playing_t;
```

Members

Members	Description
bt_av_response_t header;	< Common response header.
bt_byte status;	< The result of the request.

Description

brief Parameter to AVRCP_EVT_ADD_TO_NOW_PLAYING_COMPLETED event ingroup avrcp

details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::add_to_now_playing_status - when a local device received a response to a "add to now playing" request.

bt_av_battery_status_of_ct_t Structure

File

avrcp_command.h

C

```
typedef struct _bt_av_battery_status_of_ct_s {
  bt_av_response_t header;
  bt_byte status;
} bt_av_battery_status_of_ct_t;
```

Members

Members	Description
bt_av_response_t header;	< Common response header.
bt_byte status;	< Battery status. This can be one of the following values: < li AVC_BATTERY_STATUS_NORMAL < li AVC_BATTERY_STATUS_WARNING < li AVC_BATTERY_STATUS_CRITICAL < li AVC_BATTERY_STATUS_EXTERNAL < li AVC_BATTERY_STATUS_FULL_CHARGE

Description

brief Parameter to AVRCP_EVT_BATTERY_STATUS_OF_CT_RECEIVED event ingroup avrcp

details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::battery_status_of_ct - when a local device received a "battery status of controller" command.

bt_av_capability_company_id_t Structure

File

avrcp_command.h

C

```
typedef struct _bt_av_capability_company_id_s {
  bt_av_response_t header;
  bt_byte count;
  bt_ulong* company_id_list;
} bt_av_capability_company_id_t;
```

Members

Members	Description
bt_av_response_t header;	< Common response header.
bt_byte count;	< The number of supported company ids.
bt_ulong* company_id_list;	< List of supported company ids.

Description

brief Parameter to AVRCP_EVT_COMPANY_ID_LIST_RECEIVED event ingroup avrcp

details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::company_id - when a local device received a response to a "get company id" request.

bt_av_capability_event_id_t Structure

File

avrcp_command.h

C

```
typedef struct _bt_av_capability_event_id_s {
  bt_av_response_t header;
  bt_byte count;
  bt_byte* event_id_list;
} bt_av_capability_event_id_t;
```

Members

Members	Description
bt_av_response_t header;	< Common response header.
bt_byte count;	< The number of supported events ids.
bt_byte* event_id_list;	< List of supported event ids.

Description

brief Parameter to AVRCP_EVT_EVENT_ID_LIST_RECEIVED event ingroup avrcp

details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::supported_event_id - when a local device received a response to a "get supported events" request.

bt_av_command_t Type

File

avrcp_command.h

C

```
typedef struct _bt_av_command_t bt_av_command_t;
```

Description

This is type bt_av_command_t.

bt_av_displayable_character_set_t Structure

File

avrcp_command.h

C

```
typedef struct _bt_av_displayable_character_set_s {
  bt_av_response_t header;
  bt_byte count;
  bt_uint* charset_list;
} bt_av_displayable_character_set_t;
```

Members

Members	Description
bt_av_response_t header;	< Common response header.
bt_byte count;	< The number of supported characters sets by the controller.
bt_uint* charset_list;	< List of supported characters sets by the controller.

Description

brief Parameter to AVRCP_EVT_DISPLAYABLE_CHARACTER_SET_RECEIVED event ingroup avrcp

details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::displayable_character_set - when a local device received a "displayable chracter set command" request.

bt_av_element_attribute_t Structure

File

avrcp_command.h

C

```
typedef struct _bt_av_element_attribute_s {
  bt_ulong id;
  bt_uint charset;
  bt_uint len;
  bt_byte* value;
} bt_av_element_attribute_t;
```

Members

Members	Description
bt_ulong id;	< Attribute Id.
bt_uint charset;	< Charcater set.
bt_uint len;	< Value length.
bt_byte* value;	< Attribute value.

Description

brief Media element attribute ingroup avrcp

details This structure is used to store media element attribute.

bt_av_element_attributes_t Structure

File

avrcp_command.h

C

```
typedef struct _bt_av_element_attributes_s {
  bt_av_response_t header;
  bt_byte count;
  bt_av_element_attribute_t* attr_list;
} bt_av_element_attributes_t;
```

Members

Members	Description
bt_av_response_t header;	< Common response header.
bt_byte count;	< The number of attributes returned.
bt_av_element_attribute_t* attr_list;	< List of attribute values

Description

brief Parameter to AVRCP_EVT_GET_ELEMENT_ATTRIBUTES_RECEIVED event ingroup avrcp

details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::element_attributes - when a local device received a response to a "get media element attributes" request.

bt_av_element_id_t Structure

File

avrcp_command.h

C

```
typedef struct _bt_av_element_id_s {
  bt_ulong id_lo;
  bt_ulong id_hi;
} bt_av_element_id_t;
```

Members

Members	Description
bt_ulong id_lo;	< 4 least significant bytes of UID.
bt_ulong id_hi;	< 4 most significant bytes of UID.

Description

brief Media element UID ingroup avrcp

details This structure is used to store media element UID.

bt_av_get_element_attributes_t Structure

File

avrcp_command.h

C

```
typedef struct _bt_av_get_element_attributes_s {
  bt_av_response_t header;
  bt_av_element_id_t id;
  bt_byte attributes;
} bt_av_get_element_attributes_t;
```

Members

Members	Description
bt_av_response_t header;	< Common response header.
bt_av_element_id_t id;	< Media element UID
bt_byte attributes;	< Bitmask that defines attributes requested. This can be a combination of the following values: < li AVC_MEDIA_ATTR_FLAG_TITLE < li AVC_MEDIA_ATTR_FLAG_ARTIST < li AVC_MEDIA_ATTR_FLAG_ALBUM < li AVC_MEDIA_ATTR_FLAG_NUMBER < li AVC_MEDIA_ATTR_FLAG_TOTAL_NUMBER < li AVC_MEDIA_ATTR_FLAG_GENRE < li AVC_MEDIA_ATTR_FLAG_PLAYING_TIME

Description

brief Parameter to AVRCP_EVT_ELEMENT_ATTRIBUTES_REQUESTED event ingroup avrcp

details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::get_element_attributes - when a local device received a "get element attributes" request.

bt_av_notification_addressed_player_changed_t Structure

File

avrcp_command.h

C

```
typedef struct _bt_av_notification_addressed_player_changed_s {
  bt_uint player_id;
  bt_uint uid_counter;
} bt_av_notification_addressed_player_changed_t;
```

Members

Members	Description
bt_uint player_id;	< New player Id
bt_uint uid_counter;	< UID counter

Description

brief Parameter to AVRCP_EVT_ADDRESSED_PLAYER_CHANGED event ingroup avrcp

details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::notification::params:addressed_player - when a local device received a "addressed player changed" notification.

bt_av_notification_app_setting_changed_t Structure

File

avrcp_command.h

C

```
typedef struct _bt_av_notification_app_setting_changed_s {
  bt_byte setting_id;
  bt_byte setting_value_id;
} bt_av_notification_app_setting_changed_t;
```

Description

This is type bt_av_notification_app_setting_changed_t.

bt_av_notification_battery_status_t Structure

File

avrcp_command.h

C

```
typedef struct _bt_av_notification_battery_status_s {
  bt_byte status;
} bt_av_notification_battery_status_t;
```

Members

Members	Description
bt_byte status;	< New battery status.This can be one of the following values: < li AVC_BATTERY_STATUS_NORMAL < li AVC_BATTERY_STATUS_WARNING < li AVC_BATTERY_STATUS_CRITICAL < li AVC_BATTERY_STATUS_EXTERNAL < li AVC_BATTERY_STATUS_FULL_CHARGE

Description

brief Parameter to AVRCP_EVT_BATT_STATUS_CHANGED event ingroup avrcp

details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::notification::params:battery_status - when a local device received a "battery status changed" notification.

bt_av_notification_playback_pos_changed_t Structure

File

avrcp_command.h

C

```
typedef struct _bt_av_notification_playback_pos_changed_s {
  bt_ulong position;
} bt_av_notification_playback_pos_changed_t;
```

Members

Members	Description
bt_ulong position;	< New playback position

Description

brief Parameter to AVRCP_EVT_PLAYBACK_POS_CHANGED event ingroup avrcp

details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::notification::params:playback_pos - when a local device received a "playback position changed" notification.

bt_av_notification_playback_status_changed_t Structure

File

avrcp_command.h

C

```
typedef struct _bt_av_notification_playback_status_changed_s {
  bt_av_response_t header;
  bt_byte status;
} bt_av_notification_playback_status_changed_t;
```

Members

Members	Description
bt_av_response_t header;	< Common response header.
bt_byte status;	< Play status. This can be on of the following values: < li AVC_PLAY_STATUS_STOPPED < li AVC_PLAY_STATUS_PLAYING < li AVC_PLAY_STATUS_PAUSED < li AVC_PLAY_STATUS_FW_SEEK < li AVC_PLAY_STATUS_REV_SEEK < li AVC_PLAY_STATUS_ERROR

Description

brief Parameter to AVRCP_EVT_PLAYBACK_STATUS_CHANGED event ingroup avrcp

details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::notification::params:play_status - when a local device received a "play status changed" notification.

bt_av_notification_system_status_changed_t Structure

File

avrcp_command.h

C

```
typedef struct _bt_av_notification_system_status_changed_s {
  bt_byte status;
} bt_av_notification_system_status_changed_t;
```

Members

Members	Description
bt_byte status;	< New system status

Description

brief Parameter to AVRCP_EVT_SYSTEM_STATUS_CHANGED event ingroup avrcp

details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::notification::params:system_status - when a local device received a "system status changed" notification.

bt_av_notification_t Structure

File

avrcp_command.h

C

```
typedef struct _bt_av_notification_s {
  bt_av_response_t header;
  union {
    bt_av_notification_playback_status_changed_t play_status;
    bt_av_notification_track_changed_t track;
    bt_av_notification_playback_pos_changed_t playback_pos;
    bt_av_notification_battery_status_t battery_status;
    bt_av_notification_system_status_changed_t system_status;
    bt_av_notification_addressed_player_changed_t addressed_player;
    bt_av_notification_uids_changed_t uids;
    bt_av_notification_volume_changed_t volume;
    bt_av_notification_app_setting_changed_t app_setting;
  } params;
} bt_av_notification_t;
```

Members

Members	Description
bt_av_response_t header;	< Common response header.
bt_av_notification_playback_status_changed_t play_status;	< Valid if notification is AVRCP_EVT_PLAYBACK_STATUS_CHANGED
bt_av_notification_track_changed_t track;	< Valid if notification is AVRCP_EVT_TRACK_CHANGED
bt_av_notification_playback_pos_changed_t playback_pos;	< Valid if notification is AVRCP_EVT_PLAYBACK_POS_CHANGED
bt_av_notification_battery_status_t battery_status;	< Valid if notification is AVRCP_EVT_BATT_STATUS_CHANGED
bt_av_notification_system_status_changed_t system_status;	< Valid if notification is AVRCP_EVT_SYSTEM_STATUS_CHANGED
bt_av_notification_addressed_player_changed_t addressed_player;	< Valid if notification is AVRCP_EVT_ADDRESSED_PLAYER_CHANGED
bt_av_notification_uids_changed_t uids;	< Valid if notification is AVRCP_EVT_UIDS_CHANGED
bt_av_notification_volume_changed_t volume;	< Valid if notification is AVRCP_EVT_VOLUME_CHANGED
bt_av_notification_app_setting_changed_t app_setting;	< Valid if notification is AVRCP_EVT_PLAYER_APPLICATION_SETTING_CHANGED

Description

brief Parameter to the following events: li AVRCP_EVT_PLAYBACK_STATUS_CHANGED li AVRCP_EVT_TRACK_CHANGED li AVRCP_EVT_PLAYBACK_POS_CHANGED li AVRCP_EVT_BATT_STATUS_CHANGED li AVRCP_EVT_SYSTEM_STATUS_CHANGED li AVRCP_EVT_NOW_PLAYING_CONTENT_CHANGED li AVRCP_EVT_AVAILABLE_PLAYERS_CHANGED li AVRCP_EVT_ADDRESSED_PLAYER_CHANGED li AVRCP_EVT_UIDS_CHANGED li AVRCP_EVT_VOLUME_CHANGED ingroup avrcp

details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::notification - when a local device received one of the following notifications from the target: li Play status changed li Track changed changed li Playback position changed li Battery status changed li System status changed li Addressed player changed li UIDs changed li Volume changed li Player application setting changed The notification code defines which member of the bt_av_notification_t::params union is valid @arg AVRCP_EVT_PLAYBACK_STATUS_CHANGED c bt_av_notification_playback_status_changed_t play_status @arg

AVRCP_EVT_TRACK_CHANGED c bt_av_notification_track_changed_t track @arg AVRCP_EVT_PLAYBACK_POS_CHANGED c bt_av_notification_playback_pos_changed_t playback_pos @arg AVRCP_EVT_BATT_STATUS_CHANGED c bt_av_notification_battery_status_t battery_status @arg AVRCP_EVT_SYSTEM_STATUS_CHANGED c bt_av_notification_system_status_changed_t system_status @arg AVRCP_EVT_ADDRESSED_PLAYER_CHANGED c bt_av_notification_addressed_player_changed_t addressed_player @arg AVRCP_EVT_UIDS_CHANGED c bt_av_notification_uids_changed_t uids @arg AVRCP_EVT_VOLUME_CHANGED c bt_av_notification_volume_changed_t volume @arg AVRCP_EVT_PLAYER_APPLICATION_SETTING_CHANGED c bt_av_notification_app_setting_changed_t app_setting

bt_av_notification_track_changed_t Structure

File

avrcp_command.h

C

```
typedef struct _bt_av_notification_track_changed_s {
  bt_av_element_id_t id;
} bt_av_notification_track_changed_t;
```

Members

Members	Description
bt_av_element_id_t id;	< New track UID

Description

brief Parameter to AVRCP_EVT_TRACK_CHANGED event ingroup avrcp

details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::notification::params:track - when a local device received a "track changed" notification.

bt_av_notification_uids_changed_t Structure

File

avrcp_command.h

C

```
typedef struct _bt_av_notification_uids_changed_s {
  bt_uint uid_counter;
} bt_av_notification_uids_changed_t;
```

Members

Members	Description
bt_uint uid_counter;	< UID counter

Description

brief Parameter to AVRCP_EVT_UIDS_CHANGED event ingroup avrcp

details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::notification::params:uids - when a local device received a "UIDs changed" notification.

bt_av_notification_volume_changed_t Structure

File

avrcp_command.h

C

```
typedef struct _bt_av_notification_volume_changed_s {
  bt_byte volume;
} bt_av_notification_volume_changed_t;
```

Members

Members	Description
bt_byte volume;	< Volume

Description

brief Parameter to AVRCP_EVT_VOLUME_CHANGED event ingroup avrcp

details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::notification::params:volume - when a local device received a "UIDs changed" notification.

bt_av_play_item_t Structure

File

avrcp_command.h

C

```
typedef struct _bt_av_play_item_s {
  bt_av_response_t header;
  bt_byte status;
} bt_av_play_item_t;
```

Members

Members	Description
bt_av_response_t header;	< Common response header.
bt_byte status;	< The result of the request.

Description

brief Parameter to AVRCP_EVT_PLAY_ITEM_COMPLETED event ingroup avrcp

details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::play_item_status - when a local device received a response to a "play item" request.

bt_av_play_status_t Structure

File

avrcp_command.h

C

```
typedef struct _bt_av_play_status_s {
  bt_av_response_t header;
  bt_ulong song_length;
  bt_ulong song_position;
  bt_byte play_status;
} bt_av_play_status_t;
```

Members

Members	Description
bt_av_response_t header;	< Common response header.
bt_ulong song_length;	< Current track length
bt_ulong song_position;	< Current track position
bt_byte play_status;	< Playback status. This can be one of the following values: < li AVC_PLAY_STATUS_STOPPED < li AVC_PLAY_STATUS_PLAYING < li AVC_PLAY_STATUS_PAUSED < li AVC_PLAY_STATUS_FW_SEEK < li AVC_PLAY_STATUS_REV_SEEK < li AVC_PLAY_STATUS_ERROR

Description

brief Parameter to AVRCP_EVT_GET_PLAY_STATUS_RECEIVED event ingroup avrcp

details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::play_status - when a local device received a response to a "get play status" request.

bt_av_player_setting_current_values_t Structure

File

avrcp_command.h

C

```
typedef struct _bt_av_player_setting_current_values_s {
  bt_av_response_t header;
  bt_byte count;
  bt_byte* setting_id_list;
  bt_byte* setting_value_id_list;
} bt_av_player_setting_current_values_t;
```

Members

Members	Description
bt_av_response_t header;	< Common response header.
bt_byte count;	< The number of player setting attribute ids to be returned from the target.
bt_byte* setting_id_list;	< List of player setting attribute ids to be returned from the target.
bt_byte* setting_value_id_list;	< List of current player setting attribute value ids.

Description

brief Parameter to AVRCP_EVT_PLAYER_CURRENT_SETTING_VALUES_RECEIVED event ingroup avrcp

details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::player_setting_current_values - when a local device received a response to a "get current player setting attribute values" request.

bt_av_player_setting_values_t Structure

File

avrcp_command.h

C

```
typedef struct _bt_av_player_setting_values_s {
  bt_av_response_t header;
  bt_byte count;
  bt_byte* setting_value_id_list;
} bt_av_player_setting_values_t;
```

Members

Members	Description
bt_av_response_t header;	< Common response header.
bt_byte count;	< The number of supported player setting attribute value ids.
bt_byte* setting_value_id_list;	< List of supported player setting attribute value ids.

Description

brief Parameter to AVRCP_EVT_PLAYER_SETTING_VALUES_RECEIVED event ingroup avrcp

details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::player_setting_values - when a local device received a response to a "get player setting attribute values" request.

bt_av_player_setting_values_text_t Structure

File

avrcp_command.h

C

```
typedef struct _bt_av_player_setting_values_text_s {
  bt_av_response_t header;
  bt_byte count;
  bt_byte* setting_value_id_list;
  bt_av_player_text_t* setting_value_text_list;
} bt_av_player_setting_values_text_t;
```

Members

Members	Description
bt_av_response_t header;	< Common response header.

bt_byte count;	< The number of player setting attribute value ids for which displayable text is requested.
bt_byte* setting_value_id_list;	< List of player setting attribute value ids for which displayable text is requested.
bt_av_player_text_t* setting_value_text_list;	< List of player setting attribute values displayable text.

Description

brief Parameter to AVRCP_EVT_PLAYER_SETTING_VALUES_TEXT_RECEIVED event ingroup avrcp

details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::player_setting_values_text - when a local device received a response to a "get player setting attribute values displayable text" request.

bt_av_player_settings_t Structure

File

avrcp_command.h

C

```
typedef struct _bt_av_player_settings_s {
  bt_av_response_t header;
  bt_byte count;
  bt_byte* setting_id_list;
} bt_av_player_settings_t;
```

Members

Members	Description
bt_av_response_t header;	< Common response header.
bt_byte count;	< The number of supported player setting attribute ids.
bt_byte* setting_id_list;	< List of supported player setting attribute ids.

Description

brief Parameter to AVRCP_EVT_PLAYER_SETTING_ATTRIBUTES_RECEIVED event ingroup avrcp

details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::player_settings - when a local device received a response to a "get supported player setting attributes" request.

bt_av_player_settings_text_t Structure

File

avrcp_command.h

C

```
typedef struct _bt_av_player_settings_text_s {
  bt_av_response_t header;
  bt_byte count;
  bt_byte* setting_id_list;
  bt_av_player_text_t* setting_text_list;
} bt_av_player_settings_text_t;
```

Members

Members	Description
bt_av_response_t header;	< Common response header.
bt_byte count;	< The number of player setting attribute ids for which displayable text is requested.
bt_byte* setting_id_list;	< List of player setting attribute ids for which displayable text is requested.
bt_av_player_text_t* setting_text_list;	< List of player setting attributes displayable text.

Description

brief Parameter to AVRCP_EVT_PLAYER_SETTING_ATTRIBUTES_TEXT_RECEIVED event ingroup avrcp

details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::player_settings_text - when a local device received a response to a "get player setting attributes displayable text" request.

bt_av_player_text_t Structure

File

avrcp_command.h

C

```
typedef struct _bt_av_player_text_s {
  bt_uint charset;
  bt_byte len;
  bt_byte* text;
} bt_av_player_text_t;
```

Description

This is type bt_av_player_text_t.

bt_av_register_notification_t Structure

File

avrcp_command.h

C

```
typedef struct _bt_av_register_notification_t {
  bt_av_response_t header;
  bt_byte event_id;
  bt_ulong playback_pos;
} bt_av_register_notification_t;
```

Members

Members	Description
bt_av_response_t header;	< Common response header.
bt_byte event_id;	< Event Id. This can be one of the following values: < li AVC_EVENT_PLAYBACK_STATUS_CHANGED < li AVC_EVENT_TRACK_CHANGED < li AVC_EVENT_TRACK_REACHED_END < li AVC_EVENT_TRACK_REACHED_START < li AVC_EVENT_PLAYBACK_POS_CHANGED < li AVC_EVENT_BATT_STATUS_CHANGED < li AVC_EVENT_SYSTEM_STATUS_CHANGED < li AVC_EVENT_PLAYER_APPLICATION_SETTING_CHANGED < li AVC_EVENT_NOW_PLAYING_CONTENT_CHANGED < li AVC_EVENT_AVAILABLE_PLAYERS_CHANGED < li AVC_EVENT_ADDRESSED_PLAYER_CHANGED < li AVC_EVENT_UIDS_CHANGED < li AVC_EVENT_VOLUME_CHANGED
bt_ulong playback_pos;	< Playback position. Used only if c event_id is AVC_EVENT_PLAYBACK_POS_CHANGED

Description

brief Parameter to AVRCP_EVT_REGISTER_NOTIFICATION_REQUESTED event ingroup avrcp

details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::register_notification - when a local device received a "register notification" request.

bt_av_response_t Structure

File

avrcp_command.h

C

```
typedef struct _bt_av_response_t {
  bt_avrcp_channel_t* channel;
  bt_byte ctype;
  bt_byte tran_id;
} bt_av_response_t;
```

Members

Members	Description
bt_avrcp_channel_t* channel;	< AVRCP channel

bt_byte ctype;	< Response type. This can be one of the following values: < li AVC_RESPONSE_NOT_IMPLEMENTED < li AVC_RESPONSE_ACCEPTED < li AVC_RESPONSE_REJECTED < li AVC_RESPONSE_IN_TRANSITION < li AVC_RESPONSE_STABLE < li AVC_RESPONSE_IMPLEMENTED < li AVC_RESPONSE_CHANGED < li AVC_RESPONSE_INTERIM < li AVC_RESPONSE_TIMEOUT
bt_byte tran_id;	< Transaction Id.

Description

brief AV/C response header ingroup avrcp

details This structure is used to store fields present in every AV/C response.

bt_av_set_absolute_volume_t Structure

File

avrcp_command.h

C

```
typedef struct _bt_av_set_absolute_volume_s {
  bt_av_response_t header;
  bt_byte volume;
} bt_av_set_absolute_volume_t;
```

Members

Members	Description
bt_av_response_t header;	< Common response header.
bt_byte volume;	< Volume

Description

brief Parameter to AVRCP_EVT_SET_ABSOLUTE_VOLUME_COMPLETED event ingroup avrcp

details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::absolute_volume - when a local device received a response to a "set absolute volume" request.

bt_av_set_addressed_player_t Structure

File

avrcp_command.h

C

```
typedef struct _bt_av_set_addressed_player_s {
  bt_av_response_t header;
  bt_byte status;
} bt_av_set_addressed_player_t;
```

Members

Members	Description
bt_av_response_t header;	< Common response header.
bt_byte status;	< The result of changing the addressed player.

Description

brief Parameter to AVRCP_EVT_SET_ADDRESSED_PLAYER_COMPLETED event ingroup avrcp

details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::addressed_player - when a local device received a response to a "set addressed player" request.

bt_avrcp_channel_t Type

File

avrcp.h

C

```
typedef struct _bt_avrcp_channel_t bt_avrcp_channel_t;
```

Description

This is type bt_avrcp_channel_t.

bt_avrcp_device_t Structure

File

avrcp.h

C

```
typedef struct _bt_avrcp_device_s {
  bt_bdaddr_t address;
  bt_byte* name;
  bt_byte pg_scan_rpt_mode;
  bt_byte pg_scan_period_mode;
  bt_long cod;
  bt_int clock_offset;
  bt_byte rssi;
} bt_avrcp_device_t;
```

Description

This is type bt_avrcp_device_t.

bt_avrcp_event_t Union

File

avrcp.h

C

```
typedef union _bt_avrcp_event_u {
  bt_avrcp_evt_channel_connected_t channel_connected;
  bt_avrcp_evt_channel_disconnected_t channel_disconnected;
  bt_avrcp_evt_connection_failed_t connection_failed;
  bt_avrcp_evt_panel_response_received_t panel_response_received;
  bt_avrcp_evt_search_completed_t device_search;
  bt_av_response_t response;
  bt_av_capability_company_id_t company_id;
  bt_av_capability_event_id_t supported_event_id;
  bt_av_player_settings_t player_settings;
  bt_av_player_setting_values_t player_setting_values;
  bt_av_player_setting_current_values_t player_setting_current_values;
  bt_av_player_settings_text_t player_settings_text;
  bt_av_player_setting_values_text_t player_setting_values_text;
  bt_av_element_attributes_t element_attributes;
  bt_av_play_status_t play_status;
  bt_av_set_absolute_volume_t absolute_volume;
  bt_av_set_addressed_player_t addressed_player;
  bt_av_play_item_t play_item_status;
  bt_av_add_to_now_playing_t add_to_now_playing_status;
  bt_av_notification_t notification;
  bt_avrcp_evt_register_events_completed_t register_events;
  bt_avrcp_evt_panel_command_received_t panel_command_received;
  bt_av_battery_status_of_ct_t battery_status_of_ct;
  bt_av_displayable_character_set_t displayable_character_set;
  bt_av_get_element_attributes_t get_element_attributes;
  bt_av_register_notification_t register_notification;
} bt_avrcp_event_t;
```

Members

Members	Description
bt_avrcp_evt_channel_connected_t channel_connected;	< Valid if event is AVRCP_EVT_CONTROL_CHANNEL_CONNECTED
bt_avrcp_evt_channel_disconnected_t channel_disconnected;	< Valid if event is AVRCP_EVT_CONTROL_CHANNEL_DISCONNECTED

bt_avrcp_evt_connection_failed_t connection_failed;	< Valid if event is AVRCP_EVT_CONTROL_CONNECTION_FAILED
bt_avrcp_evt_panel_response_received_t panel_response_received;	< Valid if event is AVRCP_EVT_PANEL_RESPONSE_RECEIVED
bt_avrcp_evt_search_completed_t device_search;	< Valid if event is AVRCP_EVT_SEARCH_COMPLETED
bt_av_capability_company_id_t company_id;	< Valid if event is AVRCP_EVT_COMPANY_ID_LIST_RECEIVED
bt_av_capability_event_id_t supported_event_id;	< Valid if event is AVRCP_EVT_EVENT_ID_LIST_RECEIVED
bt_av_player_settings_t player_settings;	< Valid if event is AVRCP_EVT_PLAYER_SETTING_ATTRIBUTES_RECEIVED
bt_av_player_setting_values_t player_setting_values;	< Valid if event is AVRCP_EVT_PLAYER_SETTING_VALUES_RECEIVED
bt_av_player_setting_current_values_t player_setting_current_values;	< Valid if event is AVRCP_EVT_PLAYER_CURRENT_SETTING_VALUES_RECEIVED
bt_av_player_settings_text_t player_settings_text;	< Valid if event is AVRCP_EVT_PLAYER_SETTING_ATTRIBUTES_TEXT_RECEIVED
bt_av_player_setting_values_text_t player_setting_values_text;	< Valid if event is AVRCP_EVT_PLAYER_SETTING_VALUES_TEXT_RECEIVED
bt_av_element_attributes_t element_attributes;	< Valid if event is AVRCP_EVT_GET_ELEMENT_ATTRIBUTES_RECEIVED
bt_av_play_status_t play_status;	< Valid if event is AVRCP_EVT_GET_PLAY_STATUS_RECEIVED
bt_av_set_absolute_volume_t absolute_volume;	< Valid if event is AVRCP_EVT_SET_ABSOLUTE_VOLUME_COMPLETED
bt_av_set_addressed_player_t addressed_player;	< Valid if event is AVRCP_EVT_SET_ADDRESSED_PLAYER_COMPLETED
bt_av_play_item_t play_item_status;	< Valid if event is AVRCP_EVT_PLAY_ITEM_COMPLETED
bt_av_add_to_now_playing_t add_to_now_playing_status;	< Valid if event is AVRCP_EVT_ADD_TO_NOW_PLAYING_COMPLETED
bt_av_notification_t notification;	< Valid if event is on of the following: < li AVRCP_EVT_PLAYBACK_STATUS_CHANGED < li AVRCP_EVT_TRACK_CHANGED < li AVRCP_EVT_TRACK_REACHED_END < li AVRCP_EVT_TRACK_REACHED_START < li AVRCP_EVT_PLAYBACK_POS_CHANGED < li AVRCP_EVT_BATT_STATUS_CHANGED < li AVRCP_EVT_SYSTEM_STATUS_CHANGED < li AVRCP_EVT_PLAYER_APPLICATION_SETTING_CHANGED < li AVRCP_EVT_NOW_PLAYING_CONTENT_CHANGED < li AVRCP_EVT_AVAILABLE_PLAYERS_CHANGED < li AVRCP_EVT_ADDRESSED_PLAYER_CHANGED < li AVRCP_EVT_UIDS_CHANGED < li AVRCP_EVT_VOLUME_CHANGED
bt_avrcp_evt_register_events_completed_t register_events;	< Valid if event AVRCP_EVT_REGISTER_NOTIFICATIONS_COMPLETED
bt_avrcp_evt_panel_command_received_t panel_command_received;	< Valid if event is AVRCP_EVT_PANEL_COMMAND_RECEIVED
bt_av_battery_status_of_ct_t battery_status_of_ct;	< Valid if event is AVRCP_EVT_BATTERY_STATUS_OF_CT_RECEIVED
bt_av_displayable_character_set_t displayable_character_set;	< Valid if event is AVRCP_EVT_DISPLAYABLE_CHARACTER_SET_RECEIVED
bt_av_get_element_attributes_t get_element_attributes;	< Valid if event is AVRCP_EVT_ELEMENT_ATTRIBUTES_REQUESTED
bt_av_register_notification_t register_notification;	< Valid if event is AVRCP_EVT_REGISTER_NOTIFICATION_REQUESTED

Description

brief Parameter to an application callback. ingroup avrcp

details This union is used to pass event specific data to the AVRCP consumer. Which member of the union points to a valid structure depends on the event reported to the consumer. In general, each event has a corresponding member in the union.

bt_avrcp_evt_channel_connected_t Structure

File

avrcp.h

C

```
typedef struct _bt_avrcp_evt_channel_connected_t {
  bt_avrcp_channel_t* channel;
} bt_avrcp_evt_channel_connected_t;
```

Members

Members	Description
bt_avrcp_channel_t* channel;	< AVRCP channel

Description

brief Parameter to AVRCP_EVT_CONTROL_CHANNEL_CONNECTED event ingroup avrcp

details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::channel_connected - when a control channel between two devices has been established.

bt_avrcp_evt_channel_disconnected_t Structure

File

avrcp.h

C

```
typedef struct _bt_avrcp_evt_channel_disconnected_t {
  bt_avrcp_channel_t* channel;
} bt_avrcp_evt_channel_disconnected_t;
```

Members

Members	Description
bt_avrcp_channel_t* channel;	< AVRCP channel

Description

brief Parameter to AVRCP_EVT_CONTROL_CHANNEL_DISCONNECTED event ingroup avrcp

details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::channel_disconnected - when a control channel between two devices has been terminated.

bt_avrcp_evt_connection_failed_t Structure

File

avrcp.h

C

```
typedef struct _bt_avrcp_evt_connection_failed_t {
  bt_avrcp_channel_t* channel;
} bt_avrcp_evt_connection_failed_t;
```

Members

Members	Description
bt_avrcp_channel_t* channel;	< AVRCP channel

Description

brief Parameter to AVRCP_EVT_CONTROL_CONNECTION_FAILED event ingroup avrcp

details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::connection_failed - when a local device failed to create a control channel between two AVRCP entities.

bt_avrcp_evt_panel_command_received_t Structure

File

avrcp.h

C

```
typedef struct _bt_avrcp_evt_panel_command_received_t {
  bt_avrcp_channel_t* channel;
  bt_byte ctype;
  bt_byte button_status;
  bt_byte opcode;
  bt_byte* params;
  bt_byte params_len;
} bt_avrcp_evt_panel_command_received_t;
```

Members

Members	Description
bt_avrcp_channel_t* channel;	< AVRCP channel
bt_byte ctype;	< Command type
bt_byte button_status;	< Button status
bt_byte opcode;	< Operation Id
bt_byte* params;	< Operation parameters
bt_byte params_len;	< Length of the operation parameters

Description

brief Parameter to AVRCP_EVT_PANEL_COMMAND_RECEIVED event ingroup avrcp

details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::panel_command_received - when a local device received a PASS THROUGH command.

bt_avrcp_evt_panel_response_received_t Structure

File

avrcp.h

C

```
typedef struct _bt_avrcp_evt_panel_response_received_t {
  bt_avrcp_channel_t* channel;
  bt_byte ctype;
  bt_byte button_status;
  bt_byte opcode;
  bt_byte* params;
  bt_byte params_len;
} bt_avrcp_evt_panel_response_received_t;
```

Members

Members	Description
bt_avrcp_channel_t* channel;	< AVRCP channel
bt_byte ctype;	< Response type
bt_byte button_status;	< Button status
bt_byte opcode;	< Operation Id
bt_byte* params;	< Operation parameters
bt_byte params_len;	< Length of the operation parameters

Description

brief Parameter to AVRCP_EVT_PANEL_RESPONSE_RECEIVED event ingroup avrcp

details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::panel_response_received - when a local device received a response to a PASS THROUGH command.

bt_avrcp_evt_register_events_completed_t Structure

File

avrcp.h

C

```
typedef struct _bt_avrcp_evt_register_events_completed_t {
  bt_avrcp_channel_t* channel;
} bt_avrcp_evt_register_events_completed_t;
```

Members

Members	Description
bt_avrcp_channel_t* channel;	< AVRCP channel

Description

This is type bt_avrcp_evt_register_events_completed_t.

bt_avrcp_evt_search_completed_t Structure

File

avrcp.h

C

```
typedef struct _bt_avrcp_evt_search_completed_s {
  bt_avrcp_device_t* devices;
  bt_byte count;
} bt_avrcp_evt_search_completed_t;
```

Members

Members	Description
bt_avrcp_device_t* devices;	< list of found devices
bt_byte count;	< the number of found devices

Description

brief Parameter to AVRCP_EVT_SEARCH_COMPLETED event ingroup avrcp

details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::device_search - when searching for nearby devices has finished.

bt_avrcp_find_callback_fp Type

File

avrcp.h

C

```
typedef void (* bt_avrcp_find_callback_fp)(bt_bool found, void* param);
```

Description

brief Find Controller/Target callback. ingroup avrcp

details This callback is called when search for Controller/Target has finished.

param found This can be one of the following values: li c TRUE if Controller/Target was found. li c FALSE otherwise. param callback_param A pointer to an arbitrary data set by a call to bt_avrcp_find_target/bt_avrcp_find_controller.

bt_avrcp_mgr_callback_fp Type

File

avrcp.h

C

```
typedef void (* bt_avrcp_mgr_callback_fp)(bt_avrcp_mgr_t* mgr, bt_byte evt, bt_avrcp_event_t* evt_param,
void* callback_param);
```

Description

brief AVRCP application callback. ingroup avrcp

details In order to be notified of various events a consumer of the AVRCP layer has to register a callback function (done with bt_avrcp_start()). The stack will call that function whenever a new event has been generated.

param mgr AVRCP manager.

param evt AVRCP event. The event can be one of the following values: @arg AVRCP_EVT_CONTROL_CHANNEL_CONNECTED This event is generated when a control channel between two AVRCP entities has been established. @arg AVRCP_EVT_CONTROL_CHANNEL_DISCONNECTED This event is generated when a control channel between two AVRCP entities has been terminated. @arg AVRCP_EVT_CONTROL_CONNECTION_FAILED This event is generated when a local device failed to create a control channel between two AVRCP entities. @arg AVRCP_EVT_BROWSING_CHANNEL_CONNECTED This event is generated when a browsing channel between two AVRCP entities has been established. @arg AVRCP_EVT_BROWSING_CHANNEL_DISCONNECTED This event is generated when a browsing channel between two AVRCP entities has been terminated. @arg AVRCP_EVT_BROWSING_CONNECTION_FAILED This event is generated when a local device failed to create a browsing channel between two AVRCP entities. @arg AVRCP_EVT_SEARCH_COMPLETED This event is generated when a local device completed searching for nearby targets.

@arg AVRCP_EVT_PANEL_RESPONSE_RECEIVED This event is generated when a local device received a response to a PASS THROUGH command. @arg AVRCP_EVT_COMPANY_ID_LIST_RECEIVED This event is generated when a local device received a response to a "get company id" request. @arg AVRCP_EVT_EVENT_ID_LIST_RECEIVED This event is generated when a local device received a response to a "get supported events" request. @arg AVRCP_EVT_PLAYER_SETTING_ATTRIBUTES_RECEIVED This event is generated when a local device received a response to a "get supported player setting attributes" request. @arg AVRCP_EVT_PLAYER_SETTING_VALUES_RECEIVED This event is generated when a local device received a response to a "get player setting attribute values" request. @arg AVRCP_EVT_PLAYER_CURRENT_SETTING_VALUES_RECEIVED This event is generated when a local device received a response to a "get current player setting attribute values" request. @arg AVRCP_EVT_SET_PLAYER_SETTING_VALUES_COMPLETED This event is generated when a local device received a response to a "set player setting attribute values" request. @arg AVRCP_EVT_PLAYER_SETTING_ATTRIBUTES_TEXT_RECEIVED This event is generated when a local device received a response to a "get player setting attributes displayable text" request. @arg AVRCP_EVT_PLAYER_SETTING_VALUES_TEXT_RECEIVED This event is generated when a local device received a response to a "get player setting attribute values displayable text" request. @arg AVRCP_EVT_INFORM_DISPLAYABLE_CHARACTER_SET_COMPLETED This event is generated when a local device received a response to a "inform displayable character set" request. @arg AVRCP_EVT_INFORM_BATTERY_STATUS_OF_CT_COMPLETED This event is generated when a local device received a response to a "inform battery status" request. @arg AVRCP_EVT_GET_ELEMENT_ATTRIBUTES_RECEIVED This event is generated when a local device received a response to a "get media element attributes" request. @arg AVRCP_EVT_GET_PLAY_STATUS_RECEIVED This event is generated when a local device received a response to a "get play status" request. @arg AVRCP_EVT_PLAYBACK_STATUS_CHANGED This event is generated when a local device received a "play status changed" notification. @arg AVRCP_EVT_TRACK_CHANGED This event is generated when a local device received a "track changed changed" notification. @arg AVRCP_EVT_TRACK_REACHED_END This event is generated when a local device received a "track reached end" notification. @arg AVRCP_EVT_TRACK_REACHED_START This event is generated when a local device received a "track reached start" notification. @arg AVRCP_EVT_PLAYBACK_POS_CHANGED This event is generated when a local device received a "playback position changed" notification. @arg AVRCP_EVT_BATT_STATUS_CHANGED This event is generated when a local device received a "battery status changed" notification. @arg AVRCP_EVT_SYSTEM_STATUS_CHANGED This event is generated when a local device received a "system status changed" notification. @arg AVRCP_EVT_PLAYER_APPLICATION_SETTING_CHANGED This event is generated when a local device received a "player application setting changed" notification. @arg AVRCP_EVT_NOW_PLAYING_CONTENT_CHANGED This event is generated when a local device received a "now playing content changed" notification. @arg AVRCP_EVT_AVAILABLE_PLAYERS_CHANGED This event is generated when a local device received a "available players changed" notification. @arg AVRCP_EVT_ADDRESSED_PLAYER_CHANGED This event is generated when a local device received a "addressed player changed" notification. @arg AVRCP_EVT_UIDS_CHANGED This event is generated when a local device received a "UIDs changed" notification. @arg AVRCP_EVT_VOLUME_CHANGED This event is generated when a local device received a "volume changed" notification. @arg AVRCP_EVT_SET_ABSOLUTE_VOLUME_COMPLETED This event is generated when a local device received a response to a "set absolute volume" request. @arg AVRCP_EVT_SET_ADDRESSED_PLAYER_COMPLETED This event is generated when a local device received a response to a "set addressed player" request. @arg AVRCP_EVT_PLAY_ITEM_COMPLETED This event is generated when a local device received a response to a "play item" request. @arg AVRCP_EVT_ADD_TO_NOW_PLAYING_COMPLETED This event is generated when a local device received a response to a "add to now playing" request. @arg AVRCP_EVT_REGISTER_NOTIFICATIONS_COMPLETED This event is generated when a local device received a response to a "register notification" request.

@arg AVRCP_EVT_PANEL_COMMAND_RECEIVED This event is generated when a local device received a PASS THROUGH command. @arg AVRCP_EVT_SET_ABSOLUTE_VOLUME_REQUESTED This event is generated when a local device received a "set absolute volume" request. @arg AVRCP_EVT_BATTERY_STATUS_OF_CT_RECEIVED This event is generated when a local device received a "battery status of controller" command. @arg AVRCP_EVT_DISPLAYABLE_CHARACTER_SET_RECEIVED This event is generated when a local device received a "displayable chracter set command" request. @arg AVRCP_EVT_ELEMENT_ATTRIBUTES_REQUESTED This event is generated when a local device received a "get element attributes" request.

param evt_param Event parameter. Which member of the bt_avrcp_event_t union is valid depends on the event: @arg AVRCP_EVT_CONTROL_CHANNEL_CONNECTED c bt_avrcp_evt_channel_connected_t channel_connected @arg AVRCP_EVT_CONTROL_CHANNEL_DISCONNECTED c bt_avrcp_evt_channel_disconnected_t channel_disconnected @arg AVRCP_EVT_CONTROL_CONNECTION_FAILED c bt_avrcp_evt_connection_failed_t connection_failed @arg AVRCP_EVT_PANEL_RESPONSE_RECEIVED c bt_avrcp_evt_panel_response_received_t panel_response_received @arg AVRCP_EVT_SEARCH_COMPLETED c bt_avrcp_evt_search_completed_t device_search @arg AVRCP_EVT_COMPANY_ID_LIST_RECEIVED c bt_av_capability_company_id_t company_id @arg AVRCP_EVT_EVENT_ID_LIST_RECEIVED c bt_av_capability_event_id_t supported_event_id @arg AVRCP_EVT_PLAYER_SETTING_ATTRIBUTES_RECEIVED c bt_av_player_settings_t player_settings @arg AVRCP_EVT_PLAYER_SETTING_VALUES_RECEIVED c bt_av_player_setting_values_t player_setting_values @arg AVRCP_EVT_PLAYER_CURRENT_SETTING_VALUES_RECEIVED c bt_av_player_setting_current_values_t player_setting_current_values @arg AVRCP_EVT_PLAYER_SETTING_ATTRIBUTES_TEXT_RECEIVED c bt_av_player_settings_text_t player_settings_text @arg AVRCP_EVT_PLAYER_SETTING_VALUES_TEXT_RECEIVED c bt_av_player_setting_values_text_t player_setting_values_text @arg AVRCP_EVT_GET_ELEMENT_ATTRIBUTES_RECEIVED c bt_av_element_attributes_t element_attributes @arg AVRCP_EVT_GET_PLAY_STATUS_RECEIVED c bt_av_play_status_t play_status @arg AVRCP_EVT_SET_ABSOLUTE_VOLUME_COMPLETED c bt_av_set_absolute_volume_t absolute_volume @arg AVRCP_EVT_SET_ADDRESSED_PLAYER_COMPLETED c bt_av_set_addressed_player_t addressed_player @arg AVRCP_EVT_PLAY_ITEM_COMPLETED c bt_av_play_item_t play_item_status @arg AVRCP_EVT_ADD_TO_NOW_PLAYING_COMPLETED c bt_av_add_to_now_playing_t add_to_now_playing_status @arg AVRCP_EVT_PLAYBACK_STATUS_CHANGED c bt_av_notification_t notification @arg AVRCP_EVT_TRACK_CHANGED c bt_av_notification_t notification @arg AVRCP_EVT_TRACK_REACHED_END c bt_av_notification_t notification @arg AVRCP_EVT_TRACK_REACHED_START c bt_av_notification_t notification @arg AVRCP_EVT_PLAYBACK_POS_CHANGED c bt_av_notification_t notification @arg AVRCP_EVT_BATT_STATUS_CHANGED c bt_av_notification_t notification @arg AVRCP_EVT_SYSTEM_STATUS_CHANGED c bt_av_notification_t notification @arg AVRCP_EVT_PLAYER_APPLICATION_SETTING_CHANGED c bt_av_notification_t notification @arg AVRCP_EVT_NOW_PLAYING_CONTENT_CHANGED c bt_av_notification_t notification @arg AVRCP_EVT_AVAILABLE_PLAYERS_CHANGED c bt_av_notification_t notification @arg AVRCP_EVT_ADDRESSED_PLAYER_CHANGED c

bt_av_notification_t notification @arg AVRCP_EVT_UIDS_CHANGED c bt_av_notification_t notification @arg AVRCP_EVT_VOLUME_CHANGED c bt_av_notification_t notification

@arg AVRCP_EVT_PANEL_COMMAND_RECEIVED c bt_avrcp_evt_panel_command_received_t panel_command_received @arg AVRCP_EVT_BATTERY_STATUS_OF_CT_RECEIVED c bt_av_battery_status_of_ct_t battery_status_of_ct @arg AVRCP_EVT_DISPLAYABLE_CHARACTER_SET_RECEIVED c bt_av_displayable_character_set_t displayable_character_set @arg AVRCP_EVT_ELEMENT_ATTRIBUTES_REQUESTED c bt_av_get_element_attributes_t get_element_attributes

param callback_param A pointer to an arbitrary data set by a call to bt_avrcp_start.

bt_avrcp_mgr_t Type

File

avrcp.h

C

```
typedef struct _bt_avrcp_mgr_t bt_avrcp_mgr_t;
```

Description

This is type bt_avrcp_mgr_t.

__AVRCP_COMMAND_H Macro

File

avrcp_command.h

C

```
#define __AVRCP_COMMAND_H
```

Description

This is macro __AVRCP_COMMAND_H.

__AVRCP_H Macro

File

avrcp.h

C

```
#define __AVRCP_H
```

Description

This is macro __AVRCP_H.

__AVRCP_PRIVATE_H Macro

File

avrcp_private.h

C

```
#define __AVRCP_PRIVATE_H
```

Description

This is macro __AVRCP_PRIVATE_H.

AVC_BATTERY_STATUS_CRITICAL Macro

File

avrcp_command.h

C

```
#define AVC_BATTERY_STATUS_CRITICAL 2
```

Description

This is macro AVC_BATTERY_STATUS_CRITICAL.

AVC_BATTERY_STATUS_EXTERNAL Macro

File

avrcp_command.h

C

```
#define AVC_BATTERY_STATUS_EXTERNAL 3
```

Description

This is macro AVC_BATTERY_STATUS_EXTERNAL.

AVC_BATTERY_STATUS_FULL_CHARGE Macro

File

avrcp_command.h

C

```
#define AVC_BATTERY_STATUS_FULL_CHARGE 4
```

Description

This is macro AVC_BATTERY_STATUS_FULL_CHARGE.

AVC_BATTERY_STATUS_NORMAL Macro

File

avrcp_command.h

C

```
#define AVC_BATTERY_STATUS_NORMAL 0
```

Description

- addtogroup avrcp
- @{
- @name Battery status

AVC_BATTERY_STATUS_WARNING Macro

File

avrcp_command.h

C

```
#define AVC_BATTERY_STATUS_WARNING 1
```

Description

This is macro AVC_BATTERY_STATUS_WARNING.

AVC_CAPABILITY_COMPANY_ID Macro

File

avrcp_command.h

C

```
#define AVC_CAPABILITY_COMPANY_ID 2
```

Description

This is macro AVC_CAPABILITY_COMPANY_ID.

AVC_CAPABILITY_EVENTS_SUPPORTED Macro

File

avrcp_command.h

C

```
#define AVC_CAPABILITY_EVENTS_SUPPORTED 3
```

Description

This is macro AVC_CAPABILITY_EVENTS_SUPPORTED.

AVC_CMD_GENERAL_POWER Macro

File

avrcp_command.h

C

```
#define AVC_CMD_GENERAL_POWER 0xB2
```

Description

This is macro AVC_CMD_GENERAL_POWER.

AVC_CMD_GENERAL_RESERVE Macro

File

avrcp_command.h

C

```
#define AVC_CMD_GENERAL_RESERVE 0x01
```

Description

This is macro AVC_CMD_GENERAL_RESERVE.

AVC_CMD_GENERAL_SUBUNIT_INFO Macro

File

avrcp_command.h

C

```
#define AVC_CMD_GENERAL_SUBUNIT_INFO 0x31
```

Description

This is macro AVC_CMD_GENERAL_SUBUNIT_INFO.

AVC_CMD_GENERAL_UNIT_INFO Macro

File

avrcp_command.h

C

```
#define AVC_CMD_GENERAL_UNIT_INFO 0x30
```

Description

This is macro AVC_CMD_GENERAL_UNIT_INFO.

AVC_CMD_GENERAL_VENDOR_DEPENDENT Macro

File

avrcp_command.h

C

```
#define AVC_CMD_GENERAL_VENDOR_DEPENDENT 0x00
```

Description

This is macro AVC_CMD_GENERAL_VENDOR_DEPENDENT.

AVC_CMD_GENERAL_VERSION Macro

File

avrcp_command.h

C

```
#define AVC_CMD_GENERAL_VERSION 0xB0
```

Description

This is macro AVC_CMD_GENERAL_VERSION.

AVC_CMD_PANEL_PASS_THROUGH Macro

File

avrcp_command.h

C

```
#define AVC_CMD_PANEL_PASS_THROUGH 0x7C
```

Description

This is macro AVC_CMD_PANEL_PASS_THROUGH.

AVC_CTYPE_CONTROL Macro

File

avrcp_command.h

C

```
#define AVC_CTYPE_CONTROL 0
```

Description

- addtogroup avrcp
- @{
- @name Command types

AVC_CTYPE_GENERAL_INQUORY Macro

File

avrcp_command.h

C

```
#define AVC_CTYPE_GENERAL_INQUORY 4
```

Description

This is macro AVC_CTYPE_GENERAL_INQUORY.

AVC_CTYPE_NOTIFY Macro

File

avrcp_command.h

C

```
#define AVC_CTYPE_NOTIFY 3
```

Description

This is macro AVC_CTYPE_NOTIFY.

AVC_CTYPE_SPECIFIC_IQUIRY Macro

File

avrcp_command.h

C

```
#define AVC_CTYPE_SPECIFIC_IQUIRY 2
```

Description

This is macro AVC_CTYPE_SPECIFIC_IQUIRY.

AVC_CTYPE_STATUS Macro

File

avrcp_command.h

C

```
#define AVC_CTYPE_STATUS 1
```

Description

This is macro AVC_CTYPE_STATUS.

AVC_EVENT_ADDRESSED_PLAYER_CHANGED Macro

File

avrcp_command.h

C

```
#define AVC_EVENT_ADDRESSED_PLAYER_CHANGED 0x0b ///< The Addressed Player has been changed, see 6.9.2.
```

Description

< The Addressed Player has been changed, see 6.9.2.

AVC_EVENT_AVAILABLE_PLAYERS_CHANGED Macro

File

avrcp_command.h

C

```
#define AVC_EVENT_AVAILABLE_PLAYERS_CHANGED 0x0a ///< The available players have changed, see 6.9
```

Description

< The available players have changed, see 6.9

AVC_EVENT_BATT_STATUS_CHANGED Macro

File

avrcp_command.h

C

```
#define AVC_EVENT_BATT_STATUS_CHANGED 0x06 ///< Change in battery status
```

Description

< Change in battery status

AVC_EVENT_FLAG_ADDRESSED_PLAYER_CHANGED Macro

File

avrcp_command.h

C

```
#define AVC_EVENT_FLAG_ADDRESSED_PLAYER_CHANGED 0x0400
```

Description

This is macro AVC_EVENT_FLAG_ADDRESSED_PLAYER_CHANGED.

AVC_EVENT_FLAG_ALL Macro

File

avrcp_command.h

C

```
#define AVC_EVENT_FLAG_ALL 0x1FFF
```

Description

This is macro AVC_EVENT_FLAG_ALL.

AVC_EVENT_FLAG_AVAILABLE_PLAYERS_CHANGED Macro

File

avrcp_command.h

C

```
#define AVC_EVENT_FLAG_AVAILABLE_PLAYERS_CHANGED 0x0200
```

Description

This is macro AVC_EVENT_FLAG_AVAILABLE_PLAYERS_CHANGED.

AVC_EVENT_FLAG_BATT_STATUS_CHANGED Macro

File

avrcp_command.h

C

```
#define AVC_EVENT_FLAG_BATT_STATUS_CHANGED 0x0020
```

Description

This is macro AVC_EVENT_FLAG_BATT_STATUS_CHANGED.

AVC_EVENT_FLAG_NOW_PLAYING_CONTENT_CHANGED Macro

File

avrcp_command.h

C

```
#define AVC_EVENT_FLAG_NOW_PLAYING_CONTENT_CHANGED 0x0100
```

Description

This is macro AVC_EVENT_FLAG_NOW_PLAYING_CONTENT_CHANGED.

AVC_EVENT_FLAG_PLAYBACK_POS_CHANGED Macro

File

avrcp_command.h

C

```
#define AVC_EVENT_FLAG_PLAYBACK_POS_CHANGED 0x0010
```

Description

This is macro AVC_EVENT_FLAG_PLAYBACK_POS_CHANGED.

AVC_EVENT_FLAG_PLAYBACK_STATUS_CHANGED Macro

File

avrcp_command.h

C

```
#define AVC_EVENT_FLAG_PLAYBACK_STATUS_CHANGED 0x0001
```

Description

- addtogroup avrcp
- @{
- @name Notifications mask

AVC_EVENT_FLAG_PLAYER_APPLICATION_SETTING_CHANGED Macro

File

avrcp_command.h

C

```
#define AVC_EVENT_FLAG_PLAYER_APPLICATION_SETTING_CHANGED 0x0080
```

Description

This is macro AVC_EVENT_FLAG_PLAYER_APPLICATION_SETTING_CHANGED.

AVC_EVENT_FLAG_SYSTEM_STATUS_CHANGED Macro

File

avrcp_command.h

C

```
#define AVC_EVENT_FLAG_SYSTEM_STATUS_CHANGED 0x0040
```

Description

This is macro AVC_EVENT_FLAG_SYSTEM_STATUS_CHANGED.

AVC_EVENT_FLAG_TRACK_CHANGED Macro

File

avrcp_command.h

C

```
#define AVC_EVENT_FLAG_TRACK_CHANGED 0x0002
```

Description

This is macro AVC_EVENT_FLAG_TRACK_CHANGED.

AVC_EVENT_FLAG_TRACK_REACHED_END Macro

File

avrcp_command.h

C

```
#define AVC_EVENT_FLAG_TRACK_REACHED_END 0x0004
```

Description

This is macro AVC_EVENT_FLAG_TRACK_REACHED_END.

AVC_EVENT_FLAG_TRACK_REACHED_START Macro

File

avrcp_command.h

C

```
#define AVC_EVENT_FLAG_TRACK_REACHED_START 0x0008
```

Description

This is macro AVC_EVENT_FLAG_TRACK_REACHED_START.

AVC_EVENT_FLAG_UIDS_CHANGED Macro

File

avrcp_command.h

C

```
#define AVC_EVENT_FLAG_UIDS_CHANGED 0x0800
```

Description

This is macro AVC_EVENT_FLAG_UIDS_CHANGED.

AVC_EVENT_FLAG_VOLUME_CHANGED Macro

File

avrcp_command.h

C

```
#define AVC_EVENT_FLAG_VOLUME_CHANGED 0x1000
```

Description

This is macro AVC_EVENT_FLAG_VOLUME_CHANGED.

AVC_EVENT_NOW_PLAYING_CONTENT_CHANGED Macro

File

avrcp_command.h

C

```
#define AVC_EVENT_NOW_PLAYING_CONTENT_CHANGED 0x09 ///< The content of the Now Playing list has changed,
see 6.9.5.
```

Description

< The content of the Now Playing list has changed, see 6.9.5.

AVC_EVENT_PLAYBACK_POS_CHANGED Macro

File

avrcp_command.h

C

```
#define AVC_EVENT_PLAYBACK_POS_CHANGED 0x05 ///< Change in playback position. Returned after the specified
playback notification change notification interval
```

Description

< Change in playback position. Returned after the specified playback notification change notification interval

AVC_EVENT_PLAYBACK_STATUS_CHANGED Macro

File

avrcp_command.h

C

```
#define AVC_EVENT_PLAYBACK_STATUS_CHANGED 0x01 ///< Change in playback status of the current track.
```

Description

< Change in playback status of the current track.

AVC_EVENT_PLAYER_APPLICATION_SETTING_CHANGED Macro

File

avrcp_command.h

C

```
#define AVC_EVENT_PLAYER_APPLICATION_SETTING_CHANGED 0x08 ///< Change in player application setting
```

Description

< Change in player application setting

AVC_EVENT_SYSTEM_STATUS_CHANGED Macro

File

avrcp_command.h

C

```
#define AVC_EVENT_SYSTEM_STATUS_CHANGED 0x07 ///< Change in system status
```

Description

< Change in system status

AVC_EVENT_TRACK_CHANGED Macro

File

avrcp_command.h

C

```
#define AVC_EVENT_TRACK_CHANGED 0x02 ///< Change of current track
```

Description

< Change of current track

AVC_EVENT_TRACK_REACHED_END Macro

File

avrcp_command.h

C

```
#define AVC_EVENT_TRACK_REACHED_END 0x03 ///< Reached end of a track
```

Description

< Reached end of a track

AVC_EVENT_TRACK_REACHED_START Macro

File

avrcp_command.h

C

```
#define AVC_EVENT_TRACK_REACHED_START 0x04 ///< Reached start of a track
```

Description

< Reached start of a track

AVC_EVENT_UIDS_CHANGED Macro

File

avrcp_command.h

C

```
#define AVC_EVENT_UIDS_CHANGED 0x0c ///< The UIDs have changed, see 6.10.3.3.
```

Description

< The UIDs have changed, see 6.10.3.3.

AVC_EVENT_VOLUME_CHANGED Macro

File

avrcp_command.h

C

```
#define AVC_EVENT_VOLUME_CHANGED 0x0d ///< The volume has been changed locally on the TG, see 6.13.3
```

Description

< The volume has been changed locally on the TG, see 6.13.3

AVC_FLAG_BROWSING_CMD Macro

File

avrcp_command.h

C

```
#define AVC_FLAG_BROWSING_CMD 2
```

Description

This is macro AVC_FLAG_BROWSING_CMD.

AVC_FLAG_PANEL_CLICK Macro

File

avrcp_command.h

C

```
#define AVC_FLAG_PANEL_CLICK 1
```

Description

This is macro AVC_FLAG_PANEL_CLICK.

AVC_FLAG_RESPONSE Macro

File

avrcp_command.h

C

```
#define AVC_FLAG_RESPONSE 4
```

Description

This is macro AVC_FLAG_RESPONSE.

AVC_MAXEVENTS Macro

File

avrcp_command.h

C

```
#define AVC_MAXEVENTS 0x0d
```

Description

This is macro AVC_MAXEVENTS.

AVC_MEDIA_ATTR_FLAG_ALBUM Macro

File

avrcp_command.h

C

```
#define AVC_MEDIA_ATTR_FLAG_ALBUM 0x04
```

Description

This is macro AVC_MEDIA_ATTR_FLAG_ALBUM.

AVC_MEDIA_ATTR_FLAG_ALL Macro

File

avrcp_command.h

C

```
#define AVC_MEDIA_ATTR_FLAG_ALL 0x7F
```

Description

This is macro AVC_MEDIA_ATTR_FLAG_ALL.

AVC_MEDIA_ATTR_FLAG_ARTIST Macro

File

avrcp_command.h

C

```
#define AVC_MEDIA_ATTR_FLAG_ARTIST 0x02
```

Description

This is macro AVC_MEDIA_ATTR_FLAG_ARTIST.

AVC_MEDIA_ATTR_FLAG_GENRE Macro

File

avrcp_command.h

C

```
#define AVC_MEDIA_ATTR_FLAG_GENRE 0x20
```

Description

This is macro AVC_MEDIA_ATTR_FLAG_GENRE.

AVC_MEDIA_ATTR_FLAG_NUMBER Macro

File

avrcp_command.h

C

```
#define AVC_MEDIA_ATTR_FLAG_NUMBER 0x08
```

Description

This is macro AVC_MEDIA_ATTR_FLAG_NUMBER.

AVC_MEDIA_ATTR_FLAG_PLAYING_TIME Macro

File

avrcp_command.h

C

```
#define AVC_MEDIA_ATTR_FLAG_PLAYING_TIME 0x40
```

Description

This is macro AVC_MEDIA_ATTR_FLAG_PLAYING_TIME.

AVC_MEDIA_ATTR_FLAG_TITLE Macro

File

avrcp_command.h

C

```
#define AVC_MEDIA_ATTR_FLAG_TITLE 0x01
```

Description

- addtogroup avrcp
- @{
- @name Media attribute bitmask

AVC_MEDIA_ATTR_FLAG_TOTAL_NUMBER Macro

File

avrcp_command.h

C

```
#define AVC_MEDIA_ATTR_FLAG_TOTAL_NUMBER 0x10
```

Description

This is macro AVC_MEDIA_ATTR_FLAG_TOTAL_NUMBER.

AVC_MEDIA_ATTR_ID_ALBUM Macro

File

avrcp_command.h

C

```
#define AVC_MEDIA_ATTR_ID_ALBUM 3
```

Description

This is macro AVC_MEDIA_ATTR_ID_ALBUM.

AVC_MEDIA_ATTR_ID_ARTIST Macro

File

avrcp_command.h

C

```
#define AVC_MEDIA_ATTR_ID_ARTIST 2
```

Description

This is macro AVC_MEDIA_ATTR_ID_ARTIST.

AVC_MEDIA_ATTR_ID_GENRE Macro

File

avrcp_command.h

C

```
#define AVC_MEDIA_ATTR_ID_GENRE 6
```

Description

This is macro AVC_MEDIA_ATTR_ID_GENRE.

AVC_MEDIA_ATTR_ID_NUMBER Macro

File

avrcp_command.h

C

```
#define AVC_MEDIA_ATTR_ID_NUMBER 4
```

Description

This is macro AVC_MEDIA_ATTR_ID_NUMBER.

AVC_MEDIA_ATTR_ID_PLAYING_TIME Macro

File

avrcp_command.h

C

```
#define AVC_MEDIA_ATTR_ID_PLAYING_TIME 7
```

Description

This is macro AVC_MEDIA_ATTR_ID_PLAYING_TIME.

AVC_MEDIA_ATTR_ID_TITLE Macro

File

avrcp_command.h

C

```
#define AVC_MEDIA_ATTR_ID_TITLE 1
```

Description

- addtogroup avrcp
- @{
- @name Media attribute IDs

AVC_MEDIA_ATTR_ID_TOTAL_NUMBER Macro

File

avrcp_command.h

C

```
#define AVC_MEDIA_ATTR_ID_TOTAL_NUMBER 5
```

Description

This is macro AVC_MEDIA_ATTR_ID_TOTAL_NUMBER.

AVC_MEDIA_PLAYER_VIRTUAL_FILESYSTEM Macro

File

avrcp_command.h

C

```
#define AVC_MEDIA_PLAYER_VIRTUAL_FILESYSTEM 0x01 // Folder Item, Media Element Item  The virtual filesystem
containing the   Browsed
```

Description

Folder Item, Media Element Item The virtual filesystem containing the Browsed media content of the browsed player

AVC_NOW_PLAYING Macro

File

avrcp_command.h

C

```
#define AVC_NOW_PLAYING 0x03 // Media Element Item                The Now Playing list (or queue)
Addressed
```

Description

Media Element Item The Now Playing list (or queue) Addressed of the addressed player

AVC_PACKET_TYPE_CONTINUE Macro

File

avrcp_command.h

C

```
#define AVC_PACKET_TYPE_CONTINUE 2
```

Description

This is macro AVC_PACKET_TYPE_CONTINUE.

AVC_PACKET_TYPE_END Macro

File

avrcp_command.h

C

```
#define AVC_PACKET_TYPE_END 3
```

Description

This is macro AVC_PACKET_TYPE_END.

AVC_PACKET_TYPE_SINGLE Macro

File

avrcp_command.h

C

```
#define AVC_PACKET_TYPE_SINGLE 0
```

Description

This is macro AVC_PACKET_TYPE_SINGLE.

AVC_PACKET_TYPE_START Macro

File

avrcp_command.h

C

```
#define AVC_PACKET_TYPE_START 1
```

Description

This is macro AVC_PACKET_TYPE_START.

AVC_PANEL_BUTTON_PRESSED Macro

File

avrcp_command.h

C

```
#define AVC_PANEL_BUTTON_PRESSED 0
```

Description

This is macro AVC_PANEL_BUTTON_PRESSED.

AVC_PANEL_BUTTON_RELEASED Macro

File

avrcp_command.h

C

```
#define AVC_PANEL_BUTTON_RELEASED 1
```

Description

This is macro AVC_PANEL_BUTTON_RELEASED.

AVC_PANEL_OPID_0 Macro

File

avrcp_command.h

C

```
#define AVC_PANEL_OPID_0 0x20
```

Description

define AVC_PANEL_OPID_RESERVER 0x10 define AVC_PANEL_OPID_RESERVER 0x11 define AVC_PANEL_OPID_RESERVER 0x12 define AVC_PANEL_OPID_RESERVER 0x13 define AVC_PANEL_OPID_RESERVER 0x14 define AVC_PANEL_OPID_RESERVER 0x15 define AVC_PANEL_OPID_RESERVER 0x16 define AVC_PANEL_OPID_RESERVER 0x17 define AVC_PANEL_OPID_RESERVER 0x18 define AVC_PANEL_OPID_RESERVER 0x19 define AVC_PANEL_OPID_RESERVER 0x1A define AVC_PANEL_OPID_RESERVER 0x1B define AVC_PANEL_OPID_RESERVER 0x1C define AVC_PANEL_OPID_RESERVER 0x1D define AVC_PANEL_OPID_RESERVER 0x1E define AVC_PANEL_OPID_RESERVER 0x1F

AVC_PANEL_OPID_1 Macro

File

avrcp_command.h

C

```
#define AVC_PANEL_OPID_1 0x21
```

Description

This is macro AVC_PANEL_OPID_1.

AVC_PANEL_OPID_2 Macro

File

avrcp_command.h

C

```
#define AVC_PANEL_OPID_2 0x22
```

Description

This is macro AVC_PANEL_OPID_2.

AVC_PANEL_OPID_3 Macro

File

avrcp_command.h

C

```
#define AVC_PANEL_OPID_3 0x23
```

Description

This is macro AVC_PANEL_OPID_3.

AVC_PANEL_OPID_4 Macro

File

avrcp_command.h

C

```
#define AVC_PANEL_OPID_4 0x24
```

Description

This is macro AVC_PANEL_OPID_4.

AVC_PANEL_OPID_5 Macro

File

avrcp_command.h

C

```
#define AVC_PANEL_OPID_5 0x25
```

Description

This is macro AVC_PANEL_OPID_5.

AVC_PANEL_OPID_6 Macro

File

avrcp_command.h

C

```
#define AVC_PANEL_OPID_6 0x26
```

Description

This is macro AVC_PANEL_OPID_6.

AVC_PANEL_OPID_7 Macro

File

avrcp_command.h

C

```
#define AVC_PANEL_OPID_7 0x27
```

Description

This is macro AVC_PANEL_OPID_7.

AVC_PANEL_OPID_8 Macro

File

avrcp_command.h

C

```
#define AVC_PANEL_OPID_8 0x28
```

Description

This is macro AVC_PANEL_OPID_8.

AVC_PANEL_OPID_9 Macro

File

avrcp_command.h

C

```
#define AVC_PANEL_OPID_9 0x29
```

Description

This is macro AVC_PANEL_OPID_9.

AVC_PANEL_OPID_A Macro

File

avrcp_command.h

C

```
#define AVC_PANEL_OPID_A 0x7A
```

Description

This is macro AVC_PANEL_OPID_A.

AVC_PANEL_OPID_ANGLE Macro

File

avrcp_command.h

C

```
#define AVC_PANEL_OPID_ANGLE 0x50
```

Description

define AVC_PANEL_OPID_RESERVED 0x4E define AVC_PANEL_OPID_RESERVED 0x4F

AVC_PANEL_OPID_APPS_MENU Macro

File

avrcp_command.h

C

```
#define AVC_PANEL_OPID_APPS_MENU 0x0F
```

Description

This is macro AVC_PANEL_OPID_APPS_MENU.

AVC_PANEL_OPID_B Macro

File

avrcp_command.h

C

```
#define AVC_PANEL_OPID_B 0x7B
```

Description

This is macro AVC_PANEL_OPID_B.

AVC_PANEL_OPID_BACKWARD Macro

File

avrcp_command.h

C

```
#define AVC_PANEL_OPID_BACKWARD 0x4C
```

Description

This is macro AVC_PANEL_OPID_BACKWARD.

AVC_PANEL_OPID_C Macro

File

avrcp_command.h

C

```
#define AVC_PANEL_OPID_C 0x7C
```

Description

This is macro AVC_PANEL_OPID_C.

AVC_PANEL_OPID_CHANNEL_DOWN Macro

File

avrcp_command.h

C

```
#define AVC_PANEL_OPID_CHANNEL_DOWN 0x31
```

Description

This is macro AVC_PANEL_OPID_CHANNEL_DOWN.

AVC_PANEL_OPID_CHANNEL_UP Macro

File

avrcp_command.h

C

```
#define AVC_PANEL_OPID_CHANNEL_UP 0x30
```

Description

define AVC_PANEL_OPID_RESRVED 0x2D define AVC_PANEL_OPID_RESRVED 0x2E define AVC_PANEL_OPID_RESRVED 0x2F

AVC_PANEL_OPID_CLEAR Macro

File

avrcp_command.h

C

```
#define AVC_PANEL_OPID_CLEAR 0x2C
```

Description

This is macro AVC_PANEL_OPID_CLEAR.

AVC_PANEL_OPID_CONTENTS_MENU Macro

File

avrcp_command.h

C

```
#define AVC_PANEL_OPID_CONTENTS_MENU 0x0B
```

Description

This is macro AVC_PANEL_OPID_CONTENTS_MENU.

AVC_PANEL_OPID_D Macro

File

avrcp_command.h

C

```
#define AVC_PANEL_OPID_D 0x7D
```

Description

This is macro AVC_PANEL_OPID_D.

AVC_PANEL_OPID_DISPLAY_INFORMATION Macro

File

avrcp_command.h

C

```
#define AVC_PANEL_OPID_DISPLAY_INFORMATION 0x35
```

Description

This is macro AVC_PANEL_OPID_DISPLAY_INFORMATION.

AVC_PANEL_OPID_DOT Macro

File

avrcp_command.h

C

```
#define AVC_PANEL_OPID_DOT 0x2A
```

Description

This is macro AVC_PANEL_OPID_DOT.

AVC_PANEL_OPID_DOWN Macro

File

avrcp_command.h

C

```
#define AVC_PANEL_OPID_DOWN 0x02
```

Description

This is macro AVC_PANEL_OPID_DOWN.

AVC_PANEL_OPID_EJECT Macro

File

avrcp_command.h

C

```
#define AVC_PANEL_OPID_EJECT 0x4A
```

Description

This is macro AVC_PANEL_OPID_EJECT.

AVC_PANEL_OPID_ENTER Macro

File

avrcp_command.h

C

```
#define AVC_PANEL_OPID_ENTER 0x2B
```

Description

This is macro AVC_PANEL_OPID_ENTER.

AVC_PANEL_OPID_EXIT Macro

File

avrcp_command.h

C

```
#define AVC_PANEL_OPID_EXIT 0x0D
```

Description

This is macro AVC_PANEL_OPID_EXIT.

AVC_PANEL_OPID_F1 Macro

File

avrcp_command.h

C

```
#define AVC_PANEL_OPID_F1 0x71
```

Description

define AVC_PANEL_OPID_RESERVED 0x6D define AVC_PANEL_OPID_RESERVED 0x6E define AVC_PANEL_OPID_RESERVED 0x6F define AVC_PANEL_OPID_RESERVED 0x70

AVC_PANEL_OPID_F2 Macro

File

avrcp_command.h

C

```
#define AVC_PANEL_OPID_F2 0x72
```

Description

This is macro AVC_PANEL_OPID_F2.

AVC_PANEL_OPID_F3 Macro

File

avrcp_command.h

C

```
#define AVC_PANEL_OPID_F3 0x73
```

Description

This is macro AVC_PANEL_OPID_F3.

AVC_PANEL_OPID_F4 Macro

File

avrcp_command.h

C

```
#define AVC_PANEL_OPID_F4 0x74
```

Description

This is macro AVC_PANEL_OPID_F4.

AVC_PANEL_OPID_F5 Macro

File

avrcp_command.h

C

```
#define AVC_PANEL_OPID_F5 0x75
```

Description

This is macro AVC_PANEL_OPID_F5.

AVC_PANEL_OPID_F6 Macro

File

avrcp_command.h

C

```
#define AVC_PANEL_OPID_F6 0x76
```

Description

This is macro AVC_PANEL_OPID_F6.

AVC_PANEL_OPID_F7 Macro

File

avrcp_command.h

C

```
#define AVC_PANEL_OPID_F7 0x77
```

Description

This is macro AVC_PANEL_OPID_F7.

AVC_PANEL_OPID_F8 Macro

File

avrcp_command.h

C

```
#define AVC_PANEL_OPID_F8 0x78
```

Description

This is macro AVC_PANEL_OPID_F8.

AVC_PANEL_OPID_F9 Macro

File

avrcp_command.h

C

```
#define AVC_PANEL_OPID_F9 0x79
```

Description

This is macro AVC_PANEL_OPID_F9.

AVC_PANEL_OPID_FAST_FORWARD Macro

File

avrcp_command.h

C

```
#define AVC_PANEL_OPID_FAST_FORWARD 0x49
```

Description

This is macro AVC_PANEL_OPID_FAST_FORWARD.

AVC_PANEL_OPID_FAVORITE_MENU Macro

File

avrcp_command.h

C

```
#define AVC_PANEL_OPID_FAVORITE_MENU 0x0C
```

Description

This is macro AVC_PANEL_OPID_FAVORITE_MENU.

AVC_PANEL_OPID_FORWARD Macro

File

avrcp_command.h

C

```
#define AVC_PANEL_OPID_FORWARD 0x4B
```

Description

This is macro AVC_PANEL_OPID_FORWARD.

AVC_PANEL_OPID_HELP Macro

File

avrcp_command.h

C

```
#define AVC_PANEL_OPID_HELP 0x36
```

Description

This is macro AVC_PANEL_OPID_HELP.

AVC_PANEL_OPID_INPUT_SELECT Macro

File

avrcp_command.h

C

```
#define AVC_PANEL_OPID_INPUT_SELECT 0x34
```

Description

This is macro AVC_PANEL_OPID_INPUT_SELECT.

AVC_PANEL_OPID_KEYBORD_FUNCTION Macro

File

avrcp_command.h

C

```
#define AVC_PANEL_OPID_KEYBORD_FUNCTION 0x6C
```

Description

This is macro AVC_PANEL_OPID_KEYBORD_FUNCTION.

AVC_PANEL_OPID_LEFT Macro

File

avrcp_command.h

C

```
#define AVC_PANEL_OPID_LEFT 0x03
```

Description

This is macro AVC_PANEL_OPID_LEFT.

AVC_PANEL_OPID_LEFT_DOWN Macro

File

avrcp_command.h

C

```
#define AVC_PANEL_OPID_LEFT_DOWN 0x08
```

Description

This is macro AVC_PANEL_OPID_LEFT_DOWN.

AVC_PANEL_OPID_LEFT_UP Macro

File

avrcp_command.h

C

```
#define AVC_PANEL_OPID_LEFT_UP 0x07
```

Description

This is macro AVC_PANEL_OPID_LEFT_UP.

AVC_PANEL_OPID_LINKED_CONTENT Macro

File

avrcp_command.h

C

```
#define AVC_PANEL_OPID_LINKED_CONTENT 0x3F
```

Description

This is macro AVC_PANEL_OPID_LINKED_CONTENT.

AVC_PANEL_OPID_LIST Macro

File

avrcp_command.h

C

```
#define AVC_PANEL_OPID_LIST 0x4D
```

Description

This is macro AVC_PANEL_OPID_LIST.

AVC_PANEL_OPID_LIVE_TV Macro

File

avrcp_command.h

C

```
#define AVC_PANEL_OPID_LIVE_TV 0x39
```

Description

This is macro AVC_PANEL_OPID_LIVE_TV.

AVC_PANEL_OPID_LOCK Macro

File

avrcp_command.h

C

```
#define AVC_PANEL_OPID_LOCK 0x3B
```

Description

This is macro AVC_PANEL_OPID_LOCK.

AVC_PANEL_OPID_MUTE Macro

File

avrcp_command.h

C

```
#define AVC_PANEL_OPID_MUTE 0x43
```

Description

This is macro AVC_PANEL_OPID_MUTE.

AVC_PANEL_OPID_MUTE_FUNCTION Macro

File

avrcp_command.h

C

```
#define AVC_PANEL_OPID_MUTE_FUNCTION 0x65
```

Description

This is macro AVC_PANEL_OPID_MUTE_FUNCTION.

AVC_PANEL_OPID_NEXT_DAY Macro

File

avrcp_command.h

C

```
#define AVC_PANEL_OPID_NEXT_DAY 0x3D
```

Description

This is macro AVC_PANEL_OPID_NEXT_DAY.

AVC_PANEL_OPID_ON_DEMAND_MENU Macro

File

avrcp_command.h

C

```
#define AVC_PANEL_OPID_ON_DEMAND_MENU 0x0E
```

Description

This is macro AVC_PANEL_OPID_ON_DEMAND_MENU.

AVC_PANEL_OPID_PAGE_DOWN Macro

File

avrcp_command.h

C

```
#define AVC_PANEL_OPID_PAGE_DOWN 0x38
```

Description

This is macro AVC_PANEL_OPID_PAGE_DOWN.

AVC_PANEL_OPID_PAGE_UP Macro

File

avrcp_command.h

C

```
#define AVC_PANEL_OPID_PAGE_UP 0x37
```

Description

This is macro AVC_PANEL_OPID_PAGE_UP.

AVC_PANEL_OPID_PAUSE Macro

File

avrcp_command.h

C

```
#define AVC_PANEL_OPID_PAUSE 0x46
```

Description

This is macro AVC_PANEL_OPID_PAUSE.

AVC_PANEL_OPID_PAUSE_PLAY_FUNCTION Macro

File

avrcp_command.h

C

```
#define AVC_PANEL_OPID_PAUSE_PLAY_FUNCTION 0x61
```

Description

This is macro AVC_PANEL_OPID_PAUSE_PLAY_FUNCTION.

AVC_PANEL_OPID_PAUSE_RECORD_FUNCTION Macro

File

avrcp_command.h

C

```
#define AVC_PANEL_OPID_PAUSE_RECORD_FUNCTION 0x63
```

Description

This is macro AVC_PANEL_OPID_PAUSE_RECORD_FUNCTION.

AVC_PANEL_OPID_PIP_DOWN Macro

File

avrcp_command.h

C

```
#define AVC_PANEL_OPID_PIP_DOWN 0x53
```

Description

This is macro AVC_PANEL_OPID_PIP_DOWN.

AVC_PANEL_OPID_PIP_MOVE Macro

File

avrcp_command.h

C

```
#define AVC_PANEL_OPID_PIP_MOVE 0x52
```

Description

This is macro AVC_PANEL_OPID_PIP_MOVE.

AVC_PANEL_OPID_PIP_UP Macro

File

avrcp_command.h

C

```
#define AVC_PANEL_OPID_PIP_UP 0x54
```

Description

This is macro AVC_PANEL_OPID_PIP_UP.

AVC_PANEL_OPID_PLAY Macro

File

avrcp_command.h

C

```
#define AVC_PANEL_OPID_PLAY 0x44
```

Description

This is macro AVC_PANEL_OPID_PLAY.

AVC_PANEL_OPID_PLAY_FUNCTION Macro

File

avrcp_command.h

C

```
#define AVC_PANEL_OPID_PLAY_FUNCTION 0x60
```

Description

define AVC_PANEL_OPID_RESERVED 0x56 define AVC_PANEL_OPID_RESERVED 0x57 define AVC_PANEL_OPID_RESERVED 0x58 define AVC_PANEL_OPID_RESERVED 0x5A define AVC_PANEL_OPID_RESERVED 0x5B define AVC_PANEL_OPID_RESERVED 0x5C define AVC_PANEL_OPID_RESERVED 0x5D define AVC_PANEL_OPID_RESERVED 0x5E define AVC_PANEL_OPID_RESERVED 0x5F

AVC_PANEL_OPID_POWER_STATE_FUNCTION Macro

File

avrcp_command.h

C

```
#define AVC_PANEL_OPID_POWER_STATE_FUNCTION 0x6B
```

Description

This is macro AVC_PANEL_OPID_POWER_STATE_FUNCTION.

AVC_PANEL_OPID_POWER_TOGGLE Macro

File

avrcp_command.h

C

```
#define AVC_PANEL_OPID_POWER_TOGGLE 0x40
```

Description

This is macro AVC_PANEL_OPID_POWER_TOGGLE.

AVC_PANEL_OPID_PREVIOUS_CHANNEL Macro

File

avrcp_command.h

C

```
#define AVC_PANEL_OPID_PREVIOUS_CHANNEL 0x32
```

Description

This is macro AVC_PANEL_OPID_PREVIOUS_CHANNEL.

AVC_PANEL_OPID_PREVIOUS_DAY Macro

File

avrcp_command.h

C

```
#define AVC_PANEL_OPID_PREVIOUS_DAY 0x3E
```

Description

This is macro AVC_PANEL_OPID_PREVIOUS_DAY.

AVC_PANEL_OPID_RECORD Macro

File

avrcp_command.h

C

```
#define AVC_PANEL_OPID_RECORD 0x47
```

Description

This is macro AVC_PANEL_OPID_RECORD.

AVC_PANEL_OPID_RECORD_FUNCTION Macro

File

avrcp_command.h

C

```
#define AVC_PANEL_OPID_RECORD_FUNCTION 0x62
```

Description

This is macro AVC_PANEL_OPID_RECORD_FUNCTION.

AVC_PANEL_OPID_RESTORE_FOLUME_FUNCTION Macro

File

avrcp_command.h

C

```
#define AVC_PANEL_OPID_RESTORE_FOLUME_FUNCTION 0x66
```

Description

This is macro AVC_PANEL_OPID_RESTORE_FOLUME_FUNCTION.

AVC_PANEL_OPID_REWIND Macro

File

avrcp_command.h

C

```
#define AVC_PANEL_OPID_REWIND 0x48
```

Description

This is macro AVC_PANEL_OPID_REWIND.

AVC_PANEL_OPID_RF_BYPASS Macro

File

avrcp_command.h

C

```
#define AVC_PANEL_OPID_RF_BYPASS 0x55
```

Description

This is macro AVC_PANEL_OPID_RF_BYPASS.

AVC_PANEL_OPID_RIGHT Macro

File

avrcp_command.h

C

```
#define AVC_PANEL_OPID_RIGHT 0x04
```

Description

This is macro AVC_PANEL_OPID_RIGHT.

AVC_PANEL_OPID_RIGHT_DOWN Macro

File

avrcp_command.h

C

```
#define AVC_PANEL_OPID_RIGHT_DOWN 0x06
```

Description

This is macro AVC_PANEL_OPID_RIGHT_DOWN.

AVC_PANEL_OPID_RIGHT_UP Macro

File

avrcp_command.h

C

```
#define AVC_PANEL_OPID_RIGHT_UP 0x05
```

Description

This is macro AVC_PANEL_OPID_RIGHT_UP.

AVC_PANEL_OPID_ROOT_MENU Macro

File

avrcp_command.h

C

```
#define AVC_PANEL_OPID_ROOT_MENU 0x09
```

Description

This is macro AVC_PANEL_OPID_ROOT_MENU.

AVC_PANEL_OPID_SELECT Macro

File

avrcp_command.h

C

```
#define AVC_PANEL_OPID_SELECT 0x00
```

Description

- addtogroup avrcp
- @{
- @name AV/C Panel PASS THROUGH operation IDs

AVC_PANEL_OPID_SELECT_AUDIO_INPUT_FUNCTION Macro

File

avrcp_command.h

C

```
#define AVC_PANEL_OPID_SELECT_AUDIO_INPUT_FUNCTION 0x6A
```

Description

This is macro AVC_PANEL_OPID_SELECT_AUDIO_INPUT_FUNCTION.

AVC_PANEL_OPID_SELECT_AV_INPUT_FUNCTION Macro

File

avrcp_command.h

C

```
#define AVC_PANEL_OPID_SELECT_AV_INPUT_FUNCTION 0x69
```

Description

This is macro AVC_PANEL_OPID_SELECT_AV_INPUT_FUNCTION.

AVC_PANEL_OPID_SELECT_DISK_FUNCTION Macro

File

avrcp_command.h

C

```
#define AVC_PANEL_OPID_SELECT_DISK_FUNCTION 0x68
```

Description

This is macro AVC_PANEL_OPID_SELECT_DISK_FUNCTION.

AVC_PANEL_OPID_SETUP_MENU Macro

File

avrcp_command.h

C

```
#define AVC_PANEL_OPID_SETUP_MENU 0x0A
```

Description

This is macro AVC_PANEL_OPID_SETUP_MENU.

AVC_PANEL_OPID_SKIP Macro

File

avrcp_command.h

C

```
#define AVC_PANEL_OPID_SKIP 0x3C
```

Description

This is macro AVC_PANEL_OPID_SKIP.

AVC_PANEL_OPID_SOUND_SELECT Macro

File

avrcp_command.h

C

```
#define AVC_PANEL_OPID_SOUND_SELECT 0x33
```

Description

This is macro AVC_PANEL_OPID_SOUND_SELECT.

AVC_PANEL_OPID_STOP Macro

File

avrcp_command.h

C

```
#define AVC_PANEL_OPID_STOP 0x45
```

Description

This is macro AVC_PANEL_OPID_STOP.

AVC_PANEL_OPID_STOP_FUNCTION Macro

File

avrcp_command.h

C

```
#define AVC_PANEL_OPID_STOP_FUNCTION 0x64
```

Description

This is macro AVC_PANEL_OPID_STOP_FUNCTION.

AVC_PANEL_OPID_SUBPICTURE Macro

File

avrcp_command.h

C

```
#define AVC_PANEL_OPID_SUBPICTURE 0x51
```

Description

This is macro AVC_PANEL_OPID_SUBPICTURE.

AVC_PANEL_OPID_TUNE_FUNCTION Macro

File

avrcp_command.h

C

```
#define AVC_PANEL_OPID_TUNE_FUNCTION 0x67
```

Description

This is macro AVC_PANEL_OPID_TUNE_FUNCTION.

AVC_PANEL_OPID_UP Macro

File

avrcp_command.h

C

```
#define AVC_PANEL_OPID_UP 0x01
```

Description

This is macro AVC_PANEL_OPID_UP.

AVC_PANEL_OPID_VENDOR_UNIQUE Macro

File

avrcp_command.h

C

```
#define AVC_PANEL_OPID_VENDOR_UNIQUE 0x7E
```

Description

This is macro AVC_PANEL_OPID_VENDOR_UNIQUE.

AVC_PANEL_OPID_VOLUME_DOWN Macro

File

avrcp_command.h

C

```
#define AVC_PANEL_OPID_VOLUME_DOWN 0x42
```

Description

This is macro AVC_PANEL_OPID_VOLUME_DOWN.

AVC_PANEL_OPID_VOLUME_UP Macro

File

avrcp_command.h

C

```
#define AVC_PANEL_OPID_VOLUME_UP 0x41
```

Description

This is macro AVC_PANEL_OPID_VOLUME_UP.

AVC_PANEL_OPID_ZOOM Macro

File

avrcp_command.h

C

```
#define AVC_PANEL_OPID_ZOOM 0x3A
```

Description

This is macro AVC_PANEL_OPID_ZOOM.

AVC_PDUID_ABORT_CONTINUING_RESPONSE Macro

File

avrcp_command.h

C

```
#define AVC_PDUID_ABORT_CONTINUING_RESPONSE 0x41
```

Description

This is macro AVC_PDUID_ABORT_CONTINUING_RESPONSE.

AVC_PDUID_ADD_TO_NOW_PLAYING Macro

File

avrcp_command.h

C

```
#define AVC_PDUID_ADD_TO_NOW_PLAYING 0x90
```

Description

This is macro AVC_PDUID_ADD_TO_NOW_PLAYING.

AVC_PDUID_CHANGE_PATH Macro

File

avrcp_command.h

C

```
#define AVC_PDUID_CHANGE_PATH 0x72
```

Description

This is macro AVC_PDUID_CHANGE_PATH.

AVC_PDUID_GENERAL_REJECT Macro

File

avrcp_command.h

C

```
#define AVC_PDUID_GENERAL_REJECT 0xa0
```

Description

This is macro AVC_PDUID_GENERAL_REJECT.

AVC_PDUID_GET_CURRENT_PLAYER_APPLICATION_SETTING_VALUE Macro

File

avrcp_command.h

C

```
#define AVC_PDUID_GET_CURRENT_PLAYER_APPLICATION_SETTING_VALUE 0x13
```

Description

This is macro AVC_PDUID_GET_CURRENT_PLAYER_APPLICATION_SETTING_VALUE.

AVC_PDUID_GET_ELEMENT_ATTRIBUTES Macro

File

avrcp_command.h

C

```
#define AVC_PDUID_GET_ELEMENT_ATTRIBUTES 0x20
```

Description

This is macro AVC_PDUID_GET_ELEMENT_ATTRIBUTES.

AVC_PDUID_GET_FOLDER_ITEMS Macro

File

avrcp_command.h

C

```
#define AVC_PDUID_GET_FOLDER_ITEMS 0x71
```

Description

This is macro AVC_PDUID_GET_FOLDER_ITEMS.

AVC_PDUID_GET_ITEM_ATTRIBUTES Macro

File

avrcp_command.h

C

```
#define AVC_PDUID_GET_ITEM_ATTRIBUTES 0x73
```

Description

This is macro AVC_PDUID_GET_ITEM_ATTRIBUTES.

AVC_PDUID_GET_PLAY_STATUS Macro

File

avrcp_command.h

C

```
#define AVC_PDUID_GET_PLAY_STATUS 0x30
```

Description

This is macro AVC_PDUID_GET_PLAY_STATUS.

AVC_PDUID_GET_PLAYER_APPLICATION_SETTING_ATTRIBUTE_TEXT Macro

File

avrcp_command.h

C

```
#define AVC_PDUID_GET_PLAYER_APPLICATION_SETTING_ATTRIBUTE_TEXT 0x15
```

Description

This is macro AVC_PDUID_GET_PLAYER_APPLICATION_SETTING_ATTRIBUTE_TEXT.

AVC_PDUID_GET_PLAYER_APPLICATION_SETTING_VALUE_TEXT Macro

File

avrcp_command.h

C

```
#define AVC_PDUID_GET_PLAYER_APPLICATION_SETTING_VALUE_TEXT 0x16
```

Description

This is macro AVC_PDUID_GET_PLAYER_APPLICATION_SETTING_VALUE_TEXT.

AVC_PDUID_GETCAPABILITIES Macro

File

avrcp_command.h

C

```
#define AVC_PDUID_GETCAPABILITIES 0x10
```

Description

This is macro AVC_PDUID_GETCAPABILITIES.

AVC_PDUID_INFORM_BATTERY_STATUS_OF_CT Macro

File

avrcp_command.h

C

```
#define AVC_PDUID_INFORM_BATTERY_STATUS_OF_CT 0x18
```

Description

This is macro AVC_PDUID_INFORM_BATTERY_STATUS_OF_CT.

AVC_PDUID_INFORM_DISPLAYABLE_CHARACTER_SET Macro

File

avrcp_command.h

C

```
#define AVC_PDUID_INFORM_DISPLAYABLE_CHARACTER_SET 0x17
```

Description

This is macro AVC_PDUID_INFORM_DISPLAYABLE_CHARACTER_SET.

AVC_PDUID_LIST_PLAYER_APPLICATION_SETTING_ATTRIBUTES Macro

File

avrcp_command.h

C

```
#define AVC_PDUID_LIST_PLAYER_APPLICATION_SETTING_ATTRIBUTES 0x11
```

Description

This is macro AVC_PDUID_LIST_PLAYER_APPLICATION_SETTING_ATTRIBUTES.

AVC_PDUID_LIST_PLAYER_APPLICATION_SETTING_VALUES Macro

File

avrcp_command.h

C

```
#define AVC_PDUID_LIST_PLAYER_APPLICATION_SETTING_VALUES 0x12
```

Description

This is macro AVC_PDUID_LIST_PLAYER_APPLICATION_SETTING_VALUES.

AVC_PDUID_PLAY_ITEM Macro

File

avrcp_command.h

C

```
#define AVC_PDUID_PLAY_ITEM 0x74
```

Description

This is macro AVC_PDUID_PLAY_ITEM.

AVC_PDUID_REGISTER_NOTIFICATION Macro

File

avrcp_command.h

C

```
#define AVC_PDUID_REGISTER_NOTIFICATION 0x31
```

Description

This is macro AVC_PDUID_REGISTER_NOTIFICATION.

AVC_PDUID_REQUEST_CONTINUING_RESPONSE Macro

File

avrcp_command.h

C

```
#define AVC_PDUID_REQUEST_CONTINUING_RESPONSE 0x40
```

Description

This is macro AVC_PDUID_REQUEST_CONTINUING_RESPONSE.

AVC_PDUID_SEARCH Macro

File

avrcp_command.h

C

```
#define AVC_PDUID_SEARCH 0x80
```

Description

This is macro AVC_PDUID_SEARCH.

AVC_PDUID_SET_ABSOLUTE_VOLUME Macro

File

avrcp_command.h

C

```
#define AVC_PDUID_SET_ABSOLUTE_VOLUME 0x50
```

Description

This is macro AVC_PDUID_SET_ABSOLUTE_VOLUME.

AVC_PDUID_SET_ADDRESSED_PLAYER Macro

File

avrcp_command.h

C

```
#define AVC_PDUID_SET_ADDRESSED_PLAYER 0x60
```

Description

This is macro AVC_PDUID_SET_ADDRESSED_PLAYER.

AVC_PDUID_SET_BROWSED_PLAYER Macro

File

avrcp_command.h

C

```
#define AVC_PDUID_SET_BROWSED_PLAYER 0x70
```

Description

This is macro AVC_PDUID_SET_BROWSED_PLAYER.

AVC_PDUID_SET_PLAYER_APPLICATION_SETTING_VALUE Macro

File

avrcp_command.h

C

```
#define AVC_PDUID_SET_PLAYER_APPLICATION_SETTING_VALUE 0x14
```

Description

This is macro AVC_PDUID_SET_PLAYER_APPLICATION_SETTING_VALUE.

AVC_PLAY_STATUS_ERROR Macro

File

avrcp_command.h

C

```
#define AVC_PLAY_STATUS_ERROR 0xFF
```

Description

This is macro AVC_PLAY_STATUS_ERROR.

AVC_PLAY_STATUS_FW_SEEK Macro

File

avrcp_command.h

C

```
#define AVC_PLAY_STATUS_FW_SEEK 3
```

Description

This is macro AVC_PLAY_STATUS_FW_SEEK.

AVC_PLAY_STATUS_PAUSED Macro

File

avrcp_command.h

C

```
#define AVC_PLAY_STATUS_PAUSED 2
```

Description

This is macro AVC_PLAY_STATUS_PAUSED.

AVC_PLAY_STATUS_PLAYING Macro

File

avrcp_command.h

C

```
#define AVC_PLAY_STATUS_PLAYING 1
```

Description

This is macro AVC_PLAY_STATUS_PLAYING.

AVC_PLAY_STATUS_REV_SEEK Macro

File

avrcp_command.h

C

```
#define AVC_PLAY_STATUS_REV_SEEK 4
```

Description

This is macro AVC_PLAY_STATUS_REV_SEEK.

AVC_PLAY_STATUS_STOPPED Macro

File

avrcp_command.h

C

```
#define AVC_PLAY_STATUS_STOPPED 0
```

Description

- addtogroup avrcp
- @{
- @name Play status

AVC_PLAYER_SETTING_EQUALIZER_OFF Macro

File

avrcp_command.h

C

```
#define AVC_PLAYER_SETTING_EQUALIZER_OFF 1
```

Description

This is macro AVC_PLAYER_SETTING_EQUALIZER_OFF.

AVC_PLAYER_SETTING_EQUALIZER_ON Macro

File

avrcp_command.h

C

```
#define AVC_PLAYER_SETTING_EQUALIZER_ON 2
```

Description

This is macro AVC_PLAYER_SETTING_EQUALIZER_ON.

AVC_PLAYER_SETTING_EQUALIZER_STATUS Macro

File

avrcp_command.h

C

```
#define AVC_PLAYER_SETTING_EQUALIZER_STATUS 1
```

Description

This is macro AVC_PLAYER_SETTING_EQUALIZER_STATUS.

AVC_PLAYER_SETTING_REPEAT_ALL_TRACKS Macro

File

avrcp_command.h

C

```
#define AVC_PLAYER_SETTING_REPEAT_ALL_TRACKS 3
```

Description

This is macro AVC_PLAYER_SETTING_REPEAT_ALL_TRACKS.

AVC_PLAYER_SETTING_REPEAT_GROUP Macro

File

avrcp_command.h

C

```
#define AVC_PLAYER_SETTING_REPEAT_GROUP 4
```

Description

This is macro AVC_PLAYER_SETTING_REPEAT_GROUP.

AVC_PLAYER_SETTING_REPEAT_MODE_OFF Macro

File

avrcp_command.h

C

```
#define AVC_PLAYER_SETTING_REPEAT_MODE_OFF 1
```

Description

This is macro AVC_PLAYER_SETTING_REPEAT_MODE_OFF.

AVC_PLAYER_SETTING_REPEAT_MODE_STATUS Macro

File

avrcp_command.h

C

```
#define AVC_PLAYER_SETTING_REPEAT_MODE_STATUS 2
```

Description

This is macro AVC_PLAYER_SETTING_REPEAT_MODE_STATUS.

AVC_PLAYER_SETTING_REPEAT_SINGLE_TRACK Macro

File

avrcp_command.h

C

```
#define AVC_PLAYER_SETTING_REPEAT_SINGLE_TRACK 2
```

Description

This is macro AVC_PLAYER_SETTING_REPEAT_SINGLE_TRACK.

AVC_PLAYER_SETTING_SCAN_ALL_TRACKS Macro

File

avrcp_command.h

C

```
#define AVC_PLAYER_SETTING_SCAN_ALL_TRACKS 2
```

Description

This is macro AVC_PLAYER_SETTING_SCAN_ALL_TRACKS.

AVC_PLAYER_SETTING_SCAN_GROUP Macro

File

avrcp_command.h

C

```
#define AVC_PLAYER_SETTING_SCAN_GROUP 3
```

Description

This is macro AVC_PLAYER_SETTING_SCAN_GROUP.

AVC_PLAYER_SETTING_SCAN_OFF Macro

File

avrcp_command.h

C

```
#define AVC_PLAYER_SETTING_SCAN_OFF 1
```

Description

This is macro AVC_PLAYER_SETTING_SCAN_OFF.

AVC_PLAYER_SETTING_SCAN_STATUS Macro

File

avrcp_command.h

C

```
#define AVC_PLAYER_SETTING_SCAN_STATUS 4
```

Description

This is macro AVC_PLAYER_SETTING_SCAN_STATUS.

AVC_PLAYER_SETTING_SHUFFLE_ALL_TRACKS Macro

File

avrcp_command.h

C

```
#define AVC_PLAYER_SETTING_SHUFFLE_ALL_TRACKS 2
```

Description

This is macro AVC_PLAYER_SETTING_SHUFFLE_ALL_TRACKS.

AVC_PLAYER_SETTING_SHUFFLE_GROUP Macro

File

avrcp_command.h

C

```
#define AVC_PLAYER_SETTING_SHUFFLE_GROUP 3
```

Description

This is macro AVC_PLAYER_SETTING_SHUFFLE_GROUP.

AVC_PLAYER_SETTING_SHUFFLE_OFF Macro

File

avrcp_command.h

C

```
#define AVC_PLAYER_SETTING_SHUFFLE_OFF 1
```

Description

This is macro AVC_PLAYER_SETTING_SHUFFLE_OFF.

AVC_PLAYER_SETTING_SHUFFLE_STATUS Macro

File

avrcp_command.h

C

```
#define AVC_PLAYER_SETTING_SHUFFLE_STATUS 3
```

Description

This is macro AVC_PLAYER_SETTING_SHUFFLE_STATUS.

AVC_RESPONSE_ACCEPTED Macro

File

avrcp_command.h

C

```
#define AVC_RESPONSE_ACCEPTED 0x9
```

Description

This is macro AVC_RESPONSE_ACCEPTED.

AVC_RESPONSE_CHANGED Macro

File

avrcp_command.h

C

```
#define AVC_RESPONSE_CHANGED 0xD
```

Description

This is macro AVC_RESPONSE_CHANGED.

AVC_RESPONSE_IMPLEMENTED Macro

File

avrcp_command.h

C

```
#define AVC_RESPONSE_IMPLEMENTED 0xC
```

Description

This is macro AVC_RESPONSE_IMPLEMENTED.

AVC_RESPONSE_IN_TRANSITION Macro

File

avrcp_command.h

C

```
#define AVC_RESPONSE_IN_TRANSITION 0xB
```

Description

This is macro AVC_RESPONSE_IN_TRANSITION.

AVC_RESPONSE_INTERIM Macro

File

avrcp_command.h

C

```
#define AVC_RESPONSE_INTERIM 0xF
```

Description

This is macro AVC_RESPONSE_INTERIM.

AVC_RESPONSE_NOT_IMPLEMENTED Macro

File

avrcp_command.h

C

```
#define AVC_RESPONSE_NOT_IMPLEMENTED 0x8
```

Description

- addtogroup avrcp
- @{
- @name Response types

AVC_RESPONSE_REJECTED Macro

File

avrcp_command.h

C

```
#define AVC_RESPONSE_REJECTED 0xA
```

Description

This is macro AVC_RESPONSE_REJECTED.

AVC_RESPONSE_STABLE Macro

File

avrcp_command.h

C

```
#define AVC_RESPONSE_STABLE 0xC
```

Description

This is macro AVC_RESPONSE_STABLE.

AVC_RESPONSE_TIMEOUT Macro

File

avrcp_command.h

C

```
#define AVC_RESPONSE_TIMEOUT 0xF0
```

Description

This is macro AVC_RESPONSE_TIMEOUT.

AVC_SCOPE_MEDIA_PLAYER_LIST Macro

File

avrcp_command.h

C

```
#define AVC_SCOPE_MEDIA_PLAYER_LIST 0x00 // Media Player Item                Contains all available media
players    None
```

Description

Media Player Item Contains all available media players None

AVC_SEARCH Macro

File

avrcp_command.h

C

```
#define AVC_SEARCH 0x02 // Media Element Item              The results of a search operation       Browsed
```

Description

Media Element Item The results of a search operation Browsed on the browsed player

AVC_SUBUNIT_ID_EXTENDED_TO_NEXT_BYTE Macro

File

avrcp_command.h

C

```
#define AVC_SUBUNIT_ID_EXTENDED_TO_NEXT_BYTE 0x5
```

Description

This is macro AVC_SUBUNIT_ID_EXTENDED_TO_NEXT_BYTE.

AVC_SUBUNIT_ID_IGNORE Macro

File

avrcp_command.h

C

```
#define AVC_SUBUNIT_ID_IGNORE 0x7
```

Description

This is macro AVC_SUBUNIT_ID_IGNORE.

AVC_SUBUNIT_TYPE_AUDIO Macro

File

avrcp_command.h

C

```
#define AVC_SUBUNIT_TYPE_AUDIO 0x01
```

Description

This is macro AVC_SUBUNIT_TYPE_AUDIO.

AVC_SUBUNIT_TYPE_BULLETIN_BOARD Macro

File

avrcp_command.h

C

```
#define AVC_SUBUNIT_TYPE_BULLETIN_BOARD 0x0A
```

Description

This is macro AVC_SUBUNIT_TYPE_BULLETIN_BOARD.

AVC_SUBUNIT_TYPE_CA Macro

File

avrcp_command.h

C

```
#define AVC_SUBUNIT_TYPE_CA 0x06
```

Description

This is macro AVC_SUBUNIT_TYPE_CA.

AVC_SUBUNIT_TYPE_CAMERA Macro

File

avrcp_command.h

C

```
#define AVC_SUBUNIT_TYPE_CAMERA 0x07
```

Description

This is macro AVC_SUBUNIT_TYPE_CAMERA.

AVC_SUBUNIT_TYPE_CAMERA_STORAGE Macro

File

avrcp_command.h

C

```
#define AVC_SUBUNIT_TYPE_CAMERA_STORAGE 0x0B
```

Description

This is macro AVC_SUBUNIT_TYPE_CAMERA_STORAGE.

AVC_SUBUNIT_TYPE_DISC Macro

File

avrcp_command.h

C

```
#define AVC_SUBUNIT_TYPE_DISC 0x03
```

Description

This is macro AVC_SUBUNIT_TYPE_DISC.

AVC_SUBUNIT_TYPE_EXTENDED_TO_NEXT_BYTE Macro

File

avrcp_command.h

C

```
#define AVC_SUBUNIT_TYPE_EXTENDED_TO_NEXT_BYTE 0x1E
```

Description

This is macro AVC_SUBUNIT_TYPE_EXTENDED_TO_NEXT_BYTE.

AVC_SUBUNIT_TYPE_MONITOR Macro

File

avrcp_command.h

C

```
#define AVC_SUBUNIT_TYPE_MONITOR 0x00
```

Description

- addtogroup avrcp
- @{
- @name Subunit types

AVC_SUBUNIT_TYPE_PANEL Macro

File

avrcp_command.h

C

```
#define AVC_SUBUNIT_TYPE_PANEL 0x09
```

Description

This is macro AVC_SUBUNIT_TYPE_PANEL.

AVC_SUBUNIT_TYPE_PRINTER Macro

File

avrcp_command.h

C

```
#define AVC_SUBUNIT_TYPE_PRINTER 0x02
```

Description

This is macro AVC_SUBUNIT_TYPE_PRINTER.

AVC_SUBUNIT_TYPE_TAPE_RECORDER_PLAYER Macro

File

avrcp_command.h

C

```
#define AVC_SUBUNIT_TYPE_TAPE_RECORDER_PLAYER 0x04
```

Description

This is macro AVC_SUBUNIT_TYPE_TAPE_RECORDER_PLAYER.

AVC_SUBUNIT_TYPE_TUNER Macro

File

avrcp_command.h

C

```
#define AVC_SUBUNIT_TYPE_TUNER 0x05
```

Description

This is macro AVC_SUBUNIT_TYPE_TUNER.

AVC_SUBUNIT_TYPE_UNIT Macro

File

avrcp_command.h

C

```
#define AVC_SUBUNIT_TYPE_UNIT 0x1F
```

Description

This is macro AVC_SUBUNIT_TYPE_UNIT.

AVC_SUBUNIT_TYPE_VENDOR_UNIQUE Macro

File

avrcp_command.h

C

```
#define AVC_SUBUNIT_TYPE_VENDOR_UNIQUE 0x1C
```

Description

This is macro AVC_SUBUNIT_TYPE_VENDOR_UNIQUE.

AVC_VOLUME_MAX Macro

File

avrcp_command.h

C

```
#define AVC_VOLUME_MAX 0x7F // 100%
```

Description

100

AVC_VOLUME_MIN Macro

File

avrcp_command.h

C

```
#define AVC_VOLUME_MIN 0 // 0%
```

Description

0

AVRCP_BTSIG_COMPANY_ID Macro

File

avrcp.h

C

```
#define AVRCP_BTSIG_COMPANY_ID 0x001958
```

Description

This is macro AVRCP_BTSIG_COMPANY_ID.

AVRCP_CHANNEL_FLAG_LISTENING Macro

File

avrcp.h

C

```
#define AVRCP_CHANNEL_FLAG_LISTENING 1
```

Description

This is macro AVRCP_CHANNEL_FLAG_LISTENING.

AVRCP_CHANNEL_FLAG_PLAY_STATUS_REQUESTED Macro

File

avrcp.h

C

```
#define AVRCP_CHANNEL_FLAG_PLAY_STATUS_REQUESTED 8
```

Description

This is macro AVRCP_CHANNEL_FLAG_PLAY_STATUS_REQUESTED.

AVRCP_CHANNEL_FLAG_REGISTERING_NOTIFICATIONS Macro

File

avrcp.h

C

```
#define AVRCP_CHANNEL_FLAG_REGISTERING_NOTIFICATIONS 4
```

Description

This is macro AVRCP_CHANNEL_FLAG_REGISTERING_NOTIFICATIONS.

AVRCP_CHANNEL_FLAG_SENDING Macro

File

avrcp.h

C

```
#define AVRCP_CHANNEL_FLAG_SENDING 2
```

Description

This is macro AVRCP_CHANNEL_FLAG_SENDING.

AVRCP_CHANNEL_STATE_CONNECTED Macro

File

avrcp.h

C

```
#define AVRCP_CHANNEL_STATE_CONNECTED 3
```

Description

This is macro AVRCP_CHANNEL_STATE_CONNECTED.

AVRCP_CHANNEL_STATE_CONNECTING Macro

File

avrcp.h

C

```
#define AVRCP_CHANNEL_STATE_CONNECTING 2
```

Description

This is macro AVRCP_CHANNEL_STATE_CONNECTING.

AVRCP_CHANNEL_STATE_DISCONNECTING Macro

File

avrcp.h

C

```
#define AVRCP_CHANNEL_STATE_DISCONNECTING 4
```

Description

This is macro AVRCP_CHANNEL_STATE_DISCONNECTING.

AVRCP_CHANNEL_STATE_FREE Macro

File

avrcp.h

C

```
#define AVRCP_CHANNEL_STATE_FREE 0
```

Description

This is macro AVRCP_CHANNEL_STATE_FREE.

AVRCP_CHANNEL_STATE_IDLE Macro

File

avrcp.h

C

```
#define AVRCP_CHANNEL_STATE_IDLE 1
```

Description

This is macro AVRCP_CHANNEL_STATE_IDLE.

AVRCP_COMMAND_TYPE_BROWSING Macro

File

avrcp_command.h

C

```
#define AVRCP_COMMAND_TYPE_BROWSING 1
```

Description

This is macro AVRCP_COMMAND_TYPE_BROWSING.

AVRCP_COMMAND_TYPE_CONTROL Macro

File

avrcp_command.h

C

```
#define AVRCP_COMMAND_TYPE_CONTROL 0
```

Description

This is macro AVRCP_COMMAND_TYPE_CONTROL.

AVRCP_EVT_ADD_TO_NOW_PLAYING_COMPLETED Macro

File

avrcp.h

C

```
#define AVRCP_EVT_ADD_TO_NOW_PLAYING_COMPLETED 79    ///< This event is generated when a local device
received a response to a "add to now playing" request.
```

Description

< This event is generated when a local device received a response to a "add to now playing" request.

AVRCP_EVT_ADDRESSED_PLAYER_CHANGED Macro

File

avrcp.h

C

```
#define AVRCP_EVT_ADDRESSED_PLAYER_CHANGED 73    ///< This event is generated when a local device received
a "addressed player changed" notification.
```

Description

< This event is generated when a local device received a "addressed player changed" notification.

AVRCP_EVT_AVAILABLE_PLAYERS_CHANGED Macro

File

avrcp.h

C

```
#define AVRCP_EVT_AVAILABLE_PLAYERS_CHANGED 72    ///< This event is generated when a local device received
a "available players changed" notification.
```

Description

< This event is generated when a local device received a "available players changed" notification.

AVRCP_EVT_BATT_STATUS_CHANGED Macro

File

avrcp.h

C

```
#define AVRCP_EVT_BATT_STATUS_CHANGED 68    ///< This event is generated when a local device received a
"battery status changed" notification.
```

Description

< This event is generated when a local device received a "battery status changed" notification.

AVRCP_EVT_BATTERY_STATUS_OF_CT_RECEIVED Macro

File

avrcp.h

C

```
#define AVRCP_EVT_BATTERY_STATUS_OF_CT_RECEIVED 83    ///< This event is generated when a local device received a "battery status of controller" command.
```

Description

< This event is generated when a local device received a "battery status of controller" command.

AVRCP_EVT_BROWSING_CHANNEL_CONNECTED Macro

File

avrcp.h

C

```
#define AVRCP_EVT_BROWSING_CHANNEL_CONNECTED 4    ///< This event is generated when a browsing channel between two AVRCP entities has been established.
```

Description

< This event is generated when a browsing channel between two AVRCP entities has been established.

AVRCP_EVT_BROWSING_CHANNEL_DISCONNECTED Macro

File

avrcp.h

C

```
#define AVRCP_EVT_BROWSING_CHANNEL_DISCONNECTED 5    ///< This event is generated when a browsing channel between two AVRCP entities has been terminated.
```

Description

< This event is generated when a browsing channel between two AVRCP entities has been terminated.

AVRCP_EVT_BROWSING_CONNECTION_FAILED Macro

File

avrcp.h

C

```
#define AVRCP_EVT_BROWSING_CONNECTION_FAILED 6    ///< This event is generated when a local device failed to create a browsing channel between two AVRCP entities.
```

Description

< This event is generated when a local device failed to create a browsing channel between two AVRCP entities.

AVRCP_EVT_COMPANY_ID_LIST_RECEIVED Macro

File

avrcp.h

C

```
#define AVRCP_EVT_COMPANY_ID_LIST_RECEIVED 51    ///< This event is generated when a local device received a response to a "get company id" request.
```

Description

< This event is generated when a local device received a response to a "get company id" request.

AVRCP_EVT_CONTROL_CHANNEL_CONNECTED Macro

File

avrcp.h

C

```
#define AVRCP_EVT_CONTROL_CHANNEL_CONNECTED 1    ///< This event is generated when a control channel
between two AVRCP entities has been established.
```

Description

< This event is generated when a control channel between two AVRCP entities has been established.

AVRCP_EVT_CONTROL_CHANNEL_DISCONNECTED Macro

File

avrcp.h

C

```
#define AVRCP_EVT_CONTROL_CHANNEL_DISCONNECTED 2    ///< This event is generated when a control channel
between two AVRCP entities has been terminated.
```

Description

< This event is generated when a control channel between two AVRCP entities has been terminated.

AVRCP_EVT_CONTROL_CONNECTION_FAILED Macro

File

avrcp.h

C

```
#define AVRCP_EVT_CONTROL_CONNECTION_FAILED 3    ///< This event is generated when a local device failed to
create a control channel between two AVRCP entities.
```

Description

< This event is generated when a local device failed to create a control channel between two AVRCP entities.

AVRCP_EVT_DISPLAYABLE_CHARACTER_SET_RECEIVED Macro

File

avrcp.h

C

```
#define AVRCP_EVT_DISPLAYABLE_CHARACTER_SET_RECEIVED 84    ///< This event is generated when a local device
received a "displayable chracter set command" request.
```

Description

< This event is generated when a local device received a "displayable chracter set command" request.

AVRCP_EVT_ELEMENT_ATTRIBUTES_REQUESTED Macro

File

avrcp.h

C

```
#define AVRCP_EVT_ELEMENT_ATTRIBUTES_REQUESTED 85    ///< This event is generated when a local device
received a "get element attributes" request.
```

Description

< This event is generated when a local device received a "get element attributes" request.

AVRCP_EVT_EVENT_ID_LIST_RECEIVED Macro

File

avrcp.h

C

```
#define AVRCP_EVT_EVENT_ID_LIST_RECEIVED 52    ///< This event is generated when a local device received a
response to a "get supported events" request.
```

Description

< This event is generated when a local device received a response to a "get supported events" request.

AVRCP_EVT_GET_ELEMENT_ATTRIBUTES_RECEIVED Macro

File

avrcp.h

C

```
#define AVRCP_EVT_GET_ELEMENT_ATTRIBUTES_RECEIVED 61    ///< This event is generated when a local device
received a response to a "get media element attributes" request.
```

Description

< This event is generated when a local device received a response to a "get media element attributes" request.

AVRCP_EVT_GET_PLAY_STATUS_RECEIVED Macro

File

avrcp.h

C

```
#define AVRCP_EVT_GET_PLAY_STATUS_RECEIVED 62    ///< This event is generated when a local device received
a response to a "get play status" request.
```

Description

< This event is generated when a local device received a response to a "get play status" request.

AVRCP_EVT_INFORM_BATTERY_STATUS_OF_CT_COMPLETED Macro

File

avrcp.h

C

```
#define AVRCP_EVT_INFORM_BATTERY_STATUS_OF_CT_COMPLETED 60    ///< This event is generated when a local
device received a response to a "inform battery status" request.
```

Description

< This event is generated when a local device received a response to a "inform battery status" request.

AVRCP_EVT_INFORM_DISPLAYABLE_CHARACTER_SET_COMPLETED Macro

File

avrcp.h

C

```
#define AVRCP_EVT_INFORM_DISPLAYABLE_CHARACTER_SET_COMPLETED 59    ///< This event is generated when a
local device received a response to a "inform displayable character set" request.
```

Description

< This event is generated when a local device received a response to a "inform displayable character set" request.

AVRCP_EVT_NOTHING Macro

File

avrcp.h

C

```
#define AVRCP_EVT_NOTHING 0
```

Description

addtogroup avrcp @{

@name Events

details The following is a list of events AVRCP layer generates and can report to the upper layer when it completes executing an operation initiated by either local or remote device.

AVRCP_EVT_NOW_PLAYING_CONTENT_CHANGED Macro

File

avrcp.h

C

```
#define AVRCP_EVT_NOW_PLAYING_CONTENT_CHANGED 71    ///< This event is generated when a local device
received a "now playing content changed" notification.
```

Description

< This event is generated when a local device received a "now playing content changed" notification.

AVRCP_EVT_PANEL_COMMAND_RECEIVED Macro

File

avrcp.h

C

```
#define AVRCP_EVT_PANEL_COMMAND_RECEIVED 81    ///< This event is generated when a local device received a
PASS THROUGH command.
```

Description

< This event is generated when a local device received a PASS THROUGH command.

AVRCP_EVT_PANEL_RESPONSE_RECEIVED Macro

File

avrcp.h

C

```
#define AVRCP_EVT_PANEL_RESPONSE_RECEIVED 50    ///< This event is generated when a local device received a
response to a PASS THROUGH command.
```

Description

< This event is generated when a local device received a response to a PASS THROUGH command.

AVRCP_EVT_PLAY_ITEM_COMPLETED Macro

File

avrcp.h

C

```
#define AVRCP_EVT_PLAY_ITEM_COMPLETED 78    ///< This event is generated when a local device received a
response to a "play item" request.
```

Description

< This event is generated when a local device received a response to a "play item" request.

AVRCP_EVT_PLAYBACK_POS_CHANGED Macro

File

avrcp.h

C

```
#define AVRCP_EVT_PLAYBACK_POS_CHANGED 67    ///< This event is generated when a local device received a
"playback position changed" notification.
```

Description

< This event is generated when a local device received a "playback position changed" notification.

AVRCP_EVT_PLAYBACK_STATUS_CHANGED Macro

File

avrcp.h

C

```
#define AVRCP_EVT_PLAYBACK_STATUS_CHANGED 63    ///< This event is generated when a local device received a
"play status changed" notification.
```

Description

< This event is generated when a local device received a "play status changed" notification.

AVRCP_EVT_PLAYER_APPLICATION_SETTING_CHANGED Macro

File

avrcp.h

C

```
#define AVRCP_EVT_PLAYER_APPLICATION_SETTING_CHANGED 70    ///< This event is generated when a local device
received a "player application setting changed" notification.
```

Description

< This event is generated when a local device received a "player application setting changed" notification.

AVRCP_EVT_PLAYER_CURRENT_SETTING_VALUES_RECEIVED Macro

File

avrcp.h

C

```
#define AVRCP_EVT_PLAYER_CURRENT_SETTING_VALUES_RECEIVED 55    ///< This event is generated when a local
device received a response to a "get current player setting attribute values" request.
```

Description

< This event is generated when a local device received a response to a "get current player setting attribute values" request.

AVRCP_EVT_PLAYER_SETTING_ATTRIBUTES_RECEIVED Macro

File

avrcp.h

C

```
#define AVRCP_EVT_PLAYER_SETTING_ATTRIBUTES_RECEIVED 53    ///< This event is generated when a local device
received a response to a "get supported player setting attributes" request.
```

Description

< This event is generated when a local device received a response to a "get supported player setting attributes" request.

AVRCP_EVT_PLAYER_SETTING_ATTRIBUTES_TEXT_RECEIVED Macro

File

avrcp.h

C

```
#define AVRCP_EVT_PLAYER_SETTING_ATTRIBUTES_TEXT_RECEIVED 57    ///< This event is generated when a local
device received a response to a "get player setting attributes displayable text" request.
```

Description

< This event is generated when a local device received a response to a "get player setting attributes displayable text" request.

AVRCP_EVT_PLAYER_SETTING_VALUES_RECEIVED Macro

File

avrcp.h

C

```
#define AVRCP_EVT_PLAYER_SETTING_VALUES_RECEIVED 54    ///< This event is generated when a local device
received a response to a "get player setting attribute values" request.
```

Description

< This event is generated when a local device received a response to a "get player setting attribute values" request.

AVRCP_EVT_PLAYER_SETTING_VALUES_TEXT_RECEIVED Macro

File

avrcp.h

C

```
#define AVRCP_EVT_PLAYER_SETTING_VALUES_TEXT_RECEIVED 58    ///< This event is generated when a local
device received a response to a "get player setting attribute values displayable text" request.
```

Description

< This event is generated when a local device received a response to a "get player setting attribute values displayable text" request.

AVRCP_EVT_REGISTER_NOTIFICATION_REQUESTED Macro

File

avrcp.h

C

```
#define AVRCP_EVT_REGISTER_NOTIFICATION_REQUESTED 86    ///< This event is generated when a local device
received a "register notification" request.
```

Description

< This event is generated when a local device received a "register notification" request.

AVRCP_EVT_REGISTER_NOTIFICATIONS_COMPLETED Macro

File

avrcp.h

C

```
#define AVRCP_EVT_REGISTER_NOTIFICATIONS_COMPLETED 80    ///< This event is generated when a local device
received a response to a "register notification" request.
```

Description

< This event is generated when a local device received a response to a "register notification" request.

AVRCP_EVT_SEARCH_COMPLETED Macro

File

avrcp.h

C

```
#define AVRCP_EVT_SEARCH_COMPLETED 7    ///< This event is generated when a local device completed
searching for nearby targets.
```

Description

< This event is generated when a local device completed searching for nearby targets.

AVRCP_EVT_SET_ABSOLUTE_VOLUME_COMPLETED Macro

File

avrcp.h

C

```
#define AVRCP_EVT_SET_ABSOLUTE_VOLUME_COMPLETED 76    ///< This event is generated when a local device
received a response to a "set absolute volume" request.
```

Description

< This event is generated when a local device received a response to a "set absolute volume" request.

AVRCP_EVT_SET_ABSOLUTE_VOLUME_REQUESTED Macro

File

avrcp.h

C

```
#define AVRCP_EVT_SET_ABSOLUTE_VOLUME_REQUESTED 82    ///< This event is generated when a local device
received a "set absolute volume" request.
```

Description

< This event is generated when a local device received a "set absolute volume" request.

AVRCP_EVT_SET_ADDRESSED_PLAYER_COMPLETED Macro

File

avrcp.h

C

```
#define AVRCP_EVT_SET_ADDRESSED_PLAYER_COMPLETED 77    ///< This event is generated when a local device
received a response to a "set addressed player" request.
```

Description

< This event is generated when a local device received a response to a "set addressed player" request.

AVRCP_EVT_SET_PLAYER_SETTING_VALUES_COMPLETED Macro

File

avrcp.h

C

```
#define AVRCP_EVT_SET_PLAYER_SETTING_VALUES_COMPLETED 56    ///< This event is generated when a local
device received a response to a "set player setting attribute values" request.
```

Description

< This event is generated when a local device received a response to a "set player setting attribute values" request.

AVRCP_EVT_SYSTEM_STATUS_CHANGED Macro

File

avrcp.h

C

```
#define AVRCP_EVT_SYSTEM_STATUS_CHANGED 69    ///< This event is generated when a local device received a
"system status changed" notification.
```

Description

< This event is generated when a local device received a "system status changed" notification.

AVRCP_EVT_TRACK_CHANGED Macro

File

avrcp.h

C

```
#define AVRCP_EVT_TRACK_CHANGED 64    ///< This event is generated when a local device received a "track
changed changed" notification.
```

Description

< This event is generated when a local device received a "track changed changed" notification.

AVRCP_EVT_TRACK_REACHED_END Macro

File

avrcp.h

C

```
#define AVRCP_EVT_TRACK_REACHED_END 65    ///< This event is generated when a local device received a
"track reached end" notification.
```

Description

< This event is generated when a local device received a "track reached end" notification.

AVRCP_EVT_TRACK_REACHED_START Macro

File

avrcp.h

C

```
#define AVRCP_EVT_TRACK_REACHED_START 66    ///< This event is generated when a local device received a
"track reached start" notification.
```

Description

< This event is generated when a local device received a "track reached start" notification.

AVRCP_EVT_UIDS_CHANGED Macro

File

avrcp.h

C

```
#define AVRCP_EVT_UIDS_CHANGED 74    ///< This event is generated when a local device received a "UIDs
changed" notification.
```

Description

< This event is generated when a local device received a "UIDs changed" notification.

AVRCP_EVT_VOLUME_CHANGED Macro

File

avrcp.h

C

```
#define AVRCP_EVT_VOLUME_CHANGED 75    ///< This event is generated when a local device received a "volume
changed" notification.
```

Description

< This event is generated when a local device received a "volume changed" notification.

AVRCP_MANAGER_FLAG_SEARCHING Macro

File

avrcp.h

C

```
#define AVRCP_MANAGER_FLAG_SEARCHING 1
```

Description

This is macro AVRCP_MANAGER_FLAG_SEARCHING.

AVRCP_MANAGER_STATE_IDLE Macro

File

avrcp.h

C

```
#define AVRCP_MANAGER_STATE_IDLE 0
```

Description

This is macro AVRCP_MANAGER_STATE_IDLE.

AVRCP_MAX_ELEMENT_ATTRIBUTES Macro

File

avrcp_private.h

C

```
#define AVRCP_MAX_ELEMENT_ATTRIBUTES 7
```

Description

This is macro AVRCP_MAX_ELEMENT_ATTRIBUTES.

bt_avrcp_0_click Macro

File

avrcp_command.h

C

```
#define bt_avrcp_0_click(channel) bt_avrcp_send_button_click(channel, AVC_PANEL_OPID_0)
```

Description

This is macro bt_avrcp_0_click.

bt_avrcp_0_press Macro

File

avrcp_command.h

C

```
#define bt_avrcp_0_press(channel) bt_avrcp_send_press_panel_control(channel, AVC_PANEL_OPID_0)
```

Description

This is macro bt_avrcp_0_press.

bt_avrcp_1_click Macro

File

avrcp_command.h

C

```
#define bt_avrcp_1_click(channel) bt_avrcp_send_button_click(channel, AVC_PANEL_OPID_1)
```

Description

This is macro bt_avrcp_1_click.

bt_avrcp_1_press Macro

File

avrcp_command.h

C

```
#define bt_avrcp_1_press(channel) bt_avrcp_send_press_panel_control(channel, AVC_PANEL_OPID_1)
```

Description

This is macro bt_avrcp_1_press.

bt_avrcp_2_click Macro

File

avrcp_command.h

C

```
#define bt_avrcp_2_click(channel) bt_avrcp_send_button_click(channel, AVC_PANEL_OPID_2)
```

Description

This is macro bt_avrcp_2_click.

bt_avrcp_2_press Macro

File

avrcp_command.h

C

```
#define bt_avrcp_2_press(channel) bt_avrcp_send_press_panel_control(channel, AVC_PANEL_OPID_2)
```

Description

This is macro bt_avrcp_2_press.

bt_avrcp_3_click Macro

File

avrcp_command.h

C

```
#define bt_avrcp_3_click(channel) bt_avrcp_send_button_click(channel, AVC_PANEL_OPID_3)
```

Description

This is macro bt_avrcp_3_click.

bt_avrcp_3_press Macro

File

avrcp_command.h

C

```
#define bt_avrcp_3_press(channel) bt_avrcp_send_press_panel_control(channel, AVC_PANEL_OPID_3)
```

Description

This is macro bt_avrcp_3_press.

bt_avrcp_4_click Macro

File

avrcp_command.h

C

```
#define bt_avrcp_4_click(channel) bt_avrcp_send_button_click(channel, AVC_PANEL_OPID_4)
```

Description

This is macro bt_avrcp_4_click.

bt_avrcp_4_press Macro

File

avrcp_command.h

C

```
#define bt_avrcp_4_press(channel) bt_avrcp_send_press_panel_control(channel, AVC_PANEL_OPID_4)
```

Description

This is macro bt_avrcp_4_press.

bt_avrcp_5_click Macro

File

avrcp_command.h

C

```
#define bt_avrcp_5_click(channel) bt_avrcp_send_button_click(channel, AVC_PANEL_OPID_5)
```

Description

This is macro bt_avrcp_5_click.

bt_avrcp_5_press Macro

File

avrcp_command.h

C

```
#define bt_avrcp_5_press(channel) bt_avrcp_send_press_panel_control(channel, AVC_PANEL_OPID_5)
```

Description

This is macro bt_avrcp_5_press.

bt_avrcp_6_click Macro

File

avrcp_command.h

C

```
#define bt_avrcp_6_click(channel) bt_avrcp_send_button_click(channel, AVC_PANEL_OPID_6)
```

Description

This is macro bt_avrcp_6_click.

bt_avrcp_6_press Macro

File

avrcp_command.h

C

```
#define bt_avrcp_6_press(channel) bt_avrcp_send_press_panel_control(channel, AVC_PANEL_OPID_6)
```

Description

This is macro bt_avrcp_6_press.

bt_avrcp_7_click Macro

File

avrcp_command.h

C

```
#define bt_avrcp_7_click(channel) bt_avrcp_send_button_click(channel, AVC_PANEL_OPID_7)
```

Description

This is macro bt_avrcp_7_click.

bt_avrcp_7_press Macro

File

avrcp_command.h

C

```
#define bt_avrcp_7_press(channel) bt_avrcp_send_press_panel_control(channel, AVC_PANEL_OPID_7)
```

Description

This is macro bt_avrcp_7_press.

bt_avrcp_8_click Macro

File

avrcp_command.h

C

```
#define bt_avrcp_8_click(channel) bt_avrcp_send_button_click(channel, AVC_PANEL_OPID_8)
```

Description

This is macro bt_avrcp_8_click.

bt_avrcp_8_press Macro

File

avrcp_command.h

C

```
#define bt_avrcp_8_press(channel) bt_avrcp_send_press_panel_control(channel, AVC_PANEL_OPID_8)
```

Description

This is macro bt_avrcp_8_press.

bt_avrcp_9_click Macro

File

avrcp_command.h

C

```
#define bt_avrcp_9_click(channel) bt_avrcp_send_button_click(channel, AVC_PANEL_OPID_9)
```

Description

This is macro bt_avrcp_9_click.

bt_avrcp_9_press Macro

File

avrcp_command.h

C

```
#define bt_avrcp_9_press(channel) bt_avrcp_send_press_panel_control(channel, AVC_PANEL_OPID_9)
```

Description

This is macro bt_avrcp_9_press.

bt_avrcp_angle_click Macro

File

avrcp_command.h

C

```
#define bt_avrcp_angle_click(channel) bt_avrcp_send_button_click(channel, AVC_PANEL_OPID_ANGLE)
```

Description

This is macro bt_avrcp_angle_click.

bt_avrcp_angle_press Macro

File

avrcp_command.h

C

```
#define bt_avrcp_angle_press(channel) bt_avrcp_send_press_panel_control(channel, AVC_PANEL_OPID_ANGLE)
```

Description

This is macro bt_avrcp_angle_press.

bt_avrcp_backward_click Macro

File

avrcp_command.h

C

```
#define bt_avrcp_backward_click(channel) bt_avrcp_send_button_click(channel, AVC_PANEL_OPID_BACKWARD)
```

Description

This is macro bt_avrcp_backward_click.

bt_avrcp_backward_press Macro

File

avrcp_command.h

C

```
#define bt_avrcp_backward_press(channel) bt_avrcp_send_press_panel_control(channel, AVC_PANEL_OPID_BACKWARD)
```

Description

This is macro bt_avrcp_backward_press.

bt_avrcp_channel_down_click Macro

File

avrcp_command.h

C

```
#define bt_avrcp_channel_down_click(channel) bt_avrcp_send_button_click(channel,
AVC_PANEL_OPID_CHANNEL_DOWN)
```

Description

This is macro bt_avrcp_channel_down_click.

bt_avrcp_channel_down_press Macro

File

avrcp_command.h

C

```
#define bt_avrcp_channel_down_press(channel) bt_avrcp_send_press_panel_control(channel,
AVC_PANEL_OPID_CHANNEL_DOWN)
```

Description

This is macro bt_avrcp_channel_down_press.

bt_avrcp_channel_up_click Macro

File

avrcp_command.h

C

```
#define bt_avrcp_channel_up_click(channel) bt_avrcp_send_button_click(channel, AVC_PANEL_OPID_CHANNEL_UP)
```

Description

This is macro bt_avrcp_channel_up_click.

bt_avrcp_channel_up_press Macro

File

avrcp_command.h

C

```
#define bt_avrcp_channel_up_press(channel) bt_avrcp_send_press_panel_control(channel,
AVC_PANEL_OPID_CHANNEL_UP)
```

Description

This is macro bt_avrcp_channel_up_press.

bt_avrcp_clear_click Macro

File

avrcp_command.h

C

```
#define bt_avrcp_clear_click(channel) bt_avrcp_send_button_click(channel, AVC_PANEL_OPID_CLEAR)
```

Description

This is macro bt_avrcp_clear_click.

bt_avrcp_clear_press Macro

File

avrcp_command.h

C

```
#define bt_avrcp_clear_press(channel) bt_avrcp_send_press_panel_control(channel, AVC_PANEL_OPID_CLEAR)
```

Description

This is macro bt_avrcp_clear_press.

bt_avrcp_content_menu_click Macro

File

avrcp_command.h

C

```
#define bt_avrcp_content_menu_click(channel) bt_avrcp_send_button_click(channel,
AVC_PANEL_OPID_CONTENTS_MENU)
```

Description

This is macro bt_avrcp_content_menu_click.

bt_avrcp_contents_menu_press Macro

File

avrcp_command.h

C

```
#define bt_avrcp_contents_menu_press(channel) bt_avrcp_send_press_panel_control(channel,
AVC_PANEL_OPID_CONTENTS_MENU)
```

Description

This is macro bt_avrcp_contents_menu_press.

bt_avrcp_display_info_click Macro

File

avrcp_command.h

C

```
#define bt_avrcp_display_info_click(channel) bt_avrcp_send_button_click(channel,
AVC_PANEL_OPID_DISPLAY_INFORMATION)
```

Description

This is macro bt_avrcp_display_info_click.

bt_avrcp_display_info_press Macro

File

avrcp_command.h

C

```
#define bt_avrcp_display_info_press(channel) bt_avrcp_send_press_panel_control(channel,
AVC_PANEL_OPID_DISPLAY_INFORMATION)
```

Description

This is macro bt_avrcp_display_info_press.

bt_avrcp_dot_click Macro

File

avrcp_command.h

C

```
#define bt_avrcp_dot_click(channel) bt_avrcp_send_button_click(channel, AVC_PANEL_OPID_DOT)
```

Description

This is macro bt_avrcp_dot_click.

bt_avrcp_dot_press Macro

File

avrcp_command.h

C

```
#define bt_avrcp_dot_press(channel) bt_avrcp_send_press_panel_control(channel, AVC_PANEL_OPID_DOT)
```

Description

This is macro bt_avrcp_dot_press.

bt_avrcp_down_click Macro

File

avrcp_command.h

C

```
#define bt_avrcp_down_click(channel) bt_avrcp_send_button_click(channel, AVC_PANEL_OPID_DOWN)
```

Description

This is macro bt_avrcp_down_click.

bt_avrcp_down_press Macro

File

avrcp_command.h

C

```
#define bt_avrcp_down_press(channel) bt_avrcp_send_press_panel_control(channel, AVC_PANEL_OPID_DOWN)
```

Description

This is macro bt_avrcp_down_press.

bt_avrcp_eject_click Macro

File

avrcp_command.h

C

```
#define bt_avrcp_eject_click(channel) bt_avrcp_send_button_click(channel, AVC_PANEL_OPID_EJECT)
```

Description

This is macro bt_avrcp_eject_click.

bt_avrcp_eject_press Macro

File

avrcp_command.h

C

```
#define bt_avrcp_eject_press(channel) bt_avrcp_send_press_panel_control(channel, AVC_PANEL_OPID_EJECT)
```

Description

This is macro bt_avrcp_eject_press.

bt_avrcp_enter_click Macro

File

avrcp_command.h

C

```
#define bt_avrcp_enter_click(channel) bt_avrcp_send_button_click(channel, AVC_PANEL_OPID_ENTER)
```

Description

This is macro bt_avrcp_enter_click.

bt_avrcp_enter_press Macro

File

avrcp_command.h

C

```
#define bt_avrcp_enter_press(channel) bt_avrcp_send_press_panel_control(channel, AVC_PANEL_OPID_ENTER)
```

Description

This is macro bt_avrcp_enter_press.

bt_avrcp_exit_click Macro

File

avrcp_command.h

C

```
#define bt_avrcp_exit_click(channel) bt_avrcp_send_button_click(channel, AVC_PANEL_OPID_EXIT)
```

Description

This is macro bt_avrcp_exit_click.

bt_avrcp_exit_press Macro

File

avrcp_command.h

C

```
#define bt_avrcp_exit_press(channel) bt_avrcp_send_press_panel_control(channel, AVC_PANEL_OPID_EXIT)
```

Description

This is macro bt_avrcp_exit_press.

bt_avrcp_f1_click Macro

File

avrcp_command.h

C

```
#define bt_avrcp_f1_click(channel) bt_avrcp_send_button_click(channel, AVC_PANEL_OPID_F1)
```

Description

This is macro bt_avrcp_f1_click.

bt_avrcp_f1_press Macro

File

avrcp_command.h

C

```
#define bt_avrcp_f1_press(channel) bt_avrcp_send_press_panel_control(channel, AVC_PANEL_OPID_F1)
```

Description

This is macro bt_avrcp_f1_press.

bt_avrcp_f2_click Macro

File

avrcp_command.h

C

```
#define bt_avrcp_f2_click(channel) bt_avrcp_send_button_click(channel, AVC_PANEL_OPID_F2)
```

Description

This is macro bt_avrcp_f2_click.

bt_avrcp_f2_press Macro

File

avrcp_command.h

C

```
#define bt_avrcp_f2_press(channel) bt_avrcp_send_press_panel_control(channel, AVC_PANEL_OPID_F2)
```

Description

This is macro bt_avrcp_f2_press.

bt_avrcp_f3_click Macro

File

avrcp_command.h

C

```
#define bt_avrcp_f3_click(channel) bt_avrcp_send_button_click(channel, AVC_PANEL_OPID_F3)
```

Description

This is macro bt_avrcp_f3_click.

bt_avrcp_f3_press Macro

File

avrcp_command.h

C

```
#define bt_avrcp_f3_press(channel) bt_avrcp_send_press_panel_control(channel, AVC_PANEL_OPID_F3)
```

Description

This is macro bt_avrcp_f3_press.

bt_avrcp_f4_click Macro

File

avrcp_command.h

C

```
#define bt_avrcp_f4_click(channel) bt_avrcp_send_button_click(channel, AVC_PANEL_OPID_F4)
```

Description

This is macro bt_avrcp_f4_click.

bt_avrcp_f4_press Macro

File

avrcp_command.h

C

```
#define bt_avrcp_f4_press(channel) bt_avrcp_send_press_panel_control(channel, AVC_PANEL_OPID_F4)
```

Description

This is macro bt_avrcp_f4_press.

bt_avrcp_f5_click Macro

File

avrcp_command.h

C

```
#define bt_avrcp_f5_click(channel) bt_avrcp_send_button_click(channel, AVC_PANEL_OPID_F5)
```

Description

This is macro bt_avrcp_f5_click.

bt_avrcp_f5_press Macro

File

avrcp_command.h

C

```
#define bt_avrcp_f5_press(channel) bt_avrcp_send_press_panel_control(channel, AVC_PANEL_OPID_F5)
```

Description

This is macro bt_avrcp_f5_press.

bt_avrcp_f6_click Macro

File

avrcp_command.h

C

```
#define bt_avrcp_f6_click(channel) bt_avrcp_send_button_click(channel, AVC_PANEL_OPID_F6)
```

Description

This is macro bt_avrcp_f6_click.

bt_avrcp_f6_press Macro

File

avrcp_command.h

C

```
#define bt_avrcp_f6_press(channel) bt_avrcp_send_press_panel_control(channel, AVC_PANEL_OPID_F6)
```

Description

This is macro bt_avrcp_f6_press.

bt_avrcp_f7_click Macro

File

avrcp_command.h

C

```
#define bt_avrcp_f7_click(channel) bt_avrcp_send_button_click(channel, AVC_PANEL_OPID_F7)
```

Description

This is macro bt_avrcp_f7_click.

bt_avrcp_f7_press Macro

File

avrcp_command.h

C

```
#define bt_avrcp_f7_press(channel) bt_avrcp_send_press_panel_control(channel, AVC_PANEL_OPID_F7)
```

Description

This is macro bt_avrcp_f7_press.

bt_avrcp_f8_click Macro

File

avrcp_command.h

C

```
#define bt_avrcp_f8_click(channel) bt_avrcp_send_button_click(channel, AVC_PANEL_OPID_F8)
```

Description

This is macro bt_avrcp_f8_click.

bt_avrcp_f8_press Macro

File

avrcp_command.h

C

```
#define bt_avrcp_f8_press(channel) bt_avrcp_send_press_panel_control(channel, AVC_PANEL_OPID_F8)
```

Description

This is macro bt_avrcp_f8_press.

bt_avrcp_f9_click Macro

File

avrcp_command.h

C

```
#define bt_avrcp_f9_click(channel) bt_avrcp_send_button_click(channel, AVC_PANEL_OPID_F9)
```

Description

This is macro bt_avrcp_f9_click.

bt_avrcp_f9_press Macro

File

avrcp_command.h

C

```
#define bt_avrcp_f9_press(channel) bt_avrcp_send_press_panel_control(channel, AVC_PANEL_OPID_F9)
```

Description

This is macro bt_avrcp_f9_press.

bt_avrcp_fast_forward_click Macro

File

avrcp_command.h

C

```
#define bt_avrcp_fast_forward_click(channel) bt_avrcp_send_button_click(channel,
AVC_PANEL_OPID_FAST_FORWARD)
```

Description

This is macro bt_avrcp_fast_forward_click.

bt_avrcp_fast_forward_press Macro

File

avrcp_command.h

C

```
#define bt_avrcp_fast_forward_press(channel) bt_avrcp_send_press_panel_control(channel,
AVC_PANEL_OPID_FAST_FORWARD)
```

Description

This is macro bt_avrcp_fast_forward_press.

bt_avrcp_favorite_menu_click Macro

File

avrcp_command.h

C

```
#define bt_avrcp_favorite_menu_click(channel) bt_avrcp_send_button_click(channel,
AVC_PANEL_OPID_FAVORITE_MENU)
```

Description

This is macro bt_avrcp_favorite_menu_click.

bt_avrcp_favorive_menu_press Macro

File

avrcp_command.h

C

```
#define bt_avrcp_favorive_menu_press(channel) bt_avrcp_send_press_panel_control(channel,
AVC_PANEL_OPID_FAVORIVE_MENU)
```

Description

This is macro bt_avrcp_favorive_menu_press.

bt_avrcp_forward_click Macro

File

avrcp_command.h

C

```
#define bt_avrcp_forward_click(channel) bt_avrcp_send_button_click(channel, AVC_PANEL_OPID_FORWARD)
```

Description

This is macro bt_avrcp_forward_click.

bt_avrcp_forward_press Macro

File

avrcp_command.h

C

```
#define bt_avrcp_forward_press(channel) bt_avrcp_send_press_panel_control(channel, AVC_PANEL_OPID_FORWARD)
```

Description

This is macro bt_avrcp_forward_press.

bt_avrcp_help_click Macro

File

avrcp_command.h

C

```
#define bt_avrcp_help_click(channel) bt_avrcp_send_button_click(channel, AVC_PANEL_OPID_HELP)
```

Description

This is macro bt_avrcp_help_click.

bt_avrcp_help_press Macro

File

avrcp_command.h

C

```
#define bt_avrcp_help_press(channel) bt_avrcp_send_press_panel_control(channel, AVC_PANEL_OPID_HELP)
```

Description

This is macro bt_avrcp_help_press.

bt_avrcp_input_select_click Macro

File

avrcp_command.h

C

```
#define bt_avrcp_input_select_click(channel) bt_avrcp_send_button_click(channel,
AVC_PANEL_OPID_INPUT_SELECT)
```

Description

This is macro bt_avrcp_input_select_click.

bt_avrcp_input_select_press Macro

File

avrcp_command.h

C

```
#define bt_avrcp_input_select_press(channel) bt_avrcp_send_press_panel_control(channel,
AVC_PANEL_OPID_INPUT_SELECT)
```

Description

This is macro bt_avrcp_input_select_press.

bt_avrcp_left_click Macro

File

avrcp_command.h

C

```
#define bt_avrcp_left_click(channel) bt_avrcp_send_button_click(channel, AVC_PANEL_OPID_LEFT)
```

Description

This is macro bt_avrcp_left_click.

bt_avrcp_left_down_click Macro

File

avrcp_command.h

C

```
#define bt_avrcp_left_down_click(channel) bt_avrcp_send_button_click(channel, AVC_PANEL_OPID_LEFT_DOWN)
```

Description

This is macro bt_avrcp_left_down_click.

bt_avrcp_left_down_press Macro

File

avrcp_command.h

C

```
#define bt_avrcp_left_down_press(channel) bt_avrcp_send_press_panel_control(channel,
AVC_PANEL_OPID_LEFT_DOWN)
```

Description

This is macro bt_avrcp_left_down_press.

bt_avrcp_left_press Macro

File

avrcp_command.h

C

```
#define bt_avrcp_left_press(channel) bt_avrcp_send_press_panel_control(channel, AVC_PANEL_OPID_LEFT)
```

Description

This is macro bt_avrcp_left_press.

bt_avrcp_left_up_click Macro

File

avrcp_command.h

C

```
#define bt_avrcp_left_up_click(channel) bt_avrcp_send_button_click(channel, AVC_PANEL_OPID_LEFT_UP)
```

Description

This is macro bt_avrcp_left_up_click.

bt_avrcp_left_up_press Macro

File

avrcp_command.h

C

```
#define bt_avrcp_left_up_press(channel) bt_avrcp_send_press_panel_control(channel, AVC_PANEL_OPID_LEFT_UP)
```

Description

This is macro bt_avrcp_left_up_press.

bt_avrcp_mute_click Macro

File

avrcp_command.h

C

```
#define bt_avrcp_mute_click(channel) bt_avrcp_send_button_click(channel, AVC_PANEL_OPID_MUTE)
```

Description

This is macro bt_avrcp_mute_click.

bt_avrcp_mute_press Macro

File

avrcp_command.h

C

```
#define bt_avrcp_mute_press(channel) bt_avrcp_send_press_panel_control(channel, AVC_PANEL_OPID_MUTE)
```

Description

This is macro bt_avrcp_mute_press.

bt_avrcp_page_down_click Macro

File

avrcp_command.h

C

```
#define bt_avrcp_page_down_click(channel) bt_avrcp_send_button_click(channel, AVC_PANEL_OPID_PAGE_DOWN)
```

Description

This is macro bt_avrcp_page_down_click.

bt_avrcp_page_down_press Macro

File

avrcp_command.h

C

```
#define bt_avrcp_page_down_press(channel) bt_avrcp_send_press_panel_control(channel,
AVC_PANEL_OPID_PAGE_DOWN)
```

Description

This is macro bt_avrcp_page_down_press.

bt_avrcp_page_up_click Macro

File

avrcp_command.h

C

```
#define bt_avrcp_page_up_click(channel) bt_avrcp_send_button_click(channel, AVC_PANEL_OPID_PAGE_UP)
```

Description

This is macro bt_avrcp_page_up_click.

bt_avrcp_page_up_press Macro

File

avrcp_command.h

C

```
#define bt_avrcp_page_up_press(channel) bt_avrcp_send_press_panel_control(channel, AVC_PANEL_OPID_PAGE_UP)
```

Description

This is macro bt_avrcp_page_up_press.

bt_avrcp_pause_click Macro

File

avrcp_command.h

C

```
#define bt_avrcp_pause_click(channel) bt_avrcp_send_button_click(channel, AVC_PANEL_OPID_PAUSE)
```

Description

This is macro bt_avrcp_pause_click.

bt_avrcp_pause_press Macro

File

avrcp_command.h

C

```
#define bt_avrcp_pause_press(channel) bt_avrcp_send_press_panel_control(channel, AVC_PANEL_OPID_PAUSE)
```

Description

This is macro bt_avrcp_pause_press.

bt_avrcp_play_click Macro

File

avrcp_command.h

C

```
#define bt_avrcp_play_click(channel) bt_avrcp_send_button_click(channel, AVC_PANEL_OPID_PLAY)
```

Description

This is macro bt_avrcp_play_click.

bt_avrcp_play_press Macro

File

avrcp_command.h

C

```
#define bt_avrcp_play_press(channel) bt_avrcp_send_press_panel_control(channel, AVC_PANEL_OPID_PLAY)
```

Description

This is macro bt_avrcp_play_press.

bt_avrcp_power_click Macro

File

avrcp_command.h

C

```
#define bt_avrcp_power_click(channel) bt_avrcp_send_button_click(channel, AVC_PANEL_OPID_POWER_TOGGLE)
```

Description

This is macro bt_avrcp_power_click.

bt_avrcp_power_press Macro

File

avrcp_command.h

C

```
#define bt_avrcp_power_press(channel) bt_avrcp_send_press_panel_control(channel,
AVC_PANEL_OPID_POWER_TOGGLE)
```

Description

This is macro bt_avrcp_power_press.

bt_avrcp_previous_channel_click Macro

File

avrcp_command.h

C

```
#define bt_avrcp_previous_channel_click(channel) bt_avrcp_send_button_click(channel,
AVC_PANEL_OPID_PREVIOUS_CHANNEL)
```

Description

This is macro bt_avrcp_previous_channel_click.

bt_avrcp_previous_channel_press Macro

File

avrcp_command.h

C

```
#define bt_avrcp_previous_channel_press(channel) bt_avrcp_send_press_panel_control(channel,
AVC_PANEL_OPID_PREVIOUS_CHANNEL)
```

Description

This is macro bt_avrcp_previous_channel_press.

bt_avrcp_record_click Macro

File

avrcp_command.h

C

```
#define bt_avrcp_record_click(channel) bt_avrcp_send_button_click(channel, AVC_PANEL_OPID_RECORD)
```

Description

This is macro bt_avrcp_record_click.

bt_avrcp_record_press Macro

File

avrcp_command.h

C

```
#define bt_avrcp_record_press(channel) bt_avrcp_send_press_panel_control(channel, AVC_PANEL_OPID_RECORD)
```

Description

This is macro bt_avrcp_record_press.

bt_avrcp_rewind_click Macro

File

avrcp_command.h

C

```
#define bt_avrcp_rewind_click(channel) bt_avrcp_send_button_click(channel, AVC_PANEL_OPID_REWIND)
```

Description

This is macro bt_avrcp_rewind_click.

bt_avrcp_rewind_press Macro

File

avrcp_command.h

C

```
#define bt_avrcp_rewind_press(channel) bt_avrcp_send_press_panel_control(channel, AVC_PANEL_OPID_REWIND)
```

Description

This is macro bt_avrcp_rewind_press.

bt_avrcp_right_click Macro

File

avrcp_command.h

C

```
#define bt_avrcp_right_click(channel) bt_avrcp_send_button_click(channel, AVC_PANEL_OPID_RIGHT)
```

Description

This is macro bt_avrcp_right_click.

bt_avrcp_right_down_click Macro

File

avrcp_command.h

C

```
#define bt_avrcp_right_down_click(channel) bt_avrcp_send_button_click(channel, AVC_PANEL_OPID_RIGHT_DOWN)
```

Description

This is macro bt_avrcp_right_down_click.

bt_avrcp_right_down_press Macro

File

avrcp_command.h

C

```
#define bt_avrcp_right_down_press(channel) bt_avrcp_send_press_panel_control(channel,
AVC_PANEL_OPID_RIGHT_DOWN)
```

Description

This is macro bt_avrcp_right_down_press.

bt_avrcp_right_press Macro

File

avrcp_command.h

C

```
#define bt_avrcp_right_press(channel) bt_avrcp_send_press_panel_control(channel, AVC_PANEL_OPID_RIGHT)
```

Description

This is macro bt_avrcp_right_press.

bt_avrcp_right_up_click Macro

File

avrcp_command.h

C

```
#define bt_avrcp_right_up_click(channel) bt_avrcp_send_button_click(channel, AVC_PANEL_OPID_RIGHT_UP)
```

Description

This is macro bt_avrcp_right_up_click.

bt_avrcp_right_up_press Macro

File

avrcp_command.h

C

```
#define bt_avrcp_right_up_press(channel) bt_avrcp_send_press_panel_control(channel, AVC_PANEL_OPID_RIGHT_UP)
```

Description

This is macro bt_avrcp_right_up_press.

bt_avrcp_root_menu_click Macro

File

avrcp_command.h

C

```
#define bt_avrcp_root_menu_click(channel) bt_avrcp_send_button_click(channel, AVC_PANEL_OPID_ROOT_MENU)
```

Description

This is macro bt_avrcp_root_menu_click.

bt_avrcp_root_menu_press Macro

File

avrcp_command.h

C

```
#define bt_avrcp_root_menu_press(channel) bt_avrcp_send_press_panel_control(channel,
AVC_PANEL_OPID_ROOT_MENU)
```

Description

This is macro bt_avrcp_root_menu_press.

bt_avrcp_select_click Macro

File

avrcp_command.h

C

```
#define bt_avrcp_select_click(channel) bt_avrcp_send_button_click(channel, AVC_PANEL_OPID_SELECT)
```

Description

This is macro bt_avrcp_select_click.

bt_avrcp_select_press Macro

File

avrcp_command.h

C

```
#define bt_avrcp_select_press(channel) bt_avrcp_send_press_panel_control(channel, AVC_PANEL_OPID_SELECT)
```

Description

Panel operations

bt_avrcp_setup_menu_click Macro

File

avrcp_command.h

C

```
#define bt_avrcp_setup_menu_click(channel) bt_avrcp_send_button_click(channel, AVC_PANEL_OPID_SETUP_MENU)
```

Description

This is macro bt_avrcp_setup_menu_click.

bt_avrcp_setup_menu_press Macro

File

avrcp_command.h

C

```
#define bt_avrcp_setup_menu_press(channel) bt_avrcp_send_press_panel_control(channel,
AVC_PANEL_OPID_SETUP_MENU)
```

Description

This is macro bt_avrcp_setup_menu_press.

bt_avrcp_sound_select_click Macro

File

avrcp_command.h

C

```
#define bt_avrcp_sound_select_click(channel) bt_avrcp_send_button_click(channel,
AVC_PANEL_OPID_SOUND_SELECT)
```

Description

This is macro bt_avrcp_sound_select_click.

bt_avrcp_sound_select_press Macro

File

avrcp_command.h

C

```
#define bt_avrcp_sound_select_press(channel) bt_avrcp_send_press_panel_control(channel,
AVC_PANEL_OPID_SOUND_SELECT)
```

Description

This is macro bt_avrcp_sound_select_press.

bt_avrcp_stop_click Macro

File

avrcp_command.h

C

```
#define bt_avrcp_stop_click(channel) bt_avrcp_send_button_click(channel, AVC_PANEL_OPID_STOP)
```

Description

This is macro bt_avrcp_stop_click.

bt_avrcp_stop_press Macro

File

avrcp_command.h

C

```
#define bt_avrcp_stop_press(channel) bt_avrcp_send_press_panel_control(channel, AVC_PANEL_OPID_STOP)
```

Description

This is macro bt_avrcp_stop_press.

bt_avrcp_subpicture_click Macro

File

avrcp_command.h

C

```
#define bt_avrcp_subpicture_click(channel) bt_avrcp_send_button_click(channel, AVC_PANEL_OPID_SUBPICTURE)
```

Description

This is macro bt_avrcp_subpicture_click.

bt_avrcp_subpicture_press Macro

File

avrcp_command.h

C

```
#define bt_avrcp_subpicture_press(channel) bt_avrcp_send_press_panel_control(channel,
AVC_PANEL_OPID_SUBPICTURE)
```

Description

This is macro bt_avrcp_subpicture_press.

bt_avrcp_up_click Macro

File

avrcp_command.h

C

```
#define bt_avrcp_up_click(channel) bt_avrcp_send_button_click(channel, AVC_PANEL_OPID_UP)
```

Description

This is macro bt_avrcp_up_click.

bt_avrcp_up_press Macro

File

avrcp_command.h

C

```
#define bt_avrcp_up_press(channel) bt_avrcp_send_press_panel_control(channel, AVC_PANEL_OPID_UP)
```

Description

This is macro bt_avrcp_up_press.

bt_avrcp_volume_down_click Macro

File

avrcp_command.h

C

```
#define bt_avrcp_volume_down_click(channel) bt_avrcp_send_button_click(channel, AVC_PANEL_OPID_VOLUME_DOWN)
```

Description

This is macro bt_avrcp_volume_down_click.

bt_avrcp_volume_down_press Macro

File

avrcp_command.h

C

```
#define bt_avrcp_volume_down_press(channel) bt_avrcp_send_press_panel_control(channel,
AVC_PANEL_OPID_VOLUME_DOWN)
```

Description

This is macro bt_avrcp_volume_down_press.

bt_avrcp_volume_up_click Macro

File

avrcp_command.h

C

```
#define bt_avrcp_volume_up_click(channel) bt_avrcp_send_button_click(channel, AVC_PANEL_OPID_VOLUME_UP)
```

Description

This is macro bt_avrcp_volume_up_click.

bt_avrcp_volume_up_press Macro

File

avrcp_command.h

C

```
#define bt_avrcp_volume_up_press(channel) bt_avrcp_send_press_panel_control(channel,
AVC_PANEL_OPID_VOLUME_UP)
```

Description

This is macro bt_avrcp_volume_up_press.

bt_avrcp_0_release Macro

File

avrcp_command.h

C

```
#define bt_avrcp_0_release(channel) bt_avrcp_send_release_panel_control(channel, AVC_PANEL_OPID_0)
```

Description

This is macro bt_avrcp_0_release.

bt_avrcp_1_release Macro

File

avrcp_command.h

C

```
#define bt_avrcp_1_release(channel) bt_avrcp_send_release_panel_control(channel, AVC_PANEL_OPID_1)
```

Description

This is macro bt_avrcp_1_release.

bt_avrcp_2_release Macro

File

avrcp_command.h

C

```
#define bt_avrcp_2_release(channel) bt_avrcp_send_release_panel_control(channel, AVC_PANEL_OPID_2)
```

Description

This is macro bt_avrcp_2_release.

bt_avrcp_3_release Macro

File

avrcp_command.h

C

```
#define bt_avrcp_3_release(channel) bt_avrcp_send_release_panel_control(channel, AVC_PANEL_OPID_3)
```

Description

This is macro bt_avrcp_3_release.

bt_avrcp_4_release Macro

File

avrcp_command.h

C

```
#define bt_avrcp_4_release(channel) bt_avrcp_send_release_panel_control(channel, AVC_PANEL_OPID_4)
```

Description

This is macro bt_avrcp_4_release.

bt_avrcp_5_release Macro

File

avrcp_command.h

C

```
#define bt_avrcp_5_release(channel) bt_avrcp_send_release_panel_control(channel, AVC_PANEL_OPID_5)
```

Description

This is macro bt_avrcp_5_release.

bt_avrcp_6_release Macro

File

avrcp_command.h

C

```
#define bt_avrcp_6_release(channel) bt_avrcp_send_release_panel_control(channel, AVC_PANEL_OPID_6)
```

Description

This is macro bt_avrcp_6_release.

bt_avrcp_7_release Macro

File

avrcp_command.h

C

```
#define bt_avrcp_7_release(channel) bt_avrcp_send_release_panel_control(channel, AVC_PANEL_OPID_7)
```

Description

This is macro bt_avrcp_7_release.

bt_avrcp_8_release Macro

File

avrcp_command.h

C

```
#define bt_avrcp_8_release(channel) bt_avrcp_send_release_panel_control(channel, AVC_PANEL_OPID_8)
```

Description

This is macro bt_avrcp_8_release.

bt_avrcp_9_release Macro

File

avrcp_command.h

C

```
#define bt_avrcp_9_release(channel) bt_avrcp_send_release_panel_control(channel, AVC_PANEL_OPID_9)
```

Description

This is macro bt_avrcp_9_release.

bt_avrcp_angle_release Macro

File

avrcp_command.h

C

```
#define bt_avrcp_angle_release(channel) bt_avrcp_send_release_panel_control(channel, AVC_PANEL_OPID_ANGLE)
```

Description

This is macro bt_avrcp_angle_release.

bt_avrcp_backward_release Macro

File

avrcp_command.h

C

```
#define bt_avrcp_backward_release(channel) bt_avrcp_send_release_panel_control(channel,
AVC_PANEL_OPID_BACKWARD)
```

Description

This is macro bt_avrcp_backward_release.

bt_avrcp_channel_down_release Macro

File

avrcp_command.h

C

```
#define bt_avrcp_channel_down_release(channel) bt_avrcp_send_release_panel_control(channel,
AVC_PANEL_OPID_CHANNEL_DOWN)
```

Description

This is macro bt_avrcp_channel_down_release.

bt_avrcp_channel_up_release Macro

File

avrcp_command.h

C

```
#define bt_avrcp_channel_up_release(channel) bt_avrcp_send_release_panel_control(channel,
AVC_PANEL_OPID_CHANNEL_UP)
```

Description

This is macro bt_avrcp_channel_up_release.

bt_avrcp_clear_release Macro

File

avrcp_command.h

C

```
#define bt_avrcp_clear_release(channel) bt_avrcp_send_release_panel_control(channel, AVC_PANEL_OPID_CLEAR)
```

Description

This is macro bt_avrcp_clear_release.

bt_avrcp_content_menu_release Macro

File

avrcp_command.h

C

```
#define bt_avrcp_content_menu_release(channel) bt_avrcp_send_release_panel_control(channel,
AVC_PANEL_OPID_CONTENTS_MENU)
```

Description

This is macro bt_avrcp_content_menu_release.

bt_avrcp_display_info_release Macro

File

avrcp_command.h

C

```
#define bt_avrcp_display_info_release(channel) bt_avrcp_send_release_panel_control(channel,
AVC_PANEL_OPID_DISPLAY_INFORMATION)
```

Description

This is macro bt_avrcp_display_info_release.

bt_avrcp_dot_release Macro

File

avrcp_command.h

C

```
#define bt_avrcp_dot_release(channel) bt_avrcp_send_release_panel_control(channel, AVC_PANEL_OPID_DOT)
```

Description

This is macro bt_avrcp_dot_release.

bt_avrcp_down_release Macro

File

avrcp_command.h

C

```
#define bt_avrcp_down_release(channel) bt_avrcp_send_release_panel_control(channel, AVC_PANEL_OPID_DOWN)
```

Description

This is macro bt_avrcp_down_release.

bt_avrcp_eject_release Macro

File

avrcp_command.h

C

```
#define bt_avrcp_eject_release(channel) bt_avrcp_send_release_panel_control(channel, AVC_PANEL_OPID_EJECT)
```

Description

This is macro bt_avrcp_eject_release.

bt_avrcp_enter_release Macro

File

avrcp_command.h

C

```
#define bt_avrcp_enter_release(channel) bt_avrcp_send_release_panel_control(channel, AVC_PANEL_OPID_ENTER)
```

Description

This is macro bt_avrcp_enter_release.

bt_avrcp_exit_release Macro

File

avrcp_command.h

C

```
#define bt_avrcp_exit_release(channel) bt_avrcp_send_release_panel_control(channel, AVC_PANEL_OPID_EXIT)
```

Description

This is macro bt_avrcp_exit_release.

bt_avrcp_f1_release Macro

File

avrcp_command.h

C

```
#define bt_avrcp_f1_release(channel) bt_avrcp_send_release_panel_control(channel, AVC_PANEL_OPID_F1)
```

Description

This is macro bt_avrcp_f1_release.

bt_avrcp_f2_release Macro

File

avrcp_command.h

C

```
#define bt_avrcp_f2_release(channel) bt_avrcp_send_release_panel_control(channel, AVC_PANEL_OPID_F2)
```

Description

This is macro bt_avrcp_f2_release.

bt_avrcp_f3_release Macro

File

avrcp_command.h

C

```
#define bt_avrcp_f3_release(channel) bt_avrcp_send_release_panel_control(channel, AVC_PANEL_OPID_F3)
```

Description

This is macro bt_avrcp_f3_release.

bt_avrcp_f4_release Macro

File

avrcp_command.h

C

```
#define bt_avrcp_f4_release(channel) bt_avrcp_send_release_panel_control(channel, AVC_PANEL_OPID_F4)
```

Description

This is macro bt_avrcp_f4_release.

bt_avrcp_f5_release Macro

File

avrcp_command.h

C

```
#define bt_avrcp_f5_release(channel) bt_avrcp_send_release_panel_control(channel, AVC_PANEL_OPID_F5)
```

Description

This is macro bt_avrcp_f5_release.

bt_avrcp_f6_release Macro

File

avrcp_command.h

C

```
#define bt_avrcp_f6_release(channel) bt_avrcp_send_release_panel_control(channel, AVC_PANEL_OPID_F6)
```

Description

This is macro bt_avrcp_f6_release.

bt_avrcp_f7_release Macro

File

avrcp_command.h

C

```
#define bt_avrcp_f7_release(channel) bt_avrcp_send_release_panel_control(channel, AVC_PANEL_OPID_F7)
```

Description

This is macro bt_avrcp_f7_release.

bt_avrcp_f8_release Macro

File

avrcp_command.h

C

```
#define bt_avrcp_f8_release(channel) bt_avrcp_send_release_panel_control(channel, AVC_PANEL_OPID_F8)
```

Description

This is macro bt_avrcp_f8_release.

bt_avrcp_f9_release Macro

File

avrcp_command.h

C

```
#define bt_avrcp_f9_release(channel) bt_avrcp_send_release_panel_control(channel, AVC_PANEL_OPID_F9)
```

Description

This is macro bt_avrcp_f9_release.

bt_avrcp_fast_forward_release Macro

File

avrcp_command.h

C

```
#define bt_avrcp_fast_forward_release(channel) bt_avrcp_send_release_panel_control(channel,
AVC_PANEL_OPID_FAST_FORWARD)
```

Description

This is macro bt_avrcp_fast_forward_release.

bt_avrcp_favorite_menu_release Macro

File

avrcp_command.h

C

```
#define bt_avrcp_favorite_menu_release(channel) bt_avrcp_send_release_panel_control(channel,
AVC_PANEL_OPID_FAVORIVE_MENU)
```

Description

This is macro bt_avrcp_favorite_menu_release.

bt_avrcp_forward_release Macro

File

avrcp_command.h

C

```
#define bt_avrcp_forward_release(channel) bt_avrcp_send_release_panel_control(channel,
AVC_PANEL_OPID_FORWARD)
```

Description

This is macro bt_avrcp_forward_release.

bt_avrcp_help_release Macro

File

avrcp_command.h

C

```
#define bt_avrcp_help_release(channel) bt_avrcp_send_release_panel_control(channel, AVC_PANEL_OPID_HELP)
```

Description

This is macro bt_avrcp_help_release.

bt_avrcp_input_select_release Macro

File

avrcp_command.h

C

```
#define bt_avrcp_input_select_release(channel) bt_avrcp_send_release_panel_control(channel,
AVC_PANEL_OPID_INPUT_SELECT)
```

Description

This is macro bt_avrcp_input_select_release.

bt_avrcp_left_down_release Macro

File

avrcp_command.h

C

```
#define bt_avrcp_left_down_release(channel) bt_avrcp_send_release_panel_control(channel,
AVC_PANEL_OPID_LEFT_DOWN)
```

Description

This is macro bt_avrcp_left_down_release.

bt_avrcp_left_release Macro

File

avrcp_command.h

C

```
#define bt_avrcp_left_release(channel) bt_avrcp_send_release_panel_control(channel, AVC_PANEL_OPID_LEFT)
```

Description

This is macro bt_avrcp_left_release.

bt_avrcp_left_up_release Macro

File

avrcp_command.h

C

```
#define bt_avrcp_left_up_release(channel) bt_avrcp_send_release_panel_control(channel,
AVC_PANEL_OPID_LEFT_UP)
```

Description

This is macro bt_avrcp_left_up_release.

bt_avrcp_mute_release Macro

File

avrcp_command.h

C

```
#define bt_avrcp_mute_release(channel) bt_avrcp_send_release_panel_control(channel, AVC_PANEL_OPID_MUTE)
```

Description

This is macro bt_avrcp_mute_release.

bt_avrcp_page_down_release Macro

File

avrcp_command.h

C

```
#define bt_avrcp_page_down_release(channel) bt_avrcp_send_release_panel_control(channel,
AVC_PANEL_OPID_PAGE_DOWN)
```

Description

This is macro bt_avrcp_page_down_release.

bt_avrcp_page_up_release Macro

File

avrcp_command.h

C

```
#define bt_avrcp_page_up_release(channel) bt_avrcp_send_release_panel_control(channel,
AVC_PANEL_OPID_PAGE_UP)
```

Description

This is macro bt_avrcp_page_up_release.

bt_avrcp_pause_release Macro

File

avrcp_command.h

C

```
#define bt_avrcp_pause_release(channel) bt_avrcp_send_release_panel_control(channel, AVC_PANEL_OPID_PAUSE)
```

Description

This is macro bt_avrcp_pause_release.

bt_avrcp_play_release Macro

File

avrcp_command.h

C

```
#define bt_avrcp_play_release(channel) bt_avrcp_send_release_panel_control(channel, AVC_PANEL_OPID_PLAY)
```

Description

This is macro bt_avrcp_play_release.

bt_avrcp_power_release Macro

File

avrcp_command.h

C

```
#define bt_avrcp_power_release(channel) bt_avrcp_send_release_panel_control(channel,
AVC_PANEL_OPID_POWER_TOGGLE)
```

Description

This is macro bt_avrcp_power_release.

bt_avrcp_previous_channel_release Macro

File

avrcp_command.h

C

```
#define bt_avrcp_previous_channel_release(channel) bt_avrcp_send_release_panel_control(channel,
AVC_PANEL_OPID_PREVIOUS_CHANNEL)
```

Description

This is macro bt_avrcp_previous_channel_release.

bt_avrcp_record_release Macro

File

avrcp_command.h

C

```
#define bt_avrcp_record_release(channel) bt_avrcp_send_release_panel_control(channel, AVC_PANEL_OPID_RECORD)
```

Description

This is macro bt_avrcp_record_release.

bt_avrcp_rewind_release Macro

File

avrcp_command.h

C

```
#define bt_avrcp_rewind_release(channel) bt_avrcp_send_release_panel_control(channel, AVC_PANEL_OPID_REWIND)
```

Description

This is macro bt_avrcp_rewind_release.

bt_avrcp_right_down_release Macro

File

avrcp_command.h

C

```
#define bt_avrcp_right_down_release(channel) bt_avrcp_send_release_panel_control(channel,
AVC_PANEL_OPID_RIGHT_DOWN)
```

Description

This is macro bt_avrcp_right_down_release.

bt_avrcp_right_release Macro

File

avrcp_command.h

C

```
#define bt_avrcp_right_release(channel) bt_avrcp_send_release_panel_control(channel, AVC_PANEL_OPID_RIGHT)
```

Description

This is macro bt_avrcp_right_release.

bt_avrcp_right_up_release Macro

File

avrcp_command.h

C

```
#define bt_avrcp_right_up_release(channel) bt_avrcp_send_release_panel_control(channel,
AVC_PANEL_OPID_RIGHT_UP)
```

Description

This is macro bt_avrcp_right_up_release.

bt_avrcp_root_menu_release Macro

File

avrcp_command.h

C

```
#define bt_avrcp_root_menu_release(channel) bt_avrcp_send_release_panel_control(channel,
AVC_PANEL_OPID_ROOT_MENU)
```

Description

This is macro bt_avrcp_root_menu_release.

bt_avrcp_select_release Macro

File

avrcp_command.h

C

```
#define bt_avrcp_select_release(channel) bt_avrcp_send_release_panel_control(channel, AVC_PANEL_OPID_SELECT)
```

Description

This is macro bt_avrcp_select_release.

bt_avrcp_setup_menu_release Macro

File

avrcp_command.h

C

```
#define bt_avrcp_setup_menu_release(channel) bt_avrcp_send_release_panel_control(channel,
AVC_PANEL_OPID_SETUP_MENU)
```

Description

This is macro bt_avrcp_setup_menu_release.

bt_avrcp_sound_select_release Macro

File

avrcp_command.h

C

```
#define bt_avrcp_sound_select_release(channel) bt_avrcp_send_release_panel_control(channel,
AVC_PANEL_OPID_SOUND_SELECT)
```

Description

This is macro bt_avrcp_sound_select_release.

bt_avrcp_stop_release Macro

File

avrcp_command.h

C

```
#define bt_avrcp_stop_release(channel) bt_avrcp_send_release_panel_control(channel, AVC_PANEL_OPID_STOP)
```

Description

This is macro bt_avrcp_stop_release.

bt_avrcp_subpicture_release Macro

File

avrcp_command.h

C

```
#define bt_avrcp_subpicture_release(channel) bt_avrcp_send_release_panel_control(channel,
AVC_PANEL_OPID_SUBPICTURE)
```

Description

This is macro bt_avrcp_subpicture_release.

bt_avrcp_up_release Macro

File

avrcp_command.h

C

```
#define bt_avrcp_up_release(channel) bt_avrcp_send_release_panel_control(channel, AVC_PANEL_OPID_UP)
```

Description

This is macro bt_avrcp_up_release.

bt_avrcp_volume_down_release Macro

File

avrcp_command.h

C

```
#define bt_avrcp_volume_down_release(channel) bt_avrcp_send_release_panel_control(channel,
AVC_PANEL_OPID_VOLUME_DOWN)
```

Description

This is macro bt_avrcp_volume_down_release.

bt_avrcp_volume_up_release Macro

File

avrcp_command.h

C

```
#define bt_avrcp_volume_up_release(channel) bt_avrcp_send_release_panel_control(channel,
AVC_PANEL_OPID_VOLUME_UP)
```

Description

This is macro bt_avrcp_volume_up_release.

__AVRCP_CONFIG_EVENT_HANDLERS_H Macro

File

avrcp_config_event_handlers.h

C

```
#define __AVRCP_CONFIG_EVENT_HANDLERS_H
```

Description

This is macro __AVRCP_CONFIG_EVENT_HANDLERS_H.

AVRCP_COMMAND_HANDLER Macro

File

avrcp_config_event_handlers.h

C

```
#define AVRCP_COMMAND_HANDLER NULL
```

Description

This is macro AVRCP_COMMAND_HANDLER.

AVRCP_COMMAND_SENT_HANDLER Macro

File

avrcp_config_event_handlers.h

C

```
#define AVRCP_COMMAND_SENT_HANDLER NULL
```

Description

This is macro AVRCP_COMMAND_SENT_HANDLER.

AVRCP_RESPONSE_HANDLER Macro

File

avrcp_config_event_handlers.h

C

```
#define AVRCP_RESPONSE_HANDLER NULL
```

Description

This is macro AVRCP_RESPONSE_HANDLER.

AVRCP_RESPONSE_SENT_HANDLER Macro

File

avrcp_config_event_handlers.h

C

```
#define AVRCP_RESPONSE_SENT_HANDLER NULL
```

Description

This is macro AVRCP_RESPONSE_SENT_HANDLER.

GAP Functions

bt_gap_find_devices Function

File

gap.h

C

```
bt_bool bt_gap_find_devices(bt_device_t* devices, bt_byte max_devices, bt_byte length, bt_byte mode,
bt_find_devices_callback_fp callback, void* callback_param);
```

Description

This is function bt_gap_find_devices.

GAP Data Types and Constants

bt_device_t Structure

File

gap.h

C

```
typedef struct _bt_device_t {
  bt_bdaddr_t bdaddr;
  bt_byte* name;
  bt_byte max_name_len;
  bt_byte pg_scan_rpt_mode;
  bt_byte pg_scan_period_mode;
  bt_long cod;
  bt_int clock_offset;
  bt_byte rssi;
  bt_byte* eir;
  bt_byte eir_len;
  bt_byte max_eir_len;
} bt_device_t;
```

Description

This is type bt_device_t.

bt_find_devices_callback_fp Type

File

gap.h

C

```
typedef void (* bt_find_devices_callback_fp)(bt_byte count, void* callback_param);
```

Description

This is type bt_find_devices_callback_fp.

HCI Functions

bt_hci_add_param_bdaddr Function

File

hci.h

C

```
bt_bool bt_hci_add_param_bdaddr(bt_hci_command_p pcmd, const bt_bdaddr_t* pbdaddr);
```

Description

brief Add BD address parameter to an HCI command ingroup hci

bt_hci_add_param_byte Function

File

hci.h

C

```
bt_bool bt_hci_add_param_byte(bt_hci_command_p pcmd, bt_byte value);
```

Description

brief Add byte parameter to an HCI command ingroup hci

bt_hci_add_param_cod Function

File

hci.h

C

```
bt_bool bt_hci_add_param_cod(bt_hci_command_p pcmd, bt_long value);
```

Description

brief Add class of device parameter to an HCI command ingroup hci

bt_hci_add_param_int Function

File

hci.h

C

```
bt_bool bt_hci_add_param_int(bt_hci_command_p pcmd, bt_int value);
```

Description

brief Add int parameter to an HCI command ingroup hci

bt_hci_add_param_linkkey Function

File

hci.h

C

```
bt_bool bt_hci_add_param_linkkey(bt_hci_command_p pcmd, const bt_linkkey_t* linkkey);
```

Description

brief Add link key parameter to an HCI command ingroup hci

bt_hci_add_param_long Function

File

hci.h

C

```
bt_bool bt_hci_add_param_long(bt_hci_command_p pcmd, bt_long value);
```

Description

brief Add long parameter to an HCI command ingroup hci

bt_hci_add_param_string Function

File

hci.h

C

```
bt_bool bt_hci_add_param_string(bt_hci_command_p pcmd, const char * ps, bt_int len);
```

Description

brief Add string parameter to an HCI command ingroup hci

bt_hci_alloc_canned_command Function

File

hci.h

C

```
bt_hci_command_p bt_hci_alloc_canned_command(const bt_byte* canned_command, bt_hci_cmd_callback_fp
callback);
```

Description

brief Allocate and initialize an HCI command structure for a canned (pre-formatted) command. ingroup hci

bt_hci_alloc_command Function

File

hci.h

C

```
bt_hci_command_p bt_hci_alloc_command(bt_int opcode, bt_hci_cmd_callback_fp callback);
```

Description

brief Allocate and initialize an HCI command structure. ingroup hci

bt_hci_alloc_data_buffer Function

File

hci_data_buffer.h

C

```
bt_hci_data_p bt_hci_alloc_data_buffer();
```

Description

This is function bt_hci_alloc_data_buffer.

bt_hci_allocate_write_eir_command Function

File

hci_eir.h

C

```
bt_hci_command_p bt_hci_allocate_write_eir_command(bt_byte fec_required);
```

Description

This is function bt_hci_allocate_write_eir_command.

bt_hci_authenticate_ex Function

File

hci.h

C

```
bt_bool bt_hci_authenticate_ex(bt_hci_conn_state_t * pconn, bt_hci_cmd_callback_fp cb, void* cb_param);
```

Description

This is function bt_hci_authenticate_ex.

bt_hci_cancel_find_devices Function

File

hci.h

C

```
bt_bool bt_hci_cancel_find_devices(bt_hci_cmd_callback_fp cb);
```

Description

brief Stop inquiry

bt_hci_cancel_request_remote_name Function

File

hci.h

C

```
bt_bool bt_hci_cancel_request_remote_name(bt_bdaddr_p pbdaddr);
```

Description

brief Cancel remote device name request

bt_hci_cancel_send_acl_data Function

File

hci.h

C

```
void bt_hci_cancel_send_acl_data(bt_byte* data);
```

Description

brief Cancel sending data over ACL connection

bt_hci_connect Function

File

hci.h

C

```
bt_bool bt_hci_connect(bt_bdaddr_p dest, bt_uint packet_type, bt_byte pg_scan_rpt_mode, bt_byte
role_switch, bt_uint acl_config, bt_hci_connect_callback_fp callback, void* param);
```

Description

brief Connect to a remote device. ingroup hci

details This function tries to establish an HCI connection with a remote device specified by the Bluetooth address c dest. Upon completion, the callback function specified by the c callback parameter is called.

param dest Bluetooth address of the remote device. param packet_type param role_switch param acl_config param callback Pointer to a callback function that is called when the connect operation completes. param param Pointer to arbitrary data that is to be passed to the callback function.

return li c TRUE when the function succeeds. li c FALSE otherwise. The callback function is not called in this case.

bt_hci_connect_sco Function

File

hci.h

C

```
bt_bool bt_hci_connect_sco(bt_hci_conn_state_t* pconn, bt_ulong tx_bandwidth, bt_ulong rcv_bandwidth,
bt_uint max_latency, bt_uint content_format, bt_byte retransmission_effort, bt_uint packet_type,
```

```
bt_hci_connect_callback_fp cb, void * param);
```

Description

This is function bt_hci_connect_sco.

bt_hci_ctrl_register_data_listener Function

File

hci_ctrl_state.h

C

```
bt_bool bt_hci_ctrl_register_data_listener(bt_hci_ctrl_listener_t* listener, bt_byte listener_type);
```

Description

This is function bt_hci_ctrl_register_data_listener.

bt_hci_ctrl_register_listener Function

File

hci_ctrl_state.h

C

```
bt_bool bt_hci_ctrl_register_listener(bt_hci_ctrl_listener_t* listener);
```

Description

This is function bt_hci_ctrl_register_listener.

bt_hci_ctrl_unregister_listener Function

File

hci_ctrl_state.h

C

```
void bt_hci_ctrl_unregister_listener(bt_hci_ctrl_listener_t* listener);
```

Description

This is function bt_hci_ctrl_unregister_listener.

bt_hci_disconnect Function

File

hci.h

C

```
bt_bool bt_hci_disconnect(bt_hci_conn_state_t * pconn);
```

Description

brief Abort connection ingroup hci

bt_hci_evt_authentication_complete_handler Function

File

hci_evt_handlers.h

C

```
void bt_hci_evt_authentication_complete_handler(bt_hci_event_p pevt);
```

Description

This is function bt_hci_evt_authentication_complete_handler.

bt_hci_evt_change_conn_link_complete_handler Function

File

hci_evt_handlers.h

C

```
void bt_hci_evt_change_conn_link_complete_handler(bt_hci_event_p pevt);
```

Description

This is function bt_hci_evt_change_conn_link_complete_handler.

bt_hci_evt_command_complete_handler Function

File

hci_evt_handlers.h

C

```
void bt_hci_evt_command_complete_handler(bt_hci_event_p pevt);
```

Description

This is function bt_hci_evt_command_complete_handler.

bt_hci_evt_command_status_handler Function

File

hci_evt_handlers.h

C

```
void bt_hci_evt_command_status_handler(bt_hci_event_p pevt);
```

Description

This is function bt_hci_evt_command_status_handler.

bt_hci_evt_conn_packet_type_changed_handler Function

File

hci_evt_handlers.h

C

```
void bt_hci_evt_conn_packet_type_changed_handler(bt_hci_event_p pevt);
```

Description

This is function bt_hci_evt_conn_packet_type_changed_handler.

bt_hci_evt_connection_complete_handler Function

File

hci_evt_handlers.h

C

```
void bt_hci_evt_connection_complete_handler(bt_hci_event_p pevt);
```

Description

This is function bt_hci_evt_connection_complete_handler.

bt_hci_evt_connection_request_handler Function

File

hci_evt_handlers.h

C

```
void bt_hci_evt_connection_request_handler(bt_hci_event_p pevt);
```

Description

This is function bt_hci_evt_connection_request_handler.

bt_hci_evt_data_buffer_overflow_handler Function

File

hci_evt_handlers.h

C

```
void bt_hci_evt_data_buffer_overflow_handler(bt_hci_event_p pevt);
```

Description

This is function bt_hci_evt_data_buffer_overflow_handler.

bt_hci_evt_default_handler Function

File

hci_evt_handlers.h

C

```
void bt_hci_evt_default_handler(bt_hci_event_p pevt);
```

Description

This is function bt_hci_evt_default_handler.

bt_hci_evt_disconnection_complete_handler Function

File

hci_evt_handlers.h

C

```
void bt_hci_evt_disconnection_complete_handler(bt_hci_event_p pevt);
```

Description

This is function bt_hci_evt_disconnection_complete_handler.

bt_hci_evt_encryption_change_handler Function

File

hci_evt_handlers.h

C

```
void bt_hci_evt_encryption_change_handler(bt_hci_event_p pevt);
```

Description

This is function bt_hci_evt_encryption_change_handler.

bt_hci_evt_extended_inquiry_result_handler Function

File

hci_evt_handlers.h

C

```
void bt_hci_evt_extended_inquiry_result_handler(bt_hci_event_p pevt);
```

Description

This is function bt_hci_evt_extended_inquiry_result_handler.

bt_hci_evt_flow_specification_complete_handler Function

File

hci_evt_handlers.h

C

```
void bt_hci_evt_flow_specification_complete_handler(bt_hci_event_p pevt);
```

Description

This is function bt_hci_evt_flow_specification_complete_handler.

bt_hci_evt_flush_occured_handler Function

File

hci_evt_handlers.h

C

```
void bt_hci_evt_flush_occured_handler(bt_hci_event_p pevt);
```

Description

This is function bt_hci_evt_flush_occured_handler.

bt_hci_evt_hardware_error_handler Function

File

hci_evt_handlers.h

C

```
void bt_hci_evt_hardware_error_handler(bt_hci_event_p pevt);
```

Description

This is function bt_hci_evt_hardware_error_handler.

bt_hci_evt_inquiry_complete_handler Function

File

hci_evt_handlers.h

C

```
void bt_hci_evt_inquiry_complete_handler(bt_hci_event_p pevt);
```

Description

This is function bt_hci_evt_inquiry_complete_handler.

bt_hci_evt_inquiry_result_handler Function

File

hci_evt_handlers.h

C

```
void bt_hci_evt_inquiry_result_handler(bt_hci_event_p pevt);
```

Description

This is function bt_hci_evt_inquiry_result_handler.

bt_hci_evt_inquiry_result_with_rssi_handler Function

File

hci_evt_handlers.h

C

```
void bt_hci_evt_inquiry_result_with_rssi_handler(bt_hci_event_p pevt);
```

Description

This is function bt_hci_evt_inquiry_result_with_rssi_handler.

bt_hci_evt_link_key_notification_handler Function

File

hci_evt_handlers.h

C

```
void bt_hci_evt_link_key_notification_handler(bt_hci_event_p pevt);
```

Description

This is function bt_hci_evt_link_key_notification_handler.

bt_hci_evt_link_key_request_handler Function

File

hci_evt_handlers.h

C

```
void bt_hci_evt_link_key_request_handler(bt_hci_event_p pevt);
```

Description

This is function bt_hci_evt_link_key_request_handler.

bt_hci_evt_loopback_command_handler Function

File

hci_evt_handlers.h

C

```
void bt_hci_evt_loopback_command_handler(bt_hci_event_p pevt);
```

Description

This is function bt_hci_evt_loopback_command_handler.

bt_hci_evt_master_link_key_complete_handler Function

File

hci_evt_handlers.h

C

```
void bt_hci_evt_master_link_key_complete_handler(bt_hci_event_p pevt);
```

Description

This is function bt_hci_evt_master_link_key_complete_handler.

bt_hci_evt_max_slots_change_handler Function

File

hci_evt_handlers.h

C

```
void bt_hci_evt_max_slots_change_handler(bt_hci_event_p pevt);
```

Description

This is function bt_hci_evt_max_slots_change_handler.

bt_hci_evt_mode_change_handler Function

File

hci_evt_handlers.h

C

```
void bt_hci_evt_mode_change_handler(bt_hci_event_p pevt);
```

Description

This is function bt_hci_evt_mode_change_handler.

bt_hci_evt_num_of_completed_packets_handler Function

File

hci_evt_handlers.h

C

```
void bt_hci_evt_num_of_completed_packets_handler(bt_hci_event_p pevt);
```

Description

This is function bt_hci_evt_num_of_completed_packets_handler.

bt_hci_evt_page_scan_repet_mode_change_handler Function

File

hci_evt_handlers.h

C

```
void bt_hci_evt_page_scan_repet_mode_change_handler(bt_hci_event_p pevt);
```

Description

This is function bt_hci_evt_page_scan_repet_mode_change_handler.

bt_hci_evt_pin_code_request_handler Function

File

hci_evt_handlers.h

C

```
void bt_hci_evt_pin_code_request_handler(bt_hci_event_p pevt);
```

Description

This is function bt_hci_evt_pin_code_request_handler.

bt_hci_evt_qos_setup_complete_handler Function

File

hci_evt_handlers.h

C

```
void bt_hci_evt_qos_setup_complete_handler(bt_hci_event_p pevt);
```

Description

This is function bt_hci_evt_qos_setup_complete_handler.

bt_hci_evt_qos_violation_handler Function

File

hci_evt_handlers.h

C

```
void bt_hci_evt_qos_violation_handler(bt_hci_event_p pevt);
```

Description

This is function bt_hci_evt_qos_violation_handler.

bt_hci_evt_read_clock_offset_complete_handler Function

File

hci_evt_handlers.h

C

```
void bt_hci_evt_read_clock_offset_complete_handler(bt_hci_event_p pevt);
```

Description

This is function bt_hci_evt_read_clock_offset_complete_handler.

bt_hci_evt_read_rmt_ext_features_comp_handler Function

File

hci_evt_handlers.h

C

```
void bt_hci_evt_read_rmt_ext_features_comp_handler(bt_hci_event_p pevt);
```

Description

This is function bt_hci_evt_read_rmt_ext_features_comp_handler.

bt_hci_evt_read_rmt_sup_features_comp_handler Function

File

hci_evt_handlers.h

C

```
void bt_hci_evt_read_rmt_sup_features_comp_handler(bt_hci_event_p pevt);
```

Description

This is function bt_hci_evt_read_rmt_sup_features_comp_handler.

bt_hci_evt_read_rmt_version_info_comp_handler Function

File

hci_evt_handlers.h

C

```
void bt_hci_evt_read_rmt_version_info_comp_handler(bt_hci_event_p pevt);
```

Description

This is function bt_hci_evt_read_rmt_version_info_comp_handler.

bt_hci_evt_remote_name_request_complete_handler Function

File

hci_evt_handlers.h

C

```
void bt_hci_evt_remote_name_request_complete_handler(bt_hci_event_p pevt, bt_uint params_len);
```

Description

This is function bt_hci_evt_remote_name_request_complete_handler.

bt_hci_evt_return_link_keys_handler Function

File

hci_evt_handlers.h

C

```
void bt_hci_evt_return_link_keys_handler(bt_hci_event_p pevt);
```

Description

This is function bt_hci_evt_return_link_keys_handler.

bt_hci_evt_role_change_handler Function

File

hci_evt_handlers.h

C

```
void bt_hci_evt_role_change_handler(bt_hci_event_p pevt);
```

Description

This is function bt_hci_evt_role_change_handler.

bt_hci_evt_synch_connection_changed_handler Function

File

hci_evt_handlers.h

C

```
void bt_hci_evt_synch_connection_changed_handler(bt_hci_event_p pevt);
```

Description

This is function bt_hci_evt_synch_connection_changed_handler.

bt_hci_evt_synch_connection_complete_handler Function

File

hci_evt_handlers.h

C

```
void bt_hci_evt_synch_connection_complete_handler(bt_hci_event_p pevt);
```

Description

This is function bt_hci_evt_synch_connection_complete_handler.

bt_hci_exit_park_state Function

File

hci.h

C

```
bt_bool bt_hci_exit_park_state(bt_hci_conn_state_t * pconn, bt_hci_cmd_callback_fp cb);
```

Description

brief Exit park state

bt_hci_exit_sniff_mode_ex Function

File

hci.h

C

```
bt_bool bt_hci_exit_sniff_mode_ex(bt_hci_conn_state_t * pconn, bt_hci_cmd_callback_fp cb, void* cb_param);
```

Description

This is function bt_hci_exit_sniff_mode_ex.

bt_hci_find_devices Function

File

hci.h

C

```
bt_bool bt_hci_find_devices(bt_byte max_responses, bt_byte length, bt_hci_inquiry_callback_fp cb);
```

Description

brief Start inquiry

bt_hci_find_devices_ex Function

File

hci.h

C

```
bt_bool bt_hci_find_devices_ex(bt_byte max_responses, bt_byte length, bt_byte mode,
bt_hci_inquiry_callback_fp cb);
```

Description

This is function bt_hci_find_devices_ex.

bt_hci_free_command Function

File

hci.h

C

```
void bt_hci_free_command(bt_hci_command_p cmd);
```

Description

brief Free HCI command. ingroup hci

bt_hci_free_data_buffer Function

File

hci_data_buffer.h

C

```
void bt_hci_free_data_buffer(bt_hci_data_p p);
```

Description

This is function bt_hci_free_data_buffer.

bt_hci_get_evt_param_bdaddr Function

File

hci.h

C

```
bt_bool bt_hci_get_evt_param_bdaddr(bt_hci_event_p pevt, bt_bdaddr_p pvalue, bt_int_p poffset);
```

Description

brief Get bd address parameter from HCI event ingroup hci

bt_hci_get_evt_param_byte Function

File

hci.h

C

```
bt_bool bt_hci_get_evt_param_byte(bt_hci_event_p pevt, bt_byte* pvalue, bt_int* poffset);
```

Description

brief Get byte parameter from HCI event ingroup hci

bt_hci_get_evt_param_devclass Function

File

hci.h

C

```
bt_bool bt_hci_get_evt_param_devclass(bt_hci_event_p pevt, bt_long_p pvalue, bt_int_p poffset);
```

Description

brief Get class of device parameter from HCI event ingroup hci

bt_hci_get_evt_param_int Function

File

hci.h

C

```
bt_bool bt_hci_get_evt_param_int(bt_hci_event_p pevt, bt_int* pvalue, bt_int* poffset);
```

Description

brief Get int parameter from HCI event ingroup hci

bt_hci_get_evt_param_linkkey Function

File

hci.h

C

```
bt_bool bt_hci_get_evt_param_linkkey(bt_hci_event_p pevt, bt_linkkey_p pvalue, bt_int_p poffset);
```

Description

brief Get link key parameter from HCI event ingroup hci

bt_hci_get_evt_param_long Function

File

hci.h

C

```
bt_bool bt_hci_get_evt_param_long(bt_hci_event_p pevt, bt_long* pvalue, bt_int* poffset);
```

Description

brief Get long parameter from HCI event ingroup hci

bt_hci_get_evt_param_uint Function

File

hci.h

C

```
bt_bool bt_hci_get_evt_param_uint(bt_hci_event_p pevt, bt_uint* pvalue, bt_int* poffset);
```

Description

brief Get unsigned int parameter from HCI event ingroup hci

bt_hci_get_evt_param_ulong Function

File

hci.h

C

```
bt_bool bt_hci_get_evt_param_ulong(bt_hci_event_p pevt, bt_ulong* pvalue, bt_int* poffset);
```

Description

brief Get unsigned long parameter from HCI event ingroup hci

bt_hci_get_last_cmd_status Function

File

hci.h

C

```
bt_byte bt_hci_get_last_cmd_status();
```

Description

This is function bt_hci_get_last_cmd_status.

bt_hci_get_param_bdaddr Function

File

hci.h

C

```
bt_bool bt_hci_get_param_bdaddr(bt_hci_command_p pcm, bt_bdaddr_p pvalue, bt_int_p poffset);
```

Description

brief Get BD address parameter from HCI command ingroup hci

bt_hci_get_param_byte Function

File

hci.h

C

```
bt_bool bt_hci_get_param_byte(bt_hci_command_p pcmd, bt_byte* pvalue, bt_int* offset);
```

Description

brief Get byte parameter from HCI command ingroup hci

bt_hci_get_param_int Function

File

hci.h

C

```
bt_bool bt_hci_get_param_int(bt_hci_command_p pcmd, bt_int* pvalue, bt_int* poffset);
```

Description

brief Get int parameter from HCI command ingroup hci

bt_hci_get_param_linkkey Function

File

hci.h

C

```
bt_bool bt_hci_get_param_linkkey(bt_hci_command_p pcmd, bt_byte_p pvalue, bt_int_p poffset);
```

Description

brief Get link key parameter from HCI command ingroup hci

bt_hci_get_param_long Function

File

hci.h

C

```
bt_bool bt_hci_get_param_long(bt_hci_command_p pcmd, bt_long* pvalue, bt_int* poffset);
```

Description

brief Get long parameter from HCI command ingroup hci

bt_hci_get_recv_buffer Function

File

hci.h

C

```
bt_byte* bt_hci_get_recv_buffer();
```

Description

This is function bt_hci_get_recv_buffer.

bt_hci_get_recv_buffer_len Function

File

hci.h

C

```
bt_int bt_hci_get_recv_buffer_len();
```

Description

This is function bt_hci_get_recv_buffer_len.

bt_hci_get_send_buffer Function

File

hci.h

C

```
bt_byte* bt_hci_get_send_buffer();
```

Description

This is function bt_hci_get_send_buffer.

bt_hci_get_send_buffer_len Function

File

hci.h

C

```
bt_int bt_hci_get_send_buffer_len();
```

Description

This is function bt_hci_get_send_buffer_len.

bt_hci_init Function

File

hci.h

C

```
void bt_hci_init();
```

Description

brief Initialize the HCI layer. ingroup hci

This function initializes all internal variables of the HCI layer. The application, unless it's going to use only HCI layer, does not need to call this function as it is implicitly called by bt_sys_init.

This function essentially calls bt_hci_init_ex(HCI_LINK_POLICY_ENABLE_ALL) so all link policy setting are enabled.

bt_hci_init_data_buffers Function

File

hci_data_buffer.h

C

```
void bt_hci_init_data_buffers();
```

Description

This is function bt_hci_init_data_buffers.

bt_hci_init_data_queues Function

File

hci_data_queue.h

C

```
void bt_hci_init_data_queues();
```

Description

This is function bt_hci_init_data_queues.

bt_hci_init_linkkey_buffers Function

File

hci_linkkey_buffer.h

C

```
void bt_hci_init_linkkey_buffers();
```

Description

This is function bt_hci_init_linkkey_buffers.

bt_hci_le_add_device_to_white_list Function

File

hci_le.h

C

```
bt_bool bt_hci_le_add_device_to_white_list(bt_byte address_type, bt_bdaddr_t* address,
bt_hci_cmd_callback_fp cb);
```

Description

This is function bt_hci_le_add_device_to_white_list.

bt_hci_le_advertising_add Function

File

hci_le.h

C

```
bt_bool bt_hci_le_advertising_add(bt_byte data_type, const bt_byte* data, bt_byte length, bt_hci_command_p
pcmd);
```

Description

This is function bt_hci_le_advertising_add.

bt_hci_le_advertising_add_local_name Function

File

hci_le.h

C

```
bt_bool bt_hci_le_advertising_add_local_name(const char* local_name, bt_hci_command_t* pcmd);
```

Description

This is function bt_hci_le_advertising_add_local_name.

bt_hci_le_advertising_device_id_add Function

File

hci_le.h

C

```
bt_bool bt_hci_le_advertising_device_id_add(bt_uint vendor_id_source, bt_uint vendor_id, bt_uint
product_id, bt_uint version, bt_hci_command_p pcmd);
```

Description

This is function bt_hci_le_advertising_device_id_add.

bt_hci_le_advertising_flags_add Function

File

hci_le.h

C

```
bt_bool bt_hci_le_advertising_flags_add(bt_byte flags, bt_hci_command_p pcmd);
```

Description

This is function bt_hci_le_advertising_flags_add.

bt_hci_le_advertising_get_local_name Function

File

hci_le.h

C

```
bt_bool bt_hci_le_advertising_get_local_name(const bt_byte* data, bt_byte data_len, const bt_byte**
local_name, bt_byte* name_len, bt_byte* name_type);
```

Description

This is function bt_hci_le_advertising_get_local_name.

bt_hci_le_advertising_tx_power_level_add Function

File

hci_le.h

C

```
bt_bool bt_hci_le_advertising_tx_power_level_add(bt_byte tx_power_level, bt_hci_command_p pcmd);
```

Description

This is function bt_hci_le_advertising_tx_power_level_add.

bt_hci_le_advertising_uuid128_add Function

File

hci_le.h

C

```
bt_bool bt_hci_le_advertising_uuid128_add(bt_byte data_type, const bt_uuid_t* uuid_list, bt_byte
uuid_list_size, bt_hci_command_p pcmd);
```

Description

This is function bt_hci_le_advertising_uuid128_add.

bt_hci_le_advertising_uuid16_add Function

File

hci_le.h

C

```
bt_bool bt_hci_le_advertising_uuid16_add(bt_byte data_type, const bt_uint* uuid_list, bt_byte
uuid_list_size, bt_hci_command_p pcmd);
```

Description

This is function bt_hci_le_advertising_uuid16_add.

bt_hci_le_advertising_uuid32_add Function

File

hci_le.h

C

```
bt_bool bt_hci_le_advertising_uuid32_add(bt_byte data_type, const bt_uuid32* uuid_list, bt_byte
uuid_list_size, bt_hci_command_p pcmd);
```

Description

This is function bt_hci_le_advertising_uuid32_add.

bt_hci_le_advertising_vendor_add Function

File

hci_le.h

C

```
bt_bool bt_hci_le_advertising_vendor_add(bt_uint vendor_id, bt_byte* data, bt_byte data_len,
bt_hci_command_p pcmd);
```

Description

This is function bt_hci_le_advertising_vendor_add.

bt_hci_le_allocate_set_advertising_data_command Function

File

hci_le.h

C

```
bt_hci_command_t* bt_hci_le_allocate_set_advertising_data_command(bt_hci_cmd_callback_fp cb);
```

Description

This is function bt_hci_le_allocate_set_advertising_data_command.

bt_hci_le_cancel_find_devices Function

File

hci_le.h

C

```
bt_bool bt_hci_le_cancel_find_devices();
```

Description

This is function bt_hci_le_cancel_find_devices.

bt_hci_le_clear_white_list Function

File

hci_le.h

C

```
bt_bool bt_hci_le_clear_white_list(bt_hci_cmd_callback_fp cb);
```

Description

This is function bt_hci_le_clear_white_list.

bt_hci_le_connect_ex Function

File

hci_le.h

C

```
bt_bool bt_hci_le_connect_ex(bt_uint scan_interval, bt_uint scan_window, bt_byte filter_policy, bt_byte
peer_address_type, bt_bdaddr_t* peer_address, bt_byte own_address_type, bt_uint conn_interval_min, bt_uint
conn_interval_max, bt_uint conn_latency, bt_uint supervision_timeout, bt_uint min_ce_length, bt_uint
max_ce_length, bt_uint acl_config, bt_hci_connect_callback_fp cb, void * param);
```

Description

This is function bt_hci_le_connect_ex.

bt_hci_le_enable Function

File

hci_le.h

C

```
bt_bool bt_hci_le_enable(bt_bool enabled, bt_bool simultaneous_enabled, bt_hci_cmd_callback_fp cb);
```

Description

This is function bt_hci_le_enable.

bt_hci_le_encrypt Function

File

hci_le.h

C

```
bt_bool bt_hci_le_encrypt(const bt_byte* key, bt_byte key_len, const bt_byte* data, bt_byte data_len,
bt_hci_cmd_callback_fp cb, void* param);
```

Description

This is function bt_hci_le_encrypt.

bt_hci_le_find_devices Function

File

hci_le.h

C

```
bt_bool bt_hci_le_find_devices(bt_byte filter_duplicates, bt_hci_le_scan_callback_fp cb, void* param);
```

Description

This is function bt_hci_le_find_devices.

bt_hci_le_get_connect_parameters Function

File

hci_le.h

C

```
bt_bool bt_hci_le_get_connect_parameters(bt_hci_le_connect_parameters_t* params);
```

Description

This is function bt_hci_le_get_connect_parameters.

bt_hci_le_init Function

File

hci_private.h

C

```
void bt_hci_le_init(bt_hci_le_ctrl_state_t* le_ctrl_state);
```

Description

This is function bt_hci_le_init.

bt_hci_le_ltk_negative_reply Function

File

hci_le.h

C

```
bt_bool bt_hci_le_ltk_negative_reply(struct _bt_hci_conn_state_s* pconn, bt_hci_cmd_callback_fp cb);
```

Description

This is function bt_hci_le_ltk_negative_reply.

bt_hci_le_ltk_reply Function

File

hci_le.h

C

```
bt_bool bt_hci_le_ltk_reply(struct _bt_hci_conn_state_s* pconn, const bt_byte* long_term_key,
bt_hci_cmd_callback_fp cb);
```

Description

This is function bt_hci_le_ltk_reply.

bt_hci_le_rand Function

File

hci_le.h

C

```
bt_bool bt_hci_le_rand(bt_hci_cmd_callback_fp cb, void* param);
```

Description

This is function bt_hci_le_rand.

bt_hci_le_read_channel_map Function

File

hci_le.h

C

```
bt_bool bt_hci_le_read_channel_map(struct _bt_hci_conn_state_s* pconn, bt_hci_cmd_callback_fp cb);
```

Description

This is function bt_hci_le_read_channel_map.

bt_hci_le_read_remote_used_features Function

File

hci_le.h

C

```
bt_bool bt_hci_le_read_remote_used_features(struct _bt_hci_conn_state_s* pconn, bt_hci_cmd_callback_fp cb);
```

Description

This is function bt_hci_le_read_remote_used_features.

bt_hci_le_read_support Function

File

hci_le.h

C

```
bt_bool bt_hci_le_read_support(bt_hci_cmd_callback_fp cb);
```

Description

This is function bt_hci_le_read_support.

bt_hci_le_read_white_list_size Function

File

hci_le.h

C

```
bt_bool bt_hci_le_read_white_list_size(bt_hci_cmd_callback_fp cb);
```

Description

This is function bt_hci_le_read_white_list_size.

bt_hci_le_receiver_test Function

File

hci_le.h

C

```
bt_bool bt_hci_le_receiver_test(bt_byte frequency, bt_hci_cmd_callback_fp cb);
```

Description

This is function bt_hci_le_receiver_test.

bt_hci_le_remove_device_from_white_list Function

File

hci_le.h

C

```
bt_bool bt_hci_le_remove_device_from_white_list(bt_byte address_type, bt_bdaddr_t* address,
bt_hci_cmd_callback_fp cb);
```

Description

This is function bt_hci_le_remove_device_from_white_list.

bt_hci_le_set_adevrtising_enable_ex Function

File

hci_le.h

C

```
bt_bool bt_hci_le_set_adevrtising_enable_ex(bt_byte adv_enable, bt_hci_cmd_callback_fp cb, void* cb_param);
```

Description

This is function bt_hci_le_set_adevrtising_enable_ex.

bt_hci_le_set_advertising_parameters Function

File

hci_le.h

C

```
bt_bool bt_hci_le_set_advertising_parameters(bt_uint adv_interval_min, bt_uint adv_interval_max, bt_byte
adv_type, bt_byte own_address_type, bt_byte direct_address_type, bt_bdaddr_t* direct_address, bt_byte
adv_channel_map, bt_byte adv_filter_policy, bt_hci_cmd_callback_fp cb);
```

Description

This is function bt_hci_le_set_advertising_parameters.

bt_hci_le_set_connect_parameters Function

File

hci_le.h

C

```
bt_bool bt_hci_le_set_connect_parameters(bt_hci_le_connect_parameters_t* params);
```

Description

This is function bt_hci_le_set_connect_parameters.

bt_hci_le_set_host_channel_classification Function

File

hci_le.h

C

```
bt_bool bt_hci_le_set_host_channel_classification(const bt_byte* channel_map, bt_hci_cmd_callback_fp cb);
```

Description

This is function bt_hci_le_set_host_channel_classification.

bt_hci_le_set_random_address Function

File

hci_le.h

C

```
bt_bool bt_hci_le_set_random_address(bt_bdaddr_t* bdaddr, bt_hci_cmd_callback_fp cb);
```

Description

This is function bt_hci_le_set_random_address.

bt_hci_le_set_scan_enable Function

File

hci_le.h

C

```
bt_bool bt_hci_le_set_scan_enable(bt_byte scan_enable, bt_byte filter_duplicates, bt_hci_cmd_callback_fp
cb);
```

Description

This is function bt_hci_le_set_scan_enable.

bt_hci_le_set_scan_parameters Function

File

hci_le.h

C

```
bt_bool bt_hci_le_set_scan_parameters(bt_byte scan_type, bt_uint scan_interval, bt_uint scan_window,
bt_byte own_address_type, bt_byte filter_policy, bt_hci_cmd_callback_fp cb);
```

Description

This is function bt_hci_le_set_scan_parameters.

bt_hci_le_start_encryption Function

File

hci_le.h

C

```
bt_bool bt_hci_le_start_encryption(struct _bt_hci_conn_state_s* pconn, const bt_byte* random_number,
bt_uint encrypted_diversifier, const bt_byte* long_term_key, bt_hci_cmd_callback_fp cb, void* param);
```

Description

This is function bt_hci_le_start_encryption.

bt_hci_le_supported Function

File

hci_le.h

C

```
bt_bool bt_hci_le_supported();
```

Description

This is function bt_hci_le_supported.

bt_hci_le_test_end Function

File

hci_le.h

C

```
bt_bool bt_hci_le_test_end(bt_hci_cmd_callback_fp cb);
```

Description

This is function bt_hci_le_test_end.

bt_hci_le_transmitter_test Function

File

hci_le.h

C

```
bt_bool bt_hci_le_transmitter_test(bt_byte frequency, bt_byte test_data_len, bt_byte packet_payload,
bt_hci_cmd_callback_fp cb);
```

Description

This is function bt_hci_le_transmitter_test.

bt_hci_le_update_connection Function

File

hci_le.h

C

```
bt_bool bt_hci_le_update_connection(struct _bt_hci_conn_state_s* pconn, bt_uint conn_interval_min, bt_uint conn_interval_max, bt_uint conn_latency, bt_uint supervision_timeout, bt_uint min_ce_length, bt_uint max_ce_length, bt_hci_cmd_callback_fp cb, void * param);
```

Description

This is function bt_hci_le_update_connection.

bt_hci_le_write_support Function

File

hci_le.h

C

```
bt_bool bt_hci_le_write_support(bt_bool enabled, bt_bool simultaneous_enabled, bt_hci_cmd_callback_fp cb);
```

Description

This is function bt_hci_le_write_support.

bt_hci_listen Function

File

hci.h

C

```
bt_bool bt_hci_listen(bt_hci_connect_callback_fp cb, void* param);
```

Description

brief Listen for incoming connections. ingroup hci

details

param callback Pointer to a callback function that is called when a new incoming connection has been established. param param Pointer to arbitrary data that is to be passed to the callback function.

return li c TRUE when the function succeeds. li c FALSE otherwise. The callback function is not called in this case.

bt_hci_listen_sco Function

File

hci.h

C

```
bt_bool bt_hci_listen_sco(bt_ulong tx_bandwidth, bt_ulong rcv_bandwidth, bt_uint max_latency, bt_uint content_format, bt_byte retransmission_effort, bt_uint packet_type, bt_hci_connect_callback_fp cb, void * param);
```

Description

This is function bt_hci_listen_sco.

bt_hci_param_eir_add Function

File

hci_eir.h

C

```
bt_bool bt_hci_param_eir_add(bt_byte data_type, const bt_byte* data, bt_byte length, bt_hci_command_p pcmd);
```

Description

This is function bt_hci_param_eir_add.

bt_hci_param_eir_device_id_add Function

File

hci_eir.h

C

```
bt_bool bt_hci_param_eir_device_id_add(bt_uint vendor_id_source, bt_uint vendor_id, bt_uint product_id,
bt_uint version, bt_hci_command_p pcmd);
```

Description

This is function bt_hci_param_eir_device_id_add.

bt_hci_param_eir_local_name_add Function

File

hci_eir.h

C

```
bt_bool bt_hci_param_eir_local_name_add(const char* local_name, bt_hci_command_p pcmd);
```

Description

This is function bt_hci_param_eir_local_name_add.

bt_hci_param_eir_uuid128_add Function

File

hci_eir.h

C

```
bt_bool bt_hci_param_eir_uuid128_add(bt_byte data_type, const bt_uuid_t* uuid_list, bt_byte uuid_list_size,
bt_hci_command_p pcmd);
```

Description

This is function bt_hci_param_eir_uuid128_add.

bt_hci_param_eir_uuid16_add Function

File

hci_eir.h

C

```
bt_bool bt_hci_param_eir_uuid16_add(bt_byte data_type, const bt_uint* uuid_list, bt_byte uuid_list_size,
bt_hci_command_p pcmd);
```

Description

This is function bt_hci_param_eir_uuid16_add.

bt_hci_param_eir_uuid32_add Function

File

hci_eir.h

C

```
bt_bool bt_hci_param_eir_uuid32_add(bt_byte data_type, const bt_uuid32* uuid_list, bt_byte uuid_list_size,
bt_hci_command_p pcmd);
```

Description

This is function bt_hci_param_eir_uuid32_add.

bt_hci_param_eir_vendor_add Function

File

hci_eir.h

C

```
bt_bool bt_hci_param_eir_vendor_add(bt_uint vendor_id, bt_byte* data, bt_byte data_len, bt_hci_command_p
pcmd);
```

Description

This is function bt_hci_param_eir_vendor_add.

bt_hci_param_tx_power_level_add Function

File

hci_eir.h

C

```
bt_bool bt_hci_param_tx_power_level_add(bt_byte tx_power_level, bt_hci_command_p pcmd);
```

Description

This is function bt_hci_param_tx_power_level_add.

bt_hci_park_state Function

File

hci.h

C

```
bt_bool bt_hci_park_state(bt_hci_conn_state_t * pconn, bt_int min_interval, bt_int max_interval,
bt_hci_cmd_callback_fp cb);
```

Description

brief Put local device to park state

bt_hci_read_inquiry_mode Function

File

hci.h

C

```
bt_bool bt_hci_read_inquiry_mode(bt_hci_read_inquiry_mode_callback_fp cb);
```

Description

brief Get current inquiry mode

bt_hci_read_inquiry_scan_activity Function

File

hci.h

C

```
bt_bool bt_hci_read_inquiry_scan_activity(bt_hci_read_inquiry_scan_activity_callback_fp cb);
```

Description

brief Get current inquiry scan activity configuration

bt_hci_read_inquiry_scan_type Function

File

hci.h

C

```
bt_bool bt_hci_read_inquiry_scan_type(bt_hci_read_inquiry_scan_type_callback_fp cb);
```

Description

brief Get current inquiry scan type

bt_hci_read_page_scan_activity Function

File

hci.h

C

```
bt_bool bt_hci_read_page_scan_activity(bt_hci_read_page_scan_activity_callback_fp cb);
```

Description

brief Get current page scan activity configuration

bt_hci_read_page_scan_period_mode Function

File

hci.h

C

```
bt_bool bt_hci_read_page_scan_period_mode(bt_hci_read_page_scan_period_mode_callback_fp cb);
```

Description

brief Get current page scan period mode

bt_hci_read_page_scan_type Function

File

hci.h

C

```
bt_bool bt_hci_read_page_scan_type(bt_hci_read_page_scan_type_callback_fp cb);
```

Description

brief Get current page scan type

bt_hci_read_page_timeout Function

File

hci.h

C

```
bt_bool bt_hci_read_page_timeout(bt_hci_read_page_timeout_callback_fp cb);
```

Description

brief Get current page timeout

bt_hci_register_listener Function

File

hci_conn_state.h

C

```
bt_bool bt_hci_register_listener(bt_hci_conn_state_t* pconn, bt_hci_listener_t* listener);
```

Description

This is function bt_hci_register_listener.

bt_hci_reject_pin_code Function

File

hci.h

C

```
void bt_hci_reject_pin_code(bt_bdaddr_p pbdaddr_remote);
```

Description

This is function bt_hci_reject_pin_code.

bt_hci_request_remote_name Function

File

hci.h

C

```
bt_bool bt_hci_request_remote_name(bt_bdaddr_p pbdaddr, bt_byte pg_scan_rpt_mode, bt_int clock_offset,
bt_hci_request_remote_name_callback_fp cb);
```

Description

brief Request remote device's name

bt_hci_role_change_ex Function

File

hci.h

C

```
bt_bool bt_hci_role_change_ex(bt_bdaddr_p pbdaddr, bt_byte role, bt_hci_cmd_callback_fp cb, void* cb_param);
```

Description

brief Change local device's role

bt_hci_send_acl_data Function

File

hci.h

C

```
bt_bool bt_hci_send_acl_data(bt_hci_hconn_t h, bt_byte_p pdata, bt_int len, bt_hci_data_callback_fp cb,
void * param);
```

Description

brief Send data over ACL connection

bt_hci_send_cmd Function

File

hci.h

C

```
bt_bool bt_hci_send_cmd(bt_hci_command_p pcmd);
```

Description

brief Send a command to local device

bt_hci_send_linkkey Function

File

hci.h

C

```
bt_bool bt_hci_send_linkkey(const bt_bdaddr_t* bdaddr_remote, const bt_linkkey_t* link_key,
bt_hci_cmd_callback_fp cb);
```

Description

This is function bt_hci_send_linkkey.

bt_hci_send_sco_data Function

File

hci.h

C

```
bt_bool bt_hci_send_sco_data(bt_hci_hconn_t h, bt_byte_p pdata, bt_uint len, bt_hci_data_callback_fp cb,
void * param);
```

Description

brief Send data over SCO connection

bt_hci_set_encryption_ex Function

File

hci.h

C

```
bt_bool bt_hci_set_encryption_ex(bt_hci_conn_state_t* pconn, bt_byte encryption_enable,
bt_hci_cmd_callback_fp cb, void* cb_param);
```

Description

brief Set connection encryption

bt_hci_set_event_listener Function

File

hci_ctrl_state.h

C

```
void bt_hci_set_event_listener(bt_hci_event_listener_fp callback);
```

Description

listener is triggered by the following events: HCI_EVT_AUTHENTICATION_COMPLETE HCI_EVT_CONNECTION_COMPLETE HCI_EVT_DISCONNECTION_COMPLETE HCI_EVT_ROLE_CHANGE HCI_EVT_MODE_CHANGE

bt_hci_set_incoming_connection_role Function

File

hci.h

C

```
void bt_hci_set_incoming_connection_role(bt_byte role);
```

Description

This is function bt_hci_set_incoming_connection_role.

bt_hci_set_scan Function

File

hci.h

C

```
bt_bool bt_hci_set_scan(bt_bool discoverable, bt_ulong discoverable_period, bt_bool connectable, bt_ulong
connectable_period, bt_hci_cmd_callback_fp cb);
```

Description

This is function bt_hci_set_scan.

bt_hci_sniff_mode_ex Function

File

hci.h

C

```
bt_bool bt_hci_sniff_mode_ex(bt_hci_conn_state_t * pconn, bt_int min_interval, bt_int max_interval, bt_int
attempt_slots, bt_int timeout, bt_hci_cmd_callback_fp cb, void* cb_param);
```

Description

This is function bt_hci_sniff_mode_ex.

bt_hci_sniff_subrating_ex Function

File

hci.h

C

```
bt_bool bt_hci_sniff_subrating_ex(bt_hci_conn_state_t* pconn, bt_uint max_latency, bt_uint
min_remote_timeout, bt_uint min_local_timeout, bt_hci_cmd_callback_fp cb, void* cb_param);
```

Description

This is function bt_hci_sniff_subrating_ex.

bt_hci_start Function

File

hci.h

C

```
bt_bool bt_hci_start(bt_hci_start_callback_fp callback, void* callback_param, bt_byte enable_scan);
```

Description

brief Start HCI layer. ingroup hci

details This function starts the HCI layer of the stack. Starting the HCI layer consists essentially of two steps:

1. Make the HCI transport receive packets from the controller. This results in a call to bt_oem_recv.
2. Reset and configure the controller.

Upon completion of controller initialization the callback function passed in the c callback parameter is called.

param callback Completion callback. Called when controller initialization is complete. param callback_param A pointer to arbitrary data to be passed to the c callback callback. param enable_scan This is a bitmask that defines which scans are enabled during the controller configuration. This value can be a combination of the following values: li HCI_SCAN_INQUIRY (the controller is discoverable) li HCI_SCAN_PAGE (the controller is connectable)

return li c TRUE when the function succeeds. li c FALSE otherwise. The callback function is not called in this case.

bt_hci_start_no_init Function

File

hci.h

C

```
void bt_hci_start_no_init();
```

Description

brief Start HCI layer without controller configuration. ingroup hci

details This function is similar to bt_hci_start but unlike the former it does not perform the controller configuration. I.e., bt_hci_start_no_init simply calls the HCI transport and makes it receive packets from the controller. The main purpose of this function is make the HCI transport ready to exchange packets if controller needs some vendor specific configuration before it can be used with the stack. E.g., controllers based on CRS8811 chip need loading various values that configure its operating mode using CSR's proprietary protocol. So the application after configuring the HCI transport would call bt_hci_init(), bt_hci_start_no_init() and then load configuration values. Once the vendor specific configuration is done, the application will re-initialize the HCI transport and perform full start of the stack with bt_sys_init() and bt_sys_start().

bt_hci_stop Function

File

hci.h

C

```
void bt_hci_stop(bt_hci_stop_callback_fp callback, void* callback_param);
```

Description

brief Stop HCI layer. ingroup hci

details This function makes the HCI layer inoperable. After this call the application must perform the full reset of the HCI transport and stack.

param callback Completion callback. Called when the HCI layer has been stopped. param callback_param A pointer to arbitrary data to be passed to the c callback callback.

bt_hci_transport_recv_packet Function

File

hci_transport.h

C

```
void bt_hci_transport_recv_packet(bt_byte* buffer, bt_uint len, bt_hci_transport_recv_packet_callback_fp
callback);
```

Description

This is function bt_hci_transport_recv_packet.

bt_hci_transport_send_cmd Function

File

hci_transport.h

C

```
void bt_hci_transport_send_cmd(const bt_byte* buffer, bt_uint len, bt_hci_transport_send_packet_callback_fp
callback);
```

Description

deprecated

bt_hci_transport_send_data Function

File

hci_transport.h

C

```
void bt_hci_transport_send_data(bt_byte* buffer, bt_uint len, bt_hci_transport_send_packet_callback_fp
callback);
```

Description

This is function bt_hci_transport_send_data.

bt_hci_transport_send_packet Function

File

hci_transport.h

C

```
void bt_hci_transport_send_packet(const bt_byte* buffer, bt_uint len,
bt_hci_transport_send_packet_callback_fp callback);
```

Description

This is function bt_hci_transport_send_packet.

bt_hci_transport_set_transport Function

File

hci_transport.h

C

```
void bt_hci_transport_set_transport(const hci_transport_t* transport);
```

Description

This is function bt_hci_transport_set_transport.

bt_hci_unregister_listener Function

File

hci_conn_state.h

C

```
void bt_hci_unregister_listener(bt_hci_conn_state_t* pconn, bt_hci_listener_t* listener);
```

Description

This is function bt_hci_unregister_listener.

bt_hci_write_default_link_policy_settings Function

File

hci.h

C

```
bt_bool bt_hci_write_default_link_policy_settings(bt_uint settings, bt_hci_cmd_callback_fp cb);
```

Description

This is function bt_hci_write_default_link_policy_settings.

bt_hci_write_eir Function

File

hci_eir.h

C

```
bt_bool bt_hci_write_eir(bt_hci_command_p pcmd, bt_hci_cmd_callback_fp cb);
```

Description

This is function bt_hci_write_eir.

bt_hci_write_inquiry_mode Function

File

hci.h

C

```
bt_bool bt_hci_write_inquiry_mode(bt_byte scanMode, bt_hci_cmd_callback_fp cb);
```

Description

brief Configure inquiry mode

bt_hci_write_inquiry_scan_activity Function

File

hci.h

C

```
bt_bool bt_hci_write_inquiry_scan_activity(bt_uint interval, bt_uint window, bt_hci_cmd_callback_fp cb);
```

Description

brief Configure inquiry scan activity

bt_hci_write_inquiry_scan_type Function

File

hci.h

C

```
bt_bool bt_hci_write_inquiry_scan_type(bt_byte scanType, bt_hci_cmd_callback_fp cb);
```

Description

brief Configure inquiry scan type

bt_hci_write_local_name Function

File

hci.h

C

```
bt_bool bt_hci_write_local_name(const char * device_name, bt_hci_cmd_callback_fp cb);
```

Description

brief Write local device name ingroup hci

details The Write_Local_Name command provides the ability to modify the userfriendly name for the BR/EDR Controller.

bt_hci_write_page_scan_activity Function

File

hci.h

C

```
bt_bool bt_hci_write_page_scan_activity(bt_uint interval, bt_uint window, bt_hci_cmd_callback_fp cb);
```

Description

brief Configure page scan activity

bt_hci_write_page_scan_period_mode Function

File

hci.h

C

```
bt_bool bt_hci_write_page_scan_period_mode(bt_byte mode, bt_hci_cmd_callback_fp cb);
```

Description

brief Configure page scan period mode

bt_hci_write_page_scan_type Function

File

hci.h

C

```
bt_bool bt_hci_write_page_scan_type(bt_byte scanType, bt_hci_cmd_callback_fp cb);
```

Description

brief Configure page scan type

bt_hci_write_page_timeout Function

File

hci.h

C

```
bt_bool bt_hci_write_page_timeout(bt_uint timeout, bt_hci_cmd_callback_fp cb);
```

Description

brief Configure page timeout

bt_hcitr_3wire_start Function

File

hcitr_3wire.h

C

```
void bt_hcitr_3wire_start();
```

Description

This is function bt_hcitr_3wire_start.

bt_hcitr_bcsp_init_ex Function

File

hcitr_bcsp.h

C

```
void bt_hcitr_bcsp_init_ex(bt_ulong ack_timeout);
```

Description

This is function bt_hcitr_bcsp_init_ex.

bt_hcitr_bcsp_reset_ex Function

File

hcitr_bcsp.h

C

```
void bt_hcitr_bcsp_reset_ex(bt_ulong ack_timeout);
```

Description

This is function bt_hcitr_bcsp_reset_ex.

bt_hcitr_bcsp_start Function

File

hcitr_bcsp.h

C

```
void bt_hcitr_bcsp_start();
```

Description

This is function bt_hcitr_bcsp_start.

bt_hcitr_packet_init Function

File

hcitr_packet.h

C

```
void bt_hcitr_packet_init();
```

Description

This is function bt_hcitr_packet_init.

bt_hcitr_packet_reset Function

File

hcitr_packet.h

C

```
void bt_hcitr_packet_reset();
```

Description

This is function bt_hcitr_packet_reset.

bt_hcitr_packet_start Function

File

hcitr_packet.h

C

```
void bt_hcitr_packet_start();
```

Description

This is function bt_hcitr_packet_start.

bt_hcitr_tih4_init Function

File

hcitr_tih4.h

C

```
void bt_hcitr_tih4_init(bt_hcitr_tih4_power_callback_fp callback);
```

Description

This is function bt_hcitr_tih4_init.

bt_hcitr_tih4_reset Function

File

hcitr_tih4.h

C

```
void bt_hcitr_tih4_reset();
```

Description

This is function bt_hcitr_tih4_reset.

bt_hcitr_tih4_start Function

File

hcitr_tih4.h

C

```
void bt_hcitr_tih4_start();
```

Description

This is function bt_hcitr_tih4_start.

bt_hcitr_tih4_wake_up Function

File

hcitr_tih4.h

C

```
void bt_hcitr_tih4_wake_up();
```

Description

This is function bt_hcitr_tih4_wake_up.

bt_hcitr_uart_init Function

File

hcitr_uart.h

C

```
void bt_hcitr_uart_init();
```

Description

brief Initialize HCI UART (H4) transport protocol ingroup hcitr_uart

details This function initializes internal structures of the transport. The application must call it as early as possible before bt_hcitr_uart_start and before the stack is initialized and started with bt_sys_init and bt_sys_start.

bt_hcitr_uart_reset Function

File

hcitr_uart.h

C

```
void bt_hcitr_uart_reset();
```

Description

brief Re-initialize HCI UART (H4) transport protocol ingroup hcitr_uart

details This function re-initializes the transport. Currently it simply calls bt_hcitr_uart_init. After calling this function the application must perform the full initialization of the stack by calling bt_sys_init, bt_sys_start and initialization functions of all other profile modules the application is intending to use.

bt_hcitr_uart_start Function

File

hcitr_uart.h

C

```
void bt_hcitr_uart_start();
```

Description

brief Start HCI UART (H4) transport protocol ingroup hcitr_uart

details This function starts the transport, i.e., makes it able to receive and send packets.

hci_allocate_conn_state Function

File

hci_conn_state.h

C

```
bt_hci_conn_state_p hci_allocate_conn_state(struct _bt_hci_ctrl_state_s * pctrl);
```

Description

This is function hci_allocate_conn_state.

hci_check_aux_info Function

File

hci_private.h

C

```
bt_bool hci_check_aux_info(bt_long bdaddr_l, bt_long bdaddr_m);
```

Description

From hci_aux_info.c

hci_cq_find_by_bdaddr_and_opcode Function

File

hci_cmd_queue.h

C

```
bt_hci_command_p hci_cq_find_by_bdaddr_and_opcode(bt_queue_element_t* head, bt_bdaddr_t bdaddrFind, bt_int
opcode);
```

Description

This is function hci_cq_find_by_bdaddr_and_opcode.

hci_cq_find_by_hconn Function

File

hci_cmd_queue.h

C

```
bt_hci_command_p hci_cq_find_by_hconn(bt_queue_element_t* head, bt_hci_hconn_t hconn);
```

Description

This is function hci_cq_find_by_hconn.

hci_cq_find_by_hconn_and_opcode Function

File

hci_cmd_queue.h

C

```
bt_hci_command_p hci_cq_find_by_hconn_and_opcode(bt_queue_element_t* head, bt_hci_hconn_t hconnFind, bt_int
opcode);
```

Description

This is function hci_cq_find_by_hconn_and_opcode.

hci_cq_find_by_opcode Function

File

hci_cmd_queue.h

C

```
bt_hci_command_p hci_cq_find_by_opcode(bt_queue_element_t* head, bt_int opcode);
```

Description

This is function hci_cq_find_by_opcode.

hci_get_conn_state Function

File

hci_conn_state.h

C

```
bt_hci_conn_state_p hci_get_conn_state(struct _bt_hci_ctrl_state_s * pctrl, bt_hci_hconn_t h);
```

Description

This is function hci_get_conn_state.

hci_sleep Function

File

hci.h

C

```
void hci_sleep(pf_hci_sleep_callback cb);
```

Description

brief Put local device to sleep

hci_wakeup Function

File

hci.h

C

```
void hci_wakeup(pf_hci_wakeup_callback cb);
```

Description

brief Awaken local device

is_bdaddr_command Function

File

hci_utils.h

C

```
bt_bool is_bdaddr_command(bt_int opcode);
```

Description

This is function is_bdaddr_command.

is_hconn_command Function

File

hci_utils.h

C

```
bt_bool is_hconn_command(bt_int opcode);
```

Description

This is function is_hconn_command.

_bt_hci_ctrl_notify_data_listeners Function

File

hci_private.h

C

```
void _bt_hci_ctrl_notify_data_listeners(bt_hci_conn_state_t* connection, const bt_byte* data, bt_int len,
bt_bool sent);
```

Description

This is function _bt_hci_ctrl_notify_data_listeners.

_bt_hci_ctrl_notify_listeners Function

File

hci_private.h

C

```
void _bt_hci_ctrl_notify_listeners(bt_uint event_id, void* params);
```

Description

This is function _bt_hci_ctrl_notify_listeners.

_bt_hci_get_tick_count Function

File

hci_private.h

C

```
bt_ulong _bt_hci_get_tick_count();
```

Description

This is function _bt_hci_get_tick_count.

_bt_hci_init_signal Function

File

hci_signal.h

C

```
void _bt_hci_init_signal();
```

Description

This is function _bt_hci_init_signal.

_bt_hci_init_timer Function

File

hci_private.h

C

```
void _bt_hci_init_timer();
```

Description

From hci_timer.c

_bt_hci_init_transport Function

File

hci_private.h

C

```
void _bt_hci_init_transport();
```

Description

This is function _bt_hci_init_transport.

_bt_hci_le_command_complete_handler Function

File

hci_private.h

C

```
void _bt_hci_le_command_complete_handler(bt_int status, bt_hci_command_t* pcmd, bt_hci_event_t* pevt);
```

Description

This is function _bt_hci_le_command_complete_handler.

_bt_hci_notify_listeners Function

File

hci_private.h

C

```
void _bt_hci_notify_listeners(bt_hci_conn_state_t* pconn, bt_byte event_id, bt_hci_event_e* params);
```

Description

This is function _bt_hci_notify_listeners.

_bt_hci_set_signal Function

File

hci_signal.h

C

```
void _bt_hci_set_signal();
```

Description

This is function _bt_hci_set_signal.

_hci_allocate_buffers Function

File

hci_private.h

C

```
void _hci_allocate_buffers();
```

Description

Defined by OEM through library configuration

_hci_allocate_cmd Function

File

hci_cmd_buffer.h

C

```
bt_hci_command_t* _hci_allocate_cmd();
```

Description

This is function _hci_allocate_cmd.

_hci_free_cmd Function

File

hci_cmd_buffer.h

C

```
void _hci_free_cmd(bt_hci_command_t* pcmd);
```

Description

This is function _hci_free_cmd.

_hci_init_cmd_buffers Function

File

hci_cmd_buffer.h

C

```
void _hci_init_cmd_buffers();
```

Description

This is function _hci_init_cmd_buffers.

_hci_init_cmd_queues Function

File

hci_cmd_queue.h

C

```
void _hci_init_cmd_queues();
```

Description

This is function _hci_init_cmd_queues.

_hci_receive_start Function

File

hci_private.h

C

```
void _hci_receive_start();
```

Description

From hci_receive.c

_hci_recv_sco_data_packet Function

File

hci_private.h

C

```
void _hci_recv_sco_data_packet(bt_byte_p pbuf);
```

Description

From hci_sco.c

_hci_send_commands_from_queue Function

File

hci_private.h

C

```
void _hci_send_commands_from_queue();
```

Description

From hci_send.c

_hci_send_data Function

File

hci_private.h

C

```
bt_bool _hci_send_data(bt_hci_data_p pdata);
```

Description

This is function _hci_send_data.

_hci_send_data_fragment Function

File

hci_private.h

C

```
void _hci_send_data_fragment(bt_hci_data_p pdata);
```

Description

This is function _hci_send_data_fragment.

_hci_send_data_from_queue Function

File

hci_private.h

C

```
void _hci_send_data_from_queue();
```

Description

This is function _hci_send_data_from_queue.

_bt_le_evt_handler Function

File

hci_private.h

C

```
void _bt_le_evt_handler(bt_hci_event_t* evt);
```

Description

This is function _bt_le_evt_handler.

bt_hci_le_allocate_set_scan_response_data_command Function

File

hci_le.h

C

```
bt_hci_command_t* bt_hci_le_allocate_set_scan_response_data_command(bt_hci_cmd_callback_fp cb);
```

Description

This is function bt_hci_le_allocate_set_scan_response_data_command.

bt_hci_le_cancel_connect_ex Function

File

hci_le.h

C

```
bt_bool bt_hci_le_cancel_connect_ex(bt_hci_cmd_callback_fp cb, void * cb_param);
```

Description

This is function bt_hci_le_cancel_connect_ex.

bt_hci_le_ibeacon_add Function

File

hci_le.h

C

```
bt_bool bt_hci_le_ibeacon_add(const bt_uuid_t* uuid, bt_uint major, bt_uint minor, bt_byte tx_power_level,
bt_hci_command_p pcmd);
```

Description

This is function bt_hci_le_ibeacon_add.

bt_hci_set_vendor_specific_event_handler Function

File

hci_evt_handlers.h

C

```
void bt_hci_set_vendor_specific_event_handler(bt_hci_event_handler_fp handler);
```

Description

This is function bt_hci_set_vendor_specific_event_handler.

bt_hcitr_3wire_init_ex Function

File

hcitr_3wire.h

C

```
void bt_hcitr_3wire_init_ex(bt_ulong ack_timeout, bt_bool data_integrity_check_enabled);
```

Description

This is function bt_hcitr_3wire_init_ex.

bt_hcitr_3wire_reset_ex Function

File

hcitr_3wire.h

C

```
void bt_hcitr_3wire_reset_ex(bt_ulong ack_timeout, bt_bool data_integrity_check_enabled);
```

Description

This is function bt_hcitr_3wire_reset_ex.

_bt_hci_set_init_flags Function

File

hci_private.h

C

```
void _bt_hci_set_init_flags(bt_byte flags);
```

Description

This is function _bt_hci_set_init_flags.

hci_get_conn_state_by_bdaddr Function

File

hci_conn_state.h

C

```
bt_hci_conn_state_p hci_get_conn_state_by_bdaddr(struct _bt_hci_ctrl_state_s * pctrl, const bt_bdaddr_t*
pbdaddr_remote, bt_byte conn_type, bt_byte link_type);
```

Description

This is function hci_get_conn_state_by_bdaddr.

bt_hci_cancel_find_devices_ex Function

File

hci.h

C

```
bt_bool bt_hci_cancel_find_devices_ex(bt_hci_cmd_callback_fp cb, void* cb_param);
```

Description

brief Stop inquiry

bt_hci_get_inquiry_response_tx_power_level Function

File

hci.h

C

```
bt_byte bt_hci_get_inquiry_response_tx_power_level();
```

Description

This is function bt_hci_get_inquiry_response_tx_power_level.

bt_hci_init_ex Function

File

hci.h

C

```
void bt_hci_init_ex(bt_byte default_link_policy);
```

Description

brief Initialize the HCI layer. ingroup hci

This function initializes all internal variables of the HCI layer. The application, unless it's going to use only HCI layer, does not need to call this function as it is implicitly called by bt_sys_init_ex.

@param default_link_policy default link policy settings. This is a bitmask that defines the initial value of the link policy settings for all new BR/EDR connections. This value can be a combination of the following values: li HCI_LINK_POLICY_ENABLE_ROLE_SWITCH li HCI_LINK_POLICY_ENABLE_HOLD_MODE li HCI_LINK_POLICY_ENABLE_SNIFF_MODE li HCI_LINK_POLICY_ENABLE_PARK_STATE

To enable all settings pass HCI_LINK_POLICY_ENABLE_ALL.

bt_hci_reset Function

File

hci.h

C

```
bt_bool bt_hci_reset(bt_hci_cmd_callback_fp callback, void* callback_param);
```

Description

brief Reset controller. ingroup hci

details This function resets the BT controller.

param callback Completion callback. Called when the controller has been reset. param callback_param A pointer to arbitrary data to be passed to the c callback callback.

bt_hci_send_pin_code_ex Function

File

hci.h

C

```
bt_bool bt_hci_send_pin_code_ex(bt_bdaddr_p pbdaddr_remote, const char* pin, bt_hci_cmd_callback_fp
callback, void* callback_param);
```

Description

This is function bt_hci_send_pin_code_ex.

bt_hci_set_scan_ex Function

File

hci.h

C

```
bt_bool bt_hci_set_scan_ex(bt_bool discoverable, bt_ulong discoverable_period, bt_byte discoverable_mode,
bt_bool connectable, bt_ulong connectable_period, bt_hci_cmd_callback_fp cb, void* cb_param);
```

Description

This is function bt_hci_set_scan_ex.

bt_hci_write_link_policy_settings Function

File

hci.h

C

```
bt_bool bt_hci_write_link_policy_settings(bt_hci_conn_state_t* conn, bt_uint settings,
bt_hci_cmd_callback_fp cb, void* cb_param);
```

Description

This is function bt_hci_write_link_policy_settings.

bt_hci_write_local_name_ex Function

File

hci.h

C

```
bt_bool bt_hci_write_local_name_ex(const char * device_name, bt_hci_cmd_callback_fp cb, void*
callback_param);
```

Description

This is function bt_hci_write_local_name_ex.

HCI Data Types and Constants

bt_hci_add_param_hconn Macro

File

hci.h

C

```
#define bt_hci_add_param_hconn(pcmd, value) \
    bt_hci_add_param_int(pcmd, value)
```

Description

brief Add connection handle parameter to an HCI command ingroup hci

bt_hci_add_param_lap Macro

File

hci.h

C

```
#define bt_hci_add_param_lap(cmd, lap) bt_hci_add_param_cod(cmd, lap)
```

Description

This is macro bt_hci_add_param_lap.

bt_hci_add_param_uint Macro

File

hci.h

C

```
#define bt_hci_add_param_uint(cmd, value) bt_hci_add_param_int(cmd, (bt_int)(value))
```

Description

brief Add unsigned int parameter to an HCI command ingroup hci

bt_hci_add_param_ulong Macro

File

hci.h

C

```
#define bt_hci_add_param_ulong(cmd, value) bt_hci_add_param_long(cmd, (bt_long)(value))
```

Description

brief Add unsigned long parameter to an HCI command ingroup hci

bt_hci_authenticate Macro

File

hci.h

C

```
#define bt_hci_authenticate(pconn, cb) bt_hci_authenticate_ex(pconn, cb, NULL)
```

Description

This is macro bt_hci_authenticate.

bt_hci_exit_sniff_mode Macro

File

hci.h

C

```
#define bt_hci_exit_sniff_mode(pconn, cb) bt_hci_exit_sniff_mode_ex(pconn, cb, NULL)
```

Description

brief Cancel sniff mode

bt_hci_get_evt_param_hconn Macro

File

hci.h

C

```
#define bt_hci_get_evt_param_hconn(pevt, pvalue, poffset) \
    bt_hci_get_evt_param_int(pevt, pvalue, poffset)
```

Description

brief Get connection handle parameter from HCI event ingroup hci

bt_hci_get_evt_param_uint Macro

File

hci.h

C

```
#define bt_hci_get_evt_param_uint(pevt, pvalue, poffset) bt_hci_get_evt_param_int(pevt, (bt_int*)pvalue,
poffset)
```

Description

This is macro bt_hci_get_evt_param_uint.

bt_hci_get_evt_param_ulong Macro

File

hci.h

C

```
#define bt_hci_get_evt_param_ulong(pevt, pvalue, poffset) bt_hci_get_evt_param_long(pevt, (bt_long*)pvalue,
poffset)
```

Description

This is macro bt_hci_get_evt_param_ulong.

bt_hci_get_param_hconn Macro

File

hci.h

C

```
#define bt_hci_get_param_hconn(pcmd, pvalue, poffset) bt_hci_get_param_int(pcmd, pvalue, poffset)
```

Description

brief Get connection handle parameter from HCI command ingroup hci

bt_hci_le_set_adevrtising_enable Macro

File

hci_le.h

C

```
#define bt_hci_le_set_adevrtising_enable(adv_enable, cb) bt_hci_le_set_adevrtising_enable_ex(adv_enable,
cb, NULL)
```

Description

This is macro bt_hci_le_set_adevrtising_enable.

bt_hci_role_change Macro

File

hci.h

C

```
#define bt_hci_role_change(pbdaddr, role, cb) bt_hci_role_change_ex(pbdaddr, role, cb, NULL)
```

Description

This is macro bt_hci_role_change.

bt_hci_set_encryption Macro

File

hci.h

C

```
#define bt_hci_set_encryption(pconn, encryption_enable, cb) bt_hci_set_encryption_ex(pconn,
encryption_enable, cb, NULL)
```

Description

This is macro bt_hci_set_encryption.

bt_hci_sniff_mode Macro

File

hci.h

C

```
#define bt_hci_sniff_mode(pconn, min_interval, max_interval, attempt_slots, timeout, cb) \
    bt_hci_sniff_mode_ex(pconn, min_interval, max_interval, attempt_slots, timeout, cb, NULL)
```

Description

brief Put local device to sniff mode

bt_hci_sniff_subrating Macro

File

hci.h

C

```
#define bt_hci_sniff_subrating(pconn, max_latency, min_remote_timeout, min_local_timeout, cb) \
    bt_hci_sniff_subrating_ex(pconn, max_latency, min_remote_timeout, min_local_timeout, cb, NULL)
```

Description

This is macro bt_hci_sniff_subrating.

bt_hcitr_3wire_init Macro

File

hcitr_3wire.h

C

```
#define bt_hcitr_3wire_init bt_hcitr_3wire_init_ex(HCITR_3WIRE_DEFAULT_ACK_TIMEOUT, BT_FALSE)
```

Description

This is macro bt_hcitr_3wire_init.

bt_hcitr_3wire_reset Macro

File

hcitr_3wire.h

C

```
#define bt_hcitr_3wire_reset bt_hcitr_3wire_reset_ex(HCITR_3WIRE_DEFAULT_ACK_TIMEOUT, BT_FALSE)
```

Description

This is macro bt_hcitr_3wire_reset.

bt_hcitr_bcsp_init Macro

File

hcitr_bcsp.h

C

```
#define bt_hcitr_bcsp_init bt_hcitr_bcsp_init_ex(HCITR_BCSP_DEFAULT_ACK_TIMEOUT)
```

Description

This is macro bt_hcitr_bcsp_init.

bt_hcitr_bcsp_reset Macro

File

hcitr_bcsp.h

C

```
#define bt_hcitr_bcsp_reset bt_hcitr_bcsp_reset_ex(HCITR_BCSP_DEFAULT_ACK_TIMEOUT)
```

Description

This is macro bt_hcitr_bcsp_reset.

COD_MAJOR_AUDIO Macro

File

baseband.h

C

```
#define COD_MAJOR_AUDIO 0x0400
```

Description

This is macro COD_MAJOR_AUDIO.

COD_MAJOR_COMPUTER Macro

File

baseband.h

C

```
#define COD_MAJOR_COMPUTER 0x0100
```

Description

This is macro COD_MAJOR_COMPUTER.

COD_MAJOR_HEALTH Macro

File

baseband.h

C

```
#define COD_MAJOR_HEALTH 0x0900
```

Description

This is macro COD_MAJOR_HEALTH.

COD_MAJOR_IMAGING Macro

File

baseband.h

C

```
#define COD_MAJOR_IMAGING 0x0600
```

Description

This is macro COD_MAJOR_IMAGING.

COD_MAJOR_MISC Macro

File

baseband.h

C

```
#define COD_MAJOR_MISC 0x0000
```

Description

This is macro COD_MAJOR_MISC.

COD_MAJOR_NET_ACCESS_POINT Macro

File

baseband.h

C

```
#define COD_MAJOR_NET_ACCESS_POINT 0x0300
```

Description

This is macro COD_MAJOR_NET_ACCESS_POINT.

COD_MAJOR_PERIPHERAL Macro

File

baseband.h

C

```
#define COD_MAJOR_PERIPHERAL 0x0500
```

Description

This is macro COD_MAJOR_PERIPHERAL.

COD_MAJOR_PHONE Macro

File

baseband.h

C

```
#define COD_MAJOR_PHONE 0x0200
```

Description

This is macro COD_MAJOR_PHONE.

COD_MAJOR_TOY Macro

File

baseband.h

C

```
#define COD_MAJOR_TOY 0x0800
```

Description

This is macro COD_MAJOR_TOY.

COD_MAJOR_UNCATEGORIZED Macro

File

baseband.h

C

```
#define COD_MAJOR_UNCATEGORIZED 0x1f00
```

Description

This is macro COD_MAJOR_UNCATEGORIZED.

COD_MAJOR_WEARABLE Macro

File

baseband.h

C

```
#define COD_MAJOR_WEARABLE 0x0700
```

Description

This is macro COD_MAJOR_WEARABLE.

COD_MINOR_AV_CAMCORDER Macro

File

baseband.h

C

```
#define COD_MINOR_AV_CAMCORDER (0x0d << 2)
```

Description

This is macro COD_MINOR_AV_CAMCORDER.

COD_MINOR_AV_CAR_AUDIO Macro

File

baseband.h

C

```
#define COD_MINOR_AV_CAR_AUDIO (0x08 << 2)
```

Description

This is macro COD_MINOR_AV_CAR_AUDIO.

COD_MINOR_AV_GAMING Macro

File

baseband.h

C

```
#define COD_MINOR_AV_GAMING (0x12 << 2)
```

Description

This is macro COD_MINOR_AV_GAMING.

COD_MINOR_AV_HANDSFREE Macro

File

baseband.h

C

```
#define COD_MINOR_AV_HANDSFREE (0x02 << 2)
```

Description

This is macro COD_MINOR_AV_HANDSFREE.

COD_MINOR_AV_HEADPHONES Macro

File

baseband.h

C

```
#define COD_MINOR_AV_HEADPHONES (0x06 << 2)
```

Description

This is macro COD_MINOR_AV_HEADPHONES.

COD_MINOR_AV_HEADSET Macro

File

baseband.h

C

```
#define COD_MINOR_AV_HEADSET (0x01 << 2)
```

Description

This is macro COD_MINOR_AV_HEADSET.

COD_MINOR_AV_HIFI_AUDIO Macro

File

baseband.h

C

```
#define COD_MINOR_AV_HIFI_AUDIO (0x0a << 2)
```

Description

This is macro COD_MINOR_AV_HIFI_AUDIO.

COD_MINOR_AV_LOUDSPEAKER Macro

File

baseband.h

C

```
#define COD_MINOR_AV_LOUDSPEAKER (0x05 << 2)
```

Description

This is macro COD_MINOR_AV_LOUDSPEAKER.

COD_MINOR_AV_MICROPHONE Macro

File

baseband.h

C

```
#define COD_MINOR_AV_MICROPHONE (0x04 << 2)
```

Description

This is macro COD_MINOR_AV_MICROPHONE.

COD_MINOR_AV_PORTABLE_AUDIO Macro

File

baseband.h

C

```
#define COD_MINOR_AV_PORTABLE_AUDIO (0x07 << 2)
```

Description

This is macro COD_MINOR_AV_PORTABLE_AUDIO.

COD_MINOR_AV_RESERVED Macro

File

baseband.h

C

```
#define COD_MINOR_AV_RESERVED (0x03 << 2)
```

Description

This is macro COD_MINOR_AV_RESERVED.

COD_MINOR_AV_RESERVERD2 Macro

File

baseband.h

C

```
#define COD_MINOR_AV_RESERVERD2 (0x11 << 2)
```

Description

This is macro COD_MINOR_AV_RESERVERD2.

COD_MINOR_AV_SET_TOP_BOX Macro

File

baseband.h

C

```
#define COD_MINOR_AV_SET_TOP_BOX (0x09 << 2)
```

Description

This is macro COD_MINOR_AV_SET_TOP_BOX.

COD_MINOR_AV_UNCATEGORIZED Macro

File

baseband.h

C

```
#define COD_MINOR_AV_UNCATEGORIZED 0x00
```

Description

This is macro COD_MINOR_AV_UNCATEGORIZED.

COD_MINOR_AV_VCR Macro

File

baseband.h

C

```
#define COD_MINOR_AV_VCR (0x0b << 2)
```

Description

This is macro COD_MINOR_AV_VCR.

COD_MINOR_AV_VIDE_DISPLAY_AND_LOUDSPEAKER Macro

File

baseband.h

C

```
#define COD_MINOR_AV_VIDE_DISPLAY_AND_LOUDSPEAKER (0x0f << 2)
```

Description

This is macro COD_MINOR_AV_VIDE_DISPLAY_AND_LOUDSPEAKER.

COD_MINOR_AV_VIDEO_CAMERA Macro

File

baseband.h

C

```
#define COD_MINOR_AV_VIDEO_CAMERA (0x0c << 2)
```

Description

This is macro COD_MINOR_AV_VIDEO_CAMERA.

COD_MINOR_AV_VIDEO_CONFERENCING Macro

File

baseband.h

C

```
#define COD_MINOR_AV_VIDEO_CONFERENCING (0x10 << 2)
```

Description

This is macro COD_MINOR_AV_VIDEO_CONFERENCING.

COD_MINOR_AV_VIDEO_MONITOR Macro

File

baseband.h

C

```
#define COD_MINOR_AV_VIDEO_MONITOR (0x0e << 2)
```

Description

This is macro COD_MINOR_AV_VIDEO_MONITOR.

COD_MINOR_COMPUTER_DESKTOP Macro

File

baseband.h

C

```
#define COD_MINOR_COMPUTER_DESKTOP (0x01 << 2)
```

Description

This is macro COD_MINOR_COMPUTER_DESKTOP.

COD_MINOR_COMPUTER_HANDHELD Macro

File

baseband.h

C

```
#define COD_MINOR_COMPUTER_HANDHELD (0x04 << 2)
```

Description

This is macro COD_MINOR_COMPUTER_HANDHELD.

COD_MINOR_COMPUTER_LAPTOP Macro

File

baseband.h

C

```
#define COD_MINOR_COMPUTER_LAPTOP (0x03 << 2)
```

Description

This is macro COD_MINOR_COMPUTER_LAPTOP.

COD_MINOR_COMPUTER_PALMSIZED Macro

File

baseband.h

C

```
#define COD_MINOR_COMPUTER_PALMSIZED (0x05 << 2)
```

Description

This is macro COD_MINOR_COMPUTER_PALMSIZED.

COD_MINOR_COMPUTER_SERVER Macro

File

baseband.h

C

```
#define COD_MINOR_COMPUTER_SERVER (0x02 << 2)
```

Description

This is macro COD_MINOR_COMPUTER_SERVER.

COD_MINOR_COMPUTER_UNCATEGORIZED Macro

File

baseband.h

C

```
#define COD_MINOR_COMPUTER_UNCATEGORIZED 0x00
```

Description

This is macro COD_MINOR_COMPUTER_UNCATEGORIZED.

COD_MINOR_COMPUTER_WEARABLE Macro

File

baseband.h

C

```
#define COD_MINOR_COMPUTER_WEARABLE (0x06 << 2)
```

Description

This is macro COD_MINOR_COMPUTER_WEARABLE.

COD_MINOR_HEALTH_BPM Macro

File

baseband.h

C

```
#define COD_MINOR_HEALTH_BPM (0x01 << 2)
```

Description

This is macro COD_MINOR_HEALTH_BPM.

COD_MINOR_HEALTH_DATA_DISPLAY Macro

File

baseband.h

C

```
#define COD_MINOR_HEALTH_DATA_DISPLAY (0x07 << 2)
```

Description

This is macro COD_MINOR_HEALTH_DATA_DISPLAY.

COD_MINOR_HEALTH_GLUCOSE_METER Macro

File

baseband.h

C

```
#define COD_MINOR_HEALTH_GLUCOSE_METER (0x04 << 2)
```

Description

This is macro COD_MINOR_HEALTH_GLUCOSE_METER.

COD_MINOR_HEALTH_HEART_MONITOR Macro

File

baseband.h

C

```
#define COD_MINOR_HEALTH_HEART_MONITOR (0x06 << 2)
```

Description

This is macro COD_MINOR_HEALTH_HEART_MONITOR.

COD_MINOR_HEALTH_PULSE_OXIMETER Macro

File

baseband.h

C

```
#define COD_MINOR_HEALTH_PULSE_OXIMETER (0x05 << 2)
```

Description

This is macro COD_MINOR_HEALTH_PULSE_OXIMETER.

COD_MINOR_HEALTH_THERMOMETER Macro

File

baseband.h

C

```
#define COD_MINOR_HEALTH_THERMOMETER (0x02 << 2)
```

Description

This is macro COD_MINOR_HEALTH_THERMOMETER.

COD_MINOR_HEALTH_UNCATEGORIZED Macro

File

baseband.h

C

```
#define COD_MINOR_HEALTH_UNCATEGORIZED 0x00
```

Description

This is macro COD_MINOR_HEALTH_UNCATEGORIZED.

COD_MINOR_HEALTH_WEIGHING_SCALE Macro

File

baseband.h

C

```
#define COD_MINOR_HEALTH_WEIGHING_SCALE (0x03 << 2)
```

Description

This is macro COD_MINOR_HEALTH_WEIGHING_SCALE.

COD_MINOR_IMAGING_CAMERA Macro

File

baseband.h

C

```
#define COD_MINOR_IMAGING_CAMERA (0x01 << 5)
```

Description

This is macro COD_MINOR_IMAGING_CAMERA.

COD_MINOR_IMAGING_DISPLAY Macro

File

baseband.h

C

```
#define COD_MINOR_IMAGING_DISPLAY (0x01 << 4)
```

Description

This is macro COD_MINOR_IMAGING_DISPLAY.

COD_MINOR_IMAGING_PRINTER Macro

File

baseband.h

C

```
#define COD_MINOR_IMAGING_PRINTER (0x01 << 7)
```

Description

This is macro COD_MINOR_IMAGING_PRINTER.

COD_MINOR_IMAGING_SCANNER Macro

File

baseband.h

C

```
#define COD_MINOR_IMAGING_SCANNER (0x01 << 6)
```

Description

This is macro COD_MINOR_IMAGING_SCANNER.

COD_MINOR_IMAGING_UNCATEGORIZED Macro

File

baseband.h

C

```
#define COD_MINOR_IMAGING_UNCATEGORIZED 0x00
```

Description

This is macro COD_MINOR_IMAGING_UNCATEGORIZED.

COD_MINOR_LAN_01_17 Macro

File

baseband.h

C

```
#define COD_MINOR_LAN_01_17 (0x01 << 5)
```

Description

This is macro COD_MINOR_LAN_01_17.

COD_MINOR_LAN_17_33 Macro

File

baseband.h

C

```
#define COD_MINOR_LAN_17_33 (0x02 << 5)
```

Description

This is macro COD_MINOR_LAN_17_33.

COD_MINOR_LAN_33_50 Macro

File

baseband.h

C

```
#define COD_MINOR_LAN_33_50 (0x03 << 5)
```

Description

This is macro COD_MINOR_LAN_33_50.

COD_MINOR_LAN_50_67 Macro

File

baseband.h

C

```
#define COD_MINOR_LAN_50_67 (0x04 << 5)
```

Description

This is macro COD_MINOR_LAN_50_67.

COD_MINOR_LAN_67_83 Macro

File

baseband.h

C

```
#define COD_MINOR_LAN_67_83 (0x05 << 5)
```

Description

This is macro COD_MINOR_LAN_67_83.

COD_MINOR_LAN_83_99 Macro

File

baseband.h

C

```
#define COD_MINOR_LAN_83_99 (0x06 << 5)
```

Description

This is macro COD_MINOR_LAN_83_99.

COD_MINOR_LAN_FULLY_AVAILABLE Macro

File

baseband.h

C

```
#define COD_MINOR_LAN_FULLY_AVAILABLE 0x00
```

Description

This is macro COD_MINOR_LAN_FULLY_AVAILABLE.

COD_MINOR_LAN_NO_SERVICE Macro

File

baseband.h

C

```
#define COD_MINOR_LAN_NO_SERVICE (0x07 << 5)
```

Description

This is macro COD_MINOR_LAN_NO_SERVICE.

COD_MINOR_LAN_UNCATEGORIZED Macro

File

baseband.h

C

```
#define COD_MINOR_LAN_UNCATEGORIZED 0x00
```

Description

This is macro COD_MINOR_LAN_UNCATEGORIZED.

COD_MINOR_PERIPHERAL_CARD_READER Macro

File

baseband.h

C

```
#define COD_MINOR_PERIPHERAL_CARD_READER (0x06 << 2)
```

Description

This is macro COD_MINOR_PERIPHERAL_CARD_READER.

COD_MINOR_PERIPHERAL_COMBO Macro

File

baseband.h

C

```
#define COD_MINOR_PERIPHERAL_COMBO 0xC0
```

Description

This is macro COD_MINOR_PERIPHERAL_COMBO.

COD_MINOR_PERIPHERAL_DIGITIZER Macro

File

baseband.h

C

```
#define COD_MINOR_PERIPHERAL_DIGITIZER (0x05 << 2)
```

Description

This is macro COD_MINOR_PERIPHERAL_DIGITIZER.

COD_MINOR_PERIPHERAL_GAMEPAD Macro

File

baseband.h

C

```
#define COD_MINOR_PERIPHERAL_GAMEPAD (0x02 << 2)
```

Description

This is macro COD_MINOR_PERIPHERAL_GAMEPAD.

COD_MINOR_PERIPHERAL_JOYSTICK Macro

File

baseband.h

C

```
#define COD_MINOR_PERIPHERAL_JOYSTICK (0x01 << 2)
```

Description

This is macro COD_MINOR_PERIPHERAL_JOYSTICK.

COD_MINOR_PERIPHERAL_KEYBOARD Macro

File

baseband.h

C

```
#define COD_MINOR_PERIPHERAL_KEYBOARD 0x40
```

Description

This is macro COD_MINOR_PERIPHERAL_KEYBOARD.

COD_MINOR_PERIPHERAL_MOUSE Macro

File

baseband.h

C

```
#define COD_MINOR_PERIPHERAL_MOUSE 0x80
```

Description

This is macro COD_MINOR_PERIPHERAL_MOUSE.

COD_MINOR_PERIPHERAL_OTHER Macro

File

baseband.h

C

```
#define COD_MINOR_PERIPHERAL_OTHER 0x00
```

Description

This is macro COD_MINOR_PERIPHERAL_OTHER.

COD_MINOR_PERIPHERAL_REMOTE Macro

File

baseband.h

C

```
#define COD_MINOR_PERIPHERAL_REMOTE (0x03 << 2)
```

Description

This is macro COD_MINOR_PERIPHERAL_REMOTE.

COD_MINOR_PERIPHERAL_SENSING Macro

File

baseband.h

C

```
#define COD_MINOR_PERIPHERAL_SENSING (0x04 << 2)
```

Description

This is macro COD_MINOR_PERIPHERAL_SENSING.

COD_MINOR_PERIPHERAL_UNCATEGORIZED Macro

File

baseband.h

C

```
#define COD_MINOR_PERIPHERAL_UNCATEGORIZED 0x00
```

Description

This is macro COD_MINOR_PERIPHERAL_UNCATEGORIZED.

COD_MINOR_PHONE_CELLULAR Macro

File

baseband.h

C

```
#define COD_MINOR_PHONE_CELLULAR (0x01 << 2)
```

Description

This is macro COD_MINOR_PHONE_CELLULAR.

COD_MINOR_PHONE_CORDLESS Macro

File

baseband.h

C

```
#define COD_MINOR_PHONE_CORDLESS (0x02 << 2)
```

Description

This is macro COD_MINOR_PHONE_CORDLESS.

COD_MINOR_PHONE_ISDN Macro

File

baseband.h

C

```
#define COD_MINOR_PHONE_ISDN (0x05 << 2)
```

Description

This is macro COD_MINOR_PHONE_ISDN.

COD_MINOR_PHONE_SMART Macro

File

baseband.h

C

```
#define COD_MINOR_PHONE_SMART (0x03 << 2)
```

Description

This is macro COD_MINOR_PHONE_SMART.

COD_MINOR_PHONE_UNCATEGORIZED Macro

File

baseband.h

C

```
#define COD_MINOR_PHONE_UNCATEGORIZED 0x00
```

Description

This is macro COD_MINOR_PHONE_UNCATEGORIZED.

COD_MINOR_PHONE_WIREDMODEM Macro

File

baseband.h

C

```
#define COD_MINOR_PHONE_WIREDMODEM (0x04 << 2)
```

Description

This is macro COD_MINOR_PHONE_WIREDMODEM.

COD_MINOR_TOY_CONTROLLER Macro

File

baseband.h

C

```
#define COD_MINOR_TOY_CONTROLLER (0x04 << 2)
```

Description

This is macro COD_MINOR_TOY_CONTROLLER.

COD_MINOR_TOY_DOLL Macro

File

baseband.h

C

```
#define COD_MINOR_TOY_DOLL (0x03 << 2)
```

Description

This is macro COD_MINOR_TOY_DOLL.

COD_MINOR_TOY_GAME Macro

File

baseband.h

C

```
#define COD_MINOR_TOY_GAME (0x05 << 2)
```

Description

This is macro COD_MINOR_TOY_GAME.

COD_MINOR_TOY_ROBOT Macro

File

baseband.h

C

```
#define COD_MINOR_TOY_ROBOT (0x01 << 2)
```

Description

This is macro COD_MINOR_TOY_ROBOT.

COD_MINOR_TOY_UNCATEGORIZED Macro

File

baseband.h

C

```
#define COD_MINOR_TOY_UNCATEGORIZED 0x00
```

Description

This is macro COD_MINOR_TOY_UNCATEGORIZED.

COD_MINOR_TOY_VEHICLE Macro

File

baseband.h

C

```
#define COD_MINOR_TOY_VEHICLE (0x02 << 2)
```

Description

This is macro COD_MINOR_TOY_VEHICLE.

COD_MINOR_WEARABLE_GLASSES Macro

File

baseband.h

C

```
#define COD_MINOR_WEARABLE_GLASSES (0x05 << 2)
```

Description

This is macro COD_MINOR_WEARABLE_GLASSES.

COD_MINOR_WEARABLE_HELMET Macro

File

baseband.h

C

```
#define COD_MINOR_WEARABLE_HELMET (0x04 << 2)
```

Description

This is macro COD_MINOR_WEARABLE_HELMET.

COD_MINOR_WEARABLE_JACKET Macro

File

baseband.h

C

```
#define COD_MINOR_WEARABLE_JACKET (0x03 << 2)
```

Description

This is macro COD_MINOR_WEARABLE_JACKET.

COD_MINOR_WEARABLE_PAGER Macro

File

baseband.h

C

```
#define COD_MINOR_WEARABLE_PAGER (0x02 << 2)
```

Description

This is macro COD_MINOR_WEARABLE_PAGER.

COD_MINOR_WEARABLE_UNCATEGORIZED Macro

File

baseband.h

C

```
#define COD_MINOR_WEARABLE_UNCATEGORIZED 0x00
```

Description

This is macro COD_MINOR_WEARABLE_UNCATEGORIZED.

COD_MINOR_WEARABLE_WATCH Macro

File

baseband.h

C

```
#define COD_MINOR_WEARABLE_WATCH (0x01 << 2)
```

Description

This is macro COD_MINOR_WEARABLE_WATCH.

COS_AUDIO Macro

File

baseband.h

C

```
#define COS_AUDIO 0x200000
```

Description

This is macro COS_AUDIO.

COS_CAPTURING Macro

File

baseband.h

C

```
#define COS_CAPTURING 0x080000
```

Description

This is macro COS_CAPTURING.

COS_INFORMATION Macro

File

baseband.h

C

```
#define COS_INFORMATION 0x800000
```

Description

This is macro COS_INFORMATION.

COS_LIMITED_DISCOVERABLE_MODE Macro

File

baseband.h

C

```
#define COS_LIMITED_DISCOVERABLE_MODE 0x002000
```

Description

This is macro COS_LIMITED_DISCOVERABLE_MODE.

COS_NETWORKING Macro

File

baseband.h

C

```
#define COS_NETWORKING 0x020000
```

Description

This is macro COS_NETWORKING.

COS_OBJECTTRANSFER Macro

File

baseband.h

C

```
#define COS_OBJECTTRANSFER 0x100000
```

Description

This is macro COS_OBJECTTRANSFER.

COS_POSITIONING Macro

File

baseband.h

C

```
#define COS_POSITIONING 0x010000
```

Description

This is macro COS_POSITIONING.

COS_RENDERING Macro

File

baseband.h

C

```
#define COS_RENDERING 0x040000
```

Description

This is macro COS_RENDERING.

COS_TELEPHONY Macro

File

baseband.h

C

```
#define COS_TELEPHONY 0x400000
```

Description

This is macro COS_TELEPHONY.

HCI_ACCEPT_CONNECTION_REQUEST Macro

File

hci.h

C

```
#define HCI_ACCEPT_CONNECTION_REQUEST HCI_OPCODE(OGF_LINK_CONTROL, 0x0009)
```

Description

This is macro HCI_ACCEPT_CONNECTION_REQUEST.

HCI_ACCEPT_SYNCH_CONNECTION_REQUEST Macro

File

hci.h

C

```
#define HCI_ACCEPT_SYNCH_CONNECTION_REQUEST HCI_OPCODE(OGF_LINK_CONTROL, 0x0029)
```

Description

This is macro HCI_ACCEPT_SYNCH_CONNECTION_REQUEST.

HCI_ACL_DATA_HEADER_LEN Macro

File

hci.h

C

```
#define HCI_ACL_DATA_HEADER_LEN 4
```

Description

This is macro HCI_ACL_DATA_HEADER_LEN.

HCI_AUTHENTICATION_REQUESTED Macro

File

hci.h

C

```
#define HCI_AUTHENTICATION_REQUESTED HCI_OPCODE(OGF_LINK_CONTROL, 0x0011)
```

Description

This is macro HCI_AUTHENTICATION_REQUESTED.

HCI_BB_PACKET_TYPE_DH1 Macro

File

hci.h

C

```
#define HCI_BB_PACKET_TYPE_DH1 0x0010
```

Description

This is macro HCI_BB_PACKET_TYPE_DH1.

HCI_BB_PACKET_TYPE_DH3 Macro

File

hci.h

C

```
#define HCI_BB_PACKET_TYPE_DH3 0x0800
```

Description

This is macro HCI_BB_PACKET_TYPE_DH3.

HCI_BB_PACKET_TYPE_DH5 Macro

File

hci.h

C

```
#define HCI_BB_PACKET_TYPE_DH5 0x8000
```

Description

This is macro HCI_BB_PACKET_TYPE_DH5.

HCI_BB_PACKET_TYPE_DM1 Macro

File

hci.h

C

```
#define HCI_BB_PACKET_TYPE_DM1 0x0008
```

Description

This is macro HCI_BB_PACKET_TYPE_DM1.

HCI_BB_PACKET_TYPE_DM3 Macro

File

hci.h

C

```
#define HCI_BB_PACKET_TYPE_DM3 0x0400
```

Description

This is macro HCI_BB_PACKET_TYPE_DM3.

HCI_BB_PACKET_TYPE_DM5 Macro

File

hci.h

C

```
#define HCI_BB_PACKET_TYPE_DM5 0x4000
```

Description

This is macro HCI_BB_PACKET_TYPE_DM5.

HCI_BB_PACKET_TYPE_NO_2_DH1 Macro

File

hci.h

C

```
#define HCI_BB_PACKET_TYPE_NO_2_DH1 0x0002
```

Description

This is macro HCI_BB_PACKET_TYPE_NO_2_DH1.

HCI_BB_PACKET_TYPE_NO_2_DH3 Macro

File

hci.h

C

```
#define HCI_BB_PACKET_TYPE_NO_2_DH3 0x0100
```

Description

This is macro HCI_BB_PACKET_TYPE_NO_2_DH3.

HCI_BB_PACKET_TYPE_NO_2_DH5 Macro

File

hci.h

C

```
#define HCI_BB_PACKET_TYPE_NO_2_DH5 0x1000
```

Description

This is macro HCI_BB_PACKET_TYPE_NO_2_DH5.

HCI_BB_PACKET_TYPE_NO_3_DH1 Macro

File

hci.h

C

```
#define HCI_BB_PACKET_TYPE_NO_3_DH1 0x0004
```

Description

This is macro HCI_BB_PACKET_TYPE_NO_3_DH1.

HCI_BB_PACKET_TYPE_NO_3_DH3 Macro

File

hci.h

C

```
#define HCI_BB_PACKET_TYPE_NO_3_DH3 0x0200
```

Description

This is macro HCI_BB_PACKET_TYPE_NO_3_DH3.

HCI_BB_PACKET_TYPE_NO_3_DH5 Macro

File

hci.h

C

```
#define HCI_BB_PACKET_TYPE_NO_3_DH5 0x2000
```

Description

This is macro HCI_BB_PACKET_TYPE_NO_3_DH5.

HCI_C2H_BROADCAST_NOT_PARCKED_SLAVE Macro

File

hci.h

C

```
#define HCI_C2H_BROADCAST_NOT_PARCKED_SLAVE 0x1
```

Description

This is macro HCI_C2H_BROADCAST_NOT_PARCKED_SLAVE.

HCI_C2H_BROADCAST_P2P Macro

File

hci.h

C

```
#define HCI_C2H_BROADCAST_P2P 0x0
```

Description

This is macro HCI_C2H_BROADCAST_P2P.

HCI_C2H_BROADCAST_PARCKED_SLAVE Macro

File

hci.h

C

```
#define HCI_C2H_BROADCAST_PARCKED_SLAVE 0x2
```

Description

This is macro HCI_C2H_BROADCAST_PARCKED_SLAVE.

HCI_C2H_BROADCAST_RESERVED Macro

File

hci.h

C

```
#define HCI_C2H_BROADCAST_RESERVED 0x3
```

Description

This is macro HCI_C2H_BROADCAST_RESERVED.

HCI_CHANGE_CONNECTION_LINK_KEY Macro

File

hci.h

C

```
#define HCI_CHANGE_CONNECTION_LINK_KEY HCI_OPCODE(OGF_LINK_CONTROL, 0x0015)
```

Description

This is macro HCI_CHANGE_CONNECTION_LINK_KEY.

HCI_CHANGE_CONNECTION_PACKET_TYPE Macro

File

hci.h

C

```
#define HCI_CHANGE_CONNECTION_PACKET_TYPE HCI_OPCODE(OGF_LINK_CONTROL, 0x000F)
```

Description

This is macro HCI_CHANGE_CONNECTION_PACKET_TYPE.

HCI_CMD_HEADER_LEN Macro

File

hci.h

C

```
#define HCI_CMD_HEADER_LEN 3
```

Description

This is macro HCI_CMD_HEADER_LEN.

HCI_CMD_STATUS_BEING_SENT Macro

File

hci.h

C

```
#define HCI_CMD_STATUS_BEING_SENT 2
```

Description

This is macro HCI_CMD_STATUS_BEING_SENT.

HCI_CMD_STATUS_PENDING Macro

File

hci.h

C

```
#define HCI_CMD_STATUS_PENDING 0
```

Description

This is macro HCI_CMD_STATUS_PENDING.

HCI_CMD_STATUS_WAITING_RESPONSE Macro

File

hci.h

C

```
#define HCI_CMD_STATUS_WAITING_RESPONSE 1
```

Description

This is macro HCI_CMD_STATUS_WAITING_RESPONSE.

HCI_CONFIG_BECOME_MASTER Macro

File

hci.h

C

```
#define HCI_CONFIG_BECOME_MASTER 0x04
```

Description

This is macro HCI_CONFIG_BECOME_MASTER.

HCI_CONFIG_ENABLE_AUTHENTICATION Macro

File

hci.h

C

```
#define HCI_CONFIG_ENABLE_AUTHENTICATION 0x01
```

Description

This is macro HCI_CONFIG_ENABLE_AUTHENTICATION.

HCI_CONFIG_ENABLE_ENCRYPTION Macro

File

hci.h

C

```
#define HCI_CONFIG_ENABLE_ENCRYPTION 0x02
```

Description

This is macro HCI_CONFIG_ENABLE_ENCRYPTION.

HCI_CONN_ROLE_MASTER Macro

File

hci_conn_state.h

C

```
#define HCI_CONN_ROLE_MASTER 0
```

Description

This is macro HCI_CONN_ROLE_MASTER.

HCI_CONN_ROLE_SLAVE Macro

File

hci_conn_state.h

C

```
#define HCI_CONN_ROLE_SLAVE 1
```

Description

This is macro HCI_CONN_ROLE_SLAVE.

HCI_CONN_STATE_AUTHENTICATING Macro

File

hci_conn_state.h

C

```
#define HCI_CONN_STATE_AUTHENTICATING 1
```

Description

This is macro HCI_CONN_STATE_AUTHENTICATING.

HCI_CONN_STATE_CLOSED Macro

File

hci_conn_state.h

C

```
#define HCI_CONN_STATE_CLOSED 0
```

Description

This is macro HCI_CONN_STATE_CLOSED.

HCI_CONN_STATE_OPEN Macro

File

hci_conn_state.h

C

```
#define HCI_CONN_STATE_OPEN 2
```

Description

This is macro HCI_CONN_STATE_OPEN.

HCI_CONN_TYPE_ACL Macro

File

hci_conn_state.h

C

```
#define HCI_CONN_TYPE_ACL 1
```

Description

This is macro HCI_CONN_TYPE_ACL.

HCI_CONN_TYPE_ESCO Macro

File

hci_conn_state.h

C

```
#define HCI_CONN_TYPE_ESCO 2
```

Description

This is macro HCI_CONN_TYPE_ESCO.

HCI_CONN_TYPE_SCO Macro

File

hci_conn_state.h

C

```
#define HCI_CONN_TYPE_SCO 0
```

Description

This is macro HCI_CONN_TYPE_SCO.

HCI_CONNECTABLE Macro

File

hci.h

C

```
#define HCI_CONNECTABLE 0x02
```

Description

This is macro HCI_CONNECTABLE.

HCI_CONTROLLER Macro

File

hci.h

C

```
#define HCI_CONTROLLER _phci_ctrl
```

Description

This is macro HCI_CONTROLLER.

HCI_CREATE_CONNECTION Macro

File

hci.h

C

```
#define HCI_CREATE_CONNECTION HCI_OPCODE(OGF_LINK_CONTROL, 0x0005)
```

Description

This is macro HCI_CREATE_CONNECTION.

HCI_CREATE_CONNECTION_CANCEL Macro

File

hci.h

C

```
#define HCI_CREATE_CONNECTION_CANCEL HCI_OPCODE(OGF_LINK_CONTROL, 0x0008)
```

Description

This is macro HCI_CREATE_CONNECTION_CANCEL.

HCI_CREATE_NEW_UNIT_KEY Macro

File

hci.h

C

```
#define HCI_CREATE_NEW_UNIT_KEY HCI_OPCODE(OGF_CTRL_BASEBAND, 0x000B)
```

Description

This is macro HCI_CREATE_NEW_UNIT_KEY.

HCI_CTRL_LISTENER_ACL_DATA Macro

File

hci_ctrl_state.h

C

```
#define HCI_CTRL_LISTENER_ACL_DATA 1
```

Description

This is macro HCI_CTRL_LISTENER_ACL_DATA.

HCI_CTRL_LISTENER_EVENT Macro

File

hci_ctrl_state.h

C

```
#define HCI_CTRL_LISTENER_EVENT 0
```

Description

This is macro HCI_CTRL_LISTENER_EVENT.

HCI_CTRL_LISTENER_SCO_DATA Macro

File

hci_ctrl_state.h

C

```
#define HCI_CTRL_LISTENER_SCO_DATA 2
```

Description

This is macro HCI_CTRL_LISTENER_SCO_DATA.

HCI_CTRL_STATE_CLOSED Macro

File

hci_ctrl_state.h

C

```
#define HCI_CTRL_STATE_CLOSED 0x00
```

Description

This is macro HCI_CTRL_STATE_CLOSED.

HCI_CTRL_STATE_CONNECTABLE Macro

File

hci_ctrl_state.h

C

```
#define HCI_CTRL_STATE_CONNECTABLE 0x20
```

Description

This is macro HCI_CTRL_STATE_CONNECTABLE.

HCI_CTRL_STATE_DISCOVERABLE Macro

File

hci_ctrl_state.h

C

```
#define HCI_CTRL_STATE_DISCOVERABLE 0x10
```

Description

This is macro HCI_CTRL_STATE_DISCOVERABLE.

HCI_CTRL_STATE_INIT Macro

File

hci_ctrl_state.h

C

```
#define HCI_CTRL_STATE_INIT 0x01
```

Description

This is macro HCI_CTRL_STATE_INIT.

HCI_CTRL_STATE_READY Macro

File

hci_ctrl_state.h

C

```
#define HCI_CTRL_STATE_READY 0x02
```

Description

This is macro HCI_CTRL_STATE_READY.

HCI_CTRL_STATE_SLEEP Macro

File

hci_ctrl_state.h

C

```
#define HCI_CTRL_STATE_SLEEP 0x04
```

Description

This is macro HCI_CTRL_STATE_SLEEP.

HCI_CTRL_STATE_WAKING_UP Macro

File

hci_ctrl_state.h

C

```
#define HCI_CTRL_STATE_WAKING_UP 0x08
```

Description

This is macro HCI_CTRL_STATE_WAKING_UP.

HCI_DATA_BUFFER_STATE_FREE Macro

File

hci_data_buffer.h

C

```
#define HCI_DATA_BUFFER_STATE_FREE 0
```

Description

This is macro HCI_DATA_BUFFER_STATE_FREE.

HCI_DATA_BUFFER_STATE_USED Macro

File

hci_data_buffer.h

C

```
#define HCI_DATA_BUFFER_STATE_USED 1
```

Description

This is macro HCI_DATA_BUFFER_STATE_USED.

HCI_DATA_STATUS_BEING_SENT Macro

File

hci.h

C

```
#define HCI_DATA_STATUS_BEING_SENT 2
```

Description

This is macro HCI_DATA_STATUS_BEING_SENT.

HCI_DATA_STATUS_PENDING Macro

File

hci.h

C

```
#define HCI_DATA_STATUS_PENDING 0
```

Description

This is macro HCI_DATA_STATUS_PENDING.

HCI_DEFAULT_ACL_CONFIG Macro

File

hci.h

C

```
#define HCI_DEFAULT_ACL_CONFIG 0
```

Description

Default value for acl_config parameter of bt_hci_connect()

HCI_DELETE_STORED_LINK_KEY Macro

File

hci.h

C

```
#define HCI_DELETE_STORED_LINK_KEY HCI_OPCODE(OGF_CTRL_BASEBAND, 0x0012)
```

Description

This is macro HCI_DELETE_STORED_LINK_KEY.

HCI_DISCONNECT Macro

File

hci.h

C

```
#define HCI_DISCONNECT HCI_OPCODE(OGF_LINK_CONTROL, 0x0006)
```

Description

This is macro HCI_DISCONNECT.

HCI_DISCOVERABLE Macro

File

hci.h

C

```
#define HCI_DISCOVERABLE 0x01
```

Description

This is macro HCI_DISCOVERABLE.

HCI_EIR_FEC_NOT_REQUIRED Macro

File

hci_eir.h

C

```
#define HCI_EIR_FEC_NOT_REQUIRED 0
```

Description

This is macro HCI_EIR_FEC_NOT_REQUIRED.

HCI_EIR_FEC_REQUIRED Macro

File

hci_eir.h

C

```
#define HCI_EIR_FEC_REQUIRED 1
```

Description

This is macro HCI_EIR_FEC_REQUIRED.

HCI_EIR_TYPE_DEVICE_ID Macro

File

hci_eir.h

C

```
#define HCI_EIR_TYPE_DEVICE_ID 0x10
```

Description

This is macro HCI_EIR_TYPE_DEVICE_ID.

HCI_EIR_TYPE_FLAGS Macro

File

hci_eir.h

C

```
#define HCI_EIR_TYPE_FLAGS 0x01
```

Description

This is macro HCI_EIR_TYPE_FLAGS.

HCI_EIR_TYPE_LOCAL_NAME_COMPLETE Macro

File

hci_eir.h

C

```
#define HCI_EIR_TYPE_LOCAL_NAME_COMPLETE 0x09
```

Description

This is macro HCI_EIR_TYPE_LOCAL_NAME_COMPLETE.

HCI_EIR_TYPE_LOCAL_NAME_SHORTENED Macro

File

hci_eir.h

C

```
#define HCI_EIR_TYPE_LOCAL_NAME_SHORTENED 0x08
```

Description

This is macro HCI_EIR_TYPE_LOCAL_NAME_SHORTENED.

HCI_EIR_TYPE_MANUFACTURER_SPECIFIC Macro

File

hci_eir.h

C

```
#define HCI_EIR_TYPE_MANUFACTURER_SPECIFIC 0xFF
```

Description

This is macro HCI_EIR_TYPE_MANUFACTURER_SPECIFIC.

HCI_EIR_TYPE_OOB_COD Macro

File

hci_eir.h

C

```
#define HCI_EIR_TYPE_OOB_COD 0x0d
```

Description

This is macro HCI_EIR_TYPE_OOB_COD.

HCI_EIR_TYPE_OOB_HASH Macro

File

hci_eir.h

C

```
#define HCI_EIR_TYPE_OOB_HASH 0x0e
```

Description

This is macro HCI_EIR_TYPE_OOB_HASH.

HCI_EIR_TYPE_OOB_RANDOMIZER Macro

File

hci_eir.h

C

```
#define HCI_EIR_TYPE_OOB_RANDOMIZER 0x0f
```

Description

This is macro HCI_EIR_TYPE_OOB_RANDOMIZER.

HCI_EIR_TYPE_TX_POWER_LEVEL Macro

File

hci_eir.h

C

```
#define HCI_EIR_TYPE_TX_POWER_LEVEL 0x0a
```

Description

This is macro HCI_EIR_TYPE_TX_POWER_LEVEL.

HCI_EIR_TYPE_UUID128_LIST_COMPLETE Macro

File

hci_eir.h

C

```
#define HCI_EIR_TYPE_UUID128_LIST_COMPLETE 0x07
```

Description

This is macro HCI_EIR_TYPE_UUID128_LIST_COMPLETE.

HCI_EIR_TYPE_UUID128_LIST_MORE_AVAILABLE Macro

File

hci_eir.h

C

```
#define HCI_EIR_TYPE_UUID128_LIST_MORE_AVAILABLE 0x06
```

Description

This is macro HCI_EIR_TYPE_UUID128_LIST_MORE_AVAILABLE.

HCI_EIR_TYPE_UUID16_LIST_COMPLETE Macro

File

hci_eir.h

C

```
#define HCI_EIR_TYPE_UUID16_LIST_COMPLETE 0x03
```

Description

This is macro HCI_EIR_TYPE_UUID16_LIST_COMPLETE.

HCI_EIR_TYPE_UUID16_LIST_MORE_AVAILABLE Macro

File

hci_eir.h

C

```
#define HCI_EIR_TYPE_UUID16_LIST_MORE_AVAILABLE 0x02
```

Description

This is macro HCI_EIR_TYPE_UUID16_LIST_MORE_AVAILABLE.

HCI_EIR_TYPE_UUID32_LIST_COMPLETE Macro

File

hci_eir.h

C

```
#define HCI_EIR_TYPE_UUID32_LIST_COMPLETE 0x05
```

Description

This is macro HCI_EIR_TYPE_UUID32_LIST_COMPLETE.

HCI_EIR_TYPE_UUID32_LIST_MORE_AVAILABLE Macro

File

hci_eir.h

C

```
#define HCI_EIR_TYPE_UUID32_LIST_MORE_AVAILABLE 0x04
```

Description

This is macro HCI_EIR_TYPE_UUID32_LIST_MORE_AVAILABLE.

HCI_ENABLE_DEVICE_UNDER_TEST_MODE Macro

File

hci.h

C

```
#define HCI_ENABLE_DEVICE_UNDER_TEST_MODE HCI_OPCODE(OGF_TESTING, 0x0003)
```

Description

This is macro HCI_ENABLE_DEVICE_UNDER_TEST_MODE.

HCI_ENCRYPTION_OFF Macro

File

hci.h

C

```
#define HCI_ENCRYPTION_OFF 0x00
```

Description

This is macro HCI_ENCRYPTION_OFF.

HCI_ENCRYPTION_ON Macro

File

hci.h

C

```
#define HCI_ENCRYPTION_ON 0x01
```

Description

This is macro HCI_ENCRYPTION_ON.

HCI_ENHANCED_FLUSH Macro

File

hci.h

C

```
#define HCI_ENHANCED_FLUSH HCI_OPCODE(OGF_CTRL_BASEBAND, 0x005F)
```

Description

This is macro HCI_ENHANCED_FLUSH.

HCI_ERR_ACL_CONN_ALREADY_EXISTS Macro

File

hci_errors.h

C

```
#define HCI_ERR_ACL_CONN_ALREADY_EXISTS 0x0b
```

Description

This is macro HCI_ERR_ACL_CONN_ALREADY_EXISTS.

HCI_ERR_AUTHENTICATION_FAILURE Macro

File

hci_errors.h

C

```
#define HCI_ERR_AUTHENTICATION_FAILURE 0x05
```

Description

This is macro HCI_ERR_AUTHENTICATION_FAILURE.

HCI_ERR_CONN_REJECT_LIMITED_RESOURCES Macro

File

hci_errors.h

C

```
#define HCI_ERR_CONN_REJECT_LIMITED_RESOURCES 0x0d
```

Description

This is macro HCI_ERR_CONN_REJECT_LIMITED_RESOURCES.

HCI_ERR_INVALID_PARAMETERS Macro

File

hci_errors.h

C

```
#define HCI_ERR_INVALID_PARAMETERS 0x12
```

Description

This is macro HCI_ERR_INVALID_PARAMETERS.

HCI_ERR_MEMORY_CAPACITY_EXCEEDED Macro

File

hci_errors.h

C

```
#define HCI_ERR_MEMORY_CAPACITY_EXCEEDED 0x07
```

Description

This is macro HCI_ERR_MEMORY_CAPACITY_EXCEEDED.

HCI_ERR_SCO_CONN_LIMIT_EXCEEDED Macro

File

hci_errors.h

C

```
#define HCI_ERR_SCO_CONN_LIMIT_EXCEEDED 0x0a
```

Description

This is macro HCI_ERR_SCO_CONN_LIMIT_EXCEEDED.

HCI_ERR_SIMPLE_PAIRING_NOT_SUPPORTED Macro

File

hci_errors.h

C

```
#define HCI_ERR_SIMPLE_PAIRING_NOT_SUPPORTED 0x37
```

Description

This is macro HCI_ERR_SIMPLE_PAIRING_NOT_SUPPORTED.

HCI_ERR_SUCCESS Macro

File

hci_errors.h

C

```
#define HCI_ERR_SUCCESS 0x00
```

Description

- addtogroup hci
- @{
- @name Errors

HCI_ERR_UNSPECIFIED Macro

File

hci_errors.h

C

```
#define HCI_ERR_UNSPECIFIED 0x1F
```

Description

This is macro HCI_ERR_UNSPECIFIED.

HCI_EVT_ALL_HCI_EVENTS Macro

File

hci.h

C

```
#define HCI_EVT_ALL_HCI_EVENTS 0xffff
```

Description

Used in bt_hci_ctrl_register_event_listener to indicate that the listener is called for all HCI events (excluding dotstack internal ones)

HCI_EVT_AUTHENTICATION_COMPLETE Macro

File

hci.h

C

```
#define HCI_EVT_AUTHENTICATION_COMPLETE 0x06
```

Description

This is macro HCI_EVT_AUTHENTICATION_COMPLETE.

HCI_EVT_CHANGE_CONN_LINK_COMPLETE Macro

File

hci.h

C

```
#define HCI_EVT_CHANGE_CONN_LINK_COMPLETE 0x09
```

Description

This is macro HCI_EVT_CHANGE_CONN_LINK_COMPLETE.

HCI_EVT_CMD_SEND_FINISHED Macro

File

hci.h

C

```
#define HCI_EVT_CMD_SEND_FINISHED 0xff01
```

Description

This is macro HCI_EVT_CMD_SEND_FINISHED.

HCI_EVT_CMD_SEND_STARTED Macro

File

hci.h

C

```
#define HCI_EVT_CMD_SEND_STARTED 0xff00
```

Description

dotstack internal events

HCI_EVT_COMMAND_COMPLETE Macro

File

hci.h

C

```
#define HCI_EVT_COMMAND_COMPLETE 0x0E
```

Description

This is macro HCI_EVT_COMMAND_COMPLETE.

HCI_EVT_COMMAND_COMPLETE_PARAM_LEN Macro

File

hci.h

C

```
#define HCI_EVT_COMMAND_COMPLETE_PARAM_LEN 3
```

Description

This is macro HCI_EVT_COMMAND_COMPLETE_PARAM_LEN.

HCI_EVT_COMMAND_STATUS Macro

File

hci.h

C

```
#define HCI_EVT_COMMAND_STATUS 0x0F
```

Description

This is macro HCI_EVT_COMMAND_STATUS.

HCI_EVT_CONN_PACKET_TYPE_CHANGED Macro

File

hci.h

C

```
#define HCI_EVT_CONN_PACKET_TYPE_CHANGED 0x1D
```

Description

This is macro HCI_EVT_CONN_PACKET_TYPE_CHANGED.

HCI_EVT_CONNECTION_COMPLETE Macro

File

hci.h

C

```
#define HCI_EVT_CONNECTION_COMPLETE 0x03
```

Description

This is macro HCI_EVT_CONNECTION_COMPLETE.

HCI_EVT_CONNECTION_REQUEST Macro

File

hci.h

C

```
#define HCI_EVT_CONNECTION_REQUEST 0x04
```

Description

This is macro HCI_EVT_CONNECTION_REQUEST.

HCI_EVT_DATA_BUFFER_OVERFLOW Macro

File

hci.h

C

```
#define HCI_EVT_DATA_BUFFER_OVERFLOW 0x1A
```

Description

This is macro HCI_EVT_DATA_BUFFER_OVERFLOW.

HCI_EVT_DISCONNECTION_COMPLETE Macro

File

hci.h

C

```
#define HCI_EVT_DISCONNECTION_COMPLETE 0x05
```

Description

This is macro HCI_EVT_DISCONNECTION_COMPLETE.

HCI_EVT_ENCRYPTION_CHANGE Macro

File

hci.h

C

```
#define HCI_EVT_ENCRYPTION_CHANGE 0x08
```

Description

This is macro HCI_EVT_ENCRYPTION_CHANGE.

HCI_EVT_ENCRYPTION_KEY_REFRESH_COMPLETE Macro

File

hci.h

C

```
#define HCI_EVT_ENCRYPTION_KEY_REFRESH_COMPLETE 0x30
```

Description

This is macro HCI_EVT_ENCRYPTION_KEY_REFRESH_COMPLETE.

HCI_EVT_ENHANCED_FLUSH_COMPLETE Macro

File

hci.h

C

```
#define HCI_EVT_ENHANCED_FLUSH_COMPLETE 0x39
```

Description

This is macro HCI_EVT_ENHANCED_FLUSH_COMPLETE.

HCI_EVT_EXTENDED_INQUIRY_RESULT Macro

File

hci.h

C

```
#define HCI_EVT_EXTENDED_INQUIRY_RESULT 0x2F
```

Description

This is macro HCI_EVT_EXTENDED_INQUIRY_RESULT.

HCI_EVT_FIRST Macro

File

hci.h

C

```
#define HCI_EVT_FIRST HCI_EVT_INQUIRY_COMPLETE
```

Description

This is macro HCI_EVT_FIRST.

HCI_EVT_FLOW_SPECIFICATION_COMPLETE Macro

File

hci.h

C

```
#define HCI_EVT_FLOW_SPECIFICATION_COMPLETE 0x21
```

Description

This is macro HCI_EVT_FLOW_SPECIFICATION_COMPLETE.

HCI_EVT_FLUSH_OCCURED Macro

File

hci.h

C

```
#define HCI_EVT_FLUSH_OCCURED 0x11
```

Description

This is macro HCI_EVT_FLUSH_OCCURED.

HCI_EVT_HARDWARE_ERROR Macro

File

hci.h

C

```
#define HCI_EVT_HARDWARE_ERROR 0x10
```

Description

This is macro HCI_EVT_HARDWARE_ERROR.

HCI_EVT_INQUIRY_COMPLETE Macro

File

hci.h

C

```
#define HCI_EVT_INQUIRY_COMPLETE 0x01
```

Description

- addtogroup hci
- @{
- @name Events

HCI_EVT_INQUIRY_RESULT Macro

File

hci.h

C

```
#define HCI_EVT_INQUIRY_RESULT 0x02
```

Description

This is macro HCI_EVT_INQUIRY_RESULT.

HCI_EVT_INQUIRY_RESULT_WITH_RSSI Macro

File

hci.h

C

```
#define HCI_EVT_INQUIRY_RESULT_WITH_RSSI 0x22
```

Description

This is macro HCI_EVT_INQUIRY_RESULT_WITH_RSSI.

HCI_EVT_IO_CAPABILITY_REQUEST Macro

File

hci.h

C

```
#define HCI_EVT_IO_CAPABILITY_REQUEST 0x31
```

Description

This is macro HCI_EVT_IO_CAPABILITY_REQUEST.

HCI_EVT_IO_CAPABILITY_RESPONSE Macro

File

hci.h

C

```
#define HCI_EVT_IO_CAPABILITY_RESPONSE 0x32
```

Description

This is macro HCI_EVT_IO_CAPABILITY_RESPONSE.

HCI_EVT_KEYPRESS_NOTIFICATION Macro

File

hci.h

C

```
#define HCI_EVT_KEYPRESS_NOTIFICATION 0x3C
```

Description

This is macro HCI_EVT_KEYPRESS_NOTIFICATION.

HCI_EVT_LAST Macro

File

hci.h

C

```
#define HCI_EVT_LAST HCI_EVT_LE_META_EVENT
```

Description

This is macro HCI_EVT_LAST.

HCI_EVT_LE_META_EVENT Macro

File

hci.h

C

```
#define HCI_EVT_LE_META_EVENT 0x3E
```

Description

This is macro HCI_EVT_LE_META_EVENT.

HCI_EVT_LINK_IS_BUSY Macro

File

hci.h

C

```
#define HCI_EVT_LINK_IS_BUSY 0xff03
```

Description

This is macro HCI_EVT_LINK_IS_BUSY.

HCI_EVT_LINK_IS_IDLE Macro

File

hci.h

C

```
#define HCI_EVT_LINK_IS_IDLE 0xff02
```

Description

This is macro HCI_EVT_LINK_IS_IDLE.

HCI_EVT_LINK_KEY_NOTIFICATION Macro

File

hci.h

C

```
#define HCI_EVT_LINK_KEY_NOTIFICATION 0x18
```

Description

This is macro HCI_EVT_LINK_KEY_NOTIFICATION.

HCI_EVT_LINK_KEY_REQUEST Macro

File

hci.h

C

```
#define HCI_EVT_LINK_KEY_REQUEST 0x17
```

Description

This is macro HCI_EVT_LINK_KEY_REQUEST.

HCI_EVT_LINK_SUPERVISION_TO_CHANGED Macro

File

hci.h

C

```
#define HCI_EVT_LINK_SUPERVISION_TO_CHANGED 0x38
```

Description

This is macro HCI_EVT_LINK_SUPERVISION_TO_CHANGED.

HCI_EVT_LOOPBACK_COMMAND Macro

File

hci.h

C

```
#define HCI_EVT_LOOPBACK_COMMAND 0x19
```

Description

This is macro HCI_EVT_LOOPBACK_COMMAND.

HCI_EVT_MASTER_LINK_KEY_COMPLETE Macro

File

hci.h

C

```
#define HCI_EVT_MASTER_LINK_KEY_COMPLETE 0x0A
```

Description

This is macro HCI_EVT_MASTER_LINK_KEY_COMPLETE.

HCI_EVT_MAX_SLOTS_CHANGE Macro

File

hci.h

C

```
#define HCI_EVT_MAX_SLOTS_CHANGE 0x1B
```

Description

This is macro HCI_EVT_MAX_SLOTS_CHANGE.

HCI_EVT_MODE_CHANGE Macro

File

hci.h

C

```
#define HCI_EVT_MODE_CHANGE 0x14
```

Description

This is macro HCI_EVT_MODE_CHANGE.

HCI_EVT_NUM_OF_COMPLETED_PACKETS Macro

File

hci.h

C

```
#define HCI_EVT_NUM_OF_COMPLETED_PACKETS 0x13
```

Description

This is macro HCI_EVT_NUM_OF_COMPLETED_PACKETS.

HCI_EVT_PAGE_SCAN_REPET_MODE_CHANGE Macro

File

hci.h

C

```
#define HCI_EVT_PAGE_SCAN_REPET_MODE_CHANGE 0x20
```

Description

This is macro HCI_EVT_PAGE_SCAN_REPET_MODE_CHANGE.

HCI_EVT_PIN_CODE_REQUEST Macro

File

hci.h

C

```
#define HCI_EVT_PIN_CODE_REQUEST 0x16
```

Description

This is macro HCI_EVT_PIN_CODE_REQUEST.

HCI_EVT_QOS_SETUP_COMPLETE Macro

File

hci.h

C

```
#define HCI_EVT_QOS_SETUP_COMPLETE 0x0D
```

Description

This is macro HCI_EVT_QOS_SETUP_COMPLETE.

HCI_EVT_QOS_VIOLATION Macro

File

hci.h

C

```
#define HCI_EVT_QOS_VIOLATION 0x1E
```

Description

This is macro HCI_EVT_QOS_VIOLATION.

HCI_EVT_READ_CLOCK_OFFSET_COMPLETE Macro

File

hci.h

C

```
#define HCI_EVT_READ_CLOCK_OFFSET_COMPLETE 0x1C
```

Description

This is macro HCI_EVT_READ_CLOCK_OFFSET_COMPLETE.

HCI_EVT_READ_RMT_EXT_FEATURES_COMP Macro

File

hci.h

C

```
#define HCI_EVT_READ_RMT_EXT_FEATURES_COMP 0x23
```

Description

This is macro HCI_EVT_READ_RMT_EXT_FEATURES_COMP.

HCI_EVT_READ_RMT_SUP_FEATURES_COMP Macro

File

hci.h

C

```
#define HCI_EVT_READ_RMT_SUP_FEATURES_COMP 0x0B
```

Description

This is macro HCI_EVT_READ_RMT_SUP_FEATURES_COMP.

HCI_EVT_READ_RMT_VERSION_INFO_COMP Macro

File

hci.h

C

```
#define HCI_EVT_READ_RMT_VERSION_INFO_COMP 0x0C
```

Description

This is macro HCI_EVT_READ_RMT_VERSION_INFO_COMP.

HCI_EVT_REMOTE_NAME_REQUEST_COMPLETE Macro

File

hci.h

C

```
#define HCI_EVT_REMOTE_NAME_REQUEST_COMPLETE 0x07
```

Description

This is macro HCI_EVT_REMOTE_NAME_REQUEST_COMPLETE.

HCI_EVT_REMOTE_OOB_DATA_REQUEST Macro

File

hci.h

C

```
#define HCI_EVT_REMOTE_OOB_DATA_REQUEST 0x35
```

Description

This is macro HCI_EVT_REMOTE_OOB_DATA_REQUEST.

HCI_EVT_RETURN_LINK_KEYS Macro

File

hci.h

C

```
#define HCI_EVT_RETURN_LINK_KEYS 0x15
```

Description

This is macro HCI_EVT_RETURN_LINK_KEYS.

HCI_EVT_RMT_HOST_SUPP_FEATURES_NTF Macro

File

hci.h

C

```
#define HCI_EVT_RMT_HOST_SUPP_FEATURES_NTF 0x3D
```

Description

This is macro HCI_EVT_RMT_HOST_SUPP_FEATURES_NTF.

HCI_EVT_ROLE_CHANGE Macro

File

hci.h

C

```
#define HCI_EVT_ROLE_CHANGE 0x12
```

Description

This is macro HCI_EVT_ROLE_CHANGE.

HCI_EVT_SIMPLE_PAIRING_COMPLETE Macro

File

hci.h

C

```
#define HCI_EVT_SIMPLE_PAIRING_COMPLETE 0x36
```

Description

This is macro HCI_EVT_SIMPLE_PAIRING_COMPLETE.

HCI_EVT_SNIFF_SUBRATING Macro

File

hci.h

C

```
#define HCI_EVT_SNIFF_SUBRATING 0x2E
```

Description

This is macro HCI_EVT_SNIFF_SUBRATING.

HCI_EVT_SYNCH_CONNECTION_CHANGED Macro

File

hci.h

C

```
#define HCI_EVT_SYNCH_CONNECTION_CHANGED 0x2D
```

Description

This is macro HCI_EVT_SYNCH_CONNECTION_CHANGED.

HCI_EVT_SYNCH_CONNECTION_COMPLETE Macro

File

hci.h

C

```
#define HCI_EVT_SYNCH_CONNECTION_COMPLETE 0x2C
```

Description

This is macro HCI_EVT_SYNCH_CONNECTION_COMPLETE.

HCI_EVT_USER_CONFIRMATION_REQUEST Macro

File

hci.h

C

```
#define HCI_EVT_USER_CONFIRMATION_REQUEST 0x33
```

Description

This is macro HCI_EVT_USER_CONFIRMATION_REQUEST.

HCI_EVT_USER_PASSKEY_NOTIFICATION Macro

File

hci.h

C

```
#define HCI_EVT_USER_PASSKEY_NOTIFICATION 0x3B
```

Description

This is macro HCI_EVT_USER_PASSKEY_NOTIFICATION.

HCI_EVT_USER_PASSKEY_REQUEST Macro

File

hci.h

C

```
#define HCI_EVT_USER_PASSKEY_REQUEST 0x34
```

Description

This is macro HCI_EVT_USER_PASSKEY_REQUEST.

HCI_EXIT_PARK_STATE Macro

File

hci.h

C

```
#define HCI_EXIT_PARK_STATE HCI_OPCODE(OGF_LINK_POLICY, 0x0006)
```

Description

This is macro HCI_EXIT_PARK_STATE.

HCI_EXIT_PERIODIC_INQUIRY_MODE Macro

File

hci.h

C

```
#define HCI_EXIT_PERIODIC_INQUIRY_MODE HCI_OPCODE(OGF_LINK_CONTROL, 0x0004)
```

Description

This is macro HCI_EXIT_PERIODIC_INQUIRY_MODE.

HCI_EXIT_SNIFF_MODE Macro

File

hci.h

C

```
#define HCI_EXIT_SNIFF_MODE HCI_OPCODE(OGF_LINK_POLICY, 0x0004)
```

Description

This is macro HCI_EXIT_SNIFF_MODE.

HCI_FLOW_SPECIFICATION Macro

File

hci.h

C

```
#define HCI_FLOW_SPECIFICATION HCI_OPCODE(OGF_LINK_POLICY, 0x0010)
```

Description

This is macro HCI_FLOW_SPECIFICATION.

HCI_FLUSH Macro

File

hci.h

C

```
#define HCI_FLUSH HCI_OPCODE(OGF_CTRL_BASEBAND, 0x0008)
```

Description

This is macro HCI_FLUSH.

HCI_H2C_BROADCAST_ACTIVE_SLAVE Macro

File

hci.h

C

```
#define HCI_H2C_BROADCAST_ACTIVE_SLAVE 0x1
```

Description

This is macro HCI_H2C_BROADCAST_ACTIVE_SLAVE.

HCI_H2C_BROADCAST_NO_BROADCATS Macro

File

hci.h

C

```
#define HCI_H2C_BROADCAST_NO_BROADCATS 0x0
```

Description

This is macro HCI_H2C_BROADCAST_NO_BROADCATS.

HCI_H2C_BROADCAST_PARCKED_SLAVE Macro

File

hci.h

C

```
#define HCI_H2C_BROADCAST_PARCKED_SLAVE 0x2
```

Description

This is macro HCI_H2C_BROADCAST_PARCKED_SLAVE.

HCI_H2C_BROADCAST_RESERVED Macro

File

hci.h

C

```
#define HCI_H2C_BROADCAST_RESERVED 0x3
```

Description

This is macro HCI_H2C_BROADCAST_RESERVED.

HCI_HOLD_MODE Macro

File

hci.h

C

```
#define HCI_HOLD_MODE HCI_OPCODE(OGF_LINK_POLICY, 0x0001)
```

Description

addtogroup hci @{

@name Link policy commands

details The Link Policy Commands provide methods for the Host to affect how the Link Manager manages the piconet.

HCI_HOST_BUFFER_SIZE Macro

File

hci.h

C

```
#define HCI_HOST_BUFFER_SIZE HCI_OPCODE(OGF_CTRL_BASEBAND, 0x0033)
```

Description

This is macro HCI_HOST_BUFFER_SIZE.

HCI_HOST_NUM_OF_COMPLETED_PACKETS Macro

File

hci.h

C

```
#define HCI_HOST_NUM_OF_COMPLETED_PACKETS HCI_OPCODE(OGF_CTRL_BASEBAND, 0x0035)
```

Description

This is macro HCI_HOST_NUM_OF_COMPLETED_PACKETS.

HCI_INQUIRY Macro

File

hci.h

C

```
#define HCI_INQUIRY HCI_OPCODE(OGF_LINK_CONTROL, 0x0001)
```

Description

addtogroup hci @{

@name Link control commands

details The Link Control commands allow a Controller to control connections to other BR/EDR Controllers.

HCI_INQUIRY_CANCEL Macro

File

hci.h

C

```
#define HCI_INQUIRY_CANCEL HCI_OPCODE(OGF_LINK_CONTROL, 0x0002)
```

Description

This is macro HCI_INQUIRY_CANCEL.

HCI_INQUIRY_MODE_EXTENDED Macro

File

hci.h

C

```
#define HCI_INQUIRY_MODE_EXTENDED 2
```

Description

This is macro HCI_INQUIRY_MODE_EXTENDED.

HCI_INQUIRY_MODE_STANDARD Macro

File

hci.h

C

```
#define HCI_INQUIRY_MODE_STANDARD 0
```

Description

This is macro HCI_INQUIRY_MODE_STANDARD.

HCI_INQUIRY_MODE_WITH_RSSI Macro

File

hci.h

C

```
#define HCI_INQUIRY_MODE_WITH_RSSI 1
```

Description

This is macro HCI_INQUIRY_MODE_WITH_RSSI.

HCI_IO_CAPABILITY_REQUEST_NEGATIVE_REPLY Macro

File

hci.h

C

```
#define HCI_IO_CAPABILITY_REQUEST_NEGATIVE_REPLY HCI_OPCODE(OGF_LINK_CONTROL, 0x0034)
```

Description

This is macro HCI_IO_CAPABILITY_REQUEST_NEGATIVE_REPLY.

HCI_IO_CAPABILITY_REQUEST_REPLY Macro

File

hci.h

C

```
#define HCI_IO_CAPABILITY_REQUEST_REPLY HCI_OPCODE(OGF_LINK_CONTROL, 0x002B)
```

Description

This is macro HCI_IO_CAPABILITY_REQUEST_REPLY.

HCI_LE_ADD_DEVICE_TO_WHITE_LIST Macro

File

hci.h

C

```
#define HCI_LE_ADD_DEVICE_TO_WHITE_LIST HCI_OPCODE(OGF_LE, 0x0011)
```

Description

This is macro HCI_LE_ADD_DEVICE_TO_WHITE_LIST.

HCI_LE_ADDRESS_TYPE_PUBLIC Macro

File

hci_le.h

C

```
#define HCI_LE_ADDRESS_TYPE_PUBLIC 0
```

Description

This is macro HCI_LE_ADDRESS_TYPE_PUBLIC.

HCI_LE_ADDRESS_TYPE_RANDOM Macro

File

hci_le.h

C

```
#define HCI_LE_ADDRESS_TYPE_RANDOM 1
```

Description

This is macro HCI_LE_ADDRESS_TYPE_RANDOM.

HCI_LE_ADVERTISING_FLAG_BREDR_NOT_SUPPORTED Macro

File

hci_le.h

C

```
#define HCI_LE_ADVERTISING_FLAG_BREDR_NOT_SUPPORTED 0x04
```

Description

This is macro HCI_LE_ADVERTISING_FLAG_BREDR_NOT_SUPPORTED.

HCI_LE_ADVERTISING_FLAG_GENERAL_DISCOVERABLE_MODE Macro

File

hci_le.h

C

```
#define HCI_LE_ADVERTISING_FLAG_GENERAL_DISCOVERABLE_MODE 0x02
```

Description

This is macro HCI_LE_ADVERTISING_FLAG_GENERAL_DISCOVERABLE_MODE.

HCI_LE_ADVERTISING_FLAG_LIMITED_DISCOVERABLE_MODE Macro

File

hci_le.h

C

```
#define HCI_LE_ADVERTISING_FLAG_LIMITED_DISCOVERABLE_MODE 0x01
```

Description

This is macro HCI_LE_ADVERTISING_FLAG_LIMITED_DISCOVERABLE_MODE.

HCI_LE_ADVERTISING_FLAG_SIMULTANEOUS_LE_BREDR_CONTROLLER Macro

File

hci_le.h

C

```
#define HCI_LE_ADVERTISING_FLAG_SIMULTANEOUS_LE_BREDR_CONTROLLER 0x08
```

Description

This is macro HCI_LE_ADVERTISING_FLAG_SIMULTANEOUS_LE_BREDR_CONTROLLER.

HCI_LE_ADVERTISING_FLAG_SIMULTANEOUS_LE_BREDR_HOST Macro

File

hci_le.h

C

```
#define HCI_LE_ADVERTISING_FLAG_SIMULTANEOUS_LE_BREDR_HOST 0x10
```

Description

This is macro HCI_LE_ADVERTISING_FLAG_SIMULTANEOUS_LE_BREDR_HOST.

HCI_LE_AVERTISING_DISABLED Macro

File

hci_le.h

C

```
#define HCI_LE_AVERTISING_DISABLED 0
```

Description

This is macro HCI_LE_AVERTISING_DISABLED.

HCI_LE_AVERTISING_ENABLED Macro

File

hci_le.h

C

```
#define HCI_LE_AVERTISING_ENABLED 1
```

Description

This is macro HCI_LE_AVERTISING_ENABLED.

HCI_LE_CLEAR_WHITE_LIST Macro

File

hci.h

C

```
#define HCI_LE_CLEAR_WHITE_LIST HCI_OPCODE(OGF_LE, 0x0010)
```

Description

This is macro HCI_LE_CLEAR_WHITE_LIST.

HCI_LE_CONNECTION_UPDATE Macro

File

hci.h

C

```
#define HCI_LE_CONNECTION_UPDATE HCI_OPCODE(OGF_LE, 0x0013)
```

Description

This is macro HCI_LE_CONNECTION_UPDATE.

HCI_LE_CREATE_CONNECTION Macro

File

hci.h

C

```
#define HCI_LE_CREATE_CONNECTION HCI_OPCODE(OGF_LE, 0x000D)
```

Description

This is macro HCI_LE_CREATE_CONNECTION.

HCI_LE_CREATE_CONNECTION_CANCEL Macro

File

hci.h

C

```
#define HCI_LE_CREATE_CONNECTION_CANCEL HCI_OPCODE(OGF_LE, 0x000E)
```

Description

This is macro HCI_LE_CREATE_CONNECTION_CANCEL.

HCI_LE_DISABLED Macro

File

hci_le.h

C

```
#define HCI_LE_DISABLED 0
```

Description

This is macro HCI_LE_DISABLED.

HCI_LE_DUPLICATE_FILTERING_DISABLED Macro

File

hci_le.h

C

```
#define HCI_LE_DUPLICATE_FILTERING_DISABLED 0
```

Description

This is macro HCI_LE_DUPLICATE_FILTERING_DISABLED.

HCI_LE_DUPLICATE_FILTERING_ENABLED Macro

File

hci_le.h

C

```
#define HCI_LE_DUPLICATE_FILTERING_ENABLED 1
```

Description

This is macro HCI_LE_DUPLICATE_FILTERING_ENABLED.

HCI_LE_ENABLED Macro

File

hci_le.h

C

```
#define HCI_LE_ENABLED 1
```

Description

This is macro HCI_LE_ENABLED.

HCI_LE_ENCRYPT Macro

File

hci.h

C

```
#define HCI_LE_ENCRYPT HCI_OPCODE(OGF_LE, 0x0017)
```

Description

This is macro HCI_LE_ENCRYPT.

HCI_LE_EVT_ADVERTISING_REPORT Macro

File

hci_le.h

C

```
#define HCI_LE_EVT_ADVERTISING_REPORT 0x02
```

Description

This is macro HCI_LE_EVT_ADVERTISING_REPORT.

HCI_LE_EVT_CONNECTION_COMPLETE Macro

File

hci_le.h

C

```
#define HCI_LE_EVT_CONNECTION_COMPLETE 0x01
```

Description

This is macro HCI_LE_EVT_CONNECTION_COMPLETE.

HCI_LE_EVT_CONNECTION_UPDATE_COMPLETE Macro

File

hci_le.h

C

```
#define HCI_LE_EVT_CONNECTION_UPDATE_COMPLETE 0x03
```

Description

This is macro HCI_LE_EVT_CONNECTION_UPDATE_COMPLETE.

HCI_LE_EVT_LONG_TERM_KEY_REQUEST Macro

File

hci_le.h

C

```
#define HCI_LE_EVT_LONG_TERM_KEY_REQUEST 0x05
```

Description

This is macro HCI_LE_EVT_LONG_TERM_KEY_REQUEST.

HCI_LE_EVT_READ_REMOTE_USED_FEATURES_COMPLETE Macro

File

hci_le.h

C

```
#define HCI_LE_EVT_READ_REMOTE_USED_FEATURES_COMPLETE 0x04
```

Description

This is macro HCI_LE_EVT_READ_REMOTE_USED_FEATURES_COMPLETE.

HCI_LE_FILTER_POLICY_NOT_USED Macro

File

hci_le.h

C

```
#define HCI_LE_FILTER_POLICY_NOT_USED 0 // White list is not used to determine which advertiser to connect
to.
```

Description

White list is not used to determine which advertiser to connect to. Peer_Address_Type and Peer_Address shall be used.

HCI_LE_FILTER_POLICY_WHITE_LIST Macro

File

hci_le.h

C

```
#define HCI_LE_FILTER_POLICY_WHITE_LIST 1 // White list is used to determine which advertiser to connect to.
```

Description

White list is used to determine which advertiser to connect to. Peer_Address_Type and Peer_Address shall be ignored

HCI_LE_FLAG_SHARED_ACL_BUFFERS Macro

File

hci_le.h

C

```
#define HCI_LE_FLAG_SHARED_ACL_BUFFERS 2
```

Description

This is macro HCI_LE_FLAG_SHARED_ACL_BUFFERS.

HCI_LE_FLAG_SIMULTANEOUS_LE_BREDR Macro

File

hci_le.h

C

```
#define HCI_LE_FLAG_SIMULTANEOUS_LE_BREDR 1
```

Description

This is macro HCI_LE_FLAG_SIMULTANEOUS_LE_BREDR.

HCI_LE_LONG_TERM_KEY_REQUEST_NEGATIVE_REPLY Macro

File

hci.h

C

```
#define HCI_LE_LONG_TERM_KEY_REQUEST_NEGATIVE_REPLY HCI_OPCODE(OGF_LE, 0x001B)
```

Description

This is macro HCI_LE_LONG_TERM_KEY_REQUEST_NEGATIVE_REPLY.

HCI_LE_LONG_TERM_KEY_REQUEST_REPLY Macro

File

hci.h

C

```
#define HCI_LE_LONG_TERM_KEY_REQUEST_REPLY HCI_OPCODE(OGF_LE, 0x001A)
```

Description

This is macro HCI_LE_LONG_TERM_KEY_REQUEST_REPLY.

HCI_LE_MAX_AD_LEN Macro

File

hci.h

C

```
#define HCI_LE_MAX_AD_LEN 31
```

Description

This is macro HCI_LE_MAX_AD_LEN.

HCI_LE_RAND Macro

File

hci.h

C

```
#define HCI_LE_RAND HCI_OPCODE(OGF_LE, 0x0018)
```

Description

This is macro HCI_LE_RAND.

HCI_LE_READ_ADVERTISING_CHANNEL_TX_POWER Macro

File

hci.h

C

```
#define HCI_LE_READ_ADVERTISING_CHANNEL_TX_POWER HCI_OPCODE(OGF_LE, 0x0007)
```

Description

This is macro HCI_LE_READ_ADVERTISING_CHANNEL_TX_POWER.

HCI_LE_READ_BUFFER_SIZE Macro

File

hci.h

C

```
#define HCI_LE_READ_BUFFER_SIZE HCI_OPCODE(OGF_LE, 0x0002)
```

Description

This is macro HCI_LE_READ_BUFFER_SIZE.

HCI_LE_READ_CHANNEL_MAP Macro

File

hci.h

C

```
#define HCI_LE_READ_CHANNEL_MAP HCI_OPCODE(OGF_LE, 0x0015)
```

Description

This is macro HCI_LE_READ_CHANNEL_MAP.

HCI_LE_READ_LOCAL_SUPPORTED_FEATURES Macro

File

hci.h

C

```
#define HCI_LE_READ_LOCAL_SUPPORTED_FEATURES HCI_OPCODE(OGF_LE, 0x0003)
```

Description

This is macro HCI_LE_READ_LOCAL_SUPPORTED_FEATURES.

HCI_LE_READ_REMOTE_USED_FEATURES Macro

File

hci.h

C

```
#define HCI_LE_READ_REMOTE_USED_FEATURES HCI_OPCODE(OGF_LE, 0x0016)
```

Description

This is macro HCI_LE_READ_REMOTE_USED_FEATURES.

HCI_LE_READ_SUPPORTED_STATES Macro

File

hci.h

C

```
#define HCI_LE_READ_SUPPORTED_STATES HCI_OPCODE(OGF_LE, 0x001C)
```

Description

This is macro HCI_LE_READ_SUPPORTED_STATES.

HCI_LE_READ_WHITE_LIST_SIZE Macro

File

hci.h

C

```
#define HCI_LE_READ_WHITE_LIST_SIZE HCI_OPCODE(OGF_LE, 0x000F)
```

Description

This is macro HCI_LE_READ_WHITE_LIST_SIZE.

HCI_LE_RECEIVE_TEST Macro

File

hci.h

C

```
#define HCI_LE_RECEIVE_TEST HCI_OPCODE(OGF_LE, 0x001D)
```

Description

This is macro HCI_LE_RECEIVE_TEST.

HCI_LE_REMOVE_DEVICE_FROM_WHITE_LIST Macro

File

hci.h

C

```
#define HCI_LE_REMOVE_DEVICE_FROM_WHITE_LIST HCI_OPCODE(OGF_LE, 0x0012)
```

Description

This is macro HCI_LE_REMOVE_DEVICE_FROM_WHITE_LIST.

HCI_LE_SCAN_DISABLED Macro

File

hci_le.h

C

```
#define HCI_LE_SCAN_DISABLED 0
```

Description

This is macro HCI_LE_SCAN_DISABLED.

HCI_LE_SCAN_ENABLED Macro

File

hci_le.h

C

```
#define HCI_LE_SCAN_ENABLED 1
```

Description

This is macro HCI_LE_SCAN_ENABLED.

HCI_LE_SCAN_EVT_CANCEL_FAILED Macro

File

hci_le.h

C

```
#define HCI_LE_SCAN_EVT_CANCEL_FAILED 4
```

Description

This is macro HCI_LE_SCAN_EVT_CANCEL_FAILED.

HCI_LE_SCAN_EVT_CANCELLED Macro

File

hci_le.h

C

```
#define HCI_LE_SCAN_EVT_CANCELLED 1
```

Description

This is macro HCI_LE_SCAN_EVT_CANCELLED.

HCI_LE_SCAN_EVT_DEVICE_FOUND Macro

File

hci_le.h

C

```
#define HCI_LE_SCAN_EVT_DEVICE_FOUND 2
```

Description

This is macro HCI_LE_SCAN_EVT_DEVICE_FOUND.

HCI_LE_SCAN_EVT_START_FAILED Macro

File

hci_le.h

C

```
#define HCI_LE_SCAN_EVT_START_FAILED 3
```

Description

This is macro HCI_LE_SCAN_EVT_START_FAILED.

HCI_LE_SCAN_EVT_STARTED Macro

File

hci_le.h

C

```
#define HCI_LE_SCAN_EVT_STARTED 0
```

Description

This is macro HCI_LE_SCAN_EVT_STARTED.

HCI_LE_SCAN_FILTER_POLICY_ALL Macro

File

hci_le.h

C

```
#define HCI_LE_SCAN_FILTER_POLICY_ALL 0
```

Description

This is macro HCI_LE_SCAN_FILTER_POLICY_ALL.

HCI_LE_SCAN_FILTER_POLICY_WHITE_LIST Macro

File

hci_le.h

C

```
#define HCI_LE_SCAN_FILTER_POLICY_WHITE_LIST 1
```

Description

This is macro HCI_LE_SCAN_FILTER_POLICY_WHITE_LIST.

HCI_LE_SCAN_TYPE_ACTIVE Macro

File

hci_le.h

C

```
#define HCI_LE_SCAN_TYPE_ACTIVE 1
```

Description

This is macro HCI_LE_SCAN_TYPE_ACTIVE.

HCI_LE_SCAN_TYPE_PASSIVE Macro

File

hci_le.h

C

```
#define HCI_LE_SCAN_TYPE_PASSIVE 0
```

Description

This is macro HCI_LE_SCAN_TYPE_PASSIVE.

HCI_LE_SET_ADVERTISE_ENABLE Macro

File

hci.h

C

```
#define HCI_LE_SET_ADVERTISE_ENABLE HCI_OPCODE(OGF_LE, 0x000A)
```

Description

This is macro HCI_LE_SET_ADVERTISE_ENABLE.

HCI_LE_SET_ADVERTISING_DATA Macro

File

hci.h

C

```
#define HCI_LE_SET_ADVERTISING_DATA HCI_OPCODE(OGF_LE, 0x0008)
```

Description

This is macro HCI_LE_SET_ADVERTISING_DATA.

HCI_LE_SET_ADVERTISING_PARAMETERS Macro

File

hci.h

C

```
#define HCI_LE_SET_ADVERTISING_PARAMETERS HCI_OPCODE(OGF_LE, 0x0006)
```

Description

This is macro HCI_LE_SET_ADVERTISING_PARAMETERS.

HCI_LE_SET_EVENT_MASK Macro

File

hci.h

C

```
#define HCI_LE_SET_EVENT_MASK HCI_OPCODE(OGF_LE, 0x0001)
```

Description

addtogroup hci @{

@name LE controller commands

details The LE Controller Commands provide access and control to various capabilities of the Bluetooth hardware, as well as methods for the Host to affect how the Link Layer manages the piconet, and controls connections.

HCI_LE_SET_HOST_CHANNEL_CLASSIFICATION Macro

File

hci.h

C

```
#define HCI_LE_SET_HOST_CHANNEL_CLASSIFICATION HCI_OPCODE(OGF_LE, 0x0014)
```

Description

This is macro HCI_LE_SET_HOST_CHANNEL_CLASSIFICATION.

HCI_LE_SET_RANDOM_ADDRESS Macro

File

hci.h

C

```
#define HCI_LE_SET_RANDOM_ADDRESS HCI_OPCODE(OGF_LE, 0x0005)
```

Description

This is macro HCI_LE_SET_RANDOM_ADDRESS.

HCI_LE_SET_SCAN_ENABLE Macro

File

hci.h

C

```
#define HCI_LE_SET_SCAN_ENABLE HCI_OPCODE(OGF_LE, 0x000C)
```

Description

This is macro HCI_LE_SET_SCAN_ENABLE.

HCI_LE_SET_SCAN_PARAMETERS Macro

File

hci.h

C

```
#define HCI_LE_SET_SCAN_PARAMETERS HCI_OPCODE(OGF_LE, 0x000B)
```

Description

This is macro HCI_LE_SET_SCAN_PARAMETERS.

HCI_LE_SET_SCAN_RESPONSE_DATA Macro

File

hci.h

C

```
#define HCI_LE_SET_SCAN_RESPONSE_DATA HCI_OPCODE(OGF_LE, 0x0009)
```

Description

This is macro HCI_LE_SET_SCAN_RESPONSE_DATA.

HCI_LE_SIMULTANEOUS_DISABLED Macro

File

hci_le.h

C

```
#define HCI_LE_SIMULTANEOUS_DISABLED 0
```

Description

This is macro HCI_LE_SIMULTANEOUS_DISABLED.

HCI_LE_SIMULTANEOUS_ENABLED Macro

File

hci_le.h

C

```
#define HCI_LE_SIMULTANEOUS_ENABLED 1
```

Description

This is macro HCI_LE_SIMULTANEOUS_ENABLED.

HCI_LE_START_ENCRYPTION Macro

File

hci.h

C

```
#define HCI_LE_START_ENCRYPTION HCI_OPCODE(OGF_LE, 0x0019)
```

Description

This is macro HCI_LE_START_ENCRYPTION.

HCI_LE_TEST_END Macro

File

hci.h

C

```
#define HCI_LE_TEST_END HCI_OPCODE(OGF_LE, 0x001F)
```

Description

This is macro HCI_LE_TEST_END.

HCI_LE_TRANSMITTER_TEST Macro

File

hci.h

C

```
#define HCI_LE_TRANSMITTER_TEST HCI_OPCODE(OGF_LE, 0x001E)
```

Description

This is macro HCI_LE_TRANSMITTER_TEST.

HCI_LINK_KEY_LEN Macro

File

hci.h

C

```
#define HCI_LINK_KEY_LEN 16
```

Description

This is macro HCI_LINK_KEY_LEN.

HCI_LINK_KEY_REQUEST_NEGATIVE_REPLY Macro

File

hci.h

C

```
#define HCI_LINK_KEY_REQUEST_NEGATIVE_REPLY HCI_OPCODE(OGF_LINK_CONTROL, 0x000C)
```

Description

This is macro HCI_LINK_KEY_REQUEST_NEGATIVE_REPLY.

HCI_LINK_KEY_REQUEST_REPLY Macro

File

hci.h

C

```
#define HCI_LINK_KEY_REQUEST_REPLY HCI_OPCODE(OGF_LINK_CONTROL, 0x000B)
```

Description

This is macro HCI_LINK_KEY_REQUEST_REPLY.

HCI_LINK_KEY_TYPE_COMBINATION Macro

File

hci.h

C

```
#define HCI_LINK_KEY_TYPE_COMBINATION 0x00
```

Description

This is macro HCI_LINK_KEY_TYPE_COMBINATION.

HCI_LINK_KEY_TYPE_LOCAL_UNIT Macro

File

hci.h

C

```
#define HCI_LINK_KEY_TYPE_LOCAL_UNIT 0x01
```

Description

This is macro HCI_LINK_KEY_TYPE_LOCAL_UNIT.

HCI_LINK_KEY_TYPE_REMOTE_UNIT Macro

File

hci.h

C

```
#define HCI_LINK_KEY_TYPE_REMOTE_UNIT 0x02
```

Description

This is macro HCI_LINK_KEY_TYPE_REMOTE_UNIT.

HCI_LINK_POLICY_ENABLE_HOLD_MODE Macro

File

hci.h

C

```
#define HCI_LINK_POLICY_ENABLE_HOLD_MODE 0x0002
```

Description

This is macro HCI_LINK_POLICY_ENABLE_HOLD_MODE.

HCI_LINK_POLICY_ENABLE_PARK_STATE Macro

File

hci.h

C

```
#define HCI_LINK_POLICY_ENABLE_PARK_STATE 0x0008
```

Description

This is macro HCI_LINK_POLICY_ENABLE_PARK_STATE.

HCI_LINK_POLICY_ENABLE_ROLE_SWITCH Macro

File

hci.h

C

```
#define HCI_LINK_POLICY_ENABLE_ROLE_SWITCH 0x0001
```

Description

This is macro HCI_LINK_POLICY_ENABLE_ROLE_SWITCH.

HCI_LINK_POLICY_ENABLE_SNIFF_MODE Macro

File

hci.h

C

```
#define HCI_LINK_POLICY_ENABLE_SNIFF_MODE 0x0004
```

Description

This is macro HCI_LINK_POLICY_ENABLE_SNIFF_MODE.

HCI_LINK_TYPE_BD_EDR Macro

File

hci_conn_state.h

C

```
#define HCI_LINK_TYPE_BD_EDR 0
```

Description

This is macro HCI_LINK_TYPE_BD_EDR.

HCI_LINK_TYPE_LE Macro

File

hci_conn_state.h

C

```
#define HCI_LINK_TYPE_LE 1
```

Description

This is macro HCI_LINK_TYPE_LE.

HCI_MASTER_LINK_KEY Macro

File

hci.h

C

```
#define HCI_MASTER_LINK_KEY HCI_OPCODE(OGF_LINK_CONTROL, 0x0017)
```

Description

This is macro HCI_MASTER_LINK_KEY.

HCI_MAX_DATA_BUFFERS Macro

File

hci_data_buffer.h

C

```
#define HCI_MAX_DATA_BUFFERS 2
```

Description

This is macro HCI_MAX_DATA_BUFFERS.

HCI_MAX_EVENT_PARAM_LEN Macro

File

hci.h

C

```
#define HCI_MAX_EVENT_PARAM_LEN 255
```

Description

This is macro HCI_MAX_EVENT_PARAM_LEN.

HCI_MAX_PARAM_LEN Macro

File

hci.h

C

```
#define HCI_MAX_PARAM_LEN 248
```

Description

This is macro HCI_MAX_PARAM_LEN.

HCI_MAX_PIN_LENGTH Macro

File

hci.h

C

```
#define HCI_MAX_PIN_LENGTH 8
```

Description

This is macro HCI_MAX_PIN_LENGTH.

HCI_OPCODE Macro

File

hci.h

C

```
#define HCI_OPCODE(ogf, ocf) (((int)ogf << 10) | ((int)ocf & 0x3ff))
```

Description

This is macro HCI_OPCODE.

HCI_PACKET_BOUNDARY_CONTINUE Macro

File

hci.h

C

```
#define HCI_PACKET_BOUNDARY_CONTINUE 0x1
```

Description

This is macro HCI_PACKET_BOUNDARY_CONTINUE.

HCI_PACKET_BOUNDARY_FIRST Macro

File

hci.h

C

```
#define HCI_PACKET_BOUNDARY_FIRST 0x2
```

Description

This is macro HCI_PACKET_BOUNDARY_FIRST.

HCI_PACKET_BOUNDARY_FIRST_AUTO_FLUSH Macro

File

hci.h

C

```
#define HCI_PACKET_BOUNDARY_FIRST_AUTO_FLUSH 0x2
```

Description

This is macro HCI_PACKET_BOUNDARY_FIRST_AUTO_FLUSH.

HCI_PACKET_BOUNDARY_FIRST_NO_AUTO_FLUSH Macro

File

hci.h

C

```
#define HCI_PACKET_BOUNDARY_FIRST_NO_AUTO_FLUSH 0x0
```

Description

This is macro HCI_PACKET_BOUNDARY_FIRST_NO_AUTO_FLUSH.

HCI_PACKET_TYPE_ACL_DATA Macro

File

hci.h

C

```
#define HCI_PACKET_TYPE_ACL_DATA 0x02
```

Description

This is macro HCI_PACKET_TYPE_ACL_DATA.

HCI_PACKET_TYPE_COMMAND Macro

File

hci.h

C

```
#define HCI_PACKET_TYPE_COMMAND 0x01
```

Description

This is macro HCI_PACKET_TYPE_COMMAND.

HCI_PACKET_TYPE_EVENT Macro

File

hci.h

C

```
#define HCI_PACKET_TYPE_EVENT 0x04
```

Description

This is macro HCI_PACKET_TYPE_EVENT.

HCI_PACKET_TYPE_NONE Macro

File

hci.h

C

```
#define HCI_PACKET_TYPE_NONE 0x00
```

Description

This is macro HCI_PACKET_TYPE_NONE.

HCI_PACKET_TYPE_SCO_DATA Macro

File

hci.h

C

```
#define HCI_PACKET_TYPE_SCO_DATA 0x03
```

Description

This is macro HCI_PACKET_TYPE_SCO_DATA.

HCI_PAGE_SCAN_REPETITION_MODE_R0 Macro

File

hci.h

C

```
#define HCI_PAGE_SCAN_REPETITION_MODE_R0 0x00
```

Description

This is macro HCI_PAGE_SCAN_REPETITION_MODE_R0.

HCI_PAGE_SCAN_REPETITION_MODE_R1 Macro

File

hci.h

C

```
#define HCI_PAGE_SCAN_REPETITION_MODE_R1 0x01
```

Description

This is macro HCI_PAGE_SCAN_REPETITION_MODE_R1.

HCI_PAGE_SCAN_REPETITION_MODE_R2 Macro

File

hci.h

C

```
#define HCI_PAGE_SCAN_REPETITION_MODE_R2 0x02
```

Description

This is macro HCI_PAGE_SCAN_REPETITION_MODE_R2.

HCI_PARAM_LEN_BD_ADDR Macro

File

hci.h

C

```
#define HCI_PARAM_LEN_BD_ADDR 6
```

Description

This is macro HCI_PARAM_LEN_BD_ADDR.

HCI_PARAM_LEN_BYTE Macro

File

hci.h

C

```
#define HCI_PARAM_LEN_BYTE 1
```

Description

This is macro HCI_PARAM_LEN_BYTE.

HCI_PARAM_LEN_DEV_CLASS Macro

File

hci.h

C

```
#define HCI_PARAM_LEN_DEV_CLASS 3
```

Description

This is macro HCI_PARAM_LEN_DEV_CLASS.

HCI_PARAM_LEN_HANDLE Macro

File

hci.h

C

```
#define HCI_PARAM_LEN_HANDLE 4
```

Description

This is macro HCI_PARAM_LEN_HANDLE.

HCI_PARAM_LEN_INT Macro

File

hci.h

C

```
#define HCI_PARAM_LEN_INT 2
```

Description

This is macro HCI_PARAM_LEN_INT.

HCI_PARAM_LEN_LONG Macro

File

hci.h

C

```
#define HCI_PARAM_LEN_LONG 4
```

Description

This is macro HCI_PARAM_LEN_LONG.

HCI_PARAM_TYPE_BD_ADDR Macro

File

hci.h

C

```
#define HCI_PARAM_TYPE_BD_ADDR 4
```

Description

This is macro HCI_PARAM_TYPE_BD_ADDR.

HCI_PARAM_TYPE_BYTE Macro

File

hci.h

C

```
#define HCI_PARAM_TYPE_BYTE 0
```

Description

This is macro HCI_PARAM_TYPE_BYTE.

HCI_PARAM_TYPE_DEV_CLASS Macro

File

hci.h

C

```
#define HCI_PARAM_TYPE_DEV_CLASS 6
```

Description

This is macro HCI_PARAM_TYPE_DEV_CLASS.

HCI_PARAM_TYPE_HANDLE Macro

File

hci.h

C

```
#define HCI_PARAM_TYPE_HANDLE 5
```

Description

This is macro HCI_PARAM_TYPE_HANDLE.

HCI_PARAM_TYPE_INT Macro

File

hci.h

C

```
#define HCI_PARAM_TYPE_INT 1
```

Description

This is macro HCI_PARAM_TYPE_INT.

HCI_PARAM_TYPE_LONG Macro

File

hci.h

C

```
#define HCI_PARAM_TYPE_LONG 2
```

Description

This is macro HCI_PARAM_TYPE_LONG.

HCI_PARAM_TYPE_STRING Macro

File

hci.h

C

```
#define HCI_PARAM_TYPE_STRING 7
```

Description

This is macro HCI_PARAM_TYPE_STRING.

HCI_PARK_STATE Macro

File

hci.h

C

```
#define HCI_PARK_STATE HCI_OPCODE(OGF_LINK_POLICY, 0x0005)
```

Description

This is macro HCI_PARK_STATE.

HCI_PERIODIC_INQUIRY_MODE Macro

File

hci.h

C

```
#define HCI_PERIODIC_INQUIRY_MODE HCI_OPCODE(OGF_LINK_CONTROL, 0x0003)
```

Description

This is macro HCI_PERIODIC_INQUIRY_MODE.

HCI_PIN_CODE_REQUEST_NEGATIVE_REPLY Macro

File

hci.h

C

```
#define HCI_PIN_CODE_REQUEST_NEGATIVE_REPLY HCI_OPCODE(OGF_LINK_CONTROL, 0x000E)
```

Description

This is macro HCI_PIN_CODE_REQUEST_NEGATIVE_REPLY.

HCI_PIN_CODE_REQUEST_REPLY Macro

File

hci.h

C

```
#define HCI_PIN_CODE_REQUEST_REPLY HCI_OPCODE(OGF_LINK_CONTROL, 0x000D)
```

Description

This is macro HCI_PIN_CODE_REQUEST_REPLY.

HCI_POWER_MODE_ACTIVE Macro

File

hci.h

C

```
#define HCI_POWER_MODE_ACTIVE 0
```

Description

This is macro HCI_POWER_MODE_ACTIVE.

HCI_POWER_MODE_HOLD Macro

File

hci.h

C

```
#define HCI_POWER_MODE_HOLD 1
```

Description

This is macro HCI_POWER_MODE_HOLD.

HCI_POWER_MODE_PARK Macro

File

hci.h

C

```
#define HCI_POWER_MODE_PARK 3
```

Description

This is macro HCI_POWER_MODE_PARK.

HCI_POWER_MODE_SNIFF Macro

File

hci.h

C

```
#define HCI_POWER_MODE_SNIFF 2
```

Description

This is macro HCI_POWER_MODE_SNIFF.

HCI_QOS_SETUP Macro

File

hci.h

C

```
#define HCI_QOS_SETUP HCI_OPCODE(OGF_LINK_POLICY, 0x0007)
```

Description

This is macro HCI_QOS_SETUP.

HCI_READ_AFH_CHANNEL_ASSESSMENT_MODE Macro

File

hci.h

C

```
#define HCI_READ_AFH_CHANNEL_ASSESSMENT_MODE HCI_OPCODE(OGF_CTRL_BASEBAND, 0x0048)
```

Description

This is macro HCI_READ_AFH_CHANNEL_ASSESSMENT_MODE.

HCI_READ_AFH_CHANNEL_MAP Macro

File

hci.h

C

```
#define HCI_READ_AFH_CHANNEL_MAP HCI_OPCODE(OGF_STATUS, 0x0006)
```

Description

This is macro HCI_READ_AFH_CHANNEL_MAP.

HCI_READ_AUTHENTICATION_ENABLE Macro

File

hci.h

C

```
#define HCI_READ_AUTHENTICATION_ENABLE HCI_OPCODE(OGF_CTRL_BASEBAND, 0x001F)
```

Description

This is macro HCI_READ_AUTHENTICATION_ENABLE.

HCI_READ_AUTOMATIC_FLASH_TIMEOUT Macro

File

hci.h

C

```
#define HCI_READ_AUTOMATIC_FLASH_TIMEOUT HCI_OPCODE(OGF_CTRL_BASEBAND, 0x0027)
```

Description

This is macro HCI_READ_AUTOMATIC_FLASH_TIMEOUT.

HCI_READ_BD_ADDR Macro

File

hci.h

C

```
#define HCI_READ_BD_ADDR HCI_OPCODE(OGF_INFORMATION, 0x0009)
```

Description

This is macro HCI_READ_BD_ADDR.

HCI_READ_BUFFER_SIZE Macro

File

hci.h

C

```
#define HCI_READ_BUFFER_SIZE HCI_OPCODE(OGF_INFORMATION, 0x0005)
```

Description

This is macro HCI_READ_BUFFER_SIZE.

HCI_READ_CLASS_OF_DEVICE Macro

File

hci.h

C

```
#define HCI_READ_CLASS_OF_DEVICE HCI_OPCODE(OGF_CTRL_BASEBAND, 0x0023)
```

Description

This is macro HCI_READ_CLASS_OF_DEVICE.

HCI_READ_CLOCK_COMMAND Macro

File

hci.h

C

```
#define HCI_READ_CLOCK_COMMAND HCI_OPCODE(OGF_STATUS, 0x0007)
```

Description

This is macro HCI_READ_CLOCK_COMMAND.

HCI_READ_CLOCK_OFFSET Macro

File

hci.h

C

```
#define HCI_READ_CLOCK_OFFSET HCI_OPCODE(OGF_LINK_CONTROL, 0x001F)
```

Description

This is macro HCI_READ_CLOCK_OFFSET.

HCI_READ_CONNECTION_ACCEPT_TIMEOUT Macro

File

hci.h

C

```
#define HCI_READ_CONNECTION_ACCEPT_TIMEOUT HCI_OPCODE(OGF_CTRL_BASEBAND, 0x0015)
```

Description

This is macro HCI_READ_CONNECTION_ACCEPT_TIMEOUT.

HCI_READ_CURRENT_IAC_LAP Macro

File

hci.h

C

```
#define HCI_READ_CURRENT_IAC_LAP HCI_OPCODE(OGF_CTRL_BASEBAND, 0x0039)
```

Description

This is macro HCI_READ_CURRENT_IAC_LAP.

HCI_READ_DEFAULT_ERRONEOUS_DATA_REPORTING Macro

File

hci.h

C

```
#define HCI_READ_DEFAULT_ERRONEOUS_DATA_REPORTING HCI_OPCODE(OGF_CTRL_BASEBAND, 0x005A)
```

Description

This is macro HCI_READ_DEFAULT_ERRONEOUS_DATA_REPORTING.

HCI_READ_DEFAULT_POLICY_SETTINGS Macro

File

hci.h

C

```
#define HCI_READ_DEFAULT_POLICY_SETTINGS HCI_OPCODE(OGF_LINK_POLICY, 0x000E)
```

Description

This is macro HCI_READ_DEFAULT_POLICY_SETTINGS.

HCI_READ_ENCRYPTION_MODE Macro

File

hci.h

C

```
#define HCI_READ_ENCRYPTION_MODE HCI_OPCODE(OGF_CTRL_BASEBAND, 0x0021)
```

Description

This is macro HCI_READ_ENCRYPTION_MODE.

HCI_READ_EXTENDED_INQUIRY_RESPONSE Macro

File

hci.h

C

```
#define HCI_READ_EXTENDED_INQUIRY_RESPONSE HCI_OPCODE(OGF_CTRL_BASEBAND, 0x0051)
```

Description

This is macro HCI_READ_EXTENDED_INQUIRY_RESPONSE.

HCI_READ_FAILED_CONTACT_COUNTER Macro

File

hci.h

C

```
#define HCI_READ_FAILED_CONTACT_COUNTER HCI_OPCODE(OGF_STATUS, 0x0001)
```

Description

addtogroup hci @{

@name Status Parameters

details The Controller modifies all status parameters. These parameters provide information about the current state of the Link Manager and Baseband in the BR/EDR Controller and the PAL in an AMP Controller. The host device cannot modify any of these parameters other than to reset certain specific parameters.

HCI_READ_HOLD_MODE_ACTIVITY Macro

File

hci.h

C

```
#define HCI_READ_HOLD_MODE_ACTIVITY HCI_OPCODE(OGF_CTRL_BASEBAND, 0x002B)
```

Description

This is macro HCI_READ_HOLD_MODE_ACTIVITY.

HCI_READ_INQUIRY_MODE Macro

File

hci.h

C

```
#define HCI_READ_INQUIRY_MODE HCI_OPCODE(OGF_CTRL_BASEBAND, 0x0044)
```

Description

This is macro HCI_READ_INQUIRY_MODE.

HCI_READ_INQUIRY_RESPONSE_TX_POWER_LEVEL Macro

File

hci.h

C

```
#define HCI_READ_INQUIRY_RESPONSE_TX_POWER_LEVEL HCI_OPCODE(OGF_CTRL_BASEBAND, 0x0058)
```

Description

This is macro HCI_READ_INQUIRY_RESPONSE_TX_POWER_LEVEL.

HCI_READ_INQUIRY_SCAN_ACTIVITY Macro

File

hci.h

C

```
#define HCI_READ_INQUIRY_SCAN_ACTIVITY HCI_OPCODE(OGF_CTRL_BASEBAND, 0x001D)
```

Description

This is macro HCI_READ_INQUIRY_SCAN_ACTIVITY.

HCI_READ_INQUIRY_SCAN_TYPE Macro

File

hci.h

C

```
#define HCI_READ_INQUIRY_SCAN_TYPE HCI_OPCODE(OGF_CTRL_BASEBAND, 0x0042)
```

Description

This is macro HCI_READ_INQUIRY_SCAN_TYPE.

HCI_READ_LE_HOST_SUPPORT Macro

File

hci.h

C

```
#define HCI_READ_LE_HOST_SUPPORT HCI_OPCODE(OGF_CTRL_BASEBAND, 0x006C)
```

Description

This is macro HCI_READ_LE_HOST_SUPPORT.

HCI_READ_LINK_POLICY_SETTINGS Macro

File

hci.h

C

```
#define HCI_READ_LINK_POLICY_SETTINGS HCI_OPCODE(OGF_LINK_POLICY, 0x000C)
```

Description

This is macro HCI_READ_LINK_POLICY_SETTINGS.

HCI_READ_LINK_QUALITY Macro

File

hci.h

C

```
#define HCI_READ_LINK_QUALITY HCI_OPCODE(OGF_STATUS, 0x0003)
```

Description

This is macro HCI_READ_LINK_QUALITY.

HCI_READ_LINK_SUPERVISION_TIMEOUT Macro

File

hci.h

C

```
#define HCI_READ_LINK_SUPERVISION_TIMEOUT HCI_OPCODE(OGF_CTRL_BASEBAND, 0x0036)
```

Description

This is macro HCI_READ_LINK_SUPERVISION_TIMEOUT.

HCI_READ_LMP_HANDLE Macro

File

hci.h

C

```
#define HCI_READ_LMP_HANDLE HCI_OPCODE(OGF_LINK_CONTROL, 0x0020)
```

Description

This is macro HCI_READ_LMP_HANDLE.

HCI_READ_LOCAL_EXTENDED_FEATURES Macro

File

hci.h

C

```
#define HCI_READ_LOCAL_EXTENDED_FEATURES HCI_OPCODE(OGF_INFORMATION, 0x0004)
```

Description

This is macro HCI_READ_LOCAL_EXTENDED_FEATURES.

HCI_READ_LOCAL_NAME Macro

File

hci.h

C

```
#define HCI_READ_LOCAL_NAME HCI_OPCODE(OGF_CTRL_BASEBAND, 0x0014)
```

Description

This is macro HCI_READ_LOCAL_NAME.

HCI_READ_LOCAL_OOB_DATA Macro

File

hci.h

C

```
#define HCI_READ_LOCAL_OOB_DATA HCI_OPCODE(OGF_CTRL_BASEBAND, 0x0057)
```

Description

This is macro HCI_READ_LOCAL_OOB_DATA.

HCI_READ_LOCAL_SUPPORTED_COMMANDS Macro

File

hci.h

C

```
#define HCI_READ_LOCAL_SUPPORTED_COMMANDS HCI_OPCODE(OGF_INFORMATION, 0x0002)
```

Description

This is macro HCI_READ_LOCAL_SUPPORTED_COMMANDS.

HCI_READ_LOCAL_SUPPORTED_FEATURES Macro

File

hci.h

C

```
#define HCI_READ_LOCAL_SUPPORTED_FEATURES HCI_OPCODE(OGF_INFORMATION, 0x0003)
```

Description

This is macro HCI_READ_LOCAL_SUPPORTED_FEATURES.

HCI_READ_LOCAL_VERSION_INFORMATION Macro

File

hci.h

C

```
#define HCI_READ_LOCAL_VERSION_INFORMATION HCI_OPCODE(OGF_INFORMATION, 0x0001)
```

Description

addtogroup hci @{

@name Informational Parameters

details The Informational Parameters are fixed by the manufacturer of the Bluetooth hardware. These parameters provide information about the BR/EDR Controller and the capabilities of the Link Manager and Baseband in the BR/EDR Controller and PAL in the AMP Controller. The host device cannot modify any of these parameters.

HCI_READ_LOOPBACK_MODE Macro

File

hci.h

C

```
#define HCI_READ_LOOPBACK_MODE HCI_OPCODE(OGF_TESTING, 0x0001)
```

Description

addtogroup hci @{

@name Testing Commands

details The Testing commands are used to provide the ability to test various functional capabilities of the Bluetooth hardware.

HCI_READ_NUM_BROADCST_RETR Macro

File

hci.h

C

```
#define HCI_READ_NUM_BROADCST_RETR HCI_OPCODE(OGF_CTRL_BASEBAND, 0x0029)
```

Description

This is macro HCI_READ_NUM_BROADCST_RETR.

HCI_READ_NUM_OF_SUPPORTED_IAC Macro

File

hci.h

C

```
#define HCI_READ_NUM_OF_SUPPORTED_IAC HCI_OPCODE(OGF_CTRL_BASEBAND, 0x0038)
```

Description

This is macro HCI_READ_NUM_OF_SUPPORTED_IAC.

HCI_READ_PAGE_SCAN_ACTIVITY Macro

File

hci.h

C

```
#define HCI_READ_PAGE_SCAN_ACTIVITY HCI_OPCODE(OGF_CTRL_BASEBAND, 0x001B)
```

Description

This is macro HCI_READ_PAGE_SCAN_ACTIVITY.

HCI_READ_PAGE_SCAN_PERIOD_MODE Macro

File

hci.h

C

```
#define HCI_READ_PAGE_SCAN_PERIOD_MODE HCI_OPCODE(OGF_CTRL_BASEBAND, 0x003B)
```

Description

This is macro HCI_READ_PAGE_SCAN_PERIOD_MODE.

HCI_READ_PAGE_SCAN_TYPE Macro

File

hci.h

C

```
#define HCI_READ_PAGE_SCAN_TYPE HCI_OPCODE(OGF_CTRL_BASEBAND, 0x0046)
```

Description

This is macro HCI_READ_PAGE_SCAN_TYPE.

HCI_READ_PAGE_TIMEOUT Macro

File

hci.h

C

```
#define HCI_READ_PAGE_TIMEOUT HCI_OPCODE(OGF_CTRL_BASEBAND, 0x0017)
```

Description

This is macro HCI_READ_PAGE_TIMEOUT.

HCI_READ_PIN_TYPE Macro

File

hci.h

C

```
#define HCI_READ_PIN_TYPE HCI_OPCODE(OGF_CTRL_BASEBAND, 0x0009)
```

Description

This is macro HCI_READ_PIN_TYPE.

HCI_READ_REFRESH_ENCRYPTION_KEY Macro

File

hci.h

C

```
#define HCI_READ_REFRESH_ENCRYPTION_KEY HCI_OPCODE(OGF_CTRL_BASEBAND, 0x0053)
```

Description

This is macro HCI_READ_REFRESH_ENCRYPTION_KEY.

HCI_READ_REMOTE_EXTENDED_FEATURES Macro

File

hci.h

C

```
#define HCI_READ_REMOTE_EXTENDED_FEATURES HCI_OPCODE(OGF_LINK_CONTROL, 0x001C)
```

Description

This is macro HCI_READ_REMOTE_EXTENDED_FEATURES.

HCI_READ_REMOTE_SUPPORTED_FEATURES Macro

File

hci.h

C

```
#define HCI_READ_REMOTE_SUPPORTED_FEATURES HCI_OPCODE(OGF_LINK_CONTROL, 0x001B)
```

Description

This is macro HCI_READ_REMOTE_SUPPORTED_FEATURES.

HCI_READ_REMOTE_VERSION_INFORMATION Macro

File

hci.h

C

```
#define HCI_READ_REMOTE_VERSION_INFORMATION HCI_OPCODE(OGF_LINK_CONTROL, 0x001D)
```

Description

This is macro HCI_READ_REMOTE_VERSION_INFORMATION.

HCI_READ_RSSI Macro

File

hci.h

C

```
#define HCI_READ_RSSI HCI_OPCODE(OGF_STATUS, 0x0005)
```

Description

This is macro HCI_READ_RSSI.

HCI_READ_SCAN_ENABLE Macro

File

hci.h

C

```
#define HCI_READ_SCAN_ENABLE HCI_OPCODE(OGF_CTRL_BASEBAND, 0x0019)
```

Description

This is macro HCI_READ_SCAN_ENABLE.

HCI_READ_SIMPLE_PAIRING_MODE Macro

File

hci.h

C

```
#define HCI_READ_SIMPLE_PAIRING_MODE HCI_OPCODE(OGF_CTRL_BASEBAND, 0x0055)
```

Description

This is macro HCI_READ_SIMPLE_PAIRING_MODE.

HCI_READ_STORED_LINK_KEY Macro

File

hci.h

C

```
#define HCI_READ_STORED_LINK_KEY HCI_OPCODE(OGF_CTRL_BASEBAND, 0x000D)
```

Description

This is macro HCI_READ_STORED_LINK_KEY.

HCI_READ_SYNC_FLOW_CONTROL_ENABLE Macro

File

hci.h

C

```
#define HCI_READ_SYNC_FLOW_CONTROL_ENABLE HCI_OPCODE(OGF_CTRL_BASEBAND, 0x002E)
```

Description

This is macro HCI_READ_SYNC_FLOW_CONTROL_ENABLE.

HCI_READ_TRANSMIT_POWER_LEVEL Macro

File

hci.h

C

```
#define HCI_READ_TRANSMIT_POWER_LEVEL HCI_OPCODE(OGF_CTRL_BASEBAND, 0x002D)
```

Description

This is macro HCI_READ_TRANSMIT_POWER_LEVEL.

HCI_READ_VOICE_SETTING Macro

File

hci.h

C

```
#define HCI_READ_VOICE_SETTING HCI_OPCODE(OGF_CTRL_BASEBAND, 0x0025)
```

Description

This is macro HCI_READ_VOICE_SETTING.

HCI_REJECT_CONNECTION_REQUEST Macro

File

hci.h

C

```
#define HCI_REJECT_CONNECTION_REQUEST HCI_OPCODE(OGF_LINK_CONTROL, 0x000A)
```

Description

This is macro HCI_REJECT_CONNECTION_REQUEST.

HCI_REJECT_SYNCH_CONNECTION_REQUEST Macro

File

hci.h

C

```
#define HCI_REJECT_SYNCH_CONNECTION_REQUEST HCI_OPCODE(OGF_LINK_CONTROL, 0x002A)
```

Description

This is macro HCI_REJECT_SYNCH_CONNECTION_REQUEST.

HCI_REMOTE_NAME_REQUEST Macro

File

hci.h

C

```
#define HCI_REMOTE_NAME_REQUEST HCI_OPCODE(OGF_LINK_CONTROL, 0x0019)
```

Description

This is macro HCI_REMOTE_NAME_REQUEST.

HCI_REMOTE_NAME_REQUEST_CANCEL Macro

File

hci.h

C

```
#define HCI_REMOTE_NAME_REQUEST_CANCEL HCI_OPCODE(OGF_LINK_CONTROL, 0x001A)
```

Description

This is macro HCI_REMOTE_NAME_REQUEST_CANCEL.

HCI_REMOTE_OOB_DATA_REQUEST_NEGATIVE_REPLY Macro

File

hci.h

C

```
#define HCI_REMOTE_OOB_DATA_REQUEST_NEGATIVE_REPLY HCI_OPCODE(OGF_LINK_CONTROL, 0x0033)
```

Description

This is macro HCI_REMOTE_OOB_DATA_REQUEST_NEGATIVE_REPLY.

HCI_REMOTE_OOB_DATA_REQUEST_REPLY Macro

File

hci.h

C

```
#define HCI_REMOTE_OOB_DATA_REQUEST_REPLY HCI_OPCODE(OGF_LINK_CONTROL, 0x0030)
```

Description

This is macro HCI_REMOTE_OOB_DATA_REQUEST_REPLY.

HCI_RESET Macro

File

hci.h

C

```
#define HCI_RESET HCI_OPCODE(OGF_CTRL_BASEBAND, 0x0003)
```

Description

This is macro HCI_RESET.

HCI_RESET_FAILED_CONTACT_COUNTER Macro

File

hci.h

C

```
#define HCI_RESET_FAILED_CONTACT_COUNTER HCI_OPCODE(OGF_STATUS, 0x0002)
```

Description

This is macro HCI_RESET_FAILED_CONTACT_COUNTER.

HCI_ROLE_DISCOVERY Macro

File

hci.h

C

```
#define HCI_ROLE_DISCOVERY HCI_OPCODE(OGF_LINK_POLICY, 0x0009)
```

Description

This is macro HCI_ROLE_DISCOVERY.

HCI_ROLE_SWITCH_ALLOW Macro

File

hci.h

C

```
#define HCI_ROLE_SWITCH_ALLOW 0x01
```

Description

This is macro HCI_ROLE_SWITCH_ALLOW.

HCI_ROLE_SWITCH_DISALLOW Macro

File

hci.h

C

```
#define HCI_ROLE_SWITCH_DISALLOW 0x00
```

Description

This is macro HCI_ROLE_SWITCH_DISALLOW.

HCI_SCAN_INQUIRY Macro

File

hci.h

C

```
#define HCI_SCAN_INQUIRY 0x01
```

Description

Flags for HCI_READ_SCAN_ENABLE/HCI_WRITE_SCAN_ENABLE

HCI_SCAN_PAGE Macro

File

hci.h

C

```
#define HCI_SCAN_PAGE 0x02
```

Description

This is macro HCI_SCAN_PAGE.

HCI_SCO_CONTENT_FORMAT_AIR_CODING_A_LAW Macro

File

hci.h

C

```
#define HCI_SCO_CONTENT_FORMAT_AIR_CODING_A_LAW 0x2
```

Description

This is macro HCI_SCO_CONTENT_FORMAT_AIR_CODING_A_LAW.

HCI_SCO_CONTENT_FORMAT_AIR_CODING_CSVD Macro

File

hci.h

C

```
#define HCI_SCO_CONTENT_FORMAT_AIR_CODING_CSVD 0x0
```

Description

This is macro HCI_SCO_CONTENT_FORMAT_AIR_CODING_CSVD.

HCI_SCO_CONTENT_FORMAT_AIR_CODING_N_LAW Macro

File

hci.h

C

```
#define HCI_SCO_CONTENT_FORMAT_AIR_CODING_N_LAW 0x1
```

Description

This is macro HCI_SCO_CONTENT_FORMAT_AIR_CODING_N_LAW.

HCI_SCO_CONTENT_FORMAT_AIR_CODING_TRANSPARENT Macro

File

hci.h

C

```
#define HCI_SCO_CONTENT_FORMAT_AIR_CODING_TRANSPARENT 0x3
```

Description

This is macro HCI_SCO_CONTENT_FORMAT_AIR_CODING_TRANSPARENT.

HCI_SCO_CONTENT_FORMAT_INPUT_CODING_A_LAW Macro

File

hci.h

C

```
#define HCI_SCO_CONTENT_FORMAT_INPUT_CODING_A_LAW 0x200
```

Description

This is macro HCI_SCO_CONTENT_FORMAT_INPUT_CODING_A_LAW.

HCI_SCO_CONTENT_FORMAT_INPUT_CODING_LINEAR Macro

File

hci.h

C

```
#define HCI_SCO_CONTENT_FORMAT_INPUT_CODING_LINEAR 0
```

Description

This is macro HCI_SCO_CONTENT_FORMAT_INPUT_CODING_LINEAR.

HCI_SCO_CONTENT_FORMAT_INPUT_CODING_N_LAW Macro

File

hci.h

C

```
#define HCI_SCO_CONTENT_FORMAT_INPUT_CODING_N_LAW 0x100
```

Description

This is macro HCI_SCO_CONTENT_FORMAT_INPUT_CODING_N_LAW.

HCI_SCO_CONTENT_FORMAT_INPUT_FORMAT_1_COMPLEMENT Macro

File

hci.h

C

```
#define HCI_SCO_CONTENT_FORMAT_INPUT_FORMAT_1_COMPLEMENT 0
```

Description

This is macro HCI_SCO_CONTENT_FORMAT_INPUT_FORMAT_1_COMPLEMENT.

HCI_SCO_CONTENT_FORMAT_INPUT_FORMAT_2_COMPLEMENT Macro

File

hci.h

C

```
#define HCI_SCO_CONTENT_FORMAT_INPUT_FORMAT_2_COMPLEMENT 0x040
```

Description

This is macro HCI_SCO_CONTENT_FORMAT_INPUT_FORMAT_2_COMPLEMENT.

HCI_SCO_CONTENT_FORMAT_INPUT_FORMAT_SIGN_MAGNITUDE Macro

File

hci.h

C

```
#define HCI_SCO_CONTENT_FORMAT_INPUT_FORMAT_SIGN_MAGNITUDE 0x080
```

Description

This is macro HCI_SCO_CONTENT_FORMAT_INPUT_FORMAT_SIGN_MAGNITUDE.

HCI_SCO_CONTENT_FORMAT_INPUT_FORMAT_UNSIGNED Macro

File

hci.h

C

```
#define HCI_SCO_CONTENT_FORMAT_INPUT_FORMAT_UNSIGNED 0x0C0
```

Description

This is macro HCI_SCO_CONTENT_FORMAT_INPUT_FORMAT_UNSIGNED.

HCI_SCO_CONTENT_FORMAT_INPUT_SAMPLE_SIZE_16 Macro

File

hci.h

C

```
#define HCI_SCO_CONTENT_FORMAT_INPUT_SAMPLE_SIZE_16 0x020
```

Description

This is macro HCI_SCO_CONTENT_FORMAT_INPUT_SAMPLE_SIZE_16.

HCI_SCO_CONTENT_FORMAT_INPUT_SAMPLE_SIZE_8 Macro

File

hci.h

C

```
#define HCI_SCO_CONTENT_FORMAT_INPUT_SAMPLE_SIZE_8 0x0
```

Description

This is macro HCI_SCO_CONTENT_FORMAT_INPUT_SAMPLE_SIZE_8.

HCI_SCO_DATA_HEADER_LEN Macro

File

hci.h

C

```
#define HCI_SCO_DATA_HEADER_LEN 3
```

Description

This is macro HCI_SCO_DATA_HEADER_LEN.

HCI_SCO_MAX_DATA_LEN Macro

File

hci.h

C

```
#define HCI_SCO_MAX_DATA_LEN 255
```

Description

This is macro HCI_SCO_MAX_DATA_LEN.

HCI_SCO_MAX_LATENCY_DONTCARE Macro

File

hci.h

C

```
#define HCI_SCO_MAX_LATENCY_DONTCARE 0xFFFF
```

Description

This is macro HCI_SCO_MAX_LATENCY_DONTCARE.

HCI_SCO_PACKET_TYPE_ALL Macro

File

hci.h

C

```
#define HCI_SCO_PACKET_TYPE_ALL 0x003F
```

Description

This is macro HCI_SCO_PACKET_TYPE_ALL.

HCI_SCO_PACKET_TYPE_EV3 Macro

File

hci.h

C

```
#define HCI_SCO_PACKET_TYPE_EV3 0x0008
```

Description

This is macro HCI_SCO_PACKET_TYPE_EV3.

HCI_SCO_PACKET_TYPE_EV4 Macro

File

hci.h

C

```
#define HCI_SCO_PACKET_TYPE_EV4 0x0010
```

Description

This is macro HCI_SCO_PACKET_TYPE_EV4.

HCI_SCO_PACKET_TYPE_EV5 Macro

File

hci.h

C

```
#define HCI_SCO_PACKET_TYPE_EV5 0x0020
```

Description

This is macro HCI_SCO_PACKET_TYPE_EV5.

HCI_SCO_PACKET_TYPE_HV1 Macro

File

hci.h

C

```
#define HCI_SCO_PACKET_TYPE_HV1 0x0001
```

Description

This is macro HCI_SCO_PACKET_TYPE_HV1.

HCI_SCO_PACKET_TYPE_HV2 Macro

File

hci.h

C

```
#define HCI_SCO_PACKET_TYPE_HV2 0x0002
```

Description

This is macro HCI_SCO_PACKET_TYPE_HV2.

HCI_SCO_PACKET_TYPE_HV3 Macro

File

hci.h

C

```
#define HCI_SCO_PACKET_TYPE_HV3 0x0004
```

Description

This is macro HCI_SCO_PACKET_TYPE_HV3.

HCI_SCO_PACKET_TYPE_NO_2_EV3 Macro

File

hci.h

C

```
#define HCI_SCO_PACKET_TYPE_NO_2_EV3 0x0040
```

Description

This is macro HCI_SCO_PACKET_TYPE_NO_2_EV3.

HCI_SCO_PACKET_TYPE_NO_2_EV5 Macro

File

hci.h

C

```
#define HCI_SCO_PACKET_TYPE_NO_2_EV5 0x0100
```

Description

This is macro HCI_SCO_PACKET_TYPE_NO_2_EV5.

HCI_SCO_PACKET_TYPE_NO_3_EV3 Macro

File

hci.h

C

```
#define HCI_SCO_PACKET_TYPE_NO_3_EV3 0x0080
```

Description

This is macro HCI_SCO_PACKET_TYPE_NO_3_EV3.

HCI_SCO_PACKET_TYPE_NO_3_EV5 Macro

File

hci.h

C

```
#define HCI_SCO_PACKET_TYPE_NO_3_EV5 0x0200
```

Description

This is macro HCI_SCO_PACKET_TYPE_NO_3_EV5.

HCI_SCO_RTX_EFFORT_DONTCARE Macro

File

hci.h

C

```
#define HCI_SCO_RTX_EFFORT_DONTCARE 0xFF
```

Description

This is macro HCI_SCO_RTX_EFFORT_DONTCARE.

HCI_SCO_RTX_EFFORT_NO_RETRANSMISSION Macro

File

hci.h

C

```
#define HCI_SCO_RTX_EFFORT_NO_RETRANSMISSION 0x0
```

Description

This is macro HCI_SCO_RTX_EFFORT_NO_RETRANSMISSION.

HCI_SCO_RTX_EFFORT_OPTIMIZE_LINK_QUALITY Macro

File

hci.h

C

```
#define HCI_SCO_RTX_EFFORT_OPTIMIZE_LINK_QUALITY 0x2
```

Description

This is macro HCI_SCO_RTX_EFFORT_OPTIMIZE_LINK_QUALITY.

HCI_SCO_RTX_EFFORT_OPTIMIZE_POWER_CONSUMPTION Macro

File

hci.h

C

```
#define HCI_SCO_RTX_EFFORT_OPTIMIZE_POWER_CONSUMPTION 0x1
```

Description

This is macro HCI_SCO_RTX_EFFORT_OPTIMIZE_POWER_CONSUMPTION.

HCI_SCO_RX_BANDWIDTH_DONTCARE Macro

File

hci.h

C

```
#define HCI_SCO_RX_BANDWIDTH_DONTCARE 0xFFFFFFFF
```

Description

This is macro HCI_SCO_RX_BANDWIDTH_DONTCARE.

HCI_SCO_TX_BANDWIDTH_DONTCARE Macro

File

hci.h

C

```
#define HCI_SCO_TX_BANDWIDTH_DONTCARE 0xFFFFFFFF
```

Description

This is macro HCI_SCO_TX_BANDWIDTH_DONTCARE.

HCI_SEND_DATA_STATUS_INTERRUPTED Macro

File

hci.h

C

```
#define HCI_SEND_DATA_STATUS_INTERRUPTED 1
```

Description

This is macro HCI_SEND_DATA_STATUS_INTERRUPTED.

HCI_SEND_DATA_STATUS_SUCCESS Macro

File

hci.h

C

```
#define HCI_SEND_DATA_STATUS_SUCCESS 0
```

Description

This is macro HCI_SEND_DATA_STATUS_SUCCESS.

HCI_SEND_KEY_PRESS_NOTIFICATION Macro

File

hci.h

C

```
#define HCI_SEND_KEY_PRESS_NOTIFICATION HCI_OPCODE(OGF_CTRL_BASEBAND, 0x0060)
```

Description

This is macro HCI_SEND_KEY_PRESS_NOTIFICATION.

HCI_SET_AFH_HOST_CHANNEL_CLASSIFICATION Macro

File

hci.h

C

```
#define HCI_SET_AFH_HOST_CHANNEL_CLASSIFICATION HCI_OPCODE(OGF_CTRL_BASEBAND, 0x003F)
```

Description

This is macro HCI_SET_AFH_HOST_CHANNEL_CLASSIFICATION.

HCI_SET_CONNECTION_ENCRYPTION Macro

File

hci.h

C

```
#define HCI_SET_CONNECTION_ENCRYPTION HCI_OPCODE(OGF_LINK_CONTROL, 0x0013)
```

Description

This is macro HCI_SET_CONNECTION_ENCRYPTION.

HCI_SET_CTRL_TO_HOST_FLOW_CONTROL Macro

File

hci.h

C

```
#define HCI_SET_CTRL_TO_HOST_FLOW_CONTROL HCI_OPCODE(OGF_CTRL_BASEBAND, 0x0031)
```

Description

This is macro HCI_SET_CTRL_TO_HOST_FLOW_CONTROL.

HCI_SET_EVENT_FILTER Macro

File

hci.h

C

```
#define HCI_SET_EVENT_FILTER HCI_OPCODE(OGF_CTRL_BASEBAND, 0x0005)
```

Description

This is macro HCI_SET_EVENT_FILTER.

HCI_SET_EVENT_MASK Macro

File

hci.h

C

```
#define HCI_SET_EVENT_MASK HCI_OPCODE(OGF_CTRL_BASEBAND, 0x0001)
```

Description

addtogroup hci @{

@name Controller & Baseband commands

details The Controller & Baseband Commands provide access and control to various capabilities of the Bluetooth hardware.

HCI_SETUP_SYNCHRONOUS_CONNECTION Macro

File

hci.h

C

```
#define HCI_SETUP_SYNCHRONOUS_CONNECTION HCI_OPCODE(OGF_LINK_CONTROL, 0x0028)
```

Description

This is macro HCI_SETUP_SYNCHRONOUS_CONNECTION.

HCI_SNIFF_MODE Macro

File

hci.h

C

```
#define HCI_SNIFF_MODE HCI_OPCODE(OGF_LINK_POLICY, 0x0003)
```

Description

This is macro HCI_SNIFF_MODE.

HCI_SNIFF_SUBRATING Macro

File

hci.h

C

```
#define HCI_SNIFF_SUBRATING HCI_OPCODE(OGF_LINK_POLICY, 0x0011)
```

Description

This is macro HCI_SNIFF_SUBRATING.

HCI_SWITCH_ROLE Macro

File

hci.h

C

```
#define HCI_SWITCH_ROLE HCI_OPCODE(OGF_LINK_POLICY, 0x000B)
```

Description

This is macro HCI_SWITCH_ROLE.

HCI_TRANSPORT_HEADER_LEN Macro

File

hci_transport.h

C

```
#define HCI_TRANSPORT_HEADER_LEN 1
```

Description

This is macro HCI_TRANSPORT_HEADER_LEN.

HCI_UART_PACKET_TYPE_ACL_DATA Macro

File

hcitr_uart.h

C

```
#define HCI_UART_PACKET_TYPE_ACL_DATA 2
```

Description

This is macro HCI_UART_PACKET_TYPE_ACL_DATA.

HCI_UART_PACKET_TYPE_COMMAND Macro

File

hcitr_uart.h

C

```
#define HCI_UART_PACKET_TYPE_COMMAND 1
```

Description

defgroup hcitr_uart HCI UART (H4) transport protocol ingroup hcitr

details This module describes functions used to initialize and start HCI UART transport protocol. The transport uses common interface for exchanging data between the host CPU and HCI controller defined in bt_hcitr.h. This interface consist of two functions that must be implemented by the application: li bt_oem_send() li bt_oem_recv()

HCI_UART_PACKET_TYPE_EVENT Macro

File

hcitr_uart.h

C

```
#define HCI_UART_PACKET_TYPE_EVENT 4
```

Description

This is macro HCI_UART_PACKET_TYPE_EVENT.

HCI_UART_PACKET_TYPE_SCO_DATA Macro

File

hcitr_uart.h

C

```
#define HCI_UART_PACKET_TYPE_SCO_DATA 3
```

Description

This is macro HCI_UART_PACKET_TYPE_SCO_DATA.

HCI_USER_CONFIRMATION_REQ_NEGATIVE_REPLY Macro

File

hci.h

C

```
#define HCI_USER_CONFIRMATION_REQ_NEGATIVE_REPLY HCI_OPCODE(OGF_LINK_CONTROL, 0x002D)
```

Description

This is macro HCI_USER_CONFIRMATION_REQ_NEGATIVE_REPLY.

HCI_USER_CONFIRMATION_REQUEST_REPLY Macro

File

hci.h

C

```
#define HCI_USER_CONFIRMATION_REQUEST_REPLY HCI_OPCODE(OGF_LINK_CONTROL, 0x002C)
```

Description

This is macro HCI_USER_CONFIRMATION_REQUEST_REPLY.

HCI_USER_PASSKEY_REQUEST_NEGATIVE_REPLY Macro

File

hci.h

C

```
#define HCI_USER_PASSKEY_REQUEST_NEGATIVE_REPLY HCI_OPCODE(OGF_LINK_CONTROL, 0x002F)
```

Description

This is macro HCI_USER_PASSKEY_REQUEST_NEGATIVE_REPLY.

HCI_USER_PASSKEY_REQUEST_REPLY Macro

File

hci.h

C

```
#define HCI_USER_PASSKEY_REQUEST_REPLY HCI_OPCODE(OGF_LINK_CONTROL, 0x002E)
```

Description

This is macro HCI_USER_PASSKEY_REQUEST_REPLY.

HCI_WRITE_AFH_CHANNEL_ASSESSMENT_MODE Macro

File

hci.h

C

```
#define HCI_WRITE_AFH_CHANNEL_ASSESSMENT_MODE HCI_OPCODE(OGF_CTRL_BASEBAND, 0x0049)
```

Description

This is macro HCI_WRITE_AFH_CHANNEL_ASSESSMENT_MODE.

HCI_WRITE_AUTHENTICATION_ENABLE Macro

File

hci.h

C

```
#define HCI_WRITE_AUTHENTICATION_ENABLE HCI_OPCODE(OGF_CTRL_BASEBAND, 0x0020)
```

Description

This is macro HCI_WRITE_AUTHENTICATION_ENABLE.

HCI_WRITE_AUTOMATIC_FLASH_TIMEOUT Macro

File

hci.h

C

```
#define HCI_WRITE_AUTOMATIC_FLASH_TIMEOUT HCI_OPCODE(OGF_CTRL_BASEBAND, 0x0028)
```

Description

This is macro HCI_WRITE_AUTOMATIC_FLASH_TIMEOUT.

HCI_WRITE_CLASS_OF_DEVICE Macro

File

hci.h

C

```
#define HCI_WRITE_CLASS_OF_DEVICE HCI_OPCODE(OGF_CTRL_BASEBAND, 0x0024)
```

Description

This is macro HCI_WRITE_CLASS_OF_DEVICE.

HCI_WRITE_CONNECTION_ACCEPT_TIMEOUT Macro

File

hci.h

C

```
#define HCI_WRITE_CONNECTION_ACCEPT_TIMEOUT HCI_OPCODE(OGF_CTRL_BASEBAND, 0x0016)
```

Description

This is macro HCI_WRITE_CONNECTION_ACCEPT_TIMEOUT.

HCI_WRITE_CURRENT_IAC_LAP Macro

File

hci.h

C

```
#define HCI_WRITE_CURRENT_IAC_LAP HCI_OPCODE(OGF_CTRL_BASEBAND, 0x003A)
```

Description

This is macro HCI_WRITE_CURRENT_IAC_LAP.

HCI_WRITE_DEFAULT_ERRONEOUS_DATA_REPORTING Macro

File

hci.h

C

```
#define HCI_WRITE_DEFAULT_ERRONEOUS_DATA_REPORTING HCI_OPCODE(OGF_CTRL_BASEBAND, 0x005B)
```

Description

This is macro HCI_WRITE_DEFAULT_ERRONEOUS_DATA_REPORTING.

HCI_WRITE_DEFAULT_POLICY_SETTINGS Macro

File

hci.h

C

```
#define HCI_WRITE_DEFAULT_POLICY_SETTINGS HCI_OPCODE(OGF_LINK_POLICY, 0x000F)
```

Description

This is macro HCI_WRITE_DEFAULT_POLICY_SETTINGS.

HCI_WRITE_ENCRYPTION_MODE Macro

File

hci.h

C

```
#define HCI_WRITE_ENCRYPTION_MODE HCI_OPCODE(OGF_CTRL_BASEBAND, 0x0022)
```

Description

This is macro HCI_WRITE_ENCRYPTION_MODE.

HCI_WRITE_EXTENDED_INQUIRY_RESPONSE Macro

File

hci.h

C

```
#define HCI_WRITE_EXTENDED_INQUIRY_RESPONSE HCI_OPCODE(OGF_CTRL_BASEBAND, 0x0052)
```

Description

This is macro HCI_WRITE_EXTENDED_INQUIRY_RESPONSE.

HCI_WRITE_EXTENDED_INQUIRY_RESPONSE_PARAM_LEN Macro

File

hci.h

C

```
#define HCI_WRITE_EXTENDED_INQUIRY_RESPONSE_PARAM_LEN 241
```

Description

This is macro HCI_WRITE_EXTENDED_INQUIRY_RESPONSE_PARAM_LEN.

HCI_WRITE_HOLD_MODE_ACTIVITY Macro

File

hci.h

C

```
#define HCI_WRITE_HOLD_MODE_ACTIVITY HCI_OPCODE(OGF_CTRL_BASEBAND, 0x002C)
```

Description

This is macro HCI_WRITE_HOLD_MODE_ACTIVITY.

HCI_WRITE_INQUIRY_MODE Macro

File

hci.h

C

```
#define HCI_WRITE_INQUIRY_MODE HCI_OPCODE(OGF_CTRL_BASEBAND, 0x0045)
```

Description

This is macro HCI_WRITE_INQUIRY_MODE.

HCI_WRITE_INQUIRY_SCAN_ACTIVITY Macro

File

hci.h

C

```
#define HCI_WRITE_INQUIRY_SCAN_ACTIVITY HCI_OPCODE(OGF_CTRL_BASEBAND, 0x001E)
```

Description

This is macro HCI_WRITE_INQUIRY_SCAN_ACTIVITY.

HCI_WRITE_INQUIRY_SCAN_TYPE Macro

File

hci.h

C

```
#define HCI_WRITE_INQUIRY_SCAN_TYPE HCI_OPCODE(OGF_CTRL_BASEBAND, 0x0043)
```

Description

This is macro HCI_WRITE_INQUIRY_SCAN_TYPE.

HCI_WRITE_INQUIRY_TX_POWER_LEVEL Macro

File

hci.h

C

```
#define HCI_WRITE_INQUIRY_TX_POWER_LEVEL HCI_OPCODE(OGF_CTRL_BASEBAND, 0x0059)
```

Description

This is macro HCI_WRITE_INQUIRY_TX_POWER_LEVEL.

HCI_WRITE_LE_HOST_SUPPORT Macro

File

hci.h

C

```
#define HCI_WRITE_LE_HOST_SUPPORT HCI_OPCODE(OGF_CTRL_BASEBAND, 0x006D)
```

Description

This is macro HCI_WRITE_LE_HOST_SUPPORT.

HCI_WRITE_LINK_POLICY_SETTINGS Macro

File

hci.h

C

```
#define HCI_WRITE_LINK_POLICY_SETTINGS HCI_OPCODE(OGF_LINK_POLICY, 0x000D)
```

Description

This is macro HCI_WRITE_LINK_POLICY_SETTINGS.

HCI_WRITE_LINK_SUPERVISION_TIMEOUT Macro

File

hci.h

C

```
#define HCI_WRITE_LINK_SUPERVISION_TIMEOUT HCI_OPCODE(OGF_CTRL_BASEBAND, 0x0037)
```

Description

This is macro HCI_WRITE_LINK_SUPERVISION_TIMEOUT.

HCI_WRITE_LOCAL_NAME Macro

File

hci.h

C

```
#define HCI_WRITE_LOCAL_NAME HCI_OPCODE(OGF_CTRL_BASEBAND, 0x0013)
```

Description

This is macro HCI_WRITE_LOCAL_NAME.

HCI_WRITE_LOCAL_NAME_PARAM_LEN Macro

File

hci.h

C

```
#define HCI_WRITE_LOCAL_NAME_PARAM_LEN 248
```

Description

This is macro HCI_WRITE_LOCAL_NAME_PARAM_LEN.

HCI_WRITE_LOOPBACK_MODE Macro

File

hci.h

C

```
#define HCI_WRITE_LOOPBACK_MODE HCI_OPCODE(OGF_TESTING, 0x0002)
```

Description

This is macro HCI_WRITE_LOOPBACK_MODE.

HCI_WRITE_NUM_BROADCST_RETR Macro

File

hci.h

C

```
#define HCI_WRITE_NUM_BROADCST_RETR HCI_OPCODE(OGF_CTRL_BASEBAND, 0x002A)
```

Description

This is macro HCI_WRITE_NUM_BROADCST_RETR.

HCI_WRITE_PAGE_SCAN_ACTIVITY Macro

File

hci.h

C

```
#define HCI_WRITE_PAGE_SCAN_ACTIVITY HCI_OPCODE(OGF_CTRL_BASEBAND, 0x001C)
```

Description

This is macro HCI_WRITE_PAGE_SCAN_ACTIVITY.

HCI_WRITE_PAGE_SCAN_PERIOD_MODE Macro

File

hci.h

C

```
#define HCI_WRITE_PAGE_SCAN_PERIOD_MODE HCI_OPCODE(OGF_CTRL_BASEBAND, 0x003C)
```

Description

This is macro HCI_WRITE_PAGE_SCAN_PERIOD_MODE.

HCI_WRITE_PAGE_SCAN_TYPE Macro

File

hci.h

C

```
#define HCI_WRITE_PAGE_SCAN_TYPE HCI_OPCODE(OGF_CTRL_BASEBAND, 0x0047)
```

Description

This is macro HCI_WRITE_PAGE_SCAN_TYPE.

HCI_WRITE_PAGE_TIMEOUT Macro

File

hci.h

C

```
#define HCI_WRITE_PAGE_TIMEOUT HCI_OPCODE(OGF_CTRL_BASEBAND, 0x0018)
```

Description

This is macro HCI_WRITE_PAGE_TIMEOUT.

HCI_WRITE_PIN_TYPE Macro

File

hci.h

C

```
#define HCI_WRITE_PIN_TYPE HCI_OPCODE(OGF_CTRL_BASEBAND, 0x000A)
```

Description

This is macro HCI_WRITE_PIN_TYPE.

HCI_WRITE_SCAN_ENABLE Macro

File

hci.h

C

```
#define HCI_WRITE_SCAN_ENABLE HCI_OPCODE(OGF_CTRL_BASEBAND, 0x001A)
```

Description

This is macro HCI_WRITE_SCAN_ENABLE.

HCI_WRITE_SIMPLE_PAIRING_DEBUG_MODE Macro

File

hci.h

C

```
#define HCI_WRITE_SIMPLE_PAIRING_DEBUG_MODE HCI_OPCODE(OGF_TESTING, 0x0004)
```

Description

This is macro HCI_WRITE_SIMPLE_PAIRING_DEBUG_MODE.

HCI_WRITE_SIMPLE_PAIRING_MODE Macro

File

hci.h

C

```
#define HCI_WRITE_SIMPLE_PAIRING_MODE HCI_OPCODE(OGF_CTRL_BASEBAND, 0x0056)
```

Description

This is macro HCI_WRITE_SIMPLE_PAIRING_MODE.

HCI_WRITE_STORED_LINK_KEY Macro

File

hci.h

C

```
#define HCI_WRITE_STORED_LINK_KEY HCI_OPCODE(OGF_CTRL_BASEBAND, 0x0011)
```

Description

This is macro HCI_WRITE_STORED_LINK_KEY.

HCI_WRITE_SYNC_FLOW_CONTROL_ENABLE Macro

File

hci.h

C

```
#define HCI_WRITE_SYNC_FLOW_CONTROL_ENABLE HCI_OPCODE(OGF_CTRL_BASEBAND, 0x002F)
```

Description

This is macro HCI_WRITE_SYNC_FLOW_CONTROL_ENABLE.

HCI_WRITE_VOICE_SETTING Macro

File

hci.h

C

```
#define HCI_WRITE_VOICE_SETTING HCI_OPCODE(OGF_CTRL_BASEBAND, 0x0026)
```

Description

This is macro HCI_WRITE_VOICE_SETTING.

HCITR_3WIRE_DEFAULT_ACK_TIMEOUT Macro

File

hcitr_3wire.h

C

```
#define HCITR_3WIRE_DEFAULT_ACK_TIMEOUT 250
```

Description

This is macro HCITR_3WIRE_DEFAULT_ACK_TIMEOUT.

HCITR_BCSP_DEFAULT_ACK_TIMEOUT Macro

File

hcitr_bcsp.h

C

```
#define HCITR_BCSP_DEFAULT_ACK_TIMEOUT 250
```

Description

This is macro HCITR_BCSP_DEFAULT_ACK_TIMEOUT.

LM_PACKET_TYPE_DH1 Macro

File

lm.h

C

```
#define LM_PACKET_TYPE_DH1 0x0010
```

Description

This is macro LM_PACKET_TYPE_DH1.

LM_PACKET_TYPE_DH3 Macro

File

lm.h

C

```
#define LM_PACKET_TYPE_DH3 0x0800
```

Description

This is macro LM_PACKET_TYPE_DH3.

LM_PACKET_TYPE_DH5 Macro

File

lm.h

C

```
#define LM_PACKET_TYPE_DH5 0x8000
```

Description

This is macro LM_PACKET_TYPE_DH5.

LM_PACKET_TYPE_DM1 Macro

File

lm.h

C

```
#define LM_PACKET_TYPE_DM1 0x0008
```

Description

This is macro LM_PACKET_TYPE_DM1.

LM_PACKET_TYPE_DM3 Macro

File

lm.h

C

```
#define LM_PACKET_TYPE_DM3 0x0400
```

Description

This is macro LM_PACKET_TYPE_DM3.

LM_PACKET_TYPE_DM5 Macro

File

lm.h

C

```
#define LM_PACKET_TYPE_DM5 0x4000
```

Description

This is macro LM_PACKET_TYPE_DM5.

OGF_CTRL_BASEBAND Macro

File

hci.h

C

```
#define OGF_CTRL_BASEBAND 0x03
```

Description

This is macro OGF_CTRL_BASEBAND.

OGF_INFORMATION Macro

File

hci.h

C

```
#define OGF_INFORMATION 0x04
```

Description

This is macro OGF_INFORMATION.

OGF_LE Macro

File

hci.h

C

```
#define OGF_LE 0x08
```

Description

This is macro OGF_LE.

OGF_LINK_CONTROL Macro

File

hci.h

C

```
#define OGF_LINK_CONTROL 0x01
```

Description

This is macro OGF_LINK_CONTROL.

OGF_LINK_POLICY Macro

File

hci.h

C

```
#define OGF_LINK_POLICY 0x02
```

Description

This is macro OGF_LINK_POLICY.

OGF_STATUS Macro

File

hci.h

C

```
#define OGF_STATUS 0x05
```

Description

This is macro OGF_STATUS.

OGF_TESTING Macro

File

hci.h

C

```
#define OGF_TESTING 0x06
```

Description

This is macro OGF_TESTING.

OGF_VENDOR Macro

File

hci.h

C

```
#define OGF_VENDOR 0x3F
```

Description

This is macro OGF_VENDOR.

bt_hci_cmd_callback_fp Type

File

hci.h

C

```
typedef void (* bt_hci_cmd_callback_fp)(bt_int, struct _bt_hci_command_s*, struct _bt_hci_event_s*);
```

Description

This is type bt_hci_cmd_callback_fp.

bt_hci_cmd_listener_fp Type

File

hci_ctrl_state.h

C

```
typedef void (* bt_hci_cmd_listener_fp)(bt_uint event_id, bt_hci_command_t* cmd, void* cb_param);
```

Description

This is type bt_hci_cmd_listener_fp.

bt_hci_command_p Structure

File

hci.h

C

```
typedef struct _bt_hci_command_s {
  struct _bt_hci_command_s* next_cmd;
  bt_int opcode;
  bt_int params_len;
  bt_byte* params;
  bt_hci_cmd_callback_fp callback;
  void* callback_param;
  bt_int packetNum;
} bt_hci_command_t, * bt_hci_command_p;
```

Description

This is type bt_hci_command_p.

bt_hci_command_t Structure

File

hci.h

C

```
typedef struct _bt_hci_command_s {
  struct _bt_hci_command_s* next_cmd;
  bt_int opcode;
  bt_int params_len;
  bt_byte* params;
  bt_hci_cmd_callback_fp callback;
  void* callback_param;
  bt_int packetNum;
} bt_hci_command_t, * bt_hci_command_p;
```

Description

This is type bt_hci_command_t.

bt_hci_conn_state_p Type

File

hci.h

C

```
typedef struct _bt_hci_conn_state_s * bt_hci_conn_state_p;
```

Description

This is type bt_hci_conn_state_p.

bt_hci_conn_state_t Type

File

hci.h

C

```
typedef struct _bt_hci_conn_state_s bt_hci_conn_state_t;
```

Description

This is type bt_hci_conn_state_t.

bt_hci_connect_callback_fp Type

File

hci.h

C

```
typedef void (* bt_hci_connect_callback_fp)(bt_byte status, bt_hci_conn_state_t *pconn, void* param);
```

Description

brief HCI connect callback. ingroup hci

details This typedef defines a type for the callback function that is called when HCI connect operation initiated by a call to bt_hci_connect() is complete.

param status Operation status. c It is 0 if connection was successfully established. param pconn pointer to a structure representing the established connection. param param pointer to arbitrary data passed to the bt_hci_connect() function through its c param parameter.

bt_hci_ctrl_listener_t Type

File

hci_ctrl_state.h

C

```
typedef struct _bt_hci_ctrl_listener_t bt_hci_ctrl_listener_t;
```

Description

This is type bt_hci_ctrl_listener_t.

bt_hci_ctrl_state_t Structure

File

hci_ctrl_state.h

C

```
typedef struct _bt_hci_ctrl_state_s {
  bt_byte state;
  bt_byte default_link_policy;
  bt_bdaddr_t bdaddr;
  bt_int num_hci_cmd_packets;
  bt_int total_num_acl_data_packets;
  bt_int total_num_sco_data_packets;
  bt_int aclDataPacketLen;
  bt_byte scoDataPacketLen;
  bt_hci_conn_state_t* connections;
  bt_ulong _discoverable_period;
  bt_ulong _connectable_period;
  bt_byte _last_cmd_status;
  bt_hci_le_ctrl_state_t* le_ctrl_state;
  bt_ulong event_mask_l;
  bt_ulong event_mask_h;
  bt_byte incoming_connection_role;
  bt_byte inquiry_response_tx_power_level;
  struct _bt_l2cap_mgr_s* l2cap_mgr;
  pf_l2cap_receive_callback l2cap_data_receive_callback;
  bt_hci_start_callback_fp _init_cb;
  void* _init_param;
  bt_hci_connect_callback_fp _connect_cb;
  void* _connect_param;
  bt_uint _connect_acl_config;
  bt_hci_connect_callback_fp _listen_cb;
  void* _listen_param;
  bt_ulong sco_tx_bandwidth;
  bt_ulong sco_rcv_bandwidth;
  bt_uint sco_max_latency;
  bt_uint sco_content_format;
  bt_byte sco_retransmission_effort;
```

```
    bt_uint sco_packet_type;
    bt_hci_connect_callback_fp sco_listen_cb;
    void* _sco_listen_param;
    bt_hci_connect_callback_fp sco_connect_cb;
    void* _sco_connect_param;
    bt_hci_inquiry_callback_fp inquiry_cb;
    bt_hci_request_remote_name_callback_fp remote_name_cb;
    bt_byte inquiry_max_responses;
    bt_byte inquiry_length;
    pf_hci_sleep_callback sleep_cb;
    pf_hci_wakeup_callback wakeup_cb;
    bt_hci_ctrl_listener_t* listeners;
  } bt_hci_ctrl_state_t;
```

Description

This is type bt_hci_ctrl_state_t.

bt_hci_data_buffer_p Structure

File

hci_data_buffer.h

C

```
typedef struct _bt_hci_data_buffer_s {
  bt_int state;
  bt_hci_data_t packet;
} bt_hci_data_buffer_t, * bt_hci_data_buffer_p;
```

Description

This is type bt_hci_data_buffer_p.

bt_hci_data_buffer_t Structure

File

hci_data_buffer.h

C

```
typedef struct _bt_hci_data_buffer_s {
  bt_int state;
  bt_hci_data_t packet;
} bt_hci_data_buffer_t, * bt_hci_data_buffer_p;
```

Description

This is type bt_hci_data_buffer_t.

bt_hci_data_callback_fp Type

File

hci.h

C

```
typedef void (* bt_hci_data_callback_fp)(bt_byte status, struct _bt_hci_data_s*);
```

Description

This is type bt_hci_data_callback_fp.

bt_hci_data_listener_fp Type

File

hci_ctrl_state.h

C

```
typedef void (* bt_hci_data_listener_fp)(bt_hci_conn_state_t* connection, const bt_byte* data, bt_int len,
```

```
bt_bool sent, void* cb_param);
```

Description

This is type bt_hci_data_listener_fp.

bt_hci_data_p Structure

File

hci.h

C

```
typedef struct _bt_hci_data_s {
  struct _bt_hci_data_s* next_data;
  bt_hci_hconn_t hconn;
  bt_uint data_len;
  bt_byte_p pdata;
  bt_uint pos;
  bt_byte link_type;
  bt_hci_data_callback_fp callback;
  void * param;
  bt_int packetNum;
} bt_hci_data_t, * bt_hci_data_p;
```

Description

This is type bt_hci_data_p.

bt_hci_data_t Structure

File

hci.h

C

```
typedef struct _bt_hci_data_s {
  struct _bt_hci_data_s* next_data;
  bt_hci_hconn_t hconn;
  bt_uint data_len;
  bt_byte_p pdata;
  bt_uint pos;
  bt_byte link_type;
  bt_hci_data_callback_fp callback;
  void * param;
  bt_int packetNum;
} bt_hci_data_t, * bt_hci_data_p;
```

Description

This is type bt_hci_data_t.

bt_hci_disconnect_callback_fp Type

File

hci.h

C

```
typedef void (* bt_hci_disconnect_callback_fp)(bt_byte status, bt_byte reason, bt_hci_conn_state_t *pconn,
void* param);
```

Description

brief HCI disconnect callback. ingroup hci

details This typedef defines a type for the callback function that is called when an HCI connection has been terminated.

param status Operation status. c It is 0 if connection has been successfully terminated. param reason Reason for disconnection. param pconn pointer to a structure representing the connection. param param pointer to arbitrary data associated with an event listener.

bt_hci_event_e Union

File

hci_conn_state.h

C

```
typedef union _bt_hci_event_e {
  bt_hci_evt_disconnection_complete_t disconnect;
} bt_hci_event_e;
```

Description

This is type bt_hci_event_e.

bt_hci_event_handler_ex_fp Type

File

hci_evt_handlers.h

C

```
typedef void (* bt_hci_event_handler_ex_fp)(bt_hci_event_p, bt_uint params_len);
```

Description

This is type bt_hci_event_handler_ex_fp.

bt_hci_event_handler_fp Type

File

hci_evt_handlers.h

C

```
typedef void (* bt_hci_event_handler_fp)(bt_hci_event_p);
```

Description

This is type bt_hci_event_handler_fp.

bt_hci_event_listener_fp Type

File

hci.h

C

```
typedef void (* bt_hci_event_listener_fp)(bt_int evcode, void* evt_params, void* cb_param);
```

Description

This is type bt_hci_event_listener_fp.

bt_hci_event_p Structure

File

hci.h

C

```
typedef struct _bt_hci_event_s {
  bt_int evcode;
  bt_int params_len;
  bt_byte* params;
} bt_hci_event_t, * bt_hci_event_p;
```

Description

This is type bt_hci_event_p.

bt_hci_event_t Structure

File

hci.h

C

```
typedef struct _bt_hci_event_s {
  bt_int evcode;
  bt_int params_len;
  bt_byte* params;
} bt_hci_event_t, * bt_hci_event_p;
```

Description

This is type bt_hci_event_t.

bt_hci_evt_authentication_complete_t Structure

File

hci_evt_handlers.h

C

```
typedef struct _bt_hci_evt_authentication_complete_t {
  bt_byte status;
  bt_hci_conn_state_t* pconn;
} bt_hci_evt_authentication_complete_t;
```

Description

This is type bt_hci_evt_authentication_complete_t.

bt_hci_evt_command_complete_t Structure

File

hci_evt_handlers.h

C

```
typedef struct _bt_hci_evt_command_complete_t {
  bt_byte status;
  bt_int opcode;
  bt_hci_command_t* cmd;
} bt_hci_evt_command_complete_t;
```

Description

This is type bt_hci_evt_command_complete_t.

bt_hci_evt_command_status_t Structure

File

hci_evt_handlers.h

C

```
typedef struct _bt_hci_evt_command_status_t {
  bt_byte status;
  bt_int opcode;
  bt_hci_command_t* cmd;
} bt_hci_evt_command_status_t;
```

Description

This is type bt_hci_evt_command_status_t.

bt_hci_evt_connection_complete_t Structure

File

hci_evt_handlers.h

C

```
typedef struct _bt_hci_evt_connection_complete_t {
  bt_byte status;
  bt_hci_conn_state_t* pconn;
  bt_bool incomming;
} bt_hci_evt_connection_complete_t;
```

Description

This is type bt_hci_evt_connection_complete_t.

bt_hci_evt_connection_request_t Structure

File

hci_evt_handlers.h

C

```
typedef struct _bt_hci_evt_connection_request_t {
  bt_bdaddr_t bdaddr;
  bt_long dev_class;
  bt_byte link_type;
} bt_hci_evt_connection_request_t;
```

Description

This is type bt_hci_evt_connection_request_t.

bt_hci_evt_disconnection_complete_t Structure

File

hci_evt_handlers.h

C

```
typedef struct _bt_hci_evt_disconnection_complete_t {
  bt_byte status;
  bt_hci_conn_state_t* pconn;
  bt_byte reason;
} bt_hci_evt_disconnection_complete_t;
```

Description

This is type bt_hci_evt_disconnection_complete_t.

bt_hci_evt_encryption_change_t Structure

File

hci_evt_handlers.h

C

```
typedef struct _bt_hci_evt_encryption_change_s {
  bt_byte status;
  bt_hci_conn_state_t* pconn;
  bt_byte enabled;
} bt_hci_evt_encryption_change_t;
```

Description

This is type bt_hci_evt_encryption_change_t.

bt_hci_evt_mode_change_t Structure

File

hci_evt_handlers.h

C

```
typedef struct _bt_hci_evt_mode_change_t {
  bt_hci_conn_state_t* pconn;
  bt_byte mode;
  bt_int interval;
} bt_hci_evt_mode_change_t;
```

Description

This is type bt_hci_evt_mode_change_t.

bt_hci_evt_role_change_t Structure

File

hci_evt_handlers.h

C

```
typedef struct _bt_hci_evt_role_change_t {
  bt_hci_conn_state_t* pconn;
  bt_byte role;
} bt_hci_evt_role_change_t;
```

Description

This is type bt_hci_evt_role_change_t.

bt_hci_hconn_p Type

File

hci.h

C

```
typedef bt_int * bt_hci_hconn_p;
```

Description

This is type bt_hci_hconn_p.

bt_hci_hconn_t Type

File

hci.h

C

```
typedef bt_int bt_hci_hconn_t;
```

Description

This is type bt_hci_hconn_t.

bt_hci_inquiry_callback_fp Type

File

hci.h

C

```
typedef void (* bt_hci_inquiry_callback_fp)(bt_byte status, bt_hci_inquiry_response_t* response);
```

Description

This is type bt_hci_inquiry_callback_fp.

bt_hci_inquiry_response_t Structure

File

hci.h

C

```
typedef struct _bt_hci_inquiry_response_t {
  bt_bdaddr_t bdaddr;
  bt_byte pg_scan_rpt_mode;
  bt_byte pg_scan_period_mode;
  bt_long cod;
  bt_int clock_offset;
  bt_byte rssi;
  bt_byte* eir;
  bt_byte eir_len;
} bt_hci_inquiry_response_t;
```

Description

This is type bt_hci_inquiry_response_t.

bt_hci_le_advertising_report_t Type

File

hci_le.h

C

```
typedef struct _bt_hci_le_advertising_report_t bt_hci_le_advertising_report_t;
```

Description

This is type bt_hci_le_advertising_report_t.

bt_hci_le_conn_state_t Structure

File

hci_le.h

C

```
typedef struct _bt_hci_le_conn_state_t {
  bt_byte own_address_type;
  bt_byte peer_address_type;
  bt_uint conn_interval;
  bt_uint conn_latency;
  bt_uint supervision_timeout;
  bt_byte master_clock_accuracy;
} bt_hci_le_conn_state_t;
```

Description

This is type bt_hci_le_conn_state_t.

bt_hci_le_connect_parameters_t Structure

File

hci_le.h

C

```
typedef struct _bt_hci_le_connect_parameters_t {
  bt_uint scan_interval;
  bt_uint scan_window;
  bt_byte filter_policy;
```

```
  bt_byte own_address_type;
  bt_uint conn_interval_min;
  bt_uint conn_interval_max;
  bt_uint conn_latency;
  bt_uint supervision_timeout;
  bt_uint min_ce_length;
  bt_uint max_ce_length;
} bt_hci_le_connect_parameters_t;
```

Description

This is type bt_hci_le_connect_parameters_t.

bt_hci_le_ctrl_state_t Structure

File

hci_le.h

C

```
typedef struct _bt_hci_le_ctrl_state_t {
  bt_byte flags;
  bt_byte total_num_acl_data_packets;
  bt_int acl_data_packet_len;
  struct _bt_security_mgr_t* sm;
  bt_bdaddr_t bdaddr;
  bt_byte advertising_address_type;
  bt_byte connecting_address_type;
  bt_hci_le_connect_parameters_t def_connect_parameters;
  bt_le_evt_handler handler;
  bt_hci_le_scan_callback_fp scan_cb;
  void* scan_cb_param;
  bt_bool (* connect_ex)(bt_uint scan_interval, bt_uint scan_window, bt_byte filter_policy, bt_byte
peer_address_type, bt_bdaddr_t* peer_address, bt_byte own_address_type, bt_uint conn_interval_min, bt_uint
conn_interval_max, bt_uint conn_latency, bt_uint supervision_timeout, bt_uint min_ce_length, bt_uint
max_ce_length, bt_uint acl_config, bt_hci_connect_callback_fp cb, void *param);
  bt_bool (* set_connect_parameters)(bt_hci_le_connect_parameters_t* params);
  bt_bool (* get_connect_parameters)(bt_hci_le_connect_parameters_t* params);
  void (* sm_long_term_key_request_handler)(bt_hci_event_t* evt);
  bt_uint (* sm_get_session_state)(bt_hci_conn_state_t* conn);
  bt_byte (* sm_get_session_key_size)(bt_hci_conn_state_t* conn);
  bt_bool (* sm_authenticate)(bt_hci_conn_state_t* conn, bt_byte auth_req);
} bt_hci_le_ctrl_state_t;
```

Members

Members	Description
bt_hci_le_connect_parameters_t def_connect_parameters;	default connect parameters
bt_bool (* connect_ex)(bt_uint scan_interval, bt_uint scan_window, bt_byte filter_policy, bt_byte peer_address_type, bt_bdaddr_t* peer_address, bt_byte own_address_type, bt_uint conn_interval_min, bt_uint conn_interval_max, bt_uint conn_latency, bt_uint supervision_timeout, bt_uint min_ce_length, bt_uint max_ce_length, bt_uint acl_config, bt_hci_connect_callback_fp cb, void *param);	API calls

Description

This is type bt_hci_le_ctrl_state_t.

bt_hci_le_evt_connection_updated_t Structure

File

hci_le.h

C

```
typedef struct _bt_hci_le_evt_connection_updated_t {
  bt_byte status;
```

```
  struct _bt_hci_conn_state_s* conn;
} bt_hci_le_evt_connection_updated_t;
```

Description

This is type bt_hci_le_evt_connection_updated_t.

bt_hci_le_evt_read_remote_used_features_completed_t Structure

File

hci_le.h

C

```
typedef struct _bt_hci_le_evt_read_remote_used_features_completed_t {
  bt_byte status;
  struct _bt_hci_conn_state_s* conn;
  bt_byte features[8];
} bt_hci_le_evt_read_remote_used_features_completed_t;
```

Description

This is type bt_hci_le_evt_read_remote_used_features_completed_t.

bt_hci_le_evt_read_support_params_t Structure

File

hci_le.h

C

```
typedef struct _bt_hci_le_evt_read_support_params_t {
  bt_byte supported;
  bt_byte simultaneous_supported;
} bt_hci_le_evt_read_support_params_t;
```

Description

This is type bt_hci_le_evt_read_support_params_t.

bt_hci_le_scan_callback_fp Type

File

hci_le.h

C

```
typedef void (* bt_hci_le_scan_callback_fp)(bt_byte evt, bt_hci_le_advertising_report_t* report, void*
param);
```

Description

This is type bt_hci_le_scan_callback_fp.

bt_hci_link_key_t Structure

File

hci_linkkey_buffer.h

C

```
typedef struct _bt_hci_link_key_s {
  bt_bdaddr_t bdaddr;
  bt_byte linkkey[HCI_LINK_KEY_LEN];
} bt_hci_link_key_t;
```

Description

This is type bt_hci_link_key_t.

bt_hci_listener_t Type

File

hci_conn_state.h

C

```
typedef struct _bt_hci_listener_t bt_hci_listener_t;
```

Description

This is type bt_hci_listener_t.

bt_hci_read_inquiry_mode_callback_fp Type

File

hci.h

C

```
typedef void (* bt_hci_read_inquiry_mode_callback_fp)(bt_byte status, bt_byte mode);
```

Description

This is type bt_hci_read_inquiry_mode_callback_fp.

bt_hci_read_inquiry_scan_activity_callback_fp Type

File

hci.h

C

```
typedef void (* bt_hci_read_inquiry_scan_activity_callback_fp)(bt_byte status, bt_uint interval, bt_uint
window);
```

Description

This is type bt_hci_read_inquiry_scan_activity_callback_fp.

bt_hci_read_inquiry_scan_type_callback_fp Type

File

hci.h

C

```
typedef void (* bt_hci_read_inquiry_scan_type_callback_fp)(bt_byte status, bt_byte scanType);
```

Description

This is type bt_hci_read_inquiry_scan_type_callback_fp.

bt_hci_read_page_scan_activity_callback_fp Type

File

hci.h

C

```
typedef void (* bt_hci_read_page_scan_activity_callback_fp)(bt_byte status, bt_uint interval, bt_uint
window);
```

Description

This is type bt_hci_read_page_scan_activity_callback_fp.

bt_hci_read_page_scan_period_mode_callback_fp Type

File

hci.h

C

```
typedef void (* bt_hci_read_page_scan_period_mode_callback_fp)(bt_byte status, bt_byte mode);
```

Description

This is type bt_hci_read_page_scan_period_mode_callback_fp.

bt_hci_read_page_scan_type_callback_fp Type

File

hci.h

C

```
typedef void (* bt_hci_read_page_scan_type_callback_fp)(bt_byte status, bt_byte scanType);
```

Description

This is type bt_hci_read_page_scan_type_callback_fp.

bt_hci_read_page_timeout_callback_fp Type

File

hci.h

C

```
typedef void (* bt_hci_read_page_timeout_callback_fp)(bt_byte status, bt_uint timeout);
```

Description

This is type bt_hci_read_page_timeout_callback_fp.

bt_hci_request_remote_name_callback_fp Type

File

hci.h

C

```
typedef void (* bt_hci_request_remote_name_callback_fp)(bt_bdaddr_p pdbaddr, const char* name);
```

Description

This is type bt_hci_request_remote_name_callback_fp.

bt_hci_sco_read_data_callback_fp Type

File

hci_conn_state.h

C

```
typedef void (* bt_hci_sco_read_data_callback_fp)(bt_hci_conn_state_t* pconn, bt_byte_p data, bt_byte len);
```

Description

This is type bt_hci_sco_read_data_callback_fp.

bt_hci_start_callback_fp Type

File

hci.h

C

```
typedef void (* bt_hci_start_callback_fp)(bt_bool success, void* param);
```

Description

brief HCI initialization callback. ingroup hci

details This typedef defines a function pointer type for the HCI initialization callback functions. Such a function must be passed to the bt_hci_start() function.

param success Specifies whether HCI initialization succeeded or not.

bt_hci_stop_callback_fp Type

File

hci.h

C

```
typedef void (* bt_hci_stop_callback_fp)(void* param);
```

Description

brief HCI stop callback. ingroup hci

details This typedef defines a function pointer type for the HCI stop callback functions. Such a function must be passed to the bt_hci_stop() function.

bt_hci_transport_recv_packet_callback_fp Type

File

hci_transport.h

C

```
typedef void (* bt_hci_transport_recv_packet_callback_fp)(bt_uint len);
```

Description

This is type bt_hci_transport_recv_packet_callback_fp.

bt_hci_transport_send_packet_callback_fp Type

File

hci_transport.h

C

```
typedef void (* bt_hci_transport_send_packet_callback_fp)(void);
```

Description

This is type bt_hci_transport_send_packet_callback_fp.

bt_hcitr_tih4_power_callback_fp Type

File

hcitr_tih4.h

C

```
typedef void (* bt_hcitr_tih4_power_callback_fp)(bt_hcitr_tih4_power_event_e event);
```

Description

This is type bt_hcitr_tih4_power_callback_fp.

bt_hcitr_tih4_power_event_e Enumeration

File

hcitr_tih4.h

C

```
typedef enum _hcitr_tih4_power_event_e {
  HCITR_TIH4_POWER_EVENT_PREPARE_TO_SLEEP,
  HCITR_TIH4_POWER_EVENT_SLEEP,
  HCITR_TIH4_POWER_EVENT_WAKE_UP,
  HCITR_TIH4_POWER_EVENT_AWAKE
} bt_hcitr_tih4_power_event_e;
```

Description

This is type bt_hcitr_tih4_power_event_e.

bt_le_evt_handler Type

File

hci_le.h

C

```
typedef void (* bt_le_evt_handler)(bt_hci_event_t* evt);
```

Description

This is type bt_le_evt_handler.

hci_transport_t Structure

File

hci_transport.h

C

```
typedef struct _hci_transport_t {
  void (* send_packet)(const bt_byte* buffer, bt_uint len, bt_hci_transport_send_packet_callback_fp
callback);
  void (* recv_packet)(bt_byte* buffer, bt_uint len, bt_hci_transport_recv_packet_callback_fp callback);
} hci_transport_t;
```

Description

This is type hci_transport_t.

pf_hci_sleep_callback Type

File

hci.h

C

```
typedef void (* pf_hci_sleep_callback)(bt_bool status);
```

Description

This is type pf_hci_sleep_callback.

pf_hci_wakeup_callback Type

File

hci.h

C

```
typedef void (* pf_hci_wakeup_callback)(bt_bool status);
```

Description

This is type pf_hci_wakeup_callback.

pf_l2cap_receive_callback Type

File

hci_ctrl_state.h

C

```
typedef void (* pf_l2cap_receive_callback)(struct _bt_l2cap_mgr_s*, bt_hci_conn_state_p, bt_byte_p, bt_int
len);
```

Description

This is type pf_l2cap_receive_callback.

_bt_hci_conn_state_s Structure

File

hci_conn_state.h

C

```
struct _bt_hci_conn_state_s {
  bt_hci_hconn_t hconn;
  bt_bdaddr_t bdaddr_remote;
  bt_byte type;
  bt_byte state;
  bt_byte mode;
  bt_byte role;
  bt_long l2cap_channel_close_time;
  bt_uint acl_config;
  bt_byte link_type;
  bt_signal_t signal;
  bt_int outstanding_acl_packet_count;
  bt_hci_le_conn_state_t* le_conn_state;
  bt_ulong data_rx_time;
  bt_ulong data_tx_time;
  bt_byte idle;
  bt_queue_element_t* queue;
  bt_byte* recv_data;
  bt_int recv_data_len;
  bt_int recv_data_pos;
  bt_hci_sco_read_data_callback_fp sco_read_data_callback;
  void* sco_read_data_param;
  bt_hci_listener_t* listeners;
};
```

Members

Members	Description
bt_queue_element_t* queue;	queue for sending outgoing data packets
bt_byte* recv_data;	buffer to receive incoming packets
bt_hci_sco_read_data_callback_fp sco_read_data_callback;	bt_hci_disconnect_callback_fp hci_disconnect_callback; void *hci_disconnect_param;

Description

This is record _bt_hci_conn_state_s.

_bt_hci_ctrl_listener_t Structure

File

hci_ctrl_state.h

C

```
struct _bt_hci_ctrl_listener_t {
  bt_hci_ctrl_listener_t* next_listener;
  bt_byte listener_type;
  bt_uint event_id;
  union {
    void* ptr;
    bt_hci_event_listener_fp hci_event;
    bt_hci_cmd_listener_fp cmd_event;
    bt_hci_data_listener_fp data;
  } callback;
  void* callback_param;
};
```

Description

This is record _bt_hci_ctrl_listener_t.

_bt_hci_le_advertising_report_t Structure

File

hci_le.h

C

```
struct _bt_hci_le_advertising_report_t {
  bt_byte event_type;
  bt_byte address_type;
  bt_bdaddr_t bdaddr;
  bt_byte data_len;
  bt_byte* data;
  bt_byte rssi;
};
```

Description

This is record _bt_hci_le_advertising_report_t.

_bt_hci_listener_t Structure

File

hci_conn_state.h

C

```
struct _bt_hci_listener_t {
  bt_hci_listener_t* next_listener;
  bt_byte event_id;
  union {
    void* ptr;
    bt_hci_disconnect_callback_fp disconnect;
  } callback;
  void* callback_param;
};
```

Description

This is record _bt_hci_listener_t.

HCI_DISCOVERABLE_MODE_GENERAL Macro

File

hci.h

C

```
#define HCI_DISCOVERABLE_MODE_GENERAL 0
```

Description

This is macro HCI_DISCOVERABLE_MODE_GENERAL.

HCI_DISCOVERABLE_MODE_LIMITED Macro

File

hci.h

C

```
#define HCI_DISCOVERABLE_MODE_LIMITED 1
```

Description

This is macro HCI_DISCOVERABLE_MODE_LIMITED.

HCI_EIR_TYPE_3D_Information_Data Macro

File

hci_eir.h

C

```
#define HCI_EIR_TYPE_3D_Information_Data 0x3D
```

Description

This is macro HCI_EIR_TYPE_3D_Information_Data.

HCI_EIR_TYPE_ADVERTISING_INTERVAL Macro

File

hci_eir.h

C

```
#define HCI_EIR_TYPE_ADVERTISING_INTERVAL 0x1A
```

Description

This is macro HCI_EIR_TYPE_ADVERTISING_INTERVAL.

HCI_EIR_TYPE_APPEARANCE Macro

File

hci_eir.h

C

```
#define HCI_EIR_TYPE_APPEARANCE 0x19
```

Description

This is macro HCI_EIR_TYPE_APPEARANCE.

HCI_EIR_TYPE_LE_BLUETOOTH_DEVICE_ADDRESS Macro

File

hci_eir.h

C

```
#define HCI_EIR_TYPE_LE_BLUETOOTH_DEVICE_ADDRESS 0x1B
```

Description

This is macro HCI_EIR_TYPE_LE_BLUETOOTH_DEVICE_ADDRESS.

HCI_EIR_TYPE_LE_ROLE Macro

File

hci_eir.h

C

```
#define HCI_EIR_TYPE_LE_ROLE 0x1C
```

Description

This is macro HCI_EIR_TYPE_LE_ROLE.

HCI_EIR_TYPE_LE_SECURE_CONNECTIONS_CONFIRMATION_VALUE Macro

File

hci_eir.h

C

```
#define HCI_EIR_TYPE_LE_SECURE_CONNECTIONS_CONFIRMATION_VALUE 0x22
```

Description

This is macro HCI_EIR_TYPE_LE_SECURE_CONNECTIONS_CONFIRMATION_VALUE.

HCI_EIR_TYPE_LE_SECURE_CONNECTIONS_RANDOM_VALUE Macro

File

hci_eir.h

C

```
#define HCI_EIR_TYPE_LE_SECURE_CONNECTIONS_RANDOM_VALUE 0x23
```

Description

This is macro HCI_EIR_TYPE_LE_SECURE_CONNECTIONS_RANDOM_VALUE.

HCI_EIR_TYPE_PUBLIC_TARGET_ADDRESS Macro

File

hci_eir.h

C

```
#define HCI_EIR_TYPE_PUBLIC_TARGET_ADDRESS 0x17
```

Description

This is macro HCI_EIR_TYPE_PUBLIC_TARGET_ADDRESS.

HCI_EIR_TYPE_RANDOM_TARGET_ADDRESS Macro

File

hci_eir.h

C

```
#define HCI_EIR_TYPE_RANDOM_TARGET_ADDRESS 0x18
```

Description

This is macro HCI_EIR_TYPE_RANDOM_TARGET_ADDRESS.

HCI_EIR_TYPE_SERVICE_DATA Macro

File

hci_eir.h

C

```
#define HCI_EIR_TYPE_SERVICE_DATA 0x16
```

Description

This is macro HCI_EIR_TYPE_SERVICE_DATA.

HCI_EIR_TYPE_SERVICE_DATA_UUID128 Macro

File

hci_eir.h

C

```
#define HCI_EIR_TYPE_SERVICE_DATA_UUID128 0x21
```

Description

This is macro HCI_EIR_TYPE_SERVICE_DATA_UUID128.

HCI_EIR_TYPE_SERVICE_DATA_UUID16 Macro

File

hci_eir.h

C

```
#define HCI_EIR_TYPE_SERVICE_DATA_UUID16 0x16
```

Description

This is macro HCI_EIR_TYPE_SERVICE_DATA_UUID16.

HCI_EIR_TYPE_SERVICE_DATA_UUID32 Macro

File

hci_eir.h

C

```
#define HCI_EIR_TYPE_SERVICE_DATA_UUID32 0x20
```

Description

This is macro HCI_EIR_TYPE_SERVICE_DATA_UUID32.

HCI_EIR_TYPE_SIMPLE_PAIRING_HASH_C_256 Macro

File

hci_eir.h

C

```
#define HCI_EIR_TYPE_SIMPLE_PAIRING_HASH_C_256 0x1D
```

Description

This is macro HCI_EIR_TYPE_SIMPLE_PAIRING_HASH_C_256.

HCI_EIR_TYPE_SIMPLE_PAIRING_RANDOMIZER_R_256 Macro

File

hci_eir.h

C

```
#define HCI_EIR_TYPE_SIMPLE_PAIRING_RANDOMIZER_R_256 0x1E
```

Description

This is macro HCI_EIR_TYPE_SIMPLE_PAIRING_RANDOMIZER_R_256.

HCI_EIR_TYPE_SLAVE_CONN_INTERVAL_RANGE Macro

File

hci_eir.h

C

```
#define HCI_EIR_TYPE_SLAVE_CONN_INTERVAL_RANGE 0x12
```

Description

This is macro HCI_EIR_TYPE_SLAVE_CONN_INTERVAL_RANGE.

HCI_EIR_TYPE_SM_OOB_FLAGS Macro

File

hci_eir.h

C

```
#define HCI_EIR_TYPE_SM_OOB_FLAGS 0x11
```

Description

This is macro HCI_EIR_TYPE_SM_OOB_FLAGS.

HCI_EIR_TYPE_SM_TK_VALUE Macro

File

hci_eir.h

C

```
#define HCI_EIR_TYPE_SM_TK_VALUE 0x10
```

Description

This is macro HCI_EIR_TYPE_SM_TK_VALUE.

HCI_EIR_TYPE_SOLICITATION_UUID128_LIST Macro

File

hci_eir.h

C

```
#define HCI_EIR_TYPE_SOLICITATION_UUID128_LIST 0x15
```

Description

This is macro HCI_EIR_TYPE_SOLICITATION_UUID128_LIST.

HCI_EIR_TYPE_SOLICITATION_UUID16_LIST Macro

File

hci_eir.h

C

```
#define HCI_EIR_TYPE_SOLICITATION_UUID16_LIST 0x14
```

Description

This is macro HCI_EIR_TYPE_SOLICITATION_UUID16_LIST.

HCI_EIR_TYPE_SOLICITATION_UUID32_LIST Macro

File

hci_eir.h

C

```
#define HCI_EIR_TYPE_SOLICITATION_UUID32_LIST 0x1F
```

Description

This is macro HCI_EIR_TYPE_SOLICITATION_UUID32_LIST.

HCI_ERR_CONNECTION_TIMEOUT Macro

File

hci_errors.h

C

```
#define HCI_ERR_CONNECTION_TIMEOUT 0x08
```

Description

This is macro HCI_ERR_CONNECTION_TIMEOUT.

HCI_INIT_FLAG_IGNORE_TOTAL_NUM_ACL_DATA_PACKETS Macro

File

hci_private.h

C

```
#define HCI_INIT_FLAG_IGNORE_TOTAL_NUM_ACL_DATA_PACKETS 2 // The total number of ACL data packets read from
the controller will be ignored.
```

Description

The total number of ACL data packets read from the controller will be ignored. The stack will assume that he controller can accept only 1 ACL packet and sends HCI_EVT_NUM_OF_COMPLETED_PACKETS for each packet it has processed. This seems to be for controller working over SDIO (at least for Marvell's 88W8777).

HCI_INIT_FLAG_SEND_HCI_RESET Macro

File

hci_private.h

C

```
#define HCI_INIT_FLAG_SEND_HCI_RESET 1 // HCI reset will be sent when bt_hci_init is called
```

Description

HCI reset will be sent when bt_hci_init is called

HCI_LE_ADV_CHANNEL_MAP_ENABLE_37 Macro

File

hci_le.h

C

```
#define HCI_LE_ADV_CHANNEL_MAP_ENABLE_37 0x01
```

Description

This is macro HCI_LE_ADV_CHANNEL_MAP_ENABLE_37.

HCI_LE_ADV_CHANNEL_MAP_ENABLE_38 Macro

File

hci_le.h

C

```
#define HCI_LE_ADV_CHANNEL_MAP_ENABLE_38 0x02
```

Description

This is macro HCI_LE_ADV_CHANNEL_MAP_ENABLE_38.

HCI_LE_ADV_CHANNEL_MAP_ENABLE_39 Macro

File

hci_le.h

C

```
#define HCI_LE_ADV_CHANNEL_MAP_ENABLE_39 0x04
```

Description

This is macro HCI_LE_ADV_CHANNEL_MAP_ENABLE_39.

HCI_LE_ADV_CHANNEL_MAP_ENABLE_ALL Macro

File

hci_le.h

C

```
#define HCI_LE_ADV_CHANNEL_MAP_ENABLE_ALL 0x07
```

Description

This is macro HCI_LE_ADV_CHANNEL_MAP_ENABLE_ALL.

HCI_LE_ADV_CHANNEL_MAP_RESERVED Macro

File

hci_le.h

C

```
#define HCI_LE_ADV_CHANNEL_MAP_RESERVED 0x00
```

Description

This is macro HCI_LE_ADV_CHANNEL_MAP_RESERVED.

HCI_LE_ADV_TYPE_CONN_UNDIRECT Macro

File

hci_le.h

C

```
#define HCI_LE_ADV_TYPE_CONN_UNDIRECT 0x00 // Connectable undirected advertising (ADV_IND) (default)
```

Description

Connectable undirected advertising (ADV_IND) (default)

HCI_LE_ADV_TYPE_DIRECT_HIGH Macro

File

hci_le.h

C

```
#define HCI_LE_ADV_TYPE_DIRECT_HIGH 0x01 // Connectable high duty cycle directed advertising
(ADV_DIRECT_IND, high duty cycle)
```

Description

Connectable high duty cycle directed advertising (ADV_DIRECT_IND, high duty cycle)

HCI_LE_ADV_TYPE_DIRECT_LOW Macro

File

hci_le.h

C

```
#define HCI_LE_ADV_TYPE_DIRECT_LOW 0x04 // Connectable low duty cycle directed advertising (ADV_DIRECT_IND,
low duty cycle)
```

Description

Connectable low duty cycle directed advertising (ADV_DIRECT_IND, low duty cycle)

HCI_LE_ADV_TYPE_NONCONN Macro

File

hci_le.h

C

```
#define HCI_LE_ADV_TYPE_NONCONN 0x03 // Non connectable undirected advertising (ADV_NONCONN_IND)
```

Description

Non connectable undirected advertising (ADV_NONCONN_IND)

HCI_LE_ADV_TYPE_SCAN Macro

File

hci_le.h

C

```
#define HCI_LE_ADV_TYPE_SCAN 0x02 // Scannable undirected advertising (ADV_SCAN_IND)
```

Description

Scannable undirected advertising (ADV_SCAN_IND)

HCI_LE_RANDOM_ADDRESS_TYPE_NON_RESOLVABLE Macro

File

hci_le.h

C

```
#define HCI_LE_RANDOM_ADDRESS_TYPE_NON_RESOLVABLE 2
```

Description

This is macro HCI_LE_RANDOM_ADDRESS_TYPE_NON_RESOLVABLE.

HCI_LE_RANDOM_ADDRESS_TYPE_RESOLVABLE Macro

File

hci_le.h

C

```
#define HCI_LE_RANDOM_ADDRESS_TYPE_RESOLVABLE 3
```

Description

This is macro HCI_LE_RANDOM_ADDRESS_TYPE_RESOLVABLE.

HCI_LE_RANDOM_ADDRESS_TYPE_STATIC Macro

File

hci_le.h

C

```
#define HCI_LE_RANDOM_ADDRESS_TYPE_STATIC 1
```

Description

This is macro HCI_LE_RANDOM_ADDRESS_TYPE_STATIC.

HCI_LINK_POLICY_ENABLE_ALL Macro

File

hci.h

C

```
#define HCI_LINK_POLICY_ENABLE_ALL 0x000F
```

Description

This is macro HCI_LINK_POLICY_ENABLE_ALL.

bt_hci_le_cancel_connect Macro

File

hci_le.h

C

```
#define bt_hci_le_cancel_connect(cb) bt_hci_le_cancel_connect_ex(cb, NULL)
```

Description

This is macro bt_hci_le_cancel_connect.

bt_hci_send_pin_code Macro

File

hci.h

C

```
#define bt_hci_send_pin_code(pbdaddr_remote, pincode) bt_hci_send_pin_code_ex(pbdaddr_remote, pincode,
NULL, NULL)
```

Description

This is macro bt_hci_send_pin_code.

HCI_ERR_PAIRING_NOT_ALLOWED Macro

File

hci_errors.h

C

```
#define HCI_ERR_PAIRING_NOT_ALLOWED 0x18
```

Description

This is macro HCI_ERR_PAIRING_NOT_ALLOWED.

HCI_ERR_PIN_OR_KEY_MISSING Macro

File

hci_errors.h

C

```
#define HCI_ERR_PIN_OR_KEY_MISSING 0x06
```

Description

This is macro HCI_ERR_PIN_OR_KEY_MISSING.

HCI_PACKET_BOUNDARY_COMPLETE Macro

File

hci.h

C

```
#define HCI_PACKET_BOUNDARY_COMPLETE 0x3
```

Description

This is macro HCI_PACKET_BOUNDARY_COMPLETE.

HCI_SUCCESS Macro

File

hci_errors.h

C

```
#define HCI_SUCCESS 0x00
```

Description

This is macro HCI_SUCCESS.

L2CAP Functions

bt_l2cap_alloc_cmd_buffer Function

File

cmdbuffer.h

C

```
bt_l2cap_cmd_header_p bt_l2cap_alloc_cmd_buffer(bt_byte code);
```

Description

This is function bt_l2cap_alloc_cmd_buffer.

bt_l2cap_alloc_cmd_config_req Function

File

cmdbuffer.h

C

```
bt_l2cap_cmd_config_req_p bt_l2cap_alloc_cmd_config_req();
```

Description

This is function bt_l2cap_alloc_cmd_config_req.

bt_l2cap_alloc_cmd_config_res Function

File

cmdbuffer.h

C

```
bt_l2cap_cmd_config_res_p bt_l2cap_alloc_cmd_config_res();
```

Description

This is function bt_l2cap_alloc_cmd_config_res.

bt_l2cap_alloc_cmd_conn_param_update_req Function

File

cmdbuffer.h

C

```
bt_l2cap_cmd_conn_param_update_req_t* bt_l2cap_alloc_cmd_conn_param_update_req();
```

Description

This is function bt_l2cap_alloc_cmd_conn_param_update_req.

bt_l2cap_alloc_cmd_conn_param_update_res Function

File

cmdbuffer.h

C

```
bt_l2cap_cmd_conn_param_update_res_t* bt_l2cap_alloc_cmd_conn_param_update_res();
```

Description

This is function bt_l2cap_alloc_cmd_conn_param_update_res.

bt_l2cap_alloc_cmd_connection_req Function

File

cmdbuffer.h

C

```
bt_l2cap_cmd_connection_req_p bt_l2cap_alloc_cmd_connection_req();
```

Description

This is function bt_l2cap_alloc_cmd_connection_req.

bt_l2cap_alloc_cmd_connection_res Function

File

cmdbuffer.h

C

```
bt_l2cap_cmd_connection_res_p bt_l2cap_alloc_cmd_connection_res();
```

Description

This is function bt_l2cap_alloc_cmd_connection_res.

bt_l2cap_alloc_cmd_disconnection_req Function

File

cmdbuffer.h

C

```
bt_l2cap_cmd_disconnection_req_p bt_l2cap_alloc_cmd_disconnection_req();
```

Description

This is function bt_l2cap_alloc_cmd_disconnection_req.

bt_l2cap_alloc_cmd_disconnection_res Function

File

cmdbuffer.h

C

```
pcmd_disconnection_res bt_l2cap_alloc_cmd_disconnection_res();
```

Description

This is function bt_l2cap_alloc_cmd_disconnection_res.

bt_l2cap_alloc_cmd_echo_req Function

File

cmdbuffer.h

C

```
bt_l2cap_cmd_echo_req_p bt_l2cap_alloc_cmd_echo_req();
```

Description

This is function bt_l2cap_alloc_cmd_echo_req.

bt_l2cap_alloc_cmd_echo_res Function

File

cmdbuffer.h

C

```
bt_l2cap_cmd_echo_res_p bt_l2cap_alloc_cmd_echo_res();
```

Description

This is function bt_l2cap_alloc_cmd_echo_res.

bt_l2cap_alloc_cmd_info_req Function

File

cmdbuffer.h

C

```
bt_l2cap_cmd_info_req_p bt_l2cap_alloc_cmd_info_req();
```

Description

This is function bt_l2cap_alloc_cmd_info_req.

bt_l2cap_alloc_cmd_info_res Function

File

cmdbuffer.h

C

```
bt_l2cap_cmd_info_res_p bt_l2cap_alloc_cmd_info_res();
```

Description

This is function bt_l2cap_alloc_cmd_info_res.

bt_l2cap_alloc_cmd_reject Function

File

cmdbuffer.h

C

```
bt_l2cap_cmd_reject_p bt_l2cap_alloc_cmd_reject();
```

Description

This is function bt_l2cap_alloc_cmd_reject.

bt_l2cap_alloc_frame_buffer Function

File

frame_buffer.h

C

```
bt_byte_p bt_l2cap_alloc_frame_buffer();
```

Description

This is function bt_l2cap_alloc_frame_buffer.

bt_l2cap_allocate_channel Function

File

chmanager.h

C

```
bt_l2cap_channel_t* bt_l2cap_allocate_channel(bt_l2cap_mgr_p pmgr, bt_int mode, bt_int type);
```

Description

This is function bt_l2cap_allocate_channel.

bt_l2cap_allocate_fixed_channel Function

File

l2cap_fixed_channel.h

C

```
bt_l2cap_fixed_channel_t* bt_l2cap_allocate_fixed_channel(bt_l2cap_mgr_p pmgr, bt_id cid, bt_byte
link_type);
```

Description

This is function bt_l2cap_allocate_fixed_channel.

bt_l2cap_allocate_mgr Function

File

l2cap.h

C

```
bt_l2cap_mgr_t* bt_l2cap_allocate_mgr(bt_hci_ctrl_state_t* hci_ctrl);
```

Description

brief Allocate L2CAP manager. ingroup l2cap

details This function allocates and initializes an L2CAP manager structure. One L2CAP manager manages all L2CAP connections for a particular local device. The local device is specified by the c hci_ctrl parameter.

param hci_ctrl Pointer to the hci_ctrl_state structure that represents the local device (HCI controller) for which a L2CAP manager is to be allocated.

return A pointer to the allocated L2CAP manager structure. The returned L2CAP manager should be freed by a call to bt_l2cap_free_mgr when it is no longer needed.

bt_l2cap_allocate_psm Function

File

l2cap_psm.h

C

```
bt_l2cap_psm_t* bt_l2cap_allocate_psm(bt_l2cap_mgr_p pmgr, bt_int psmId);
```

Description

This is function bt_l2cap_allocate_psm.

bt_l2cap_connect_ext Function

File

l2cap.h

C

```
bt_bool bt_l2cap_connect_ext(bt_l2cap_mgr_t* mgr, bt_bdaddr_t* remote_addr, bt_int psm, bt_byte chmode,
bt_uint acl_config, bt_l2cap_connect_callback_fp connect_cb, void* param,
```

```
bt_l2cap_state_changed_callback_fp state_cb);
```

Description

brief Connect to a remote device. ingroup l2cap

details This function establishes an L2CAP connection with a remote device on a specific PSM. When connect operation completes a callback function is called.

param mgr The L2CAP manager. param remote_addr The address of the remote device. param psm The PSM for the connection. param connect_cb The Callback function that is called when the connect operation completes. param param A pointer to arbitrary data to be passed to the c connect_cb callback. param state_cb The callback function that is called when the state of the established connection changes.

return li c TRUE when the function succeeds. li c FALSE otherwise. None of the callback functions is called in this case.

bt_l2cap_connect_fixed_channel Function

File

l2cap.h

C

```
bt_bool bt_l2cap_connect_fixed_channel(bt_l2cap_mgr_p pmgr, bt_bdaddr_p pbdaddr_remote, bt_id cid, bt_uint
acl_config, bt_l2cap_connect_callback_fp cb, void* param, bt_l2cap_state_changed_callback_fp state_cb);
```

Description

This is function bt_l2cap_connect_fixed_channel.

bt_l2cap_disconnect Function

File

l2cap.h

C

```
bt_bool bt_l2cap_disconnect(bt_l2cap_channel_t* ch);
```

Description

brief Close L2CAP channel. ingroup l2cap

details This function closes an L2CAP channel. The channel can be established either by a call to bt_l2cap_connect() or by an incoming connection request.

param ch The L2CAP channel to be closed.

return li c TRUE when the function succeeds. li c FALSE otherwise. The callback function is not called in this case.

bt_l2cap_echo Function

File

l2cap.h

C

```
bt_bool bt_l2cap_echo(bt_l2cap_mgr_p pmgr, bt_hci_conn_state_p pconn, bt_byte_p data, bt_int len);
```

Description

brief Send echo command

bt_l2cap_find_fixed_channel Function

File

l2cap_fixed_channel.h

C

```
bt_l2cap_fixed_channel_t* bt_l2cap_find_fixed_channel(const bt_l2cap_mgr_p pmgr, bt_id cid, bt_byte
link_type);
```

Description

This is function bt_l2cap_find_fixed_channel.

bt_l2cap_find_psm Function

File

l2cap_psm.h

C

```
bt_l2cap_psm_t* bt_l2cap_find_psm(const bt_l2cap_mgr_p pmgr, bt_int psmId);
```

Description

This is function bt_l2cap_find_psm.

bt_l2cap_free_cmd_buffer Function

File

cmdbuffer.h

C

```
void bt_l2cap_free_cmd_buffer(void* p);
```

Description

This is function bt_l2cap_free_cmd_buffer.

bt_l2cap_free_fixed_channel Function

File

l2cap_fixed_channel.h

C

```
void bt_l2cap_free_fixed_channel(bt_l2cap_fixed_channel_t* fc);
```

Description

This is function bt_l2cap_free_fixed_channel.

bt_l2cap_free_frame_buffer Function

File

frame_buffer.h

C

```
void bt_l2cap_free_frame_buffer(bt_byte_p data);
```

Description

This is function bt_l2cap_free_frame_buffer.

bt_l2cap_free_mgr Function

File

l2cap.h

C

```
void bt_l2cap_free_mgr(bt_l2cap_mgr_t* mgr);
```

Description

brief Release L2CAP manger. ingroup l2cap

details This function releases the L2CAP manager structure.

param mgr The L2CAP manager structure to be released.

bt_l2cap_free_psm Function

File

l2cap_psm.h

C

```
void bt_l2cap_free_psm(bt_l2cap_psm_t* psm);
```

Description

This is function bt_l2cap_free_psm.

bt_l2cap_get_channel Function

File

chmanager.h

C

```
bt_l2cap_channel_t* bt_l2cap_get_channel(const bt_l2cap_mgr_p pmgr, bt_id cid);
```

Description

This is function bt_l2cap_get_channel.

bt_l2cap_get_channel_by_bdaddr_cid Function

File

chmanager.h

C

```
bt_l2cap_channel_t* bt_l2cap_get_channel_by_bdaddr_cid(const bt_l2cap_mgr_p pmgr, bt_bdaddr_t* bdaddr,
bt_id cid);
```

Description

This is function bt_l2cap_get_channel_by_bdaddr_cid.

bt_l2cap_get_channel_by_hconn_cid Function

File

chmanager.h

C

```
bt_l2cap_channel_t* bt_l2cap_get_channel_by_hconn_cid(const bt_l2cap_mgr_p pmgr, bt_hci_hconn_t hconn,
bt_id cid_dest);
```

Description

This is function bt_l2cap_get_channel_by_hconn_cid.

bt_l2cap_get_channel_by_hconn_dest_cid Function

File

chmanager.h

C

```
bt_l2cap_channel_t* bt_l2cap_get_channel_by_hconn_dest_cid(const bt_l2cap_mgr_p pmgr, bt_hci_hconn_t hconn,
bt_id cid_dest);
```

Description

This is function bt_l2cap_get_channel_by_hconn_dest_cid.

bt_l2cap_get_channel_by_psm Function

File

chmanager.h

C

```
bt_l2cap_channel_t* bt_l2cap_get_channel_by_psm(const bt_l2cap_mgr_p pmgr, bt_int psm);
```

Description

This is function bt_l2cap_get_channel_by_psm.

bt_l2cap_get_mgr Function

File

l2cap.h

C

```
bt_l2cap_mgr_p bt_l2cap_get_mgr(bt_hci_hconn_t h);
```

Description

This is function bt_l2cap_get_mgr.

bt_l2cap_init Function

File

l2cap.h

C

```
void bt_l2cap_init();
```

Description

brief Initialize the L2CAP layer. ingroup l2cap

details This function initializes the L2CAP layer of the stack. It must be called prior to any other L2CAP function can be called.

bt_l2cap_init_channels Function

File

chmanager.h

C

```
void bt_l2cap_init_channels(bt_l2cap_mgr_p pmgr);
```

Description

This is function bt_l2cap_init_channels.

bt_l2cap_init_cmd_buffers Function

File

cmdbuffer.h

C

```
void bt_l2cap_init_cmd_buffers();
```

Description

This is function bt_l2cap_init_cmd_buffers.

bt_l2cap_init_frame_buffers Function

File

frame_buffer.h

C

```
void bt_l2cap_init_frame_buffers();
```

Description

This is function bt_l2cap_init_frame_buffers.

bt_l2cap_init_psms Function

File

l2cap_psm.h

C

```
void bt_l2cap_init_psms(bt_l2cap_mgr_p pmgr);
```

Description

This is function bt_l2cap_init_psms.

bt_l2cap_listen_ext Function

File

l2cap.h

C

```
bt_bool bt_l2cap_listen_ext(bt_l2cap_mgr_t* mgr, bt_int psm, bt_byte chmode, bt_uint acl_config,
bt_l2cap_listen_callback_fp callback, void* param);
```

Description

brief Listen for incoming connections. ingroup l2cap

details This function tells the L2CAP manager to listen for incoming connections on a specific PSM. When a connection is established a callback function is called.

param mgr The L2CAP manager. param psm The PSM on which the manager will listen and accept incoming connections. param callback The callback function that will be called when an incoming connection is established. param param An arbitrary data pointer that will be passed to the callback function specified by the c callback parameter.

return li c TRUE when the function succeeds. li c FALSE otherwise. The callback function is not called in this case.

bt_l2cap_listen_fixed_channel Function

File

l2cap.h

C

```
bt_bool bt_l2cap_listen_fixed_channel(bt_l2cap_mgr_t* mgr, bt_id cid, bt_byte link_type,
bt_l2cap_listen_callback_fp callback, void* param);
```

Description

This is function bt_l2cap_listen_fixed_channel.

bt_l2cap_packet_cmd_assembler Function

File

l2cap_packet.h

C

```
bt_int bt_l2cap_packet_cmd_assembler(bt_packet_t* packet, bt_byte* buffer, bt_int buffer_len);
```

Description

This is function bt_l2cap_packet_cmd_assembler.

bt_l2cap_packet_data_assembler Function

File

l2cap_packet.h

C

```
bt_int bt_l2cap_packet_data_assembler(bt_packet_t* packet, bt_byte* buffer, bt_int buffer_len);
```

Description

This is function bt_l2cap_packet_data_assembler.

bt_l2cap_read_data Function

File

channel.h

C

```
bt_bool bt_l2cap_read_data(bt_l2cap_channel_t* channel, bt_l2cap_read_data_callback_fp callback);
```

Description

This is function bt_l2cap_read_data.

bt_l2cap_reject Function

File

l2cap.h

C

```
bt_bool bt_l2cap_reject(bt_l2cap_mgr_p pmgr, bt_hci_conn_state_p pconn, bt_byte cmd_id, bt_byte reason,
bt_l2cap_cmd_reject_param_t* param);
```

Description

brief Send reject command (used to reject unknown or invalid commands)

bt_l2cap_send_cmd Function

File

channel.h

C

```
bt_bool bt_l2cap_send_cmd(struct _bt_l2cap_mgr_s * mgr, struct _bt_hci_conn_state_s * conn,
bt_l2cap_cmd_header_p cmd, pf_l2cap_cmd_callback cb);
```

Description

This is function bt_l2cap_send_cmd.

bt_l2cap_send_data Function

File

channel.h

C

```
bt_bool bt_l2cap_send_data(bt_l2cap_channel_t* channel, const bt_byte_p data, bt_int len,
bt_l2cap_send_data_callback_fp callback, void* cb_param);
```

Description

brief Send data over an L2CAP channel ingroup l2cap

details This function sends data over the specified L2CAP channel.

param channel The L2CAP channel to send data over. param data The pointer to the data. param len The length of the data. param callback The callback function that is called when sending the data has been completed.

return li c TRUE if the function succeeds. li c FALSE otherwise. The callback function is not called in this case.

bt_l2cap_update_conn_parameters Function

File

l2cap.h

C

```
bt_bool bt_l2cap_update_conn_parameters(bt_l2cap_channel_t* channel, bt_uint min_interval, bt_uint
max_interval, bt_uint slave_latency, bt_uint supervision_timeout);
```

Description

This is function bt_l2cap_update_conn_parameters.

_bt_l2cap_clear_channel_cmd_queue Function

File

l2cap_private.h

C

```
void _bt_l2cap_clear_channel_cmd_queue(bt_l2cap_channel_t* channel);
```

Description

This is function _bt_l2cap_clear_channel_cmd_queue.

_bt_l2cap_eretr_handle_xmit_event Function

File

l2cap_eretr.h

C

```
bt_bool _bt_l2cap_eretr_handle_xmit_event(bt_l2cap_xmit_event_param_t* param);
```

Description

This is function _bt_l2cap_eretr_handle_xmit_event.

_bt_l2cap_eretr_retr_frames Function

File

l2cap_eretr.h

C

```
bt_bool _bt_l2cap_eretr_retr_frames(bt_l2cap_channel_t* pch, bt_byte fbit);
```

Description

This is function _bt_l2cap_eretr_retr_frames.

_bt_l2cap_fcs Function

File

l2cap_eretr.h

C

```
bt_uint _bt_l2cap_fcs(const bt_byte* message, bt_int len, bt_uint start_crc);
```

Description

This is function _bt_l2cap_fcs.

_bt_l2cap_get_tick_count Function

File

l2cap_timer.h

C

```
bt_long _bt_l2cap_get_tick_count();
```

Description

This is function _bt_l2cap_get_tick_count.

_bt_l2cap_init_signal Function

File

l2cap_signal.h

C

```
void _bt_l2cap_init_signal();
```

Description

This is function _bt_l2cap_init_signal.

_bt_l2cap_init_timer Function

File

l2cap_timer.h

C

```
void _bt_l2cap_init_timer();
```

Description

This is function _bt_l2cap_init_timer.

_bt_l2cap_notify_and_remove Function

File

l2cap_command_queue.h

C

```
void _bt_l2cap_notify_and_remove(struct _bt_l2cap_mgr_s* pmgr, bt_l2cap_cmd_header_p pcmd);
```

Description

This is function _bt_l2cap_notify_and_remove.

_bt_l2cap_send_commands_from_queue Function

File

l2cap_command_queue.h

C

```
void _bt_l2cap_send_commands_from_queue(struct _bt_l2cap_mgr_s* pmgr);
```

Description

This is function _bt_l2cap_send_commands_from_queue.

_bt_l2cap_set_signal Function

File

l2cap_signal.h

C

```
void _bt_l2cap_set_signal();
```

Description

This is function _bt_l2cap_set_signal.

_l2cap_allocate_buffers Function

File

l2cap_private.h

C

```
void _l2cap_allocate_buffers();
```

Description

Defined by OEM through library configuration

_l2cap_data_receive_callback Function

File

l2cap_private.h

C

```
void _l2cap_data_receive_callback(bt_l2cap_mgr_p pmg, bt_hci_conn_state_p pconn, bt_byte_p pdata, bt_int
len);
```

Description

From l2cap_recv.c

_pack_cmd_reject Function

File

l2cap_private.h

C

```
bt_int _pack_cmd_reject(bt_l2cap_cmd_header_t* pcmd, bt_byte_p buffer, bt_int buffer_len, bt_int_p poffset);
```

Description

This is function _pack_cmd_reject.

_pack_config_request Function

File

l2cap_private.h

C

```
bt_int _pack_config_request(bt_l2cap_cmd_header_t* pcmd, bt_byte_p buffer, bt_int buffer_len, bt_int_p
poffset);
```

Description

This is function _pack_config_request.

_pack_config_response Function

File

l2cap_private.h

C

```
bt_int _pack_config_response(bt_l2cap_cmd_header_t* pcmd, bt_byte_p buffer, bt_int buffer_len, bt_int_p
poffset);
```

Description

This is function _pack_config_response.

_pack_conn_param_update_request Function

File

l2cap_private.h

C

```
bt_int _pack_conn_param_update_request(bt_l2cap_cmd_header_t* pcmd, bt_byte_p buffer, bt_int buffer_len,
bt_int_p poffset);
```

Description

This is function _pack_conn_param_update_request.

_pack_conn_param_update_response Function

File

l2cap_private.h

C

```
bt_int _pack_conn_param_update_response(bt_l2cap_cmd_header_t* pcmd, bt_byte_p buffer, bt_int buffer_len,
bt_int_p poffset);
```

Description

This is function _pack_conn_param_update_response.

_pack_conn_request Function

File

l2cap_private.h

C

```
bt_int _pack_conn_request(bt_l2cap_cmd_header_t* pcmd, bt_byte_p buffer, bt_int buffer_len, bt_int_p
poffset);
```

Description

This is function _pack_conn_request.

_pack_conn_response Function

File

l2cap_private.h

C

```
bt_int _pack_conn_response(bt_l2cap_cmd_header_t* pcmd, bt_byte_p buffer, bt_int buffer_len, bt_int_p
poffset);
```

Description

This is function _pack_conn_response.

_pack_dconn_request Function

File

l2cap_private.h

C

```
bt_int _pack_dconn_request(bt_l2cap_cmd_header_t* pcmd, bt_byte_p buffer, bt_int buffer_len, bt_int_p
poffset);
```

Description

This is function _pack_dconn_request.

_pack_dconn_response Function

File

l2cap_private.h

C

```
bt_int _pack_dconn_response(bt_l2cap_cmd_header_t* pcmd, bt_byte_p buffer, bt_int buffer_len, bt_int_p
poffset);
```

Description

This is function _pack_dconn_response.

_pack_echo_request Function

File

l2cap_private.h

C

```
bt_int _pack_echo_request(bt_l2cap_cmd_header_t* pcmd, bt_byte_p buffer, bt_int buffer_len, bt_int_p
poffset);
```

Description

This is function _pack_echo_request.

_pack_echo_response Function

File

l2cap_private.h

C

```
bt_int _pack_echo_response(bt_l2cap_cmd_header_t* pcmd, bt_byte_p buffer, bt_int buffer_len, bt_int_p
poffset);
```

Description

This is function _pack_echo_response.

_pack_info_request Function

File

l2cap_private.h

C

```
bt_int _pack_info_request(bt_l2cap_cmd_header_t* pcmd, bt_byte_p buffer, bt_int buffer_len, bt_int_p
poffset);
```

Description

This is function _pack_info_request.

_pack_info_response Function

File

l2cap_private.h

C

```
bt_int _pack_info_response(bt_l2cap_cmd_header_t* pcmd, bt_byte_p buffer, bt_int buffer_len, bt_int_p
poffset);
```

Description

This is function _pack_info_response.

_process_config_req Function

File

l2cap_private.h

C

```
void _process_config_req(bt_l2cap_mgr_p pmgr, bt_hci_conn_state_p pconn, bt_l2cap_cmd_header_p pcmd);
```

Description

From cmd_config.c

_process_config_res Function

File

l2cap_private.h

C

```
void _process_config_res(bt_l2cap_mgr_p pmgr, bt_l2cap_cmd_header_p pcmd);
```

Description

This is function _process_config_res.

_process_conn_param_update_req Function

File

l2cap_private.h

C

```
void _process_conn_param_update_req(bt_l2cap_mgr_p pmgr, bt_hci_conn_state_p pconn, bt_l2cap_cmd_header_t*
pcmd);
```

Description

This is function _process_conn_param_update_req.

_process_conn_param_update_res Function

File

l2cap_private.h

C

```
void _process_conn_param_update_res(bt_l2cap_mgr_p pmgr, bt_l2cap_cmd_header_t* pcmd);
```

Description

This is function _process_conn_param_update_res.

_process_conn_req Function

File

l2cap_private.h

C

```
void _process_conn_req(bt_l2cap_mgr_p pmgr, bt_hci_conn_state_p pconn, bt_l2cap_cmd_header_p pcmd);
```

Description

From cmd_connect.c

_process_conn_res Function

File

l2cap_private.h

C

```
void _process_conn_res(bt_l2cap_mgr_p pmgr, bt_l2cap_cmd_header_p pcmd);
```

Description

This is function _process_conn_res.

_process_dconn_req Function

File

l2cap_private.h

C

```
void _process_dconn_req(bt_l2cap_mgr_p pmgr, bt_hci_conn_state_p pconn, bt_l2cap_cmd_header_p pcmd);
```

Description

From cmd_disconnect.c

_process_dconn_res Function

File

l2cap_private.h

C

```
void _process_dconn_res(bt_l2cap_mgr_p pmgr, bt_l2cap_cmd_header_p pcmd);
```

Description

This is function _process_dconn_res.

_process_echo_req Function

File

l2cap_private.h

C

```
void _process_echo_req(bt_l2cap_mgr_p pmgr, bt_hci_conn_state_p pconn, bt_l2cap_cmd_header_p pcmd);
```

Description

From cmd_echo.c

_process_echo_res Function

File

l2cap_private.h

C

```
void _process_echo_res(bt_l2cap_mgr_p pmgr, bt_l2cap_cmd_header_p pcmd);
```

Description

This is function _process_echo_res.

_process_info_req Function

File

l2cap_private.h

C

```
void _process_info_req(bt_l2cap_mgr_p pmgr, bt_hci_conn_state_p pconn, bt_l2cap_cmd_header_p pcmd);
```

Description

From cmd_info.c

_process_info_res Function

File

l2cap_private.h

C

```
void _process_info_res(bt_l2cap_mgr_p pmgr, bt_l2cap_cmd_header_p pcmd);
```

Description

This is function _process_info_res.

_process_reject Function

File

l2cap_private.h

C

```
void _process_reject(bt_l2cap_mgr_p pmgr, bt_l2cap_cmd_header_p pcmd);
```

Description

From cmd_reject.c

_process_unknown_req Function

File

l2cap_private.h

C

```
void _process_unknown_req(bt_l2cap_mgr_p pmgr, bt_hci_conn_state_p pconn, bt_l2cap_cmd_header_p pcmd);
```

Description

From cmd_unknown.c

_process_unknown_res Function

File

l2cap_private.h

C

```
void _process_unknown_res(bt_l2cap_mgr_p pmgr, bt_l2cap_cmd_header_p pcmd);
```

Description

This is function _process_unknown_res.

_read_cmd_reject Function

File

l2cap_private.h

C

```
bt_l2cap_cmd_header_p _read_cmd_reject(bt_byte_p pdata, bt_int len, bt_int_p poffset);
```

Description

This is function _read_cmd_reject.

_read_config_request Function

File

l2cap_private.h

C

```
bt_l2cap_cmd_header_p _read_config_request(bt_byte_p pdata, bt_int len, bt_int_p poffset);
```

Description

This is function _read_config_request.

_read_config_response Function

File

l2cap_private.h

C

```
bt_l2cap_cmd_header_p _read_config_response(bt_byte_p pdata, bt_int len, bt_int_p poffset);
```

Description

This is function _read_config_response.

_read_conn_param_update_request Function

File

l2cap_private.h

C

```
bt_l2cap_cmd_header_p _read_conn_param_update_request(bt_byte_p pdata, bt_int len, bt_int_p poffset);
```

Description

This is function _read_conn_param_update_request.

_read_conn_param_update_response Function

File

l2cap_private.h

C

```
bt_l2cap_cmd_header_p _read_conn_param_update_response(bt_byte_p pdata, bt_int len, bt_int_p poffset);
```

Description

This is function _read_conn_param_update_response.

_read_conn_request Function

File

l2cap_private.h

C

```
bt_l2cap_cmd_header_p _read_conn_request(bt_byte_p pdata, bt_int len, bt_int_p poffset);
```

Description

This is function _read_conn_request.

_read_conn_response Function

File

l2cap_private.h

C

```
bt_l2cap_cmd_header_p _read_conn_response(bt_byte_p pdata, bt_int len, bt_int_p poffset);
```

Description

This is function _read_conn_response.

_read_dconn_request Function

File

l2cap_private.h

C

```
bt_l2cap_cmd_header_p _read_dconn_request(bt_byte_p pdata, bt_int len, bt_int_p poffset);
```

Description

This is function _read_dconn_request.

_read_dconn_response Function

File

l2cap_private.h

C

```
bt_l2cap_cmd_header_p _read_dconn_response(bt_byte_p pdata, bt_int len, bt_int_p poffset);
```

Description

This is function _read_dconn_response.

_read_echo_request Function

File

l2cap_private.h

C

```
bt_l2cap_cmd_header_p _read_echo_request(bt_byte_p pdata, bt_int len, bt_int_p poffset);
```

Description

This is function _read_echo_request.

_read_echo_response Function

File

l2cap_private.h

C

```
bt_l2cap_cmd_header_p _read_echo_response(bt_byte_p pdata, bt_int len, bt_int_p poffset);
```

Description

This is function _read_echo_response.

_read_info_request Function

File

l2cap_private.h

C

```
bt_l2cap_cmd_header_p _read_info_request(bt_byte_p pdata, bt_int len, bt_int_p poffset);
```

Description

This is function _read_info_request.

_read_info_response Function

File

l2cap_private.h

C

```
bt_l2cap_cmd_header_p _read_info_response(bt_byte_p pdata, bt_int len, bt_int_p poffset);
```

Description

This is function _read_info_response.

_bt_l2cap_eretr_pack_config_request Function

File

l2cap_private.h

C

```
void _bt_l2cap_eretr_pack_config_request(bt_l2cap_channel_t* channel, bt_byte* buffer, bt_int buffer_len,
bt_int* offset);
```

Description

This is function _bt_l2cap_eretr_pack_config_request.

_bt_l2cap_eretr_recv Function

File

l2cap_eretr.h

C

```
void _bt_l2cap_eretr_recv(bt_l2cap_mgr_p pmgr, bt_l2cap_channel_t* pch, bt_byte* pdata, bt_int len);
```

Description

This is function _bt_l2cap_eretr_recv.

_bt_l2cap_eretr_send_data Function

File

l2cap_eretr.h

C

```
bt_bool _bt_l2cap_eretr_send_data(bt_l2cap_channel_t* pch, bt_byte* data, bt_int len,
bt_l2cap_send_data_callback_fp cb, void* cb_param);
```

Description

This is function _bt_l2cap_eretr_send_data.

_bt_l2cap_eretr_send_pending_frames Function

File

l2cap_eretr.h

C

```
bt_bool _bt_l2cap_eretr_send_pending_frames(bt_l2cap_channel_t* pch, bt_byte fbit);
```

Description

This is function _bt_l2cap_eretr_send_pending_frames.

_bt_l2cap_eretr_send_smart_data Function

File

l2cap_eretr.h

C

```
bt_bool _bt_l2cap_eretr_send_smart_data(bt_l2cap_channel_t* pch, bt_packet_t* packet, bt_int len,
bt_l2cap_send_data_callback_fp cb, void* cb_param);
```

Description

This is function _bt_l2cap_eretr_send_smart_data.

_bt_l2cap_recv_req_seq_and_fbit Function

File

l2cap_eretr.h

C

```
void _bt_l2cap_recv_req_seq_and_fbit(bt_l2cap_mgr_p pmgr, bt_l2cap_channel_t* pch, bt_byte req_seq, bt_byte fbit);
```

Description

This is function _bt_l2cap_recv_req_seq_and_fbit.

_bt_l2cap_send_ack Function

File

l2cap_eretr.h

C

```
bt_bool _bt_l2cap_send_ack(bt_l2cap_channel_t* pch, bt_byte pbit, bt_byte fbit);
```

Description

This is function _bt_l2cap_send_ack.

_bt_l2cap_send_i_or_rr_or_rnr Function

File

l2cap_eretr.h

C

```
bt_bool _bt_l2cap_send_i_or_rr_or_rnr(bt_l2cap_channel_t* pch, bt_byte pbit, bt_byte fbit);
```

Description

This is function _bt_l2cap_send_i_or_rr_or_rnr.

_bt_l2cap_send_rej Function

File

l2cap_eretr.h

C

```
bt_bool _bt_l2cap_send_rej(bt_l2cap_channel_t* pch, bt_byte pbit, bt_byte fbit);
```

Description

This is function _bt_l2cap_send_rej.

_bt_l2cap_send_rnr Function

File

l2cap_eretr.h

C

```
bt_bool _bt_l2cap_send_rnr(bt_l2cap_channel_t* pch, bt_byte pbit, bt_byte fbit);
```

Description

This is function _bt_l2cap_send_rnr.

_bt_l2cap_send_rr Function

File

l2cap_eretr.h

C

```
bt_bool _bt_l2cap_send_rr(bt_l2cap_channel_t* pch, bt_byte pbit, bt_byte fbit);
```

Description

This is function _bt_l2cap_send_rr.

_bt_l2cap_send_rr_or_rnr Function

File

l2cap_eretr.h

C

```
bt_bool _bt_l2cap_send_rr_or_rnr(bt_l2cap_channel_t* pch, bt_byte pbit, bt_byte fbit);
```

Description

This is function _bt_l2cap_send_rr_or_rnr.

bt_l2cap_disconnect_ex Function

File

l2cap.h

C

```
bt_bool bt_l2cap_disconnect_ex(bt_l2cap_channel_t* pch, bt_bool force_hci_disconnect);
```

Description

This is function bt_l2cap_disconnect_ex.

bt_l2cap_free_channel Function

File

chmanager.h

C

```
void bt_l2cap_free_channel(bt_l2cap_channel_t* ch);
```

Description

This is function bt_l2cap_free_channel.

bt_l2cap_hci_has_open_channels Function

File

l2cap.h

C

```
bt_bool bt_l2cap_hci_has_open_channels(bt_hci_conn_state_t* conn);
```

Description

This is function bt_l2cap_hci_has_open_channels.

bt_l2cap_is_channel_open Function

File

l2cap.h

C

```
bt_bool bt_l2cap_is_channel_open(bt_l2cap_channel_t* channel);
```

Description

This is function bt_l2cap_is_channel_open.

bt_l2cap_send_config Function

File

l2cap.h

C

```
bt_bool bt_l2cap_send_config(bt_l2cap_mgr_p pmgr, bt_l2cap_channel_t* pch);
```

Description

This is function bt_l2cap_send_config.

bt_l2cap_send_smart_data Function

File

channel.h

C

```
bt_bool bt_l2cap_send_smart_data(bt_l2cap_channel_t* pch, bt_packet_t* packet, bt_int len,
bt_l2cap_send_data_callback_fp cb, void* cb_param);
```

Description

This is function bt_l2cap_send_smart_data.

_bt_l2cap_process_connect_signal Function

File

l2cap_private.h

C

```
void _bt_l2cap_process_connect_signal(bt_l2cap_mgr_t* mgr);
```

Description

This is function _bt_l2cap_process_connect_signal.

L2CAP Data Types and Constants

bt_l2cap_connect Macro

File

l2cap.h

C

```
#define bt_l2cap_connect(mgr, remote_addr, psm, acl_config, connect_cb, param, state_cb)
bt_l2cap_connect_ext(mgr, remote_addr, psm, CMODE_BASIC, acl_config, connect_cb, param, state_cb)
```

Description

This is macro bt_l2cap_connect.

bt_l2cap_listen Macro

File

l2cap.h

C

```
#define bt_l2cap_listen(mgr, psm, acl_config, callback, param) bt_l2cap_listen_ext(mgr, psm, CMODE_BASIC,
acl_config, callback, param)
```

Description

This is macro bt_l2cap_listen.

bt_l2cap_test_enable_local_config Macro

File

l2cap_test.h

C

```
#define bt_l2cap_test_enable_local_config(enable)
```

Description

This is macro bt_l2cap_test_enable_local_config.

bt_l2cap_test_enable_remote_config Macro

File

l2cap_test.h

C

```
#define bt_l2cap_test_enable_remote_config(enable)
```

Description

This is macro bt_l2cap_test_enable_remote_config.

CHANNEL_SIGNAL_CMD_DISCONNECT_FIXED Macro

File

l2cap_private.h

C

```
#define CHANNEL_SIGNAL_CMD_DISCONNECT_FIXED 0
```

Description

This is macro CHANNEL_SIGNAL_CMD_DISCONNECT_FIXED.

CID_ATT Macro

File

channel.h

C

```
#define CID_ATT 0x0004
```

Description

This is macro CID_ATT.

CID_LE_SIG Macro

File

channel.h

C

```
#define CID_LE_SIG 0x0005
```

Description

This is macro CID_LE_SIG.

CID_MAX Macro

File

channel.h

C

```
#define CID_MAX 0xffff
```

Description

This is macro CID_MAX.

CID_MAX_FIXED Macro

File

channel.h

C

```
#define CID_MAX_FIXED 0x003f
```

Description

This is macro CID_MAX_FIXED.

CID_NULL Macro

File

channel.h

C

```
#define CID_NULL 0x0000
```

Description

This is macro CID_NULL.

CID_RECV Macro

File

channel.h

C

```
#define CID_RECV 0x0002
```

Description

This is macro CID_RECV.

CID_SIG Macro

File

channel.h

C

```
#define CID_SIG 0x0001
```

Description

This is macro CID_SIG.

CID_SM Macro

File

channel.h

C

```
#define CID_SM 0x0006
```

Description

This is macro CID_SM.

CMODE_BASIC Macro

File

channel.h

C

```
#define CMODE_BASIC 0
```

Description

This is macro CMODE_BASIC.

CMODE_ERETR Macro

File

channel.h

C

```
#define CMODE_ERETR 3
```

Description

This is macro CMODE_ERETR.

CMODE_FLOW Macro

File

channel.h

C

```
#define CMODE_FLOW 2
```

Description

This is macro CMODE_FLOW.

CMODE_RETR Macro

File

channel.h

C

```
#define CMODE_RETR 1
```

Description

This is macro CMODE_RETR.

CMODE_STRM Macro

File

channel.h

C

```
#define CMODE_STRM 4
```

Description

This is macro CMODE_STRM.

CSTATE_CLOSED Macro

File

channel.h

C

```
#define CSTATE_CLOSED 0x01
```

Description

This is macro CSTATE_CLOSED.

CSTATE_FREE Macro

File

channel.h

C

```
#define CSTATE_FREE 0x00
```

Description

This is macro CSTATE_FREE.

CSTATE_OPEN Macro

File

channel.h

C

```
#define CSTATE_OPEN 0x40
```

Description

This is macro CSTATE_OPEN.

CSTATE_WAIT_CONFIG Macro

File

channel.h

C

```
#define CSTATE_WAIT_CONFIG (CSTATE_WAIT_CONFIG_RSP | CSTATE_WAIT_CONFIG_REQ)
```

Description

This is macro CSTATE_WAIT_CONFIG.

CSTATE_WAIT_CONFIG_REQ Macro

File

channel.h

C

```
#define CSTATE_WAIT_CONFIG_REQ 0x10
```

Description

This is macro CSTATE_WAIT_CONFIG_REQ.

CSTATE_WAIT_CONFIG_RSP Macro

File

channel.h

C

```
#define CSTATE_WAIT_CONFIG_RSP 0x08
```

Description

This is macro CSTATE_WAIT_CONFIG_RSP.

CSTATE_WAIT_CONNECT Macro

File

channel.h

C

```
#define CSTATE_WAIT_CONNECT 0x02
```

Description

This is macro CSTATE_WAIT_CONNECT.

CSTATE_WAIT_CONNECT_RSP Macro

File

channel.h

C

```
#define CSTATE_WAIT_CONNECT_RSP 0x04
```

Description

This is macro CSTATE_WAIT_CONNECT_RSP.

CSTATE_WAIT_DISCONNECT Macro

File

channel.h

C

```
#define CSTATE_WAIT_DISCONNECT 0x20
```

Description

This is macro CSTATE_WAIT_DISCONNECT.

CTYPE_CL Macro

File

channel.h

C

```
#define CTYPE_CL 1 // connectionless bt_l2cap_channel
```

Description

connectionless bt_l2cap_channel

CTYPE_CO Macro

File

channel.h

C

```
#define CTYPE_CO 0 // connection-oriented bt_l2cap_channel
```

Description

connection-oriented bt_l2cap_channel

GET_F_BIT Macro

File

l2cap_eretr.h

C

```
#define GET_F_BIT(ctl) ((ctl & 0x80) >> 7)
```

Description

This is macro GET_F_BIT.

GET_FRAME_TYPE Macro

File

l2cap_eretr.h

C

```
#define GET_FRAME_TYPE(ctl) (ctl & 1)
```

Description

This is macro GET_FRAME_TYPE.

GET_P_BIT Macro

File

l2cap_eretr.h

C

```
#define GET_P_BIT(ctl) ((ctl & 0x10) >> 4)
```

Description

This is macro GET_P_BIT.

GET_REQ_SEQ Macro

File

l2cap_eretr.h

C

```
#define GET_REQ_SEQ(ctl) ((ctl & 0x3F00) >> 8)
```

Description

This is macro GET_REQ_SEQ.

GET_S_FUNCTION Macro

File

l2cap_eretr.h

C

```
#define GET_S_FUNCTION(ctl) ((ctl & 0x0C) >> 2)
```

Description

This is macro GET_S_FUNCTION.

GET_SAR Macro

File

l2cap_eretr.h

C

```
#define GET_SAR(ctl) ((ctl & 0xC000) >> 14)
```

Description

This is macro GET_SAR.

GET_TX_SEQ Macro

File

l2cap_eretr.h

C

```
#define GET_TX_SEQ(ctl) ((ctl & 0x7E) >> 1)
```

Description

This is macro GET_TX_SEQ.

L2CAP_CHANNEL_FLAG_INCOMING Macro

File

channel.h

C

```
#define L2CAP_CHANNEL_FLAG_INCOMING 0x02
```

Description

This is macro L2CAP_CHANNEL_FLAG_INCOMING.

L2CAP_CHANNEL_FLAG_SENDING Macro

File

channel.h

C

```
#define L2CAP_CHANNEL_FLAG_SENDING 0x01
```

Description

This is macro L2CAP_CHANNEL_FLAG_SENDING.

L2CAP_CMD_CONFIG_REQUEST Macro

File

command.h

C

```
#define L2CAP_CMD_CONFIG_REQUEST 0x04
```

Description

This is macro L2CAP_CMD_CONFIG_REQUEST.

L2CAP_CMD_CONFIG_RESPONSE Macro

File

command.h

C

```
#define L2CAP_CMD_CONFIG_RESPONSE 0x05
```

Description

This is macro L2CAP_CMD_CONFIG_RESPONSE.

L2CAP_CMD_CONN_PARAM_UPDATE_REQUEST Macro

File

command.h

C

```
#define L2CAP_CMD_CONN_PARAM_UPDATE_REQUEST 0x12
```

Description

This is macro L2CAP_CMD_CONN_PARAM_UPDATE_REQUEST.

L2CAP_CMD_CONN_PARAM_UPDATE_RESPONSE Macro

File

command.h

C

```
#define L2CAP_CMD_CONN_PARAM_UPDATE_RESPONSE 0x13
```

Description

This is macro L2CAP_CMD_CONN_PARAM_UPDATE_RESPONSE.

L2CAP_CMD_CONN_REQUEST Macro

File

command.h

C

```
#define L2CAP_CMD_CONN_REQUEST 0x02
```

Description

This is macro L2CAP_CMD_CONN_REQUEST.

L2CAP_CMD_CONN_RESPONSE Macro

File

command.h

C

```
#define L2CAP_CMD_CONN_RESPONSE 0x03
```

Description

This is macro L2CAP_CMD_CONN_RESPONSE.

L2CAP_CMD_DATA_LEN_CMD_REJECT Macro

File

command.h

C

```
#define L2CAP_CMD_DATA_LEN_CMD_REJECT 2
```

Description

This is macro L2CAP_CMD_DATA_LEN_CMD_REJECT.

L2CAP_CMD_DATA_LEN_CONFIG_REQUEST Macro

File

command.h

C

```
#define L2CAP_CMD_DATA_LEN_CONFIG_REQUEST 4
```

Description

This is macro L2CAP_CMD_DATA_LEN_CONFIG_REQUEST.

L2CAP_CMD_DATA_LEN_CONFIG_RESPONSE Macro

File

command.h

C

```
#define L2CAP_CMD_DATA_LEN_CONFIG_RESPONSE 6
```

Description

This is macro L2CAP_CMD_DATA_LEN_CONFIG_RESPONSE.

L2CAP_CMD_DATA_LEN_CONN_REQUEST Macro

File

command.h

C

```
#define L2CAP_CMD_DATA_LEN_CONN_REQUEST 4
```

Description

This is macro L2CAP_CMD_DATA_LEN_CONN_REQUEST.

L2CAP_CMD_DATA_LEN_CONN_RESPONSE Macro

File

command.h

C

```
#define L2CAP_CMD_DATA_LEN_CONN_RESPONSE 8
```

Description

This is macro L2CAP_CMD_DATA_LEN_CONN_RESPONSE.

L2CAP_CMD_DATA_LEN_DCONN_REQUEST Macro

File

command.h

C

```
#define L2CAP_CMD_DATA_LEN_DCONN_REQUEST 4
```

Description

This is macro L2CAP_CMD_DATA_LEN_DCONN_REQUEST.

L2CAP_CMD_DATA_LEN_DCONN_RESPONSE Macro

File

command.h

C

```
#define L2CAP_CMD_DATA_LEN_DCONN_RESPONSE 4
```

Description

This is macro L2CAP_CMD_DATA_LEN_DCONN_RESPONSE.

L2CAP_CMD_DATA_LEN_ECHO_REQUEST Macro

File

command.h

C

```
#define L2CAP_CMD_DATA_LEN_ECHO_REQUEST 0
```

Description

This is macro L2CAP_CMD_DATA_LEN_ECHO_REQUEST.

L2CAP_CMD_DATA_LEN_ECHO_RESPONSE Macro

File

command.h

C

```
#define L2CAP_CMD_DATA_LEN_ECHO_RESPONSE 0
```

Description

This is macro L2CAP_CMD_DATA_LEN_ECHO_RESPONSE.

L2CAP_CMD_DATA_LEN_INFO_REQUEST Macro

File

command.h

C

```
#define L2CAP_CMD_DATA_LEN_INFO_REQUEST 2
```

Description

This is macro L2CAP_CMD_DATA_LEN_INFO_REQUEST.

L2CAP_CMD_DATA_LEN_INFO_RESPONSE Macro

File

command.h

C

```
#define L2CAP_CMD_DATA_LEN_INFO_RESPONSE 4
```

Description

This is macro L2CAP_CMD_DATA_LEN_INFO_RESPONSE.

L2CAP_CMD_DCONN_REQUEST Macro

File

command.h

C

```
#define L2CAP_CMD_DCONN_REQUEST 0x06
```

Description

This is macro L2CAP_CMD_DCONN_REQUEST.

L2CAP_CMD_DCONN_RESPONSE Macro

File

command.h

C

```
#define L2CAP_CMD_DCONN_RESPONSE 0x07
```

Description

This is macro L2CAP_CMD_DCONN_RESPONSE.

L2CAP_CMD_ECHO_REQUEST Macro

File

command.h

C

```
#define L2CAP_CMD_ECHO_REQUEST 0x08
```

Description

This is macro L2CAP_CMD_ECHO_REQUEST.

L2CAP_CMD_ECHO_RESPONSE Macro

File

command.h

C

```
#define L2CAP_CMD_ECHO_RESPONSE 0x09
```

Description

This is macro L2CAP_CMD_ECHO_RESPONSE.

L2CAP_CMD_HEADER_LEN Macro

File

command.h

C

```
#define L2CAP_CMD_HEADER_LEN 4
```

Description

This is macro L2CAP_CMD_HEADER_LEN.

L2CAP_CMD_INFO_REQUEST Macro

File

command.h

C

```
#define L2CAP_CMD_INFO_REQUEST 0x0A
```

Description

This is macro L2CAP_CMD_INFO_REQUEST.

L2CAP_CMD_INFO_RESPONSE Macro

File

command.h

C

```
#define L2CAP_CMD_INFO_RESPONSE 0x0B
```

Description

This is macro L2CAP_CMD_INFO_RESPONSE.

L2CAP_CMD_LAST Macro

File

command.h

C

```
#define L2CAP_CMD_LAST L2CAP_CMD_CONN_PARAM_UPDATE_RESPONSE
```

Description

This is macro L2CAP_CMD_LAST.

L2CAP_CMD_REJECT Macro

File

command.h

C

```
#define L2CAP_CMD_REJECT 0x01
```

Description

This is macro L2CAP_CMD_REJECT.

L2CAP_CMD_RESERVED Macro

File

command.h

C

```
#define L2CAP_CMD_RESERVED 0x00
```

Description

This is macro L2CAP_CMD_RESERVED.

L2CAP_CMD_STATUS_BEING_SENT Macro

File

command.h

C

```
#define L2CAP_CMD_STATUS_BEING_SENT 2
```

Description

This is macro L2CAP_CMD_STATUS_BEING_SENT.

L2CAP_CMD_STATUS_PENDING Macro

File

command.h

C

```
#define L2CAP_CMD_STATUS_PENDING 0
```

Description

This is macro L2CAP_CMD_STATUS_PENDING.

L2CAP_CMD_STATUS_WAITING_RESPONSE Macro

File

command.h

C

```
#define L2CAP_CMD_STATUS_WAITING_RESPONSE 1
```

Description

This is macro L2CAP_CMD_STATUS_WAITING_RESPONSE.

L2CAP_CONFIG_RESULT_REJECTED Macro

File

command.h

C

```
#define L2CAP_CONFIG_RESULT_REJECTED 0x0002
```

Description

This is macro L2CAP_CONFIG_RESULT_REJECTED.

L2CAP_CONFIG_RESULT_SUCCESS Macro

File

command.h

C

```
#define L2CAP_CONFIG_RESULT_SUCCESS 0x0000
```

Description

This is macro L2CAP_CONFIG_RESULT_SUCCESS.

L2CAP_CONFIG_RESULT_UNACCEPTABLE_PARAMETER Macro

File

command.h

C

```
#define L2CAP_CONFIG_RESULT_UNACCEPTABLE_PARAMETER 0x0001
```

Description

This is macro L2CAP_CONFIG_RESULT_UNACCEPTABLE_PARAMETER.

L2CAP_CONFIG_RESULT_UNKNOWN_OPTION Macro

File

command.h

C

```
#define L2CAP_CONFIG_RESULT_UNKNOWN_OPTION 0x0003
```

Description

This is macro L2CAP_CONFIG_RESULT_UNKNOWN_OPTION.

L2CAP_CONN_REQ_RESULT_INVALID_PSM Macro

File

command.h

C

```
#define L2CAP_CONN_REQ_RESULT_INVALID_PSM 0x0002
```

Description

This is macro L2CAP_CONN_REQ_RESULT_INVALID_PSM.

L2CAP_CONN_REQ_RESULT_NO_RESOURCES Macro

File

command.h

C

```
#define L2CAP_CONN_REQ_RESULT_NO_RESOURCES 0x0004
```

Description

This is macro L2CAP_CONN_REQ_RESULT_NO_RESOURCES.

L2CAP_CONN_REQ_RESULT_PENDING Macro

File

command.h

C

```
#define L2CAP_CONN_REQ_RESULT_PENDING 0x0001
```

Description

This is macro L2CAP_CONN_REQ_RESULT_PENDING.

L2CAP_CONN_REQ_RESULT_SECURITY_BLOCK Macro

File

command.h

C

```
#define L2CAP_CONN_REQ_RESULT_SECURITY_BLOCK 0x0003
```

Description

This is macro L2CAP_CONN_REQ_RESULT_SECURITY_BLOCK.

L2CAP_CONN_REQ_RESULT_SUCCESS Macro

File

command.h

C

```
#define L2CAP_CONN_REQ_RESULT_SUCCESS 0x0000
```

Description

This is macro L2CAP_CONN_REQ_RESULT_SUCCESS.

L2CAP_CONN_REQ_STATUS_AUTHENTICATION_PENDING Macro

File

command.h

C

```
#define L2CAP_CONN_REQ_STATUS_AUTHENTICATION_PENDING 0x0001
```

Description

This is macro L2CAP_CONN_REQ_STATUS_AUTHENTICATION_PENDING.

L2CAP_CONN_REQ_STATUS_AUTHORIZATION_PENDING Macro

File

command.h

C

```
#define L2CAP_CONN_REQ_STATUS_AUTHORIZATION_PENDING 0x0002
```

Description

This is macro L2CAP_CONN_REQ_STATUS_AUTHORIZATION_PENDING.

L2CAP_CONN_REQ_STATUS_NO_INFO Macro

File

command.h

C

```
#define L2CAP_CONN_REQ_STATUS_NO_INFO 0x0000
```

Description

This is macro L2CAP_CONN_REQ_STATUS_NO_INFO.

L2CAP_DEFAULT_ERTX Macro

File

l2cap.h

C

```
#define L2CAP_DEFAULT_ERTX 60 /* seconds */
```

Description

seconds

L2CAP_DEFAULT_RTX Macro

File

l2cap.h

C

```
#define L2CAP_DEFAULT_RTX 3 /* seconds */
```

Description

seconds

L2CAP_ECHO_MAX_DATA_LEN Macro

File

command.h

C

```
#define L2CAP_ECHO_MAX_DATA_LEN 20
```

Description

echo request and response

L2CAP_ERETR_RECV_STATE_RECV Macro

File

channel.h

C

```
#define L2CAP_ERETR_RECV_STATE_RECV 0
```

Description

This is macro L2CAP_ERETR_RECV_STATE_RECV.

L2CAP_ERETR_RECV_STATE_REJ_SENT Macro

File

channel.h

C

```
#define L2CAP_ERETR_RECV_STATE_REJ_SENT 1
```

Description

This is macro L2CAP_ERETR_RECV_STATE_REJ_SENT.

L2CAP_ERETR_RECV_STATE_SREJ_SENT Macro

File

channel.h

C

```
#define L2CAP_ERETR_RECV_STATE_SREJ_SENT 2
```

Description

This is macro L2CAP_ERETR_RECV_STATE_SREJ_SENT.

L2CAP_ERETR_XMIT_STATE_WAIT_ACK Macro

File

channel.h

C

```
#define L2CAP_ERETR_XMIT_STATE_WAIT_ACK 1
```

Description

This is macro L2CAP_ERETR_XMIT_STATE_WAIT_ACK.

L2CAP_ERETR_XMIT_STATE_WAIT_F Macro

File

channel.h

C

```
#define L2CAP_ERETR_XMIT_STATE_WAIT_F 2
```

Description

This is macro L2CAP_ERETR_XMIT_STATE_WAIT_F.

L2CAP_ERETR_XMIT_STATE_XMIT Macro

File

channel.h

C

```
#define L2CAP_ERETR_XMIT_STATE_XMIT 0
```

Description

This is macro L2CAP_ERETR_XMIT_STATE_XMIT.

L2CAP_EXT_BI_QOS Macro

File

command.h

C

```
#define L2CAP_EXT_BI_QOS 0x00000004
```

Description

This is macro L2CAP_EXT_BI_QOS.

L2CAP_EXT_ENHANCED_RETRANSMISSION Macro

File

command.h

C

```
#define L2CAP_EXT_ENHANCED_RETRANSMISSION 0x00000008
```

Description

This is macro L2CAP_EXT_ENHANCED_RETRANSMISSION.

L2CAP_EXT_FCS_OPTION Macro

File

command.h

C

```
#define L2CAP_EXT_FCS_OPTION 0x00000020
```

Description

This is macro L2CAP_EXT_FCS_OPTION.

L2CAP_EXT_FLOW_CONTROL Macro

File

command.h

C

```
#define L2CAP_EXT_FLOW_CONTROL 0x00000001
```

Description

This is macro L2CAP_EXT_FLOW_CONTROL.

L2CAP_EXT_RETRANSMISSION Macro

File

command.h

C

```
#define L2CAP_EXT_RETRANSMISSION 0x00000002
```

Description

This is macro L2CAP_EXT_RETRANSMISSION.

L2CAP_EXT_STREAMING Macro

File

command.h

C

```
#define L2CAP_EXT_STREAMING 0x00000010
```

Description

This is macro L2CAP_EXT_STREAMING.

L2CAP_FRAME_TYPE_I Macro

File

l2cap.h

C

```
#define L2CAP_FRAME_TYPE_I 0x0
```

Description

This is macro L2CAP_FRAME_TYPE_I.

L2CAP_FRAME_TYPE_S Macro

File

l2cap.h

C

```
#define L2CAP_FRAME_TYPE_S 0x1
```

Description

This is macro L2CAP_FRAME_TYPE_S.

L2CAP_HCI_CONNECT_PACKET_TYPE Macro

File

l2cap.h

C

```
#define L2CAP_HCI_CONNECT_PACKET_TYPE _l2cap_hci_connect_packet_type
```

Description

This is macro L2CAP_HCI_CONNECT_PACKET_TYPE.

L2CAP_HCI_CONNECT_PAGE_SCAN_REPETITION_MODE Macro

File

l2cap.h

C

```
#define L2CAP_HCI_CONNECT_PAGE_SCAN_REPETITION_MODE _l2cap_hci_page_scan_repetition_mode
```

Description

This is macro L2CAP_HCI_CONNECT_PAGE_SCAN_REPETITION_MODE.

L2CAP_HCI_CONNECT_ROLE_SWITCH Macro

File

l2cap.h

C

```
#define L2CAP_HCI_CONNECT_ROLE_SWITCH _l2cap_hci_role_switch
```

Description

This is macro L2CAP_HCI_CONNECT_ROLE_SWITCH.

L2CAP_HEADER_LEN Macro

File

l2cap.h

C

```
#define L2CAP_HEADER_LEN 4
```

Description

This is macro L2CAP_HEADER_LEN.

L2CAP_IDLE_HCI_CONNECTION_TIMEOUT Macro

File

l2cap.h

C

```
#define L2CAP_IDLE_HCI_CONNECTION_TIMEOUT _l2cap_idle_hci_connection_timeout
```

Description

This is macro L2CAP_IDLE_HCI_CONNECTION_TIMEOUT.

L2CAP_INFO_NOT_SUPPORTED Macro

File

command.h

C

```
#define L2CAP_INFO_NOT_SUPPORTED 0x0000     // REDFLAG: should it be non zero?
```

Description

REDFLAG: should it be non zero?

L2CAP_INFO_RESULT_SUCCESS Macro

File

command.h

C

```
#define L2CAP_INFO_RESULT_SUCCESS 0x0000
```

Description

This is macro L2CAP_INFO_RESULT_SUCCESS.

L2CAP_INFO_TYPE_CONNECTIONLESS_MTU Macro

File

command.h

C

```
#define L2CAP_INFO_TYPE_CONNECTIONLESS_MTU 0x0001
```

Description

information request

L2CAP_INFO_TYPE_EXTENDED_SUPPORT Macro

File

command.h

C

```
#define L2CAP_INFO_TYPE_EXTENDED_SUPPORT 0x0002
```

Description

This is macro L2CAP_INFO_TYPE_EXTENDED_SUPPORT.

L2CAP_LINK_TYPE_BD_EDR Macro

File

l2cap_fixed_channel.h

C

```
#define L2CAP_LINK_TYPE_BD_EDR HCI_LINK_TYPE_BD_EDR
```

Description

This is macro L2CAP_LINK_TYPE_BD_EDR.

L2CAP_LINK_TYPE_LE Macro

File

l2cap_fixed_channel.h

C

```
#define L2CAP_LINK_TYPE_LE HCI_LINK_TYPE_LE
```

Description

This is macro L2CAP_LINK_TYPE_LE.

L2CAP_MAX_CMD_QUEUE_LEN Macro

File

l2cap_command_queue.h

C

```
#define L2CAP_MAX_CMD_QUEUE_LEN 5
```

Description

This is macro L2CAP_MAX_CMD_QUEUE_LEN.

L2CAP_MAX_FRAME_BUFFERS Macro

File

frame_buffer.h

C

```
#define L2CAP_MAX_FRAME_BUFFERS 2
```

Description

This is macro L2CAP_MAX_FRAME_BUFFERS.

L2CAP_MAX_MANAGERS Macro

File

l2cap.h

C

```
#define L2CAP_MAX_MANAGERS 1
```

Description

This is macro L2CAP_MAX_MANAGERS.

L2CAP_MAX_MTU Macro

File

l2cap.h

C

```
#define L2CAP_MAX_MTU (_hci_l2cap_buffer_len - L2CAP_HEADER_LEN)
```

Description

This is macro L2CAP_MAX_MTU.

L2CAP_MAX_OPTIONS Macro

File

command.h

C

```
#define L2CAP_MAX_OPTIONS 4
```

Description

configuration request/response

L2CAP_MAX_RTX Macro

File

l2cap.h

C

```
#define L2CAP_MAX_RTX 4
```

Description

This is macro L2CAP_MAX_RTX.

L2CAP_MGR_STATE_FREE Macro

File

l2cap.h

C

```
#define L2CAP_MGR_STATE_FREE 0
```

Description

This is macro L2CAP_MGR_STATE_FREE.

L2CAP_MGR_STATE_LISTENING_L2CAP Macro

File

l2cap.h

C

```
#define L2CAP_MGR_STATE_LISTENING_L2CAP 2
```

Description

This is macro L2CAP_MGR_STATE_LISTENING_L2CAP.

L2CAP_MGR_STATE_USED Macro

File

l2cap.h

C

```
#define L2CAP_MGR_STATE_USED 1
```

Description

This is macro L2CAP_MGR_STATE_USED.

L2CAP_MIN_MTU Macro

File

l2cap.h

C

```
#define L2CAP_MIN_MTU 48
```

Description

This is macro L2CAP_MIN_MTU.

L2CAP_MONITOR_TIMEOUT Macro

File

l2cap.h

C

```
#define L2CAP_MONITOR_TIMEOUT ((L2CAP_RFC_MONITOR_TIMEOUT / 1000) - 2)
```

Description

This is macro L2CAP_MONITOR_TIMEOUT.

L2CAP_OPTION_LEN_FLASH_QOS Macro

File

command.h

C

```
#define L2CAP_OPTION_LEN_FLASH_QOS 22
```

Description

This is macro L2CAP_OPTION_LEN_FLASH_QOS.

L2CAP_OPTION_LEN_FLASH_RFC Macro

File

command.h

C

```
#define L2CAP_OPTION_LEN_FLASH_RFC 9
```

Description

This is macro L2CAP_OPTION_LEN_FLASH_RFC.

L2CAP_OPTION_LEN_FLASH_TIMEOUT Macro

File

command.h

C

```
#define L2CAP_OPTION_LEN_FLASH_TIMEOUT 2
```

Description

This is macro L2CAP_OPTION_LEN_FLASH_TIMEOUT.

L2CAP_OPTION_LEN_MAX_MTU Macro

File

command.h

C

```
#define L2CAP_OPTION_LEN_MAX_MTU 2
```

Description

This is macro L2CAP_OPTION_LEN_MAX_MTU.

L2CAP_OPTION_LEN_UNKNOWN Macro

File

command.h

C

```
#define L2CAP_OPTION_LEN_UNKNOWN 22
```

Description

This is macro L2CAP_OPTION_LEN_UNKNOWN.

L2CAP_OPTION_LEN_UNKNOWN_DATA Macro

File

command.h

C

```
#define L2CAP_OPTION_LEN_UNKNOWN_DATA 9//21
```

Description

21

L2CAP_OPTION_TYPE_FLASH_QOS Macro

File

command.h

C

```
#define L2CAP_OPTION_TYPE_FLASH_QOS 0x03
```

Description

This is macro L2CAP_OPTION_TYPE_FLASH_QOS.

L2CAP_OPTION_TYPE_FLASH_QOS_FLAG Macro

File

command.h

C

```
#define L2CAP_OPTION_TYPE_FLASH_QOS_FLAG 0x04
```

Description

This is macro L2CAP_OPTION_TYPE_FLASH_QOS_FLAG.

L2CAP_OPTION_TYPE_FLASH_RFC Macro

File

command.h

C

```
#define L2CAP_OPTION_TYPE_FLASH_RFC 0x04
```

Description

This is macro L2CAP_OPTION_TYPE_FLASH_RFC.

L2CAP_OPTION_TYPE_FLASH_RFC_FLAG Macro

File

command.h

C

```
#define L2CAP_OPTION_TYPE_FLASH_RFC_FLAG 0x08
```

Description

This is macro L2CAP_OPTION_TYPE_FLASH_RFC_FLAG.

L2CAP_OPTION_TYPE_FLASH_TIMEOUT Macro

File

command.h

C

```
#define L2CAP_OPTION_TYPE_FLASH_TIMEOUT 0x02
```

Description

This is macro L2CAP_OPTION_TYPE_FLASH_TIMEOUT.

L2CAP_OPTION_TYPE_FLASH_TIMEOUT_FLAG Macro

File

command.h

C

```
#define L2CAP_OPTION_TYPE_FLASH_TIMEOUT_FLAG 0x02
```

Description

This is macro L2CAP_OPTION_TYPE_FLASH_TIMEOUT_FLAG.

L2CAP_OPTION_TYPE_MAX_MTU Macro

File

command.h

C

```
#define L2CAP_OPTION_TYPE_MAX_MTU 0x01
```

Description

This is macro L2CAP_OPTION_TYPE_MAX_MTU.

L2CAP_OPTION_TYPE_MAX_MTU_FLAG Macro

File

command.h

C

```
#define L2CAP_OPTION_TYPE_MAX_MTU_FLAG 0x01
```

Description

This is macro L2CAP_OPTION_TYPE_MAX_MTU_FLAG.

L2CAP_OPTION_TYPE_UNKNOWN_FLAG Macro

File

command.h

C

```
#define L2CAP_OPTION_TYPE_UNKNOWN_FLAG 0x10
```

Description

This is macro L2CAP_OPTION_TYPE_UNKNOWN_FLAG.

L2CAP_PACKET_DATA_TYPE_RAW Macro

File

l2cap_packet.h

C

```
#define L2CAP_PACKET_DATA_TYPE_RAW 0
```

Description

This is macro L2CAP_PACKET_DATA_TYPE_RAW.

L2CAP_PACKET_DATA_TYPE_SMART Macro

File

l2cap_packet.h

C

```
#define L2CAP_PACKET_DATA_TYPE_SMART 1
```

Description

This is macro L2CAP_PACKET_DATA_TYPE_SMART.

L2CAP_REJECT_REASON_INVALID_CHANNEL Macro

File

command.h

C

```
#define L2CAP_REJECT_REASON_INVALID_CHANNEL 0x0002
```

Description

This is macro L2CAP_REJECT_REASON_INVALID_CHANNEL.

L2CAP_REJECT_REASON_MTU_EXCEEDED Macro

File

command.h

C

```
#define L2CAP_REJECT_REASON_MTU_EXCEEDED 0x0001
```

Description

This is macro L2CAP_REJECT_REASON_MTU_EXCEEDED.

L2CAP_REJECT_REASON_NOT_UNDERSTOOD Macro

File

command.h

C

```
#define L2CAP_REJECT_REASON_NOT_UNDERSTOOD 0x0000
```

Description

This is macro L2CAP_REJECT_REASON_NOT_UNDERSTOOD.

L2CAP_RETR_TIMEOUT Macro

File

l2cap.h

C

```
#define L2CAP_RETR_TIMEOUT (L2CAP_RFC_ERETR_TIMEOUT / 1000)
```

Description

This is macro L2CAP_RETR_TIMEOUT.

L2CAP_RFC_BASIC Macro

File

command.h

C

```
#define L2CAP_RFC_BASIC 0x00
```

Description

This is macro L2CAP_RFC_BASIC.

L2CAP_RFC_ENHANCED_RETRANSMISSION Macro

File

command.h

C

```
#define L2CAP_RFC_ENHANCED_RETRANSMISSION 0x03
```

Description

This is macro L2CAP_RFC_ENHANCED_RETRANSMISSION.

L2CAP_RFC_ERETR_TIMEOUT Macro

File

command.h

C

```
#define L2CAP_RFC_ERETR_TIMEOUT (3 * 1000)  // 3 secs
```

Description

3 secs

L2CAP_RFC_ERETR_TX_WINDOW Macro

File

command.h

C

```
#define L2CAP_RFC_ERETR_TX_WINDOW 1
```

Description

This is macro L2CAP_RFC_ERETR_TX_WINDOW.

L2CAP_RFC_FLOW_CONTROL Macro

File

command.h

C

```
#define L2CAP_RFC_FLOW_CONTROL 0x02
```

Description

This is macro L2CAP_RFC_FLOW_CONTROL.

L2CAP_RFC_MONITOR_TIMEOUT Macro

File

command.h

C

```
#define L2CAP_RFC_MONITOR_TIMEOUT (12 * 1000) // 12 secs
```

Description

12 secs

L2CAP_RFC_RETRANSMISSION Macro

File

command.h

C

```
#define L2CAP_RFC_RETRANSMISSION 0x01
```

Description

This is macro L2CAP_RFC_RETRANSMISSION.

L2CAP_RFC_STREAMING Macro

File

command.h

C

```
#define L2CAP_RFC_STREAMING 0x04
```

Description

This is macro L2CAP_RFC_STREAMING.

L2CAP_SAR_SDU_CONTINUE Macro

File

l2cap.h

C

```
#define L2CAP_SAR_SDU_CONTINUE 0x3
```

Description

This is macro L2CAP_SAR_SDU_CONTINUE.

L2CAP_SAR_SDU_START Macro

File

l2cap.h

C

```
#define L2CAP_SAR_SDU_START 0x1
```

Description

This is macro L2CAP_SAR_SDU_START.

L2CAP_SAR_SDU_STOP Macro

File

l2cap.h

C

```
#define L2CAP_SAR_SDU_STOP 0x2
```

Description

This is macro L2CAP_SAR_SDU_STOP.

L2CAP_SAR_UNSEGMENTED Macro

File

l2cap.h

C

```
#define L2CAP_SAR_UNSEGMENTED 0x0
```

Description

This is macro L2CAP_SAR_UNSEGMENTED.

L2CAP_SERVICE_TYPE_BEST_EFFORT Macro

File

command.h

C

```
#define L2CAP_SERVICE_TYPE_BEST_EFFORT 0x01
```

Description

This is macro L2CAP_SERVICE_TYPE_BEST_EFFORT.

L2CAP_SERVICE_TYPE_GUARANTEED Macro

File

command.h

C

```
#define L2CAP_SERVICE_TYPE_GUARANTEED 0x02
```

Description

This is macro L2CAP_SERVICE_TYPE_GUARANTEED.

L2CAP_SERVICE_TYPE_NO_TRAFFIC Macro

File

command.h

C

```
#define L2CAP_SERVICE_TYPE_NO_TRAFFIC 0x00
```

Description

This is macro L2CAP_SERVICE_TYPE_NO_TRAFFIC.

L2CAP_SFUNCTION_REJ Macro

File

l2cap.h

C

```
#define L2CAP_SFUNCTION_REJ 0x1
```

Description

This is macro L2CAP_SFUNCTION_REJ.

L2CAP_SFUNCTION_RNR Macro

File

l2cap.h

C

```
#define L2CAP_SFUNCTION_RNR 0x2
```

Description

This is macro L2CAP_SFUNCTION_RNR.

L2CAP_SFUNCTION_RR Macro

File

l2cap.h

C

```
#define L2CAP_SFUNCTION_RR 0x0
```

Description

This is macro L2CAP_SFUNCTION_RR.

L2CAP_SFUNCTION_SREJ Macro

File

l2cap.h

C

```
#define L2CAP_SFUNCTION_SREJ 0x3
```

Description

This is macro L2CAP_SFUNCTION_SREJ.

PSM_ATT Macro

File

l2cap.h

C

```
#define PSM_ATT 0x001F
```

Description

This is macro PSM_ATT.

PSM_AVCTP Macro

File

l2cap.h

C

```
#define PSM_AVCTP 0x0017
```

Description

This is macro PSM_AVCTP.

PSM_AVCTP_Browsing Macro

File

l2cap.h

C

```
#define PSM_AVCTP_Browsing 0x001B
```

Description

This is macro PSM_AVCTP_Browsing.

PSM_AVDTP Macro

File

l2cap.h

C

```
#define PSM_AVDTP 0x0019
```

Description

This is macro PSM_AVDTP.

PSM_BNEP Macro

File

l2cap.h

C

```
#define PSM_BNEP 0x000F
```

Description

This is macro PSM_BNEP.

PSM_HID_Control Macro

File

l2cap.h

C

```
#define PSM_HID_Control 0x0011
```

Description

This is macro PSM_HID_Control.

PSM_HID_Interrupt Macro

File

l2cap.h

C

```
#define PSM_HID_Interrupt 0x0013
```

Description

This is macro PSM_HID_Interrupt.

PSM_RFCOMM Macro

File

l2cap.h

C

```
#define PSM_RFCOMM 0x0003
```

Description

This is macro PSM_RFCOMM.

PSM_SDP Macro

File

l2cap.h

C

```
#define PSM_SDP 0x0001
```

Description

This is macro PSM_SDP.

SET_F_BIT Macro

File

l2cap_eretr.h

C

```
#define SET_F_BIT(ctl, fbit) ctl = ((ctl & ~0x80) | ((fbit & 1) << 7))
```

Description

This is macro SET_F_BIT.

SET_FRAME_TYPE Macro

File

l2cap_eretr.h

C

```
#define SET_FRAME_TYPE(ctl, type) ctl = ((ctl & ~1) | (type & 1))
```

Description

This is macro SET_FRAME_TYPE.

SET_P_BIT Macro

File

l2cap_eretr.h

C

```
#define SET_P_BIT(ctl, pbit) ctl = ((ctl & ~0x10) | ((pbit & 1) << 4))
```

Description

This is macro SET_P_BIT.

SET_REQ_SEQ Macro

File

l2cap_eretr.h

C

```
#define SET_REQ_SEQ(ctl, seq) ctl = ((ctl & ~0x3F00) | ((seq & 0x3F) << 8))
```

Description

This is macro SET_REQ_SEQ.

SET_S_FUNCTION Macro

File

l2cap_eretr.h

C

```
#define SET_S_FUNCTION(ctl, s) ctl = ((ctl & ~0x0C) | ((s & 0x3) << 2))
```

Description

This is macro SET_S_FUNCTION.

SET_SAR Macro

File

l2cap_eretr.h

C

```
#define SET_SAR(ctl, sar) ctl = ((ctl & ~0xC000) | ((seq & 0x3) << 14))
```

Description

This is macro SET_SAR.

SET_TX_SEQ Macro

File

l2cap_eretr.h

C

```
#define SET_TX_SEQ(ctl, seq) ctl = ((ctl & ~0x7E) | ((seq & 0x3F) << 1))
```

Description

This is macro SET_TX_SEQ.

_bt_l2cap_start_monitor_timer Macro

File

l2cap_eretr.h

C

```
#define _bt_l2cap_start_monitor_timer(pch) pch->ext->monitor_timer_start_time = L2CAP_MONITOR_TIMEOUT
```

Description

This is macro _bt_l2cap_start_monitor_timer.

_bt_l2cap_start_monitor_timer_if_not_running Macro

File

l2cap_eretr.h

C

```
#define _bt_l2cap_start_monitor_timer_if_not_running(pch) { if (pch->ext->monitor_timer_start_time == 0)
pch->ext->monitor_timer_start_time = L2CAP_MONITOR_TIMEOUT; }
```

Description

This is macro _bt_l2cap_start_monitor_timer_if_not_running.

_bt_l2cap_start_retr_timer Macro

File

l2cap_eretr.h

C

```
#define _bt_l2cap_start_retr_timer(pch) pch->ext->retr_timer_start_time = L2CAP_RETR_TIMEOUT
```

Description

This is macro _bt_l2cap_start_retr_timer.

_bt_l2cap_start_retr_timer_if_not_running Macro

File

l2cap_eretr.h

C

```
#define _bt_l2cap_start_retr_timer_if_not_running(pch) { if (pch->ext->retr_timer_start_time == 0)
pch->ext->retr_timer_start_time = L2CAP_RETR_TIMEOUT; }
```

Description

This is macro _bt_l2cap_start_retr_timer_if_not_running.

_bt_l2cap_stop_monitor_timer Macro

File

l2cap_eretr.h

C

```
#define _bt_l2cap_stop_monitor_timer(pch) pch->ext->monitor_timer_start_time = 0
```

Description

This is macro _bt_l2cap_stop_monitor_timer.

_bt_l2cap_stop_retr_timer Macro

File

l2cap_eretr.h

C

```
#define _bt_l2cap_stop_retr_timer(pch) pch->ext->retr_timer_start_time = 0
```

Description

This is macro _bt_l2cap_stop_retr_timer.

_bt_l2cap_channel_t Structure

File

channel.h

C

```
struct _bt_l2cap_channel_t {
  bt_id cid;
  struct _bt_l2cap_mgr_s * l2cap_mgr;
  bt_byte mode;
  bt_byte type;
  bt_byte state;
  bt_byte flags;
  bt_id cid_destination;
  bt_int psm;
  bt_int cfg_try_cnt;
  bt_l2cap_read_data_callback_fp _read_data_cb;
  bt_l2cap_send_data_callback_fp _send_data_cb;
  void* _send_data_cb_param;
  bt_l2cap_state_changed_callback_fp _state_changed_cb;
  void* _state_changed_param;
  struct _bt_hci_conn_state_s * hci_conn;
  bt_l2cap_packet_t tx_packet;
  bt_uint response_config_options;
  bt_uint response_mtu;
  bt_int response_flash_timeout;
  bt_l2cap_option_rfc_t response_option_rfc;
  bt_byte response_option_unknown_type;
  bt_l2cap_option_unknown_t response_option_unknown;
  bt_uint request_config_options;
  bt_int request_mtu;
  bt_int request_flash_timeout;
  bt_l2cap_option_rfc_t request_option_rfc;
  bt_byte connect_cmd_id;
  bt_byte signal_command;
  bt_signal_t signal;
  bt_l2cap_channel_ext_t* ext;
};
```

Members

Members	Description
bt_uint response_config_options;	configuration

Description

This is record _bt_l2cap_channel_t.

bt_l2cap_cfg_option_p Structure

File

command.h

C

```
typedef struct _bt_l2cap_cfg_option_t {
  bt_byte type;
  union {
    bt_l2cap_option_max_mtu_t max_mtu;
    bt_l2cap_option_flash_timeout_t flash_timeout;
    bt_l2cap_option_rfc_t rfc;
    bt_l2cap_option_unknown_t unknown;
  } opt;
} bt_l2cap_cfg_option_t, * bt_l2cap_cfg_option_p;
```

Members

Members	Description
bt_l2cap_option_rfc_t rfc;	option_qos qos;

Description

This is type bt_l2cap_cfg_option_p.

bt_l2cap_cfg_option_t Structure

File

command.h

C

```
typedef struct _bt_l2cap_cfg_option_t {
  bt_byte type;
  union {
    bt_l2cap_option_max_mtu_t max_mtu;
    bt_l2cap_option_flash_timeout_t flash_timeout;
    bt_l2cap_option_rfc_t rfc;
    bt_l2cap_option_unknown_t unknown;
  } opt;
} bt_l2cap_cfg_option_t, * bt_l2cap_cfg_option_p;
```

Members

Members	Description
bt_l2cap_option_rfc_t rfc;	option_qos qos;

Description

This is type bt_l2cap_cfg_option_t.

bt_l2cap_channel_t Type

File

channel.h

C

```
typedef struct _bt_l2cap_channel_t bt_l2cap_channel_t;
```

Description

This is type bt_l2cap_channel_t.

bt_l2cap_cmd_assembler_fp Type

File

l2cap_private.h

C

```
typedef bt_bool (* bt_l2cap_cmd_assembler_fp)(bt_l2cap_cmd_header_t* pcmd, bt_byte* buffer, bt_int
buffer_len, bt_int* poffset);
```

Description

This is type bt_l2cap_cmd_assembler_fp.

bt_l2cap_cmd_config_req_p Structure

File

command.h

C

```
typedef struct _bt_l2cap_cmd_config_req_t {
  bt_l2cap_cmd_header_t header;
  bt_id cid_destination;
  bt_int flags;
  bt_int option_count;
```

```
  struct _bt_l2cap_channel_t* channel;
  bt_l2cap_cfg_option_t* options;
} bt_l2cap_cmd_config_req_t, * bt_l2cap_cmd_config_req_p;
```

Description

This is type bt_l2cap_cmd_config_req_p.

bt_l2cap_cmd_config_req_t Structure

File

command.h

C

```
typedef struct _bt_l2cap_cmd_config_req_t {
  bt_l2cap_cmd_header_t header;
  bt_id cid_destination;
  bt_int flags;
  bt_int option_count;
  struct _bt_l2cap_channel_t* channel;
  bt_l2cap_cfg_option_t* options;
} bt_l2cap_cmd_config_req_t, * bt_l2cap_cmd_config_req_p;
```

Description

This is type bt_l2cap_cmd_config_req_t.

bt_l2cap_cmd_config_res_p Structure

File

command.h

C

```
typedef struct _bt_l2cap_cmd_config_res_t {
  bt_l2cap_cmd_header_t header;
  bt_id cid_source;
  bt_int flags;
  bt_int result;
  bt_int option_count;
  struct _bt_l2cap_channel_t* channel;
  bt_l2cap_cfg_option_t* options;
} bt_l2cap_cmd_config_res_t, * bt_l2cap_cmd_config_res_p;
```

Description

This is type bt_l2cap_cmd_config_res_p.

bt_l2cap_cmd_config_res_t Structure

File

command.h

C

```
typedef struct _bt_l2cap_cmd_config_res_t {
  bt_l2cap_cmd_header_t header;
  bt_id cid_source;
  bt_int flags;
  bt_int result;
  bt_int option_count;
  struct _bt_l2cap_channel_t* channel;
  bt_l2cap_cfg_option_t* options;
} bt_l2cap_cmd_config_res_t, * bt_l2cap_cmd_config_res_p;
```

Description

This is type bt_l2cap_cmd_config_res_t.

bt_l2cap_cmd_conn_param_update_req_t Structure

File

command.h

C

```
typedef struct _bt_l2cap_cmd_conn_param_update_req_t {
  bt_l2cap_cmd_header_t header;
  bt_uint min_interval;
  bt_uint max_interval;
  bt_uint slave_latency;
  bt_uint timeout_multiplier;
} bt_l2cap_cmd_conn_param_update_req_t;
```

Description

connection parameter update

bt_l2cap_cmd_conn_param_update_res_t Structure

File

command.h

C

```
typedef struct _bt_l2cap_cmd_conn_param_update_res_t {
  bt_l2cap_cmd_header_t header;
  bt_int result;
} bt_l2cap_cmd_conn_param_update_res_t;
```

Description

This is type bt_l2cap_cmd_conn_param_update_res_t.

bt_l2cap_cmd_connection_req_p Structure

File

command.h

C

```
typedef struct _bt_l2cap_cmd_connection_req_t {
  bt_l2cap_cmd_header_t header;
  bt_int psm;
  bt_id cid_source;
} bt_l2cap_cmd_connection_req_t, * bt_l2cap_cmd_connection_req_p;
```

Description

This is type bt_l2cap_cmd_connection_req_p.

bt_l2cap_cmd_connection_req_t Structure

File

command.h

C

```
typedef struct _bt_l2cap_cmd_connection_req_t {
  bt_l2cap_cmd_header_t header;
  bt_int psm;
  bt_id cid_source;
} bt_l2cap_cmd_connection_req_t, * bt_l2cap_cmd_connection_req_p;
```

Description

This is type bt_l2cap_cmd_connection_req_t.

bt_l2cap_cmd_connection_res_p Structure

File

command.h

C

```
typedef struct _bt_l2cap_cmd_connection_res_t {
  bt_l2cap_cmd_header_t header;
  bt_id cid_destination;
  bt_id cid_source;
  bt_int result;
  bt_int status;
} bt_l2cap_cmd_connection_res_t, * bt_l2cap_cmd_connection_res_p;
```

Description

This is type bt_l2cap_cmd_connection_res_p.

bt_l2cap_cmd_connection_res_t Structure

File

command.h

C

```
typedef struct _bt_l2cap_cmd_connection_res_t {
  bt_l2cap_cmd_header_t header;
  bt_id cid_destination;
  bt_id cid_source;
  bt_int result;
  bt_int status;
} bt_l2cap_cmd_connection_res_t, * bt_l2cap_cmd_connection_res_p;
```

Description

This is type bt_l2cap_cmd_connection_res_t.

bt_l2cap_cmd_disconnection_req_p Structure

File

command.h

C

```
typedef struct _bt_l2cap_cmd_disconnection_req_t {
  bt_l2cap_cmd_header_t header;
  bt_id cid_destination;
  bt_id cid_source;
} bt_l2cap_cmd_disconnection_req_t, * bt_l2cap_cmd_disconnection_req_p;
```

Description

This is type bt_l2cap_cmd_disconnection_req_p.

bt_l2cap_cmd_disconnection_req_t Structure

File

command.h

C

```
typedef struct _bt_l2cap_cmd_disconnection_req_t {
  bt_l2cap_cmd_header_t header;
  bt_id cid_destination;
  bt_id cid_source;
} bt_l2cap_cmd_disconnection_req_t, * bt_l2cap_cmd_disconnection_req_p;
```

Description

This is type bt_l2cap_cmd_disconnection_req_t.

bt_l2cap_cmd_echo_req_p Structure

File

command.h

C

```
typedef struct _bt_l2cap_cmd_echo_req_t {
  bt_l2cap_cmd_header_t header;
  bt_int len;
  bt_byte data[L2CAP_ECHO_MAX_DATA_LEN];
} bt_l2cap_cmd_echo_req_t, * bt_l2cap_cmd_echo_req_p;
```

Description

This is type bt_l2cap_cmd_echo_req_p.

bt_l2cap_cmd_echo_req_t Structure

File

command.h

C

```
typedef struct _bt_l2cap_cmd_echo_req_t {
  bt_l2cap_cmd_header_t header;
  bt_int len;
  bt_byte data[L2CAP_ECHO_MAX_DATA_LEN];
} bt_l2cap_cmd_echo_req_t, * bt_l2cap_cmd_echo_req_p;
```

Description

This is type bt_l2cap_cmd_echo_req_t.

bt_l2cap_cmd_echo_res_p Structure

File

command.h

C

```
typedef struct _bt_l2cap_cmd_echo_res_t {
  bt_l2cap_cmd_header_t header;
  bt_int len;
  bt_byte data[L2CAP_ECHO_MAX_DATA_LEN];
} bt_l2cap_cmd_echo_res_t, * bt_l2cap_cmd_echo_res_p;
```

Description

This is type bt_l2cap_cmd_echo_res_p.

bt_l2cap_cmd_echo_res_t Structure

File

command.h

C

```
typedef struct _bt_l2cap_cmd_echo_res_t {
  bt_l2cap_cmd_header_t header;
  bt_int len;
  bt_byte data[L2CAP_ECHO_MAX_DATA_LEN];
} bt_l2cap_cmd_echo_res_t, * bt_l2cap_cmd_echo_res_p;
```

Description

This is type bt_l2cap_cmd_echo_res_t.

bt_l2cap_cmd_header_p Structure

File

command.h

C

```
typedef struct _bt_l2cap_cmd_header_t {
  struct _bt_l2cap_cmd_header_t* next_cmd;
  bt_byte code;
  bt_byte id;
  bt_byte status;
  bt_long send_time;
  bt_int rtx;
  bt_byte rtx_count;
  pf_l2cap_cmd_callback cb;
  struct _bt_hci_conn_state_s * phci_conn;
} bt_l2cap_cmd_header_t, * bt_l2cap_cmd_header_p;
```

Description

This is type bt_l2cap_cmd_header_p.

bt_l2cap_cmd_header_t Structure

File

command.h

C

```
typedef struct _bt_l2cap_cmd_header_t {
  struct _bt_l2cap_cmd_header_t* next_cmd;
  bt_byte code;
  bt_byte id;
  bt_byte status;
  bt_long send_time;
  bt_int rtx;
  bt_byte rtx_count;
  pf_l2cap_cmd_callback cb;
  struct _bt_hci_conn_state_s * phci_conn;
} bt_l2cap_cmd_header_t, * bt_l2cap_cmd_header_p;
```

Description

This is type bt_l2cap_cmd_header_t.

bt_l2cap_cmd_info_req_p Structure

File

command.h

C

```
typedef struct _bt_l2cap_cmd_info_req_t {
  bt_l2cap_cmd_header_t header;
  bt_int type;
} bt_l2cap_cmd_info_req_t, * bt_l2cap_cmd_info_req_p;
```

Description

This is type bt_l2cap_cmd_info_req_p.

bt_l2cap_cmd_info_req_t Structure

File

command.h

C

```
typedef struct _bt_l2cap_cmd_info_req_t {
  bt_l2cap_cmd_header_t header;
  bt_int type;
} bt_l2cap_cmd_info_req_t, * bt_l2cap_cmd_info_req_p;
```

Description

This is type bt_l2cap_cmd_info_req_t.

bt_l2cap_cmd_info_res_p Structure

File

command.h

C

```
typedef struct _bt_l2cap_cmd_info_res_t {
  bt_l2cap_cmd_header_t header;
  bt_int type;
  bt_int result;
  union {
    bt_int mtu;
    bt_long mask;
    bt_long fixed_channles[2];
  } data;
} bt_l2cap_cmd_info_res_t, * bt_l2cap_cmd_info_res_p;
```

Description

This is type bt_l2cap_cmd_info_res_p.

bt_l2cap_cmd_info_res_t Structure

File

command.h

C

```
typedef struct _bt_l2cap_cmd_info_res_t {
  bt_l2cap_cmd_header_t header;
  bt_int type;
  bt_int result;
  union {
    bt_int mtu;
    bt_long mask;
    bt_long fixed_channles[2];
  } data;
} bt_l2cap_cmd_info_res_t, * bt_l2cap_cmd_info_res_p;
```

Description

This is type bt_l2cap_cmd_info_res_t.

bt_l2cap_cmd_parser_fp Type

File

l2cap_private.h

C

```
typedef bt_l2cap_cmd_header_t* (* bt_l2cap_cmd_parser_fp)(bt_byte* pdata, bt_int len, bt_int* poffset);
```

Description

This is type bt_l2cap_cmd_parser_fp.

bt_l2cap_cmd_reject_p Structure

File

command.h

C

```
typedef struct _bt_l2cap_cmd_reject_t {
  bt_l2cap_cmd_header_t header;
  bt_uint reason;
  union {
    bt_int actual_mtu;
    struct {
      bt_id cid_local;
      bt_id cid_remote;
    } cid;
  } data;
} bt_l2cap_cmd_reject_t, * bt_l2cap_cmd_reject_p;
```

Description

This is type bt_l2cap_cmd_reject_p.

bt_l2cap_cmd_reject_param_t Union

File

command.h

C

```
typedef union _bt_l2cap_cmd_reject_param_t {
  bt_int actual_mtu;
  struct {
    bt_id cid_local;
    bt_id cid_remote;
  } cid;
} bt_l2cap_cmd_reject_param_t;
```

Description

This is type bt_l2cap_cmd_reject_param_t.

bt_l2cap_cmd_reject_t Structure

File

command.h

C

```
typedef struct _bt_l2cap_cmd_reject_t {
  bt_l2cap_cmd_header_t header;
  bt_uint reason;
  union {
    bt_int actual_mtu;
    struct {
      bt_id cid_local;
      bt_id cid_remote;
    } cid;
  } data;
} bt_l2cap_cmd_reject_t, * bt_l2cap_cmd_reject_p;
```

Description

This is type bt_l2cap_cmd_reject_t.

bt_l2cap_command_t Union

File

command.h

C

```
typedef union _bt_l2cap_command_t {
  bt_l2cap_cmd_reject_t reject;
  bt_l2cap_cmd_connection_req_t connection_req;
  bt_l2cap_cmd_connection_res_t connection_res;
  bt_l2cap_cmd_disconnection_req_t disconnection_req;
  cmd_disconnection_res disconnection_res;
  bt_l2cap_cmd_config_req_t config_req;
  bt_l2cap_cmd_config_res_t config_res;
  bt_l2cap_cmd_echo_req_t echo_req;
  bt_l2cap_cmd_echo_res_t echo_res;
  bt_l2cap_cmd_info_req_t info_req;
  bt_l2cap_cmd_info_res_t info_res;
  bt_l2cap_cmd_conn_param_update_req_t conn_update_req;
  bt_l2cap_cmd_conn_param_update_res_t conn_update_res;
} bt_l2cap_command_t;
```

Description

This is type bt_l2cap_command_t.

bt_l2cap_connect_callback_fp Type

File

l2cap.h

C

```
typedef void (* bt_l2cap_connect_callback_fp)(bt_l2cap_channel_t* pch, void* param);
```

Description

This is type bt_l2cap_connect_callback_fp.

bt_l2cap_eretr_xmit_event_e Enumeration

File

l2cap_eretr.h

C

```
typedef enum _bt_l2cap_eretr_xmit_event_e {
  L2CAP_ERETR_EVENT_DATA_REQUEST,
  L2CAP_ERETR_EVENT_LOCAL_BUSY_DETECTED,
  L2CAP_ERETR_EVENT_LOCAL_BUSY_CLEAR,
  L2CAP_ERETR_EVENT_RECV_REQSEQ_AND_FBIT,
  L2CAP_ERETR_EVENT_RECV_FBIT,
  L2CAP_ERETR_EVENT_RETRANSMIT_TIMER_EXPIRED,
  L2CAP_ERETR_EVENT_MONITOR_TIMER_EXPIRED
} bt_l2cap_eretr_xmit_event_e;
```

Description

This is type bt_l2cap_eretr_xmit_event_e.

bt_l2cap_fixed_channel_t Structure

File

l2cap.h

C

```
typedef struct _bt_l2cap_fixed_channel_s {
```

```
  bt_id cid;
  bt_byte link_type;
  bt_l2cap_listen_callback_fp listen_cb;
  bt_l2cap_connect_callback_fp connect_cb;
  void* listen_cb_param;
  void* connect_cb_param;
} bt_l2cap_fixed_channel_t;
```

Description

This is type bt_l2cap_fixed_channel_t.

bt_l2cap_frame_desc_t Structure

File

channel.h

C

```
typedef struct _bt_l2cap_frame_desc_s {
  bt_uint ctl;
  bt_byte data_type;
  union {
    bt_byte* raw;
    bt_packet_t* packet;
  } data;
  bt_int len;
  bt_byte tx_seq;
  bt_byte retry_count;
  bt_uint fcs;
  bt_l2cap_send_data_callback_fp callback;
  void* callback_param;
} bt_l2cap_frame_desc_t;
```

Description

typedef union _bt_l2cap_frame_control_s { struct { bt_byte frame_type:1; bt_byte txSeq:6; bt_byte f:1; bt_byte reqSeq:6; bt_byte sar:2; } iframe; struct { bt_byte frame_type:1; bt_byte reserved:1; bt_byte s:2; bt_byte p:1; bt_byte reserved2:2; bt_byte f:1; bt_byte reqSeq:6; bt_byte reserved3:2; } sframe; } bt_l2cap_frame_control_t;

bt_l2cap_listen_callback_fp Type

File

l2cap.h

C

```
typedef void (* bt_l2cap_listen_callback_fp)(bt_l2cap_channel_t* pch, void* param);
```

Description

This is type bt_l2cap_listen_callback_fp.

bt_l2cap_mgr_p Structure

File

l2cap.h

C

```
typedef struct _bt_l2cap_mgr_s {
  bt_int state;
  bt_l2cap_psm_t* _psms;
  bt_l2cap_fixed_channel_t* _fixed_channels;
  bt_l2cap_channel_t* _channels;
  bt_queue_element_t* cmd_queue;
  bt_queue_element_t* cmd_res_queue;
  bt_byte max_rtx;
  bt_hci_ctrl_state_t* hci_ctrl;
  bt_l2cap_packet_t cmd_tx_packet;
  bt_queue_element_t* connect_queue;
  bt_buffer_mgr_t connect_params_mgr;
```

```
  struct {
    unsigned int connecting : 1;
    unsigned int sig_cmd : 1;
    unsigned int sig_connect : 1;
    unsigned int sending : 1;
  } flags;
  bt_hci_listener_t hci_disconnect_listener;
  bt_byte connect_chmode;
} bt_l2cap_mgr_t, * bt_l2cap_mgr_p;
```

Members

Members	Description
bt_queue_element_t* connect_queue;	bt_int connect_psm; bt_l2cap_connect_callback_fp connect_cb; void* connect_param; bt_l2cap_state_changed_callback_fp connect_state_cb;

Description

This is type bt_l2cap_mgr_p.

bt_l2cap_mgr_t Structure

File

l2cap.h

C

```
typedef struct _bt_l2cap_mgr_s {
  bt_int state;
  bt_l2cap_psm_t* _psms;
  bt_l2cap_fixed_channel_t* _fixed_channels;
  bt_l2cap_channel_t* _channels;
  bt_queue_element_t* cmd_queue;
  bt_queue_element_t* cmd_res_queue;
  bt_byte max_rtx;
  bt_hci_ctrl_state_t* hci_ctrl;
  bt_l2cap_packet_t cmd_tx_packet;
  bt_queue_element_t* connect_queue;
  bt_buffer_mgr_t connect_params_mgr;
  struct {
    unsigned int connecting : 1;
    unsigned int sig_cmd : 1;
    unsigned int sig_connect : 1;
    unsigned int sending : 1;
  } flags;
  bt_hci_listener_t hci_disconnect_listener;
  bt_byte connect_chmode;
} bt_l2cap_mgr_t, * bt_l2cap_mgr_p;
```

Members

Members	Description
bt_queue_element_t* connect_queue;	bt_int connect_psm; bt_l2cap_connect_callback_fp connect_cb; void* connect_param; bt_l2cap_state_changed_callback_fp connect_state_cb;

Description

This is type bt_l2cap_mgr_t.

bt_l2cap_option_flash_timeout_t Structure

File

command.h

C

```
typedef struct _bt_l2cap_option_flash_timeout_t {
  bt_int timeout;
} bt_l2cap_option_flash_timeout_t;
```

Description

This is type bt_l2cap_option_flash_timeout_t.

bt_l2cap_option_max_mtu_t Structure

File

command.h

C

```
typedef struct _bt_l2cap_option_max_mtu_t {
  bt_uint mtu;
} bt_l2cap_option_max_mtu_t;
```

Description

This is type bt_l2cap_option_max_mtu_t.

bt_l2cap_option_qos_t Structure

File

command.h

C

```
typedef struct _bt_l2cap_option_qos_t {
  bt_byte flags;
  bt_byte service_type;
  bt_long token_rate;
  bt_long token_bucket_size;
  bt_long peak_bandwidth;
  bt_long latency;
  bt_long delay_variation;
} bt_l2cap_option_qos_t;
```

Description

This is type bt_l2cap_option_qos_t.

bt_l2cap_option_rfc_t Structure

File

command.h

C

```
typedef struct _bt_l2cap_option_rfc_t {
  bt_byte mode;
  bt_byte tx_window_size;
  bt_byte max_transmit;
  bt_int retr_timeout;
  bt_int monitor_timeout;
  bt_int mps;
} bt_l2cap_option_rfc_t;
```

Description

This is type bt_l2cap_option_rfc_t.

bt_l2cap_option_unknown_t Structure

File

command.h

C

```
typedef struct _bt_l2cap_option_unknown_t {
  bt_byte data_len;
  bt_byte data[L2CAP_OPTION_LEN_UNKNOWN_DATA];
} bt_l2cap_option_unknown_t;
```

Description

This is type bt_l2cap_option_unknown_t.

bt_l2cap_packet_t Structure

File

l2cap_packet.h

C

```
typedef struct _bt_l2cap_packet_t {
  bt_packet_t header;
  union {
    struct _bt_l2cap_channel_t* channel;
    bt_hci_hconn_t hconn;
  } destination;
  bt_byte data_type;
} bt_l2cap_packet_t;
```

Members

Members	Description
bt_byte data_type;	bt_uint fcs; bt_int ctl;

Description

This is type bt_l2cap_packet_t.

bt_l2cap_psm_t Structure

File

l2cap.h

C

```
typedef struct _bt_l2cap_psm_s {
  bt_int psm;
  bt_uint acl_config;
  bt_byte chmode;
  bt_l2cap_listen_callback_fp listen_cb;
  bt_l2cap_connect_callback_fp connect_cb;
  void* listen_cb_param;
  void* connect_cb_param;
} bt_l2cap_psm_t;
```

Description

This is type bt_l2cap_psm_t.

bt_l2cap_read_data_callback_fp Type

File

channel.h

C

```
typedef void (* bt_l2cap_read_data_callback_fp)(bt_l2cap_channel_t* pch, bt_byte_p pdata, bt_int len);
```

Description

This is type bt_l2cap_read_data_callback_fp.

bt_l2cap_request_handler_fp Type

File

l2cap_private.h

C

```
typedef void (* bt_l2cap_request_handler_fp)(bt_l2cap_mgr_p pmgr, bt_hci_conn_state_p pconn,
bt_l2cap_cmd_header_p pcmd);
```

Description

This is type bt_l2cap_request_handler_fp.

bt_l2cap_response_handler_fp Type

File

l2cap_private.h

C

```
typedef void (* bt_l2cap_response_handler_fp)(bt_l2cap_mgr_p pmgr, bt_l2cap_cmd_header_p pcmd);
```

Description

This is type bt_l2cap_response_handler_fp.

bt_l2cap_send_data_callback_fp Type

File

channel.h

C

```
typedef void (* bt_l2cap_send_data_callback_fp)(bt_l2cap_channel_t* pch, bt_byte_p pdata, bt_int len, void*
param);
```

Description

This is type bt_l2cap_send_data_callback_fp.

bt_l2cap_state_changed_callback_fp Type

File

channel.h

C

```
typedef void (* bt_l2cap_state_changed_callback_fp)(bt_l2cap_channel_t* pch, bt_int new_state, void* param);
```

Description

This is type bt_l2cap_state_changed_callback_fp.

bt_l2cap_xmit_event_param_t Structure

File

l2cap_eretr.h

C

```
typedef struct _bt_l2cap_xmit_event_param_t {
  bt_l2cap_eretr_xmit_event_e ev;
  bt_l2cap_mgr_p pmgr;
  bt_l2cap_channel_t* pch;
  union {
    struct {
      bt_byte req_seq;
      bt_byte fbit;
    } recv_reqseq_and_fbit;
    struct {
      bt_byte fbit;
    } recv_bfit;
    struct {
      bt_l2cap_frame_desc_t* frame;
```

```
      } data_request;
    } params;
  } bt_l2cap_xmit_event_param_t;
```

Description

This is type bt_l2cap_xmit_event_param_t.

cmd_disconnection_res Structure

File

command.h

C

```
typedef struct _cmd_disconnection_res {
  bt_l2cap_cmd_header_t header;
  bt_id cid_destination;
  bt_id cid_source;
} cmd_disconnection_res, * pcmd_disconnection_res;
```

Description

This is type cmd_disconnection_res.

pcmd_disconnection_res Structure

File

command.h

C

```
typedef struct _cmd_disconnection_res {
  bt_l2cap_cmd_header_t header;
  bt_id cid_destination;
  bt_id cid_source;
} cmd_disconnection_res, * pcmd_disconnection_res;
```

Description

This is type pcmd_disconnection_res.

pf_l2cap_cmd_callback Type

File

command.h

C

```
typedef void (* pf_l2cap_cmd_callback)(struct _bt_l2cap_mgr_s *ppmg, struct _bt_l2cap_cmd_header_t *pcmd);
```

Description

This is type pf_l2cap_cmd_callback.

L2CAP_CHANNEL_FLAG_FORCE_HCI_DISCONNECT Macro

File

channel.h

C

```
#define L2CAP_CHANNEL_FLAG_FORCE_HCI_DISCONNECT 0x04
```

Description

This is macro L2CAP_CHANNEL_FLAG_FORCE_HCI_DISCONNECT.

L2CAP_EXT_EXT_FLOW_SPEC_BR_EDR Macro

File

command.h

C

```
#define L2CAP_EXT_EXT_FLOW_SPEC_BR_EDR 0x00000040
```

Description

This is macro L2CAP_EXT_EXT_FLOW_SPEC_BR_EDR.

L2CAP_EXT_EXT_WINDOW_SIZE Macro

File

command.h

C

```
#define L2CAP_EXT_EXT_WINDOW_SIZE 0x00000100
```

Description

This is macro L2CAP_EXT_EXT_WINDOW_SIZE.

L2CAP_EXT_FEATURES_ENABLED Macro

File

l2cap_private.h

C

```
#define L2CAP_EXT_FEATURES_ENABLED (_l2cap_eretr_recv_fp != NULL)
```

Description

This is macro L2CAP_EXT_FEATURES_ENABLED.

L2CAP_EXT_FIXED_CHANNELS Macro

File

command.h

C

```
#define L2CAP_EXT_FIXED_CHANNELS 0x00000080
```

Description

This is macro L2CAP_EXT_FIXED_CHANNELS.

L2CAP_EXT_UNICAST_CONNLESS_DATA_RCPT Macro

File

command.h

C

```
#define L2CAP_EXT_UNICAST_CONNLESS_DATA_RCPT 0x00000200
```

Description

This is macro L2CAP_EXT_UNICAST_CONNLESS_DATA_RCPT.

L2CAP_INFO_TYPE_FIXED_CHANNELS Macro

File

command.h

C

```
#define L2CAP_INFO_TYPE_FIXED_CHANNELS 0x0003
```

Description

This is macro L2CAP_INFO_TYPE_FIXED_CHANNELS.

bt_l2cap_channel_ext_t Type

File

channel.h

C

```
typedef struct _bt_l2cap_channel_ext_t bt_l2cap_channel_ext_t;
```

Description

This is type bt_l2cap_channel_ext_t.

_bt_l2cap_channel_ext_t Structure

File

channel.h

C

```
struct _bt_l2cap_channel_ext_t {
  bt_bool check_fcs;
  bt_byte max_transmit;
  bt_byte next_tx_seq;
  bt_byte expected_ack_seq;
  bt_byte expected_tx_seq;
  bt_bool remote_busy;
  bt_bool local_busy;
  bt_l2cap_frame_desc_t unacked_frame;
  bt_l2cap_frame_desc_t pending_frame;
  bt_l2cap_frame_desc_t s_frame;
  bt_byte retry_count;
  bt_bool rnr_sent;
  bt_bool rej_actioned;
  bt_byte frames_sent;
  bt_ulong retr_timer_start_time;
  bt_ulong monitor_timer_start_time;
  bt_byte recv_state;
  bt_byte xmit_state;
  bt_l2cap_packet_t tx_s_packet;
};
```

Members

Members	Description
bt_byte expected_tx_seq;	bt_byte req_seq;
bt_bool remote_busy;	bt_byte buffer_seq;
bt_byte retry_count;	bt_byte srej_list[RFC_ERETR_TXWINDOW];
bt_byte frames_sent;	bt_bool srej_actioned; bt_byte srej_save_req_seq; bt_bool send_srej; bt_byte buffer_seq_srej;

Description

This is record _bt_l2cap_channel_ext_t.

_bt_l2cap_connect_params_s Structure

File

l2cap.h

C

```
struct _bt_l2cap_connect_params_s {
  bt_l2cap_connect_params_t* next;
  bt_bdaddr_t addr;
  bt_int psm;
  bt_l2cap_connect_callback_fp callback;
  void* param;
  bt_l2cap_state_changed_callback_fp state_cb;
  bt_byte chmode;
  bt_bool fixed_channel;
  bt_uint acl_config;
};
```

Description

This is record _bt_l2cap_connect_params_s.

L2CAP_CONN_REQ_RESULT_INVALID_SOURCE_CID Macro

File

command.h

C

```
#define L2CAP_CONN_REQ_RESULT_INVALID_SOURCE_CID 0x0006
```

Description

This is macro L2CAP_CONN_REQ_RESULT_INVALID_SOURCE_CID.

L2CAP_CONN_REQ_RESULT_SRC_CID_ALREADY_ALLOCATED Macro

File

command.h

C

```
#define L2CAP_CONN_REQ_RESULT_SRC_CID_ALREADY_ALLOCATED 0x0007
```

Description

This is macro L2CAP_CONN_REQ_RESULT_SRC_CID_ALREADY_ALLOCATED.

bt_l2cap_connect_params_t Type

File

l2cap.h

C

```
typedef struct _bt_l2cap_connect_params_s bt_l2cap_connect_params_t;
```

Description

This is type bt_l2cap_connect_params_t.

Platform Data Types and Constants

BYTE_SIZE Macro

File

types.h

C

```
#define BYTE_SIZE 1
```

Description

This is macro BYTE_SIZE.

bt_byte Type

File

types.h

C

```
typedef unsigned char bt_byte;
```

Description

This is type bt_byte.

bt_int Type

File

types.h

C

```
typedef short bt_int;
```

Description

This is type bt_int.

bt_long Type

File

types.h

C

```
typedef long bt_long;
```

Description

This is type bt_long.

bt_uint Type

File

types.h

C

```
typedef unsigned short bt_uint;
```

Description

This is type bt_uint.

bt_ulong Type

File

types.h

C

```
typedef unsigned long bt_ulong;
```

Description

This is type bt_ulong.

RFCOMM Functions

bt_rfcomm_allocate_dlc Function

File

rfcomm.h

C

```
bt_rfcomm_dlc_t* bt_rfcomm_allocate_dlc(bt_rfcomm_session_t* session, bt_byte dlci);
```

Description

brief Allocate DLC. ingroup rfcomm

details This function allocates a new DLC on the specified RFCOMM session.

param session The RFCOMM session. param dlci DLCI of the new DLC.

return li A pointer to the new DLC if the function succeeds. li c NULL otherwise.

bt_rfcomm_allocate_session Function

File

rfcomm.h

C

```
bt_rfcomm_session_t* bt_rfcomm_allocate_session(bt_l2cap_mgr_t* l2cap_mgr);
```

Description

brief Allocate RFCOMM session. ingroup rfcomm

details This function allocates a new RFCOMM session.

param l2cap_mgr The L2CAP manager on which the RFCOMM session is to be created. param callback The callback function that is called when session state changes. param param An arbitrary data pointer that will be passed to the callback function specified by the c callback parameter.

return li A pointer to the new RFCOMM session structure if the function succeeds. li c NULL otherwise.

bt_rfcomm_close_dlc Function

File

rfcomm.h

C

```
void bt_rfcomm_close_dlc(bt_rfcomm_dlc_t* dlc);
```

Description

brief Close DLC. ingroup rfcomm

details This function closes a DLC. If DLCI = 0, the parent RFCOMM session is also closed.

bt_rfcomm_connect Function

File

rfcomm.h

C

```
bt_bool bt_rfcomm_connect(bt_bdaddr_p remote_addr, bt_byte server_channel, bt_uint acl_config,
bt_rfcomm_dlc_state_callback_fp callback, void* param);
```

Description

brief Connect to a remote device. ingroup rfcomm

details This function establishes an RFCOMM connection with a remote device and opens a data DLC. Changes in the session state are reported through a callback function specified when the session has been allocated via call to bt_rfcomm_allocate_session. Changes in the data DLC are reported through a callback function specified in this call.

param remote_addr Address of the remote device. param server_channel A server channel of the remote RFCOMM server. param callback The callback function for reporting changes in DLC state opened by this call. param param An arbitrary data pointer that will be passed to the callback function specified by the c callback parameter.

return li c TRUE if the function succeeds. li c FALSE otherwise. The callback function is not called in this case.

bt_rfcomm_find_dlc Function

File

rfcomm.h

C

```
bt_rfcomm_dlc_t* bt_rfcomm_find_dlc(bt_rfcomm_session_t* session, bt_byte address);
```

Description

brief Find DLC

bt_rfcomm_free_dlc Function

File

rfcomm.h

C

```
void bt_rfcomm_free_dlc(bt_rfcomm_dlc_t* dlc);
```

Description

brief Release DLC. ingroup rfcomm

details This function releases the specified DLC.

param dlc The DLC to be released.

bt_rfcomm_free_session Function

File

rfcomm.h

C

```
void bt_rfcomm_free_session(bt_rfcomm_session_t* session);
```

Description

brief Release RFCOMM session. ingroup rfcomm

details This function deallocates the specified RFCOMM session. This function does not disconnect the session. It just frees the memory used by the bt_rfcomm_session structure. The session has to be disconnected by calling bt_rfcomm_close_dlc with DLCI = 0 first.

param session The RFCOMM session to be deallocated.

bt_rfcomm_init Function

File

rfcomm.h

C

```
bt_bool bt_rfcomm_init();
```

Description

brief Initialize the RFCOMM layer. ingroup rfcomm

details This function initializes the RFCOMM layer of the stack. It must be called prior to any other RFCOMM function can be called.

bt_rfcomm_listen Function

File

rfcomm.h

C

```
bt_bool bt_rfcomm_listen(bt_byte server_channel, bt_uint acl_config, bt_rfcomm_dlc_state_callback_fp
callback, void* param);
```

Description

brief Listen for incoming connections. ingroup rfcomm

details This function enables incoming connections on the specified RFCOMM session. Changes in the session state are reported through a callback function.

param server_channel A server channel on which the RFCOMM session will listen and accept incoming connections. param callback The callback function that is called when session state changes. param param An arbitrary data pointer that will be passed to the callback function specified by the c callback parameter.

return li c TRUE if the function succeeds. li c FALSE otherwise. The callback function is not called in this case.

bt_rfcomm_open_dlc Function

File

rfcomm.h

C

```
bt_bool bt_rfcomm_open_dlc(bt_rfcomm_dlc_t* dlc, bt_rfcomm_dlc_state_callback_fp callback, void* param);
```

Description

brief Open DLC. ingroup rfcomm

details This function opens the specified DLC. Before calling this function the RFCOMM session must be already open. This function is not to be used with DLCI = 0. Changes in DLC state are reported through a callback function.

param dlc The DLC to open. param callback The callback function for reporting changes in DLC state. param param An arbitrary data pointer that will be passed to the callback function specified by the c callback parameter.

return li c TRUE if the function succeeds. li c FALSE otherwise. The callback function is not called in this case.

bt_rfcomm_register_listener Function

File

rfcomm_mgr.h

C

```
bt_bool bt_rfcomm_register_listener(bt_rfcomm_session_listener_t* listener);
```

Description

This is function bt_rfcomm_register_listener.

bt_rfcomm_send_credit Function

File

rfcomm.h

C

```
bt_bool bt_rfcomm_send_credit(bt_rfcomm_dlc_t* dlc, bt_byte credit);
```

Description

This is function bt_rfcomm_send_credit.

bt_rfcomm_send_data Function

File

rfcomm.h

C

```
bt_bool bt_rfcomm_send_data(bt_rfcomm_dlc_t* dlc, void * data, bt_int len, bt_rfcomm_send_data_callback_fp
callback);
```

Description

brief Send data over a DLC. ingroup rfcomm

details This function sends data over the specified DLC. Operation completion is reported through callback function.

param dlc The DLC. param data A pointer to the data to be sent. param len Data length. param callback The callback function that is called when operation completes.

return li c TRUE if the function succeeds. li c FALSE otherwise. The callback function is not called in this case.

bt_rfcomm_unregister_listener Function

File

rfcomm_mgr.h

C

```
void bt_rfcomm_unregister_listener(bt_rfcomm_session_listener_t* listener);
```

Description

This is function bt_rfcomm_unregister_listener.

check_fcs Function

File

rfcomm_private.h

C

```
bt_bool check_fcs(bt_byte_p buffer, bt_byte len, bt_byte recv_fcs);
```

Description

From rfcomm_fcs.c

read_command Function

File

l2cap_private.h

C

```
bt_l2cap_cmd_header_p read_command(bt_byte_p pdata, bt_int len, bt_int* offset, bt_l2cap_cmd_header_t*
pheader);
```

Description

From channel_cmd_recv.c

rfcomm_cq_ack_cmd Function

File

rfcomm_cmd_queue.h

C

```
struct _bt_rfcomm_command_t* rfcomm_cq_ack_cmd(struct _bt_rfcomm_dlc_t* pdlc, struct _bt_rfcomm_command_t*
pres);
```

Description

This is function rfcomm_cq_ack_cmd.

rfcomm_cq_ack_mx_cmd Function

File

rfcomm_cmd_queue.h

C

```
void rfcomm_cq_ack_mx_cmd(struct _bt_rfcomm_dlc_t* pdlc, struct _bt_rfcomm_command_t* pres);
```

Description

This is function rfcomm_cq_ack_mx_cmd.

rfcomm_send_cmd Function

File

rfcomm_private.h

C

```
bt_bool rfcomm_send_cmd(bt_rfcomm_dlc_p pdlc, bt_rfcomm_command_p pcmd);
```

Description

This is function rfcomm_send_cmd.

rfcomm_send_commands_from_queue Function

File

rfcomm_cmd_queue.h

C

```
void rfcomm_send_commands_from_queue(struct _bt_rfcomm_session_t * psess);
```

Description

This is function rfcomm_send_commands_from_queue.

rfcomm_send_mx_msc_cmd Function

File

rfcomm_mx.h

C

```
bt_bool rfcomm_send_mx_msc_cmd(bt_rfcomm_dlc_p pdlc, bt_rfcomm_cmd_callback_fp cb);
```

Description

This is function rfcomm_send_mx_msc_cmd.

rfcomm_send_mx_pn_cmd Function

File

rfcomm_mx.h

C

```
void rfcomm_send_mx_pn_cmd(bt_rfcomm_dlc_p pdlc, bt_rfcomm_cmd_callback_fp cb);
```

Description

This is function rfcomm_send_mx_pn_cmd.

_rfcomm_alloc_cmd_buffer Function

File

rfcomm_private.h

C

```
bt_rfcomm_command_p _rfcomm_alloc_cmd_buffer();
```

Description

This is function _rfcomm_alloc_cmd_buffer.

_rfcomm_allocate_buffers Function

File

rfcomm_private.h

C

```
void _rfcomm_allocate_buffers();
```

Description

Defined by OEM through library configuration

_rfcomm_allocate_mx_cmd Function

File

rfcomm_mx.h

C

```
bt_rfcomm_command_p _rfcomm_allocate_mx_cmd(bt_rfcomm_session_p psess, bt_rfcomm_cmd_callback_fp cb);
```

Description

This is function _rfcomm_allocate_mx_cmd.

_rfcomm_free_cmd_buffer Function

File

rfcomm_private.h

C

```
void _rfcomm_free_cmd_buffer(void* p);
```

Description

This is function _rfcomm_free_cmd_buffer.

_rfcomm_get_tick_count Function

File

rfcomm_timer.h

C

```
bt_long _rfcomm_get_tick_count();
```

Description

This is function _rfcomm_get_tick_count.

_rfcomm_init_cmd_buffers Function

File

rfcomm_private.h

C

```
bt_bool _rfcomm_init_cmd_buffers();
```

Description

From rfcomm_cmdbuffer.c

_rfcomm_init_mgr Function

File

rfcomm_mgr.h

C

```
void _rfcomm_init_mgr();
```

Description

This is function _rfcomm_init_mgr.

_rfcomm_init_sessions Function

File

rfcomm_private.h

C

```
void _rfcomm_init_sessions();
```

Description

From rfcomm_session.c

_rfcomm_init_timer Function

File

rfcomm_timer.h

C

```
void _rfcomm_init_timer();
```

Description

This is function _rfcomm_init_timer.

_rfcomm_l2cap_read_data_callback Function

File

rfcomm_private.h

C

```
void _rfcomm_l2cap_read_data_callback(struct _bt_l2cap_channel_t * pch, bt_byte_p pdata, bt_int len);
```

Description

From rfcomm_cmd_recv.c

_rfcomm_mx_process_fc Function

File

rfcomm_mx.h

C

```
void _rfcomm_mx_process_fc(bt_rfcomm_dlc_p pdlc, bt_rfcomm_command_p pcmd, bt_byte mx_msg_type);
```

Description

This is function _rfcomm_mx_process_fc.

_rfcomm_mx_process_pn Function

File

rfcomm_mx.h

C

```
void _rfcomm_mx_process_pn(bt_rfcomm_dlc_p pdlc, bt_rfcomm_command_p pcmd, bt_int data_start, bt_int len);
```

Description

This is function _rfcomm_mx_process_pn.

_rfcomm_mx_process_rpn Function

File

rfcomm_mx.h

C

```
void _rfcomm_mx_process_rpn(bt_rfcomm_dlc_p pdlc, bt_rfcomm_command_p pcmd, bt_int data_start, bt_int len);
```

Description

This is function _rfcomm_mx_process_rpn.

_rfcomm_process_cmd_frame_disc Function

File

rfcomm_private.h

C

```
void _rfcomm_process_cmd_frame_disc(bt_rfcomm_session_p psess, bt_rfcomm_command_p pcmd);
```

Description

From frame_disc.c

_rfcomm_process_cmd_frame_dm Function

File

rfcomm_private.h

C

```
void _rfcomm_process_cmd_frame_dm(bt_rfcomm_session_p psess, bt_rfcomm_command_p pcmd);
```

Description

This is function _rfcomm_process_cmd_frame_dm.

_rfcomm_process_cmd_frame_sabm Function

File

rfcomm_private.h

C

```
void _rfcomm_process_cmd_frame_sabm(bt_rfcomm_session_p psess, bt_rfcomm_command_p pcmd);
```

Description

This is function _rfcomm_process_cmd_frame_sabm.

_rfcomm_process_cmd_frame_ua Function

File

rfcomm_private.h

C

```
void _rfcomm_process_cmd_frame_ua(bt_rfcomm_session_p psess, bt_rfcomm_command_p pcmd);
```

Description

This is function _rfcomm_process_cmd_frame_ua.

_rfcomm_process_cmd_frame_uih Function

File

rfcomm_private.h

C

```
void _rfcomm_process_cmd_frame_uih(bt_rfcomm_session_p psess, bt_rfcomm_command_p pcmd);
```

Description

From frame_uih.c

_rfcomm_process_mx_fc_response Function

File

rfcomm_mx.h

C

```
void _rfcomm_process_mx_fc_response(bt_rfcomm_dlc_p pdlc, bt_rfcomm_command_p pres);
```

Description

This is function _rfcomm_process_mx_fc_response.

_rfcomm_process_mx_msc_response Function

File

rfcomm_mx.h

C

```
void _rfcomm_process_mx_msc_response(bt_rfcomm_dlc_p pdlc, bt_rfcomm_command_p pcmd, bt_int data_start,
bt_int len);
```

Description

This is function _rfcomm_process_mx_msc_response.

_rfcomm_process_mx_pn_response Function

File

rfcomm_mx.h

C

```
void _rfcomm_process_mx_pn_response(bt_rfcomm_dlc_p pdlc, bt_rfcomm_command_p pres, bt_int data_start,
bt_int len);
```

Description

This is function _rfcomm_process_mx_pn_response.

_rfcomm_process_mx_rls_response Function

File

rfcomm_mx.h

C

```
void _rfcomm_process_mx_rls_response(bt_rfcomm_dlc_p pdlc, bt_rfcomm_command_p pcmd, bt_int data_start,
bt_int len);
```

Description

This is function _rfcomm_process_mx_rls_response.

_rfcomm_process_res_frame_disc Function

File

rfcomm_private.h

C

```
void _rfcomm_process_res_frame_disc(bt_rfcomm_session_p psess, bt_rfcomm_command_p pres);
```

Description

This is function _rfcomm_process_res_frame_disc.

_rfcomm_process_res_frame_dm Function

File

rfcomm_private.h

C

```
void _rfcomm_process_res_frame_dm(bt_rfcomm_session_p psess, bt_rfcomm_command_p pres);
```

Description

This is function _rfcomm_process_res_frame_dm.

_rfcomm_process_res_frame_sabm Function

File

rfcomm_private.h

C

```
void _rfcomm_process_res_frame_sabm(bt_rfcomm_session_p psess, bt_rfcomm_command_p pres);
```

Description

This is function _rfcomm_process_res_frame_sabm.

_rfcomm_process_res_frame_ua Function

File

rfcomm_private.h

C

```
void _rfcomm_process_res_frame_ua(bt_rfcomm_session_p psess, bt_rfcomm_command_p pres);
```

Description

This is function _rfcomm_process_res_frame_ua.

_rfcomm_process_res_frame_uih Function

File

rfcomm_private.h

C

```
void _rfcomm_process_res_frame_uih(bt_rfcomm_session_p psess, bt_rfcomm_command_p pres);
```

Description

This is function _rfcomm_process_res_frame_uih.

_rfcomm_send_command Function

File

rfcomm_private.h

C

```
bt_bool _rfcomm_send_command(bt_rfcomm_dlc_p pdlc, bt_rfcomm_command_p pcmd);
```

Description

From rfcomm_cmd_send.c

_rfcomm_send_dm_response Function

File

rfcomm_private.h

C

```
void _rfcomm_send_dm_response(bt_rfcomm_dlc_p pdlc, bt_rfcomm_command_p pcmd);
```

Description

From frame_dm.c

_rfcomm_send_mx_fc_cmd Function

File

rfcomm_mx.h

C

```
void _rfcomm_send_mx_fc_cmd(bt_rfcomm_dlc_p pdlc, bt_byte mx_msg_type);
```

Description

This is function _rfcomm_send_mx_fc_cmd.

_rfcomm_send_mx_msc_response Function

File

rfcomm_mx.h

C

```
void _rfcomm_send_mx_msc_response(bt_rfcomm_dlc_p pdlc, bt_rfcomm_command_p pcmd, bt_int data_start, bt_int
len);
```

Description

This is function _rfcomm_send_mx_msc_response.

_rfcomm_send_mx_nsc_response Function

File

rfcomm_mx.h

C

```
void _rfcomm_send_mx_nsc_response(bt_rfcomm_dlc_p pdlc, bt_rfcomm_command_p pcmd, bt_byte mx_msg_type);
```

Description

This is function _rfcomm_send_mx_nsc_response.

_rfcomm_send_mx_rls_cmd Function

File

rfcomm_mx.h

C

```
void _rfcomm_send_mx_rls_cmd(bt_rfcomm_dlc_p pdlc);
```

Description

This is function _rfcomm_send_mx_rls_cmd.

_rfcomm_send_mx_rls_response Function

File

rfcomm_mx.h

C

```
void _rfcomm_send_mx_rls_response(bt_rfcomm_dlc_p pdlc, bt_rfcomm_command_p pcmd, bt_int data_start, bt_int
len);
```

Description

This is function _rfcomm_send_mx_rls_response.

_rfcomm_send_mx_test_response Function

File

rfcomm_mx.h

C

```
void _rfcomm_send_mx_test_response(bt_rfcomm_dlc_p pdlc, bt_rfcomm_command_p pcmd, bt_int data_start,
bt_int len);
```

Description

This is function _rfcomm_send_mx_test_response.

_rfcomm_send_sabm_cmd Function

File

rfcomm_private.h

C

```
void _rfcomm_send_sabm_cmd(bt_rfcomm_dlc_p pdlc, bt_rfcomm_cmd_callback_fp cb);
```

Description

From frame_sabm.c

_rfcomm_send_ua_response Function

File

rfcomm_private.h

C

```
void _rfcomm_send_ua_response(bt_rfcomm_dlc_p pdlc, bt_rfcomm_command_p pcmd, bt_rfcomm_cmd_callback_fp cb);
```

Description

From frame_ua.c

_bt_rfcomm_allocate_channel Function

File

rfcomm_mgr.h

C

```
bt_rfcomm_server_channel_t* _bt_rfcomm_allocate_channel(bt_byte id);
```

Description

This is function _bt_rfcomm_allocate_channel.

_bt_rfcomm_clear_queue Function

File

rfcomm_private.h

C

```
void _bt_rfcomm_clear_queue(bt_rfcomm_dlc_t* pdlc);
```

Description

From command_queue.c

_bt_rfcomm_find_channel Function

File

rfcomm_mgr.h

C

```
bt_rfcomm_server_channel_t* _bt_rfcomm_find_channel(bt_byte id);
```

Description

This is function _bt_rfcomm_find_channel.

_bt_rfcomm_get_mgr Function

File

rfcomm_mgr.h

C

```
bt_rfcomm_mgr_t* _bt_rfcomm_get_mgr();
```

Description

This is function _bt_rfcomm_get_mgr.

_bt_rfcomm_init_signal Function

File

rfcomm_signal.h

C

```
void _bt_rfcomm_init_signal();
```

Description

This is function _bt_rfcomm_init_signal.

_bt_rfcomm_mgr_allocate_session Function

File

rfcomm_private.h

C

```
bt_rfcomm_session_t* _bt_rfcomm_mgr_allocate_session(bt_bdaddr_t* bdaddr_remote);
```

Description

This is function _bt_rfcomm_mgr_allocate_session.

_bt_rfcomm_mgr_l2cap_listen_callback Function

File

rfcomm_mgr.h

C

```
void _bt_rfcomm_mgr_l2cap_listen_callback(bt_l2cap_channel_t* pch, void* param);
```

Description

This is function _bt_rfcomm_mgr_l2cap_listen_callback.

_bt_rfcomm_mgr_notify_listeners Function

File

rfcomm_mgr.h

C

```
void _bt_rfcomm_mgr_notify_listeners(bt_rfcomm_session_t* session, bt_int evt);
```

Description

This is function _bt_rfcomm_mgr_notify_listeners.

_bt_rfcomm_set_signal Function

File

rfcomm_signal.h

C

```
void _bt_rfcomm_set_signal();
```

Description

This is function _bt_rfcomm_set_signal.

_calc_fcs Function

File

rfcomm_private.h

C

```
bt_byte _calc_fcs(bt_byte_p buffer, bt_byte len);
```

Description

This is function _calc_fcs.

rfcomm_find_session Function

File

rfcomm_private.h

C

```
bt_rfcomm_session_p rfcomm_find_session(bt_l2cap_channel_t* pch);
```

Description

This is function rfcomm_find_session.

_rfcomm_l2cap_state_changed_callback Function

File

rfcomm_private.h

C

```
void _rfcomm_l2cap_state_changed_callback(bt_l2cap_channel_t* pch, bt_int new_state, void* param);
```

Description

From rfcomm.c

bt_rfcomm_cancel_listen Function

File

rfcomm.h

C

```
bt_bool bt_rfcomm_cancel_listen(bt_byte server_channel);
```

Description

This is function bt_rfcomm_cancel_listen.

bt_rfcomm_get_frame_length Function

File

rfcomm.h

C

```
bt_int bt_rfcomm_get_frame_length(bt_rfcomm_dlc_t* dlc);
```

Description

This is function bt_rfcomm_get_frame_length.

rfcomm_cq_find_failed_pn Function

File

rfcomm_cmd_queue.h

C

```
struct _bt_rfcomm_command_t* rfcomm_cq_find_failed_pn(struct _bt_rfcomm_dlc_t* pdlc, struct
_bt_rfcomm_command_t* dm);
```

Description

This is function rfcomm_cq_find_failed_pn.

RFCOMM Data Types and Constants

MK_CMD_ADDRESS Macro

File

rfcomm.h

C

```
#define MK_CMD_ADDRESS(addr, pdlc) ((addr & 0xfc) | pdlc->psess->role | RFCOMM_FLAG_EA)
```

Description

define RFCOMM_MS_RTC

MK_DLCI Macro

File

rfcomm.h

C

```
#define MK_DLCI(bt_server_channel, rfcomm_session) (bt_server_channel << 3 | ((~rfcomm_session->role & 2)
<< 1))
```

Description

This is macro MK_DLCI.

RFCOMM_CFC_ENABLED Macro

File

rfcomm.h

C

```
#define RFCOMM_CFC_ENABLED (pdlc->psess->fc_type == RFCOMM_FC_TYPE_CREDIT)
```

Description

This is macro RFCOMM_CFC_ENABLED.

RFCOMM_CFC_LOCAL_CREDIT Macro

File

rfcomm.h

C

```
#define RFCOMM_CFC_LOCAL_CREDIT _rfcomm_local_credit
```

Description

This is macro RFCOMM_CFC_LOCAL_CREDIT.

RFCOMM_CFC_MAX_INITIAL_CREDIT Macro

File

rfcomm.h

C

```
#define RFCOMM_CFC_MAX_INITIAL_CREDIT 7
```

Description

This is macro RFCOMM_CFC_MAX_INITIAL_CREDIT.

RFCOMM_CMD_STATUS_FC_PENDING Macro

File

rfcomm.h

C

```
#define RFCOMM_CMD_STATUS_FC_PENDING 2
```

Description

This is macro RFCOMM_CMD_STATUS_FC_PENDING.

RFCOMM_CMD_STATUS_PENDING Macro

File

rfcomm.h

C

```
#define RFCOMM_CMD_STATUS_PENDING 0
```

Description

This is macro RFCOMM_CMD_STATUS_PENDING.

RFCOMM_CMD_STATUS_WAITING_RESPONSE Macro

File

rfcomm.h

C

```
#define RFCOMM_CMD_STATUS_WAITING_RESPONSE 1
```

Description

This is macro RFCOMM_CMD_STATUS_WAITING_RESPONSE.

RFCOMM_COMMAND Macro

File

rfcomm.h

C

```
#define RFCOMM_COMMAND 0x02
```

Description

This is macro RFCOMM_COMMAND.

RFCOMM_CTL_MSG_CLD Macro

File

rfcomm.h

C

```
#define RFCOMM_CTL_MSG_CLD 0xc0
```

Description

This is macro RFCOMM_CTL_MSG_CLD.

RFCOMM_CTL_MSG_FCOFF Macro

File

rfcomm.h

C

```
#define RFCOMM_CTL_MSG_FCOFF 0x60
```

Description

This is macro RFCOMM_CTL_MSG_FCOFF.

RFCOMM_CTL_MSG_FCON Macro

File

rfcomm.h

C

```
#define RFCOMM_CTL_MSG_FCON 0x10
```

Description

This is macro RFCOMM_CTL_MSG_FCON.

RFCOMM_CTL_MSG_MSC Macro

File

rfcomm.h

C

```
#define RFCOMM_CTL_MSG_MSC 0xe0
```

Description

This is macro RFCOMM_CTL_MSG_MSC.

RFCOMM_CTL_MSG_NSC Macro

File

rfcomm.h

C

```
#define RFCOMM_CTL_MSG_NSC 0x10
```

Description

This is macro RFCOMM_CTL_MSG_NSC.

RFCOMM_CTL_MSG_PN Macro

File

rfcomm.h

C

```
#define RFCOMM_CTL_MSG_PN 0x80
```

Description

This is macro RFCOMM_CTL_MSG_PN.

RFCOMM_CTL_MSG_PSC Macro

File

rfcomm.h

C

```
#define RFCOMM_CTL_MSG_PSC 0x40
```

Description

This is macro RFCOMM_CTL_MSG_PSC.

RFCOMM_CTL_MSG_RLS Macro

File

rfcomm.h

C

```
#define RFCOMM_CTL_MSG_RLS 0x50
```

Description

This is macro RFCOMM_CTL_MSG_RLS.

RFCOMM_CTL_MSG_RPN Macro

File

rfcomm.h

C

```
#define RFCOMM_CTL_MSG_RPN 0x90
```

Description

This is macro RFCOMM_CTL_MSG_RPN.

RFCOMM_CTL_MSG_SNC Macro

File

rfcomm.h

C

```
#define RFCOMM_CTL_MSG_SNC 0xd0
```

Description

This is macro RFCOMM_CTL_MSG_SNC.

RFCOMM_CTL_MSG_TEST Macro

File

rfcomm.h

C

```
#define RFCOMM_CTL_MSG_TEST 0x20
```

Description

This is macro RFCOMM_CTL_MSG_TEST.

RFCOMM_DLC_CHANGED_CONN_STATE Macro

File

rfcomm.h

C

```
#define RFCOMM_DLC_CHANGED_CONN_STATE 0
```

Description

This is macro RFCOMM_DLC_CHANGED_CONN_STATE.

RFCOMM_DLC_CHANGED_REMOTE_MSC Macro

File

rfcomm.h

C

```
#define RFCOMM_DLC_CHANGED_REMOTE_MSC 1
```

Description

This is macro RFCOMM_DLC_CHANGED_REMOTE_MSC.

RFCOMM_DLC_CONNECTION_FAILED Macro

File

rfcomm.h

C

```
#define RFCOMM_DLC_CONNECTION_FAILED 2
```

Description

This is macro RFCOMM_DLC_CONNECTION_FAILED.

RFCOMM_DLC_STATE_CLOSED Macro

File

rfcomm.h

C

```
#define RFCOMM_DLC_STATE_CLOSED 0
```

Description

This is macro RFCOMM_DLC_STATE_CLOSED.

RFCOMM_DLC_STATE_OPEN Macro

File

rfcomm.h

C

```
#define RFCOMM_DLC_STATE_OPEN 1
```

Description

This is macro RFCOMM_DLC_STATE_OPEN.

RFCOMM_DLCI_CONTROL Macro

File

rfcomm.h

C

```
#define RFCOMM_DLCI_CONTROL 0
```

Description

This is macro RFCOMM_DLCI_CONTROL.

RFCOMM_DLCI_FREE Macro

File

rfcomm.h

C

```
#define RFCOMM_DLCI_FREE 0xff
```

Description

This is macro RFCOMM_DLCI_FREE.

RFCOMM_ERR_DM Macro

File

rfcomm.h

C

```
#define RFCOMM_ERR_DM 0x02
```

Description

This is macro RFCOMM_ERR_DM.

RFCOMM_ERR_SUCCESS Macro

File

rfcomm.h

C

```
#define RFCOMM_ERR_SUCCESS 0x00
```

Description

This is macro RFCOMM_ERR_SUCCESS.

RFCOMM_ERR_TIMEOUT Macro

File

rfcomm.h

C

```
#define RFCOMM_ERR_TIMEOUT 0x01
```

Description

This is macro RFCOMM_ERR_TIMEOUT.

RFCOMM_FC_TYPE_AGREGATE Macro

File

rfcomm.h

C

```
#define RFCOMM_FC_TYPE_AGREGATE 0
```

Description

This is macro RFCOMM_FC_TYPE_AGREGATE.

RFCOMM_FC_TYPE_CREDIT Macro

File

rfcomm.h

C

```
#define RFCOMM_FC_TYPE_CREDIT 1
```

Description

This is macro RFCOMM_FC_TYPE_CREDIT.

RFCOMM_FLAG_CR Macro

File

rfcomm.h

C

```
#define RFCOMM_FLAG_CR 0x02
```

Description

This is macro RFCOMM_FLAG_CR.

RFCOMM_FLAG_EA Macro

File

rfcomm.h

C

```
#define RFCOMM_FLAG_EA 0x01
```

Description

This is macro RFCOMM_FLAG_EA.

RFCOMM_FLAG_PF Macro

File

rfcomm.h

C

```
#define RFCOMM_FLAG_PF 0x10
```

Description

This is macro RFCOMM_FLAG_PF.

RFCOMM_FRAME_HEADER_LEN Macro

File

rfcomm.h

C

```
#define RFCOMM_FRAME_HEADER_LEN 5
```

Description

This is macro RFCOMM_FRAME_HEADER_LEN.

RFCOMM_FRAME_TYPE_DISC Macro

File

rfcomm.h

C

```
#define RFCOMM_FRAME_TYPE_DISC 0x43
```

Description

This is macro RFCOMM_FRAME_TYPE_DISC.

RFCOMM_FRAME_TYPE_DM Macro

File

rfcomm.h

C

```
#define RFCOMM_FRAME_TYPE_DM 0x0f
```

Description

This is macro RFCOMM_FRAME_TYPE_DM.

RFCOMM_FRAME_TYPE_SABM Macro

File

rfcomm.h

C

```
#define RFCOMM_FRAME_TYPE_SABM 0x2f
```

Description

This is macro RFCOMM_FRAME_TYPE_SABM.

RFCOMM_FRAME_TYPE_UA Macro

File

rfcomm.h

C

```
#define RFCOMM_FRAME_TYPE_UA 0x63
```

Description

This is macro RFCOMM_FRAME_TYPE_UA.

RFCOMM_FRAME_TYPE_UI Macro

File

rfcomm.h

C

```
#define RFCOMM_FRAME_TYPE_UI 0x03
```

Description

This is macro RFCOMM_FRAME_TYPE_UI.

RFCOMM_FRAME_TYPE_UIH Macro

File

rfcomm.h

C

```
#define RFCOMM_FRAME_TYPE_UIH 0xef
```

Description

This is macro RFCOMM_FRAME_TYPE_UIH.

RFCOMM_LINE_STATUS_FRAMING Macro

File

rfcomm_mx.h

C

```
#define RFCOMM_LINE_STATUS_FRAMING x09
```

Description

This is macro RFCOMM_LINE_STATUS_FRAMING.

RFCOMM_LINE_STATUS_OVERRUN Macro

File

rfcomm_mx.h

C

```
#define RFCOMM_LINE_STATUS_OVERRUN x03
```

Description

This is macro RFCOMM_LINE_STATUS_OVERRUN.

RFCOMM_LINE_STATUS_PARITY Macro

File

rfcomm_mx.h

C

```
#define RFCOMM_LINE_STATUS_PARITY x05
```

Description

This is macro RFCOMM_LINE_STATUS_PARITY.

RFCOMM_MAX_INFO_LEN Macro

File

rfcomm.h

C

```
#define RFCOMM_MAX_INFO_LEN _rfcomm_info_len
```

Description

This is macro RFCOMM_MAX_INFO_LEN.

RFCOMM_MODEM_STATUS_DV Macro

File

rfcomm_mx.h

C

```
#define RFCOMM_MODEM_STATUS_DV 0x80
```

Description

This is macro RFCOMM_MODEM_STATUS_DV.

RFCOMM_MODEM_STATUS_FC Macro

File

rfcomm_mx.h

C

```
#define RFCOMM_MODEM_STATUS_FC 0x02
```

Description

This is macro RFCOMM_MODEM_STATUS_FC.

RFCOMM_MODEM_STATUS_IC Macro

File

rfcomm_mx.h

C

```
#define RFCOMM_MODEM_STATUS_IC 0x40
```

Description

This is macro RFCOMM_MODEM_STATUS_IC.

RFCOMM_MODEM_STATUS_RTC Macro

File

rfcomm_mx.h

C

```
#define RFCOMM_MODEM_STATUS_RTC 0x04
```

Description

This is macro RFCOMM_MODEM_STATUS_RTC.

RFCOMM_MODEM_STATUS_RTR Macro

File

rfcomm_mx.h

C

```
#define RFCOMM_MODEM_STATUS_RTR 0x08
```

Description

This is macro RFCOMM_MODEM_STATUS_RTR.

RFCOMM_MX_MSG_FCOFF Macro

File

rfcomm_mx.h

C

```
#define RFCOMM_MX_MSG_FCOFF 0x60
```

Description

This is macro RFCOMM_MX_MSG_FCOFF.

RFCOMM_MX_MSG_FCON Macro

File

rfcomm_mx.h

C

```
#define RFCOMM_MX_MSG_FCON 0xA0
```

Description

This is macro RFCOMM_MX_MSG_FCON.

RFCOMM_MX_MSG_MSC Macro

File

rfcomm_mx.h

C

```
#define RFCOMM_MX_MSG_MSC 0xE0
```

Description

This is macro RFCOMM_MX_MSG_MSC.

RFCOMM_MX_MSG_NSC Macro

File

rfcomm_mx.h

C

```
#define RFCOMM_MX_MSG_NSC 0x10
```

Description

This is macro RFCOMM_MX_MSG_NSC.

RFCOMM_MX_MSG_PN Macro

File

rfcomm_mx.h

C

```
#define RFCOMM_MX_MSG_PN 0x80
```

Description

This is macro RFCOMM_MX_MSG_PN.

RFCOMM_MX_MSG_RLS Macro

File

rfcomm_mx.h

C

```
#define RFCOMM_MX_MSG_RLS 0x50
```

Description

This is macro RFCOMM_MX_MSG_RLS.

RFCOMM_MX_MSG_RPN Macro

File

rfcomm_mx.h

C

```
#define RFCOMM_MX_MSG_RPN 0x90
```

Description

This is macro RFCOMM_MX_MSG_RPN.

RFCOMM_MX_MSG_TEST Macro

File

rfcomm_mx.h

C

```
#define RFCOMM_MX_MSG_TEST 0x20
```

Description

This is macro RFCOMM_MX_MSG_TEST.

RFCOMM_RESPONSE Macro

File

rfcomm.h

C

```
#define RFCOMM_RESPONSE 0x00
```

Description

This is macro RFCOMM_RESPONSE.

RFCOMM_ROLE_INITIATOR Macro

File

rfcomm.h

C

```
#define RFCOMM_ROLE_INITIATOR 0x02
```

Description

This is macro RFCOMM_ROLE_INITIATOR.

RFCOMM_ROLE_RESPONDER Macro

File

rfcomm.h

C

```
#define RFCOMM_ROLE_RESPONDER 0x00
```

Description

This is macro RFCOMM_ROLE_RESPONDER.

RFCOMM_RPN_BAUD_RATE_1152 Macro

File

rfcomm.h

C

```
#define RFCOMM_RPN_BAUD_RATE_1152 0x07
```

Description

This is macro RFCOMM_RPN_BAUD_RATE_1152.

RFCOMM_RPN_BAUD_RATE_192 Macro

File

rfcomm.h

C

```
#define RFCOMM_RPN_BAUD_RATE_192 0x04
```

Description

This is macro RFCOMM_RPN_BAUD_RATE_192.

RFCOMM_RPN_BAUD_RATE_2304 Macro

File

rfcomm.h

C

```
#define RFCOMM_RPN_BAUD_RATE_2304 0x08
```

Description

This is macro RFCOMM_RPN_BAUD_RATE_2304.

RFCOMM_RPN_BAUD_RATE_24 Macro

File

rfcomm.h

C

```
#define RFCOMM_RPN_BAUD_RATE_24 0x00
```

Description

This is macro RFCOMM_RPN_BAUD_RATE_24.

RFCOMM_RPN_BAUD_RATE_384 Macro

File

rfcomm.h

C

```
#define RFCOMM_RPN_BAUD_RATE_384 0x05
```

Description

This is macro RFCOMM_RPN_BAUD_RATE_384.

RFCOMM_RPN_BAUD_RATE_48 Macro

File

rfcomm.h

C

```
#define RFCOMM_RPN_BAUD_RATE_48 0x01
```

Description

This is macro RFCOMM_RPN_BAUD_RATE_48.

RFCOMM_RPN_BAUD_RATE_576 Macro

File

rfcomm.h

C

```
#define RFCOMM_RPN_BAUD_RATE_576 0x06
```

Description

This is macro RFCOMM_RPN_BAUD_RATE_576.

RFCOMM_RPN_BAUD_RATE_72 Macro

File

rfcomm.h

C

```
#define RFCOMM_RPN_BAUD_RATE_72 0x02
```

Description

This is macro RFCOMM_RPN_BAUD_RATE_72.

RFCOMM_RPN_BAUD_RATE_96 Macro

File

rfcomm.h

C

```
#define RFCOMM_RPN_BAUD_RATE_96 0x03 // default
```

Description

default

RFCOMM_RPN_DATA_BIT_5 Macro

File

rfcomm.h

C

```
#define RFCOMM_RPN_DATA_BIT_5 0x00
```

Description

This is macro RFCOMM_RPN_DATA_BIT_5.

RFCOMM_RPN_DATA_BIT_6 Macro

File

rfcomm.h

C

```
#define RFCOMM_RPN_DATA_BIT_6 0x01
```

Description

This is macro RFCOMM_RPN_DATA_BIT_6.

RFCOMM_RPN_DATA_BIT_7 Macro

File

rfcomm.h

C

```
#define RFCOMM_RPN_DATA_BIT_7 0x02
```

Description

This is macro RFCOMM_RPN_DATA_BIT_7.

RFCOMM_RPN_DATA_BIT_8 Macro

File

rfcomm.h

C

```
#define RFCOMM_RPN_DATA_BIT_8 0x03 // default
```

Description

default

RFCOMM_RPN_FLC_N Macro

File

rfcomm.h

C

```
#define RFCOMM_RPN_FLC_N 0x00
```

Description

This is macro RFCOMM_RPN_FLC_N.

RFCOMM_RPN_FLC_RTC_INPUT Macro

File

rfcomm.h

C

```
#define RFCOMM_RPN_FLC_RTC_INPUT 0x10
```

Description

This is macro RFCOMM_RPN_FLC_RTC_INPUT.

RFCOMM_RPN_FLC_RTC_OUTPUT Macro

File

rfcomm.h

C

```
#define RFCOMM_RPN_FLC_RTC_OUTPUT 0x20
```

Description

This is macro RFCOMM_RPN_FLC_RTC_OUTPUT.

RFCOMM_RPN_FLC_RTR_INPUT Macro

File

rfcomm.h

C

```
#define RFCOMM_RPN_FLC_RTR_INPUT 0x04
```

Description

This is macro RFCOMM_RPN_FLC_RTR_INPUT.

RFCOMM_RPN_FLC_RTR_OUTPUT Macro

File

rfcomm.h

C

```
#define RFCOMM_RPN_FLC_RTR_OUTPUT 0x08
```

Description

This is macro RFCOMM_RPN_FLC_RTR_OUTPUT.

RFCOMM_RPN_FLC_XONOFF_INPUT Macro

File

rfcomm.h

C

```
#define RFCOMM_RPN_FLC_XONOFF_INPUT 0x01
```

Description

This is macro RFCOMM_RPN_FLC_XONOFF_INPUT.

RFCOMM_RPN_FLC_XONOFF_OUTPUT Macro

File

rfcomm.h

C

```
#define RFCOMM_RPN_FLC_XONOFF_OUTPUT 0x02
```

Description

This is macro RFCOMM_RPN_FLC_XONOFF_OUTPUT.

RFCOMM_RPN_PARITY_EVEN Macro

File

rfcomm.h

C

```
#define RFCOMM_RPN_PARITY_EVEN 0x10
```

Description

This is macro RFCOMM_RPN_PARITY_EVEN.

RFCOMM_RPN_PARITY_MARK Macro

File

rfcomm.h

C

```
#define RFCOMM_RPN_PARITY_MARK 0x20
```

Description

This is macro RFCOMM_RPN_PARITY_MARK.

RFCOMM_RPN_PARITY_N Macro

File

rfcomm.h

C

```
#define RFCOMM_RPN_PARITY_N 0x00 // default
```

Description

default

RFCOMM_RPN_PARITY_ODD Macro

File

rfcomm.h

C

```
#define RFCOMM_RPN_PARITY_ODD 0x00
```

Description

This is macro RFCOMM_RPN_PARITY_ODD.

RFCOMM_RPN_PARITY_SPACE Macro

File

rfcomm.h

C

```
#define RFCOMM_RPN_PARITY_SPACE 0x30
```

Description

This is macro RFCOMM_RPN_PARITY_SPACE.

RFCOMM_RPN_PARITY_Y Macro

File

rfcomm.h

C

```
#define RFCOMM_RPN_PARITY_Y 0x08
```

Description

This is macro RFCOMM_RPN_PARITY_Y.

RFCOMM_RPN_STOP_BIT_1 Macro

File

rfcomm.h

C

```
#define RFCOMM_RPN_STOP_BIT_1 0x00 // default
```

Description

default

RFCOMM_RPN_STOP_BIT_1_5 Macro

File

rfcomm.h

C

```
#define RFCOMM_RPN_STOP_BIT_1_5 0x04
```

Description

This is macro RFCOMM_RPN_STOP_BIT_1_5.

RFCOMM_RPN_XOFF_DEFAULT Macro

File

rfcomm.h

C

```
#define RFCOMM_RPN_XOFF_DEFAULT 0x13
```

Description

This is macro RFCOMM_RPN_XOFF_DEFAULT.

RFCOMM_RPN_XON_DEFAULT Macro

File

rfcomm.h

C

```
#define RFCOMM_RPN_XON_DEFAULT 0x11
```

Description

This is macro RFCOMM_RPN_XON_DEFAULT.

RFCOMM_SERIAL_PORT_CH_1 Macro

File

rfcomm.h

C

```
#define RFCOMM_SERIAL_PORT_CH_1 1
```

Description

This is macro RFCOMM_SERIAL_PORT_CH_1.

RFCOMM_SERIAL_PORT_CH_10 Macro

File

rfcomm.h

C

```
#define RFCOMM_SERIAL_PORT_CH_10 10
```

Description

This is macro RFCOMM_SERIAL_PORT_CH_10.

RFCOMM_SERIAL_PORT_CH_11 Macro

File

rfcomm.h

C

```
#define RFCOMM_SERIAL_PORT_CH_11 11
```

Description

This is macro RFCOMM_SERIAL_PORT_CH_11.

RFCOMM_SERIAL_PORT_CH_12 Macro

File

rfcomm.h

C

```
#define RFCOMM_SERIAL_PORT_CH_12 12
```

Description

This is macro RFCOMM_SERIAL_PORT_CH_12.

RFCOMM_SERIAL_PORT_CH_13 Macro

File

rfcomm.h

C

```
#define RFCOMM_SERIAL_PORT_CH_13 13
```

Description

This is macro RFCOMM_SERIAL_PORT_CH_13.

RFCOMM_SERIAL_PORT_CH_14 Macro

File

rfcomm.h

C

```
#define RFCOMM_SERIAL_PORT_CH_14 14
```

Description

This is macro RFCOMM_SERIAL_PORT_CH_14.

RFCOMM_SERIAL_PORT_CH_15 Macro

File

rfcomm.h

C

```
#define RFCOMM_SERIAL_PORT_CH_15 15
```

Description

This is macro RFCOMM_SERIAL_PORT_CH_15.

RFCOMM_SERIAL_PORT_CH_16 Macro

File

rfcomm.h

C

```
#define RFCOMM_SERIAL_PORT_CH_16 16
```

Description

This is macro RFCOMM_SERIAL_PORT_CH_16.

RFCOMM_SERIAL_PORT_CH_17 Macro

File

rfcomm.h

C

```
#define RFCOMM_SERIAL_PORT_CH_17 17
```

Description

This is macro RFCOMM_SERIAL_PORT_CH_17.

RFCOMM_SERIAL_PORT_CH_18 Macro

File

rfcomm.h

C

```
#define RFCOMM_SERIAL_PORT_CH_18 18
```

Description

This is macro RFCOMM_SERIAL_PORT_CH_18.

RFCOMM_SERIAL_PORT_CH_19 Macro

File

rfcomm.h

C

```
#define RFCOMM_SERIAL_PORT_CH_19 19
```

Description

This is macro RFCOMM_SERIAL_PORT_CH_19.

RFCOMM_SERIAL_PORT_CH_2 Macro

File

rfcomm.h

C

```
#define RFCOMM_SERIAL_PORT_CH_2 2
```

Description

This is macro RFCOMM_SERIAL_PORT_CH_2.

RFCOMM_SERIAL_PORT_CH_20 Macro

File

rfcomm.h

C

```
#define RFCOMM_SERIAL_PORT_CH_20 20
```

Description

This is macro RFCOMM_SERIAL_PORT_CH_20.

RFCOMM_SERIAL_PORT_CH_21 Macro

File

rfcomm.h

C

```
#define RFCOMM_SERIAL_PORT_CH_21 21
```

Description

This is macro RFCOMM_SERIAL_PORT_CH_21.

RFCOMM_SERIAL_PORT_CH_22 Macro

File

rfcomm.h

C

```
#define RFCOMM_SERIAL_PORT_CH_22 22
```

Description

This is macro RFCOMM_SERIAL_PORT_CH_22.

RFCOMM_SERIAL_PORT_CH_23 Macro

File

rfcomm.h

C

```
#define RFCOMM_SERIAL_PORT_CH_23 23
```

Description

This is macro RFCOMM_SERIAL_PORT_CH_23.

RFCOMM_SERIAL_PORT_CH_24 Macro

File

rfcomm.h

C

```
#define RFCOMM_SERIAL_PORT_CH_24 24
```

Description

This is macro RFCOMM_SERIAL_PORT_CH_24.

RFCOMM_SERIAL_PORT_CH_25 Macro

File

rfcomm.h

C

```
#define RFCOMM_SERIAL_PORT_CH_25 25
```

Description

This is macro RFCOMM_SERIAL_PORT_CH_25.

RFCOMM_SERIAL_PORT_CH_26 Macro

File

rfcomm.h

C

```
#define RFCOMM_SERIAL_PORT_CH_26 26
```

Description

This is macro RFCOMM_SERIAL_PORT_CH_26.

RFCOMM_SERIAL_PORT_CH_27 Macro

File

rfcomm.h

C

```
#define RFCOMM_SERIAL_PORT_CH_27 27
```

Description

This is macro RFCOMM_SERIAL_PORT_CH_27.

RFCOMM_SERIAL_PORT_CH_28 Macro

File

rfcomm.h

C

```
#define RFCOMM_SERIAL_PORT_CH_28 28
```

Description

This is macro RFCOMM_SERIAL_PORT_CH_28.

RFCOMM_SERIAL_PORT_CH_29 Macro

File

rfcomm.h

C

```
#define RFCOMM_SERIAL_PORT_CH_29 29
```

Description

This is macro RFCOMM_SERIAL_PORT_CH_29.

RFCOMM_SERIAL_PORT_CH_3 Macro

File

rfcomm.h

C

```
#define RFCOMM_SERIAL_PORT_CH_3 3
```

Description

This is macro RFCOMM_SERIAL_PORT_CH_3.

RFCOMM_SERIAL_PORT_CH_30 Macro

File

rfcomm.h

C

```
#define RFCOMM_SERIAL_PORT_CH_30 30
```

Description

This is macro RFCOMM_SERIAL_PORT_CH_30.

RFCOMM_SERIAL_PORT_CH_4 Macro

File

rfcomm.h

C

```
#define RFCOMM_SERIAL_PORT_CH_4 4
```

Description

This is macro RFCOMM_SERIAL_PORT_CH_4.

RFCOMM_SERIAL_PORT_CH_5 Macro

File

rfcomm.h

C

```
#define RFCOMM_SERIAL_PORT_CH_5 5
```

Description

This is macro RFCOMM_SERIAL_PORT_CH_5.

RFCOMM_SERIAL_PORT_CH_6 Macro

File

rfcomm.h

C

```
#define RFCOMM_SERIAL_PORT_CH_6 6
```

Description

This is macro RFCOMM_SERIAL_PORT_CH_6.

RFCOMM_SERIAL_PORT_CH_7 Macro

File

rfcomm.h

C

```
#define RFCOMM_SERIAL_PORT_CH_7 7
```

Description

This is macro RFCOMM_SERIAL_PORT_CH_7.

RFCOMM_SERIAL_PORT_CH_8 Macro

File

rfcomm.h

C

```
#define RFCOMM_SERIAL_PORT_CH_8 8
```

Description

This is macro RFCOMM_SERIAL_PORT_CH_8.

RFCOMM_SERIAL_PORT_CH_9 Macro

File

rfcomm.h

C

```
#define RFCOMM_SERIAL_PORT_CH_9 9
```

Description

This is macro RFCOMM_SERIAL_PORT_CH_9.

RFCOMM_SESSION_CHANGED_AFC Macro

File

rfcomm.h

C

```
#define RFCOMM_SESSION_CHANGED_AFC 1
```

Description

This is macro RFCOMM_SESSION_CHANGED_AFC.

RFCOMM_SESSION_CHANGED_CONN_STATE Macro

File

rfcomm.h

C

```
#define RFCOMM_SESSION_CHANGED_CONN_STATE 0
```

Description

This is macro RFCOMM_SESSION_CHANGED_CONN_STATE.

RFCOMM_SESSION_STATE_CONNECTED Macro

File

rfcomm.h

C

```
#define RFCOMM_SESSION_STATE_CONNECTED 1
```

Description

This is macro RFCOMM_SESSION_STATE_CONNECTED.

RFCOMM_SESSION_STATE_DISCONNECTED Macro

File

rfcomm.h

C

```
#define RFCOMM_SESSION_STATE_DISCONNECTED 2
```

Description

This is macro RFCOMM_SESSION_STATE_DISCONNECTED.

RFCOMM_SESSION_STATE_FREE Macro

File

rfcomm.h

C

```
#define RFCOMM_SESSION_STATE_FREE 0
```

Description

This is macro RFCOMM_SESSION_STATE_FREE.

RFCOMM_TIMEOUT Macro

File

rfcomm.h

C

```
#define RFCOMM_TIMEOUT 60
```

Description

This is macro RFCOMM_TIMEOUT.

bt_rfcomm_cmd_callback_fp Type

File

rfcomm.h

C

```
typedef void (* bt_rfcomm_cmd_callback_fp)(struct _bt_rfcomm_dlc_t* dlc, struct _bt_rfcomm_command_t *cmd,
bt_int status);
```

Description

This is type bt_rfcomm_cmd_callback_fp.

bt_rfcomm_command_p Structure

File

rfcomm.h

C

```
typedef struct _bt_rfcomm_command_t {
  struct _bt_rfcomm_command_t* next_cmd;
  bt_byte address;
  bt_byte control;
  bt_byte credit;
  bt_byte_p pdata;
  bt_int len;
  bt_byte status;
  bt_rfcomm_cmd_callback_fp cb;
  bt_rfcomm_send_data_callback_fp send_data_cb;
  bt_long send_time;
  bt_byte mx_params[RFCOMM_MX_MSG_MAX_DATA_LEN];
} bt_rfcomm_command_t, * bt_rfcomm_command_p;
```

Description

This is type bt_rfcomm_command_p.

bt_rfcomm_command_t Structure

File

rfcomm.h

C

```
typedef struct _bt_rfcomm_command_t {
  struct _bt_rfcomm_command_t* next_cmd;
  bt_byte address;
  bt_byte control;
  bt_byte credit;
  bt_byte_p pdata;
  bt_int len;
  bt_byte status;
  bt_rfcomm_cmd_callback_fp cb;
  bt_rfcomm_send_data_callback_fp send_data_cb;
```

```
  bt_long send_time;
  bt_byte mx_params[RFCOMM_MX_MSG_MAX_DATA_LEN];
} bt_rfcomm_command_t, * bt_rfcomm_command_p;
```

Description

This is type bt_rfcomm_command_t.

bt_rfcomm_ctl_msg_p Structure

File

rfcomm.h

C

```
typedef struct _bt_rfcomm_ctl_msg_t {
  bt_byte type;
  bt_int len;
  bt_byte_p pdata;
} bt_rfcomm_ctl_msg_t, * bt_rfcomm_ctl_msg_p;
```

Description

This is type bt_rfcomm_ctl_msg_p.

bt_rfcomm_ctl_msg_t Structure

File

rfcomm.h

C

```
typedef struct _bt_rfcomm_ctl_msg_t {
  bt_byte type;
  bt_int len;
  bt_byte_p pdata;
} bt_rfcomm_ctl_msg_t, * bt_rfcomm_ctl_msg_p;
```

Description

This is type bt_rfcomm_ctl_msg_t.

bt_rfcomm_dlc_p Structure

File

rfcomm.h

C

```
typedef struct _bt_rfcomm_dlc_t {
  bt_byte id;
  bt_byte priority;
  bt_uint max_frame_size;
  bt_byte cfc_remote_credit;
  bt_byte cfc_freed_local_buffers;
  bt_byte state;
  bt_byte local_ms;
  bt_byte remote_ms;
  bt_byte local_ls;
  bt_byte remote_ls;
  bt_bool buffered;
  bt_rfcomm_read_data_callback_fp read_data_cb;
  bt_rfcomm_dlc_state_callback_fp state_cb;
  void* cb_param;
  bt_rfcomm_packet_t tx_packet;
  struct _bt_rfcomm_session_t * psess;
  bt_queue_element_t* cmd_queue;
} bt_rfcomm_dlc_t, * bt_rfcomm_dlc_p;
```

Description

This is type bt_rfcomm_dlc_p.

bt_rfcomm_dlc_state_callback_fp Type

File

rfcomm.h

C

```
typedef void (* bt_rfcomm_dlc_state_callback_fp)(struct _bt_rfcomm_dlc_t* dlc, bt_int what, void* param);
```

Description

This is type bt_rfcomm_dlc_state_callback_fp.

bt_rfcomm_dlc_t Structure

File

rfcomm.h

C

```
typedef struct _bt_rfcomm_dlc_t {
  bt_byte id;
  bt_byte priority;
  bt_uint max_frame_size;
  bt_byte cfc_remote_credit;
  bt_byte cfc_freed_local_buffers;
  bt_byte state;
  bt_byte local_ms;
  bt_byte remote_ms;
  bt_byte local_ls;
  bt_byte remote_ls;
  bt_bool buffered;
  bt_rfcomm_read_data_callback_fp read_data_cb;
  bt_rfcomm_dlc_state_callback_fp state_cb;
  void* cb_param;
  bt_rfcomm_packet_t tx_packet;
  struct _bt_rfcomm_session_t * psess;
  bt_queue_element_t* cmd_queue;
} bt_rfcomm_dlc_t, * bt_rfcomm_dlc_p;
```

Description

This is type bt_rfcomm_dlc_t.

bt_rfcomm_mgr_t Structure

File

rfcomm_mgr.h

C

```
typedef struct _bt_rfcomm_mgr_t {
  bt_rfcomm_session_listener_t* listeners;
  bt_rfcomm_server_channel_t* channels;
} bt_rfcomm_mgr_t;
```

Description

This is type bt_rfcomm_mgr_t.

bt_rfcomm_read_data_callback_fp Type

File

rfcomm.h

C

```
typedef void (* bt_rfcomm_read_data_callback_fp)(struct _bt_rfcomm_dlc_t* dlc, bt_byte_p data, bt_int len);
```

Description

This is type bt_rfcomm_read_data_callback_fp.

bt_rfcomm_send_data_callback_fp Type

File

rfcomm.h

C

```
typedef void (* bt_rfcomm_send_data_callback_fp)(struct _bt_rfcomm_dlc_t* dlc, bt_byte_p data, bt_int len,
bt_int status);
```

Description

This is type bt_rfcomm_send_data_callback_fp.

bt_rfcomm_server_channel_t Structure

File

rfcomm_mgr.h

C

```
typedef struct _bt_rfcomm_server_channel_t {
  bt_byte id;
  bt_rfcomm_dlc_state_callback_fp listen_cb;
  void* listen_param;
} bt_rfcomm_server_channel_t;
```

Description

This is type bt_rfcomm_server_channel_t.

bt_rfcomm_session_listener_t Structure

File

rfcomm_mgr.h

C

```
typedef struct _bt_rfcomm_session_listener_t {
  struct _bt_rfcomm_session_listener_t* next_listener;
  bt_rfcomm_state_callback_fp callback;
  void* callback_param;
} bt_rfcomm_session_listener_t;
```

Description

This is type bt_rfcomm_session_listener_t.

bt_rfcomm_session_p Structure

File

rfcomm.h

C

```
typedef struct _bt_rfcomm_session_t {
  bt_byte state;
  bt_l2cap_channel_t* pch;
  bt_byte role;
  bt_byte fc_type;
  bt_bool afc_off;
  bt_rfcomm_dlc_t* dlcs;
  bt_rfcomm_dlc_state_callback_fp connect_cb;
  void* connect_param;
  bt_byte connect_server_channel;
  bt_rfcomm_command_p pcmd_cur;
```

```
  bt_l2cap_mgr_p l2cap_mgr;
} bt_rfcomm_session_t, * bt_rfcomm_session_p;
```

Description

This is type bt_rfcomm_session_p.

bt_rfcomm_session_t Structure

File

rfcomm.h

C

```
typedef struct _bt_rfcomm_session_t {
  bt_byte state;
  bt_l2cap_channel_t* pch;
  bt_byte role;
  bt_byte fc_type;
  bt_bool afc_off;
  bt_rfcomm_dlc_t* dlcs;
  bt_rfcomm_dlc_state_callback_fp connect_cb;
  void* connect_param;
  bt_byte connect_server_channel;
  bt_rfcomm_command_p pcmd_cur;
  bt_l2cap_mgr_p l2cap_mgr;
} bt_rfcomm_session_t, * bt_rfcomm_session_p;
```

Description

This is type bt_rfcomm_session_t.

bt_rfcomm_state_callback_fp Type

File

rfcomm.h

C

```
typedef void (* bt_rfcomm_state_callback_fp)(struct _bt_rfcomm_session_t* session, bt_int what, void*
param);
```

Description

This is type bt_rfcomm_state_callback_fp.

RFCOMM_ERR_INTERRUPTED Macro

File

rfcomm.h

C

```
#define RFCOMM_ERR_INTERRUPTED 0x03
```

Description

This is macro RFCOMM_ERR_INTERRUPTED.

RFCOMM_MIX_INFO_LEN Macro

File

rfcomm.h

C

```
#define RFCOMM_MIX_INFO_LEN 31
```

Description

This is macro RFCOMM_MIX_INFO_LEN.

RFCOMM_MX_MSG_MAX_DATA_LEN Macro

File

rfcomm.h

C

```
#define RFCOMM_MX_MSG_MAX_DATA_LEN 10
```

Description

This is macro RFCOMM_MX_MSG_MAX_DATA_LEN.

SBC Decoder Functions

sbc_decode Function

Decodes the encoded SBC frame.

File

sbc.h

C

```
ssize_t sbc_decode(sbc_t * sbc, const void * input, size_t input_len, void * output, size_t output_len,
size_t * written);
```

Returns

This function returns the length of one encoded frame that has just been decoded.

Description

This function decodes the encoded SBC frame.

Remarks

This function is called once to decode one frame and will be called in an infinite loop. The decoder decodes one frame at a time even though there can be several frames in the input buffer. This function returns the length of the encoded frame that has just been decoded. The decoder also populates the sbc_t *sbc structure with the sampling rate, number of sub-bands, bitpool value, mode, and the allocation method.

Parameters

Parameters	Description
*sbc	Pointer to the SBC data structure.
*input	The pointer to the input buffer, where the encoded bit stream is available.
input_len	The number of valid input data bytes available in the above buffer.
*output	The pointer to the output sample buffer where the decoded PCM samples are to be written.
output_len	Maximum length of the output buffer.
*written	The number of output bytes written in the above buffer.

Function

ssize_t sbc_decode(sbc_t *sbc, const void *input, size_t input_len,
void *output, size_t output_len, size_t *written);

sbc_get_codesize Function

Gets the input block size.

File

sbc.h

C

```
size_t sbc_get_codesize(sbc_t * sbc);
```

Returns

This function returns the size of the encoded frame in bytes including the 4-byte header.

Description

This function gets the input block size.

Parameters

Parameters	Description
*sbc	Pointer to the allocated decoder state is passed.

Function

size_t sbc_get_codesize(sbc_t *sbc);

sbc_get_frame_duration Function

Gets the time one input/output block takes to play in microseconds.

File

sbc.h

C

```
uint32_t sbc_get_frame_duration(sbc_t * sbc);
```

Returns

This function returns the duration of the last decoded frame to be played. The duration is in microseconds.

Description

This function gets the time one input/output block takes to play in microseconds.

Parameters

Parameters	Description
*sbc	Pointer to the allocated decoder state is passed.

Function

uint32_t sbc_get_frame_duration(sbc_t *sbc);

sbc_get_frame_length Function

Gets the output block size.

File

sbc.h

C

```
size_t sbc_get_frame_length(sbc_t * sbc);
```

Returns

This function returns the size of the output frame (in bytes) that was decoded last.

Description

This function gets the output block size.

Parameters

Parameters	Description
*sbc	Pointer to the allocated decoder state is passed.

Function

size_t sbc_get_frame_length(sbc_t *sbc);

sbc_get_state_size Function

Gets the SBC Decoder's state data structure.

File

sbc.h

C

```
int16_t sbc_get_state_size();
```

Returns

This function returns the size of the SBC Decoder state structure in bytes.

Description

This function gets the SBC Decoder's state data structure size.

Function

int16_t sbc_get_state_size();

sbc_init Function

Initializes the SBC Decoder.

File

sbc.h

C

```
int32_t sbc_init(sbc_t * sbc, uint32_t flags);
```

Returns

This function always returns the value '0'; however, this return value is not used in the application.

Description

This function initializes the SBC Decoder.

Remarks

This function views the received pointer as structure pointer and clears the entire structure. This function is called only once for each channel before starting the decoding of the first frame.

Parameters

Parameters	Description
*sbc	Pointer to the SBC data structure.
flags	Unused. 0L is always passed.

Function

int32_t sbc_init(sbc_t *sbc, uint32_t flags);

SBC Decoder Data Types and Constants

sbc_struct Structure

File

sbc.h

C

```
struct sbc_struct {
  uint32_t flags;
```

```
    uint8_t frequency;
    uint8_t blocks;
    uint8_t subbands;
    uint8_t mode;
    uint8_t allocation;
    uint8_t bitpool;
    uint8_t endian;
    void * priv;
    void * priv_alloc_base;
    int32_t v_vec_fix[2][256+160];
    int32_t SamplingFreq;
    uint32_t Dword;
    int16_t bitcount;
    int16_t Pos;
    int16_t codeSize;
    int16_t outSize;
    int8_t BlocksIndex;
    int8_t SubBandsIndex;
    int8_t nrof_Chnls;
    int8_t Bits;
    uint8_t CrcCheck;
    uint8_t CrcData[2];
    uint8_t join[8];
    int8_t bits[2][8];
    uint8_t * Buffer;
};
```

Members

Members	Description
int32_t v_vec_fix[2][256+160];	Below variables / buffers added

Section

Data Types

sbc_t Type

File

sbc.h

C

```
typedef struct sbc_struct sbc_t;
```

Description

This is type sbc_t.

__SBC_H Macro

File

sbc.h

C

```
#define __SBC_H
```

Description

This is macro __SBC_H.

INP_BUF_SIZE Macro

File

sbc.h

C

```
#define INP_BUF_SIZE 512     // Bytes
```

Description

Bytes

MAX_SBC_DEC_STATE_SIZE Macro

File

sbc.h

C

```
#define MAX_SBC_DEC_STATE_SIZE 3500    // Bytes
```

Description

Bytes

OUT_BUF_SIZE Macro

File

sbc.h

C

```
#define OUT_BUF_SIZE 512     // Bytes
```

Description

Bytes

SBC_AM_LOUDNESS Macro

File

sbc.h

C

```
#define SBC_AM_LOUDNESS 0x00
```

Description

Allocation method

SBC_AM_SNR Macro

File

sbc.h

C

```
#define SBC_AM_SNR 0x01
```

Description

This is macro SBC_AM_SNR.

SBC_BE Macro

File

sbc.h

C

```
#define SBC_BE 0x01
```

Description

This is macro SBC_BE.

SBC_BLK_12 Macro

File

sbc.h

C

```
#define SBC_BLK_12 0x02
```

Description

This is macro SBC_BLK_12.

SBC_BLK_16 Macro

File

sbc.h

C

```
#define SBC_BLK_16 0x03
```

Description

This is macro SBC_BLK_16.

SBC_BLK_4 Macro

File

sbc.h

C

```
#define SBC_BLK_4 0x00
```

Description

Blocks

SBC_BLK_8 Macro

File

sbc.h

C

```
#define SBC_BLK_8 0x01
```

Description

This is macro SBC_BLK_8.

SBC_FREQ_16000 Macro

File

sbc.h

C

```
#define SBC_FREQ_16000 0x00
```

Description

Sampling frequency

SBC_FREQ_32000 Macro

File

sbc.h

C

```
#define SBC_FREQ_32000 0x01
```

Description

This is macro SBC_FREQ_32000.

SBC_FREQ_44100 Macro

File

sbc.h

C

```
#define SBC_FREQ_44100 0x02
```

Description

This is macro SBC_FREQ_44100.

SBC_FREQ_48000 Macro

File

sbc.h

C

```
#define SBC_FREQ_48000 0x03
```

Description

This is macro SBC_FREQ_48000.

SBC_LE Macro

File

sbc.h

C

```
#define SBC_LE 0x00
```

Description

Data endianness

SBC_MODE_DUAL_CHANNEL Macro

File

sbc.h

C

```
#define SBC_MODE_DUAL_CHANNEL 0x01
```

Description

This is macro SBC_MODE_DUAL_CHANNEL.

SBC_MODE_JOINT_STEREO Macro

File

sbc.h

C

```
#define SBC_MODE_JOINT_STEREO 0x03
```

Description

This is macro SBC_MODE_JOINT_STEREO.

SBC_MODE_MONO Macro

File

sbc.h

C

```
#define SBC_MODE_MONO 0x00
```

Description

Channel mode

SBC_MODE_STEREO Macro

File

sbc.h

C

```
#define SBC_MODE_STEREO 0x02
```

Description

This is macro SBC_MODE_STEREO.

SBC_SB_4 Macro

File

sbc.h

C

```
#define SBC_SB_4 0x00
```

Description

Sub-bands

SBC_SB_8 Macro

File

sbc.h

C

```
#define SBC_SB_8 0x01
```

Description

This is macro SBC_SB_8.

ssize_t Macro

File

sbc.h

C

```
#define ssize_t __int32_t
```

Section

Constants

SDP Functions

bt_sdp_de_to_uuid Function

File

sdp_utils.h

C

```
void bt_sdp_de_to_uuid(bt_sdp_data_element_cp pde, bt_uuid_p puuid);
```

Description

This is function bt_sdp_de_to_uuid.

bt_sdp_packet_assembler Function

File

sdp_packet.h

C

```
bt_int bt_sdp_packet_assembler(bt_packet_t* packet, bt_byte* buffer, bt_int buffer_len);
```

Description

This is function bt_sdp_packet_assembler.

bt_sdp_read_attribute Function

File

sdp_private.h

C

```
bt_bool bt_sdp_read_attribute(bt_sdp_server_attribute_t* attr, const bt_byte* buffer, bt_int len, bt_int*
offset);
```

Description

From sdp_db_utils.c

bt_sdp_start Function

File

sdp.h

C

```
bt_bool bt_sdp_start(bt_l2cap_mgr_p l2cap_mgr, const bt_byte* sdp_db, bt_uint sdp_db_len);
```

Description

brief Start SDP server ingroup sdp

details This function starts the SDP server.

param l2cap_mgr The L2CAP manager on which the SDP server is to be started. param sdp_db A pointer to the SDP database define with BEGIN_SDP_DB and END_SDP_DB macros.

return li c TRUE if the function succeeds. li c FALSE otherwise.

sdp_compare_uuid_de Function

File

sdp_private.h

C

```
bt_bool sdp_compare_uuid_de(bt_sdp_data_element_cp pde1, bt_sdp_data_element_cp pde2);
```

Description

This is function sdp_compare_uuid_de.

sdp_find_attributes Function

File

sdp_private.h

C

```
bt_bool sdp_find_attributes(bt_sr_handle_p h_list, bt_int h_count, bt_byte_p pparams, bt_int params_len,
bt_int offsetInit, bt_byte patternCount, bt_sdp_transaction_t* ptran, bt_int max_attrs);
```

Description

This is function sdp_find_attributes.

sdp_find_service_records2 Function

File

sdp_private.h

C

```
int sdp_find_service_records2(bt_byte_p pparams, bt_int params_len, bt_int offsetInit, bt_byte
patternCount, bt_sdp_service_transaction_p ptran, bt_int max_handles);
```

Description

From sdp_utils.c

_sdp_alloc_svc_tran_buffer Function

File

sdp_private.h

C

```
bt_sdp_service_transaction_p _sdp_alloc_svc_tran_buffer(bt_int id);
```

Description

This is function _sdp_alloc_svc_tran_buffer.

_sdp_alloc_tran_buffer Function

File

sdp_private.h

C

```
bt_sdp_transaction_t* _sdp_alloc_tran_buffer(bt_int id);
```

Description

This is function _sdp_alloc_tran_buffer.

_sdp_find_svc_transaction Function

File

sdp_private.h

C

```
bt_sdp_service_transaction_p _sdp_find_svc_transaction(bt_int id);
```

Description

This is function _sdp_find_svc_transaction.

_sdp_find_transaction Function

File

sdp_private.h

C

```
bt_sdp_transaction_t* _sdp_find_transaction(bt_int id);
```

Description

This is function _sdp_find_transaction.

_sdp_free_svc_tran_buffer Function

File

sdp_private.h

C

```
void _sdp_free_svc_tran_buffer(bt_sdp_service_transaction_p ptran);
```

Description

This is function _sdp_free_svc_tran_buffer.

_sdp_free_tran_buffer Function

File

sdp_private.h

C

```
void _sdp_free_tran_buffer(bt_sdp_transaction_t* ptran);
```

Description

This is function _sdp_free_tran_buffer.

_sdp_get_de_data_len Function

File

sdp_utils.h

C

```
bt_ulong _sdp_get_de_data_len(bt_sdp_data_element_cp pde);
```

Description

This is function _sdp_get_de_data_len.

_sdp_get_de_hdr_len Function

File

sdp_utils.h

C

```
bt_ulong _sdp_get_de_hdr_len(bt_sdp_data_element_cp pde);
```

Description

This is function _sdp_get_de_hdr_len.

_sdp_init_tran_buffers Function

File

sdp_private.h

C

```
bt_bool _sdp_init_tran_buffers();
```

Description

From sdp_tran_buffer.c

_sdp_read_de_header Function

File

sdp_utils.h

C

```
bt_bool _sdp_read_de_header(bt_byte_cp buffer, bt_int len, bt_int_p poffset, bt_byte_p pde_type, bt_byte_p
pde_size_index, bt_ulong_p pde_data_len);
```

Description

This is function _sdp_read_de_header.

_sdp_write_data_element Function

File

sdp_utils.h

C

```
bt_bool _sdp_write_data_element(bt_sdp_data_element_cp pde, bt_byte_p buffer, bt_int len, bt_int_p poffset);
```

Description

This is function _sdp_write_data_element.

bt_sdp_request_service_attribute Function

File

sdp_client.h

C

```
bt_bool bt_sdp_request_service_attribute(bt_l2cap_channel_t* channel, bt_sr_handle_t sr,
bt_sdp_data_element_p pattern, bt_sdp_service_attribute_callback_fp callback, void* callback_param);
```

Description

brief Search attributes ingroup sdp

details This function retrieves attribute values from a service record.

param channel The L2CAP channel used to communicate to the remote SDP server. param sr The service record handle specifies the service record from which attribute values are to be retrieved. param pattern The attribute search pattern is a data element sequence where each element in the list is either an attribute ID or a range of attribute IDs. The pattern buffer must be valid for the duration of the search operation, i.e. until c callback is called for the first time. To define a data element sequence use the BEGIN_DE_SEQUENCE and END_DE_SEQUENCE macros. These macros will define a variable whose name is the id of the data element sequence passed to the macros prefixed with "seq_". A pointer to this variable can be used as the value for the pattern parameter.

param callback The callback function that will be called when search has completed.

param callback_param A pointer to arbitrary data to be passed to the c callback callback.

return li c TRUE if the function succeeds. li c FALSE otherwise. The callback function is not called in this case.

bt_sdp_request_service_search Function

File

sdp_client.h

C

```
bt_bool bt_sdp_request_service_search(bt_l2cap_channel_t* channel, bt_sdp_data_element_p pattern,
bt_sdp_service_search_callback_fp callback, void* callback_param);
```

Description

brief Search service records ingroup sdp

details This function locates service records on a remote SDP server that match the given service search pattern.

param channel The L2CAP channel used to communicate to the remote SDP server. param pattern The service search pattern is a data element sequence where each element in the sequence is a UUID. The sequence must contain at least one UUID. The maximum number of UUIDs in the sequence is 12. The pattern buffer must be valid for the duration of the search operation, i.e. until c callback is called. To define a data element sequence use the BEGIN_DE_SEQUENCE and END_DE_SEQUENCE macros. These macros will define a variable whose name is the id of the data element sequence passed to the macros prefixed with "seq_". A pointer to this variable can be used as the value for the pattern parameter. param callback The callback function that will be called when search has completed. param callback_param A pointer to arbitrary data to be passed to the c callback callback.

return li c TRUE if the function succeeds. li c FALSE otherwise. The callback function is not called in this case.

_bt_sdp_client_init Function

File

sdp_private.h

C

```
void _bt_sdp_client_init();
```

Description

This is function _bt_sdp_client_init.

SDP Data Types and Constants

BEGIN_DE_SEQUENCE Macro

File

sdp.h

C

```
#define BEGIN_DE_SEQUENCE(id, len) \
    static bt_sdp_data_element_t id[len];    \
    static bt_sdp_sequence_t seq_##id = { len, (bt_sdp_data_element_p)id};    \
    static void init_de_sequence_##id() {    \
        static bt_bool initialized = FALSE;    \
```

```
        if (!initialized) {    \
            bt_int i = 0;    \
            bt_int max_len = len;    \
            bt_sdp_data_element_t* cur_de = id;    \
            initialized = TRUE;
```

Description

- brief Begin a data element sequence
- ingroup sdp

*

- details BEGIN_DE_SEQUENCE and END_DE_SEQUENCE are used to define a data element sequence which is an array of sdp_data_element structures.
- The array is used a search pattern in bt_sdp_request_service_search() and bt_sdp_request_service_attribute().
- For example, to find a AVRCP Target the following code can be used:

code const bt_uuid_t AVRCP_AV_REMOTE_CONTROL_CLSID = { 0x5F9B34FB, 0x80000080, 0x00001000, SDP_CLSID_AV_REMOTE_CONTROL }; const bt_uuid_t AVRCP_AV_REMOTE_CONTROL_TARGET_CLSID = { 0x5F9B34FB, 0x80000080, 0x00001000, SDP_CLSID_AV_REMOTE_CONTROL_TARGET };

BEGIN_DE_SEQUENCE(avrcp_target_service_search, 2) DE_UUID128(AVRCP_AV_REMOTE_CONTROL_CLSID) DE_UUID128(AVRCP_AV_REMOTE_CONTROL_TARGET_CLSID) END_DE_SEQUENCE(avrcp_target_service_search)

. . .

void findAvrcpTarget(void) { INIT_DE_SEQUENCE(avrcp_target_service_search);

bt_sdp_request_service_search(channel, &seq_avrcp_target_service_search, &callback, NULL); }

endcode

*

- param id The data element sequence identifier.
- param len The number of elements in the data element sequence.

DE_BOOL Macro

File

sdp.h

C

```
#define DE_BOOL(value) cur_de->type = SDP_DATA_TYPE_BOOL; cur_de->data.b = value; cur_de++; if (++i ==
max_len) return;
```

Description

brief Declare a boolean data element ingroup sdp

details This macro adds a boolean data element to a data element sequence. This macro is to be used between BEGIN_DE_SEQUENCE and END_DE_SEQUENCE.

param value The data element value.

DE_INT Macro

File

sdp.h

C

```
#define DE_INT(value) cur_de->type = SDP_DATA_TYPE_INT; cur_de->data.b = value; cur_de++; if (++i ==
max_len) return;
```

Description

brief Declare a 1-byte signed integer data element ingroup sdp

details This macro adds a 1-byte signed integer data element to a data element sequence. This macro is to be used between BEGIN_DE_SEQUENCE and END_DE_SEQUENCE.

param value The data element value.

DE_STRING Macro

File

sdp.h

C

```
#define DE_STRING(value) cur_de->type = SDP_DATA_TYPE_STRING; cur_de->data.pstr = value; cur_de++;
```

Description

brief Declare a text string data element ingroup sdp

details This macro adds a text string data element to a data element sequence. The length of the generated data element will be the actual length of the string. This macro is to be used between BEGIN_DE_SEQUENCE and END_DE_SEQUENCE.

param value The data element value.

DE_STRING2 Macro

File

sdp.h

C

```
#define DE_STRING2(value, len) cur_de->type = SDP_DATA_TYPE_UINT; cur_de->data.pstr = value;
cur_de->bytecount = len; cur_de++; if (++i == max_len) return;
```

Description

brief Declare a text string data element ingroup sdp

details This macro adds a text string data element to a data element sequence. The length of the generated data element will be the value specified by the "len" parameter even if the actual length of the string is not equal to the "len" value. This macro is to be used between BEGIN_DE_SEQUENCE and END_DE_SEQUENCE.

param value The data element value. param len The length of the data element value.

DE_UINT Macro

File

sdp.h

C

```
#define DE_UINT(value) cur_de->type = SDP_DATA_TYPE_UINT; cur_de->data.b = value; cur_de++; if (++i ==
max_len) return;
```

Description

brief Declare a 1-byte unsigned integer data element ingroup sdp

details This macro adds a 1-byte unsigned integer data element to a data element sequence. This macro is to be used between BEGIN_DE_SEQUENCE and END_DE_SEQUENCE.

param value The data element value.

DE_UINT16 Macro

File

sdp.h

C

```
#define DE_UINT16(value) cur_de->type = SDP_DATA_TYPE_UINT16; cur_de->data.ui = value; cur_de++; if (++i ==
max_len) return;
```

Description

brief Declare a 2-byte unsigned integer data element ingroup sdp

details This macro adds a 2-byte unsigned integer data element to a data element sequence. This macro is to be used between BEGIN_DE_SEQUENCE and END_DE_SEQUENCE.

param value The data element value.

DE_URL Macro

File

sdp.h

C

```
#define DE_URL(value) cur_de->type = SDP_DATA_TYPE_URL; cur_de->data.purl = value; cur_de++; if (++i ==
max_len) return;
```

Description

brief Declare a URL data element ingroup sdp

details This macro adds a URL data element to a data element sequence. This macro is to be used between BEGIN_DE_SEQUENCE and END_DE_SEQUENCE.

param value The data element value which is a pointer to a string.

DE_UUID128 Macro

File

sdp.h

C

```
#define DE_UUID128(value) cur_de->type = SDP_DATA_TYPE_UUID128; cur_de->data.uuid128 = (bt_uuid_t*)&value;
cur_de++; if (++i == max_len) return;
```

Description

brief Declare a 128-bit UUID data element ingroup sdp

details This macro adds a 128-bit UUID data element to a data element sequence. This macro is to be used between BEGIN_DE_SEQUENCE and END_DE_SEQUENCE.

param value The data element value. The value must be a name of a variable of type bt_uuid.

DE_UUID16 Macro

File

sdp.h

C

```
#define DE_UUID16(value) cur_de->type = SDP_DATA_TYPE_UUID16; cur_de->data.uuid16 = value; cur_de++; if
(++i == max_len) return;
```

Description

brief Declare a 16-bit UUID data element ingroup sdp

details This macro adds a 16-bit UUID data element to a data element sequence. This macro is to be used between BEGIN_DE_SEQUENCE and END_DE_SEQUENCE.

param value The data element value.

DE_UUID32 Macro

File

sdp.h

C

```
#define DE_UUID32(value) cur_de->type = SDP_DATA_TYPE_UUID32; cur_de->data.uuid32 = value; cur_de++; if
(++i == max_len) return;
```

Description

brief Declare a 32-bit UUID data element ingroup sdp

details This macro adds a 32-bit UUID data element to a data element sequence. This macro is to be used between BEGIN_DE_SEQUENCE and

END_DE_SEQUENCE.

param value The data element value.

END_DE_SEQUENCE Macro

File

sdp.h

C

```
#define END_DE_SEQUENCE(id) }}
```

Description

brief End a data element sequence ingroup sdp

details BEGIN_DE_SEQUENCE and END_DE_SEQUENCE are used to define a data element sequence which is an array of bt_sdp_data_element structures.

param id The data element sequence identifier.

INIT_DE_SEQUENCE Macro

File

sdp.h

C

```
#define INIT_DE_SEQUENCE(id) init_de_sequence_##id();
```

Description

brief Initialize a data element sequence ingroup sdp

details This macro calls a function defined in BEGIN_DE_SEQUENCE which initializes the data element sequence.

param id The data element sequence identifier.

SDP_ATTRID_AdditionalProtocolDescriptorLists Macro

File

sdp.h

C

```
#define SDP_ATTRID_AdditionalProtocolDescriptorLists 0x000D
```

Description

This is macro SDP_ATTRID_AdditionalProtocolDescriptorLists.

SDP_ATTRID_BluetoothProfileDescriptorList Macro

File

sdp.h

C

```
#define SDP_ATTRID_BluetoothProfileDescriptorList 0x0009
```

Description

This is macro SDP_ATTRID_BluetoothProfileDescriptorList.

SDP_ATTRID_BrowseGroupList Macro

File

sdp.h

C

```
#define SDP_ATTRID_BrowseGroupList 0x0005
```

Description

This is macro SDP_ATTRID_BrowseGroupList.

SDP_ATTRID_ClientExecutableURL Macro

File

sdp.h

C

```
#define SDP_ATTRID_ClientExecutableURL 0x000B
```

Description

This is macro SDP_ATTRID_ClientExecutableURL.

SDP_ATTRID_DIPrimaryRecord Macro

File

sdp.h

C

```
#define SDP_ATTRID_DIPrimaryRecord 0x204
```

Description

This is macro SDP_ATTRID_DIPrimaryRecord.

SDP_ATTRID_DIProductId Macro

File

sdp.h

C

```
#define SDP_ATTRID_DIProductId 0x202
```

Description

This is macro SDP_ATTRID_DIProductId.

SDP_ATTRID_DISpecificationId Macro

File

sdp.h

C

```
#define SDP_ATTRID_DISpecificationId 0x200
```

Description

This is macro SDP_ATTRID_DISpecificationId.

SDP_ATTRID_DIVendorId Macro

File

sdp.h

C

```
#define SDP_ATTRID_DIVendorId 0x201
```

Description

This is macro SDP_ATTRID_DIVendorId.

SDP_ATTRID_DIVendorIdSource Macro

File

sdp.h

C

```
#define SDP_ATTRID_DIVendorIdSource 0x205
```

Description

This is macro SDP_ATTRID_DIVendorIdSource.

SDP_ATTRID_DIVersion Macro

File

sdp.h

C

```
#define SDP_ATTRID_DIVersion 0x203
```

Description

This is macro SDP_ATTRID_DIVersion.

SDP_ATTRID_DocumentationURL Macro

File

sdp.h

C

```
#define SDP_ATTRID_DocumentationURL 0x000A
```

Description

This is macro SDP_ATTRID_DocumentationURL.

SDP_ATTRID_GAPRemoteAudioVolumeControl Macro

File

sdp.h

C

```
#define SDP_ATTRID_GAPRemoteAudioVolumeControl 0x302
```

Description

This is macro SDP_ATTRID_GAPRemoteAudioVolumeControl.

SDP_ATTRID_GroupID Macro

File

sdp.h

C

```
#define SDP_ATTRID_GroupID 0x0200
```

Description

This is macro SDP_ATTRID_GroupID.

SDP_ATTRID_HDPDataExchangeSpecification Macro

File

sdp.h

C

```
#define SDP_ATTRID_HDPDataExchangeSpecification 0x301
```

Description

This is macro SDP_ATTRID_HDPDataExchangeSpecification.

SDP_ATTRID_HDPMCAPSupportedProcedures Macro

File

sdp.h

C

```
#define SDP_ATTRID_HDPMCAPSupportedProcedures 0x302
```

Description

This is macro SDP_ATTRID_HDPMCAPSupportedProcedures.

SDP_ATTRID_HDPSuportedFeatures Macro

File

sdp.h

C

```
#define SDP_ATTRID_HDPSuportedFeatures 0x200
```

Description

This is macro SDP_ATTRID_HDPSuportedFeatures.

SDP_ATTRID_HFPAGNetwork Macro

File

sdp.h

C

```
#define SDP_ATTRID_HFPAGNetwork 0x301
```

Description

This is macro SDP_ATTRID_HFPAGNetwork.

SDP_ATTRID_HFPSupportedFeatures Macro

File

sdp.h

C

```
#define SDP_ATTRID_HFPSupportedFeatures 0x311
```

Description

This is macro SDP_ATTRID_HFPSupportedFeatures.

SDP_ATTRID_HIDBatteryPower Macro

File

sdp.h

C

```
#define SDP_ATTRID_HIDBatteryPower 0x209
```

Description

This is macro SDP_ATTRID_HIDBatteryPower.

SDP_ATTRID_HIDBootDevice Macro

File

sdp.h

C

```
#define SDP_ATTRID_HIDBootDevice 0x20E
```

Description

This is macro SDP_ATTRID_HIDBootDevice.

SDP_ATTRID_HIDCountryCode Macro

File

sdp.h

C

```
#define SDP_ATTRID_HIDCountryCode 0x203
```

Description

This is macro SDP_ATTRID_HIDCountryCode.

SDP_ATTRID_HIDDescriptorList Macro

File

sdp.h

C

```
#define SDP_ATTRID_HIDDescriptorList 0x206
```

Description

This is macro SDP_ATTRID_HIDDescriptorList.

SDP_ATTRID_HIDDeviceReleaseNumber Macro

File

sdp.h

C

```
#define SDP_ATTRID_HIDDeviceReleaseNumber 0x200
```

Description

This is macro SDP_ATTRID_HIDDeviceReleaseNumber.

SDP_ATTRID_HIDDeviceSubclass Macro

File

sdp.h

C

```
#define SDP_ATTRID_HIDDeviceSubclass 0x202
```

Description

This is macro SDP_ATTRID_HIDDeviceSubclass.

SDP_ATTRID_HIDLANGIDBaseList Macro

File

sdp.h

C

```
#define SDP_ATTRID_HIDLANGIDBaseList 0x207
```

Description

This is macro SDP_ATTRID_HIDLANGIDBaseList.

SDP_ATTRID_HIDNormallyConnectable Macro

File

sdp.h

C

```
#define SDP_ATTRID_HIDNormallyConnectable 0x20D
```

Description

This is macro SDP_ATTRID_HIDNormallyConnectable.

SDP_ATTRID_HIDParserVersion Macro

File

sdp.h

C

```
#define SDP_ATTRID_HIDParserVersion 0x201
```

Description

This is macro SDP_ATTRID_HIDParserVersion.

SDP_ATTRID_HIDProfileVersion Macro

File

sdp.h

C

```
#define SDP_ATTRID_HIDProfileVersion 0x20B
```

Description

This is macro SDP_ATTRID_HIDProfileVersion.

SDP_ATTRID_HIDReconnectInitiate Macro

File

sdp.h

C

```
#define SDP_ATTRID_HIDReconnectInitiate 0x205
```

Description

This is macro SDP_ATTRID_HIDReconnectInitiate.

SDP_ATTRID_HIDRemoteWake Macro

File

sdp.h

C

```
#define SDP_ATTRID_HIDRemoteWake 0x20A
```

Description

This is macro SDP_ATTRID_HIDRemoteWake.

SDP_ATTRID_HIDSDPDisable Macro

File

sdp.h

C

```
#define SDP_ATTRID_HIDSDPDisable 0x208
```

Description

This is macro SDP_ATTRID_HIDSDPDisable.

SDP_ATTRID_HIDSupervisionTimeout Macro

File

sdp.h

C

```
#define SDP_ATTRID_HIDSupervisionTimeout 0x20C
```

Description

This is macro SDP_ATTRID_HIDSupervisionTimeout.

SDP_ATTRID_HIDVirtualCable Macro

File

sdp.h

C

```
#define SDP_ATTRID_HIDVirtualCable 0x204
```

Description

This is macro SDP_ATTRID_HIDVirtualCable.

SDP_ATTRID_IconURL Macro

File

sdp.h

C

```
#define SDP_ATTRID_IconURL 0x000C
```

Description

This is macro SDP_ATTRID_IconURL.

SDP_ATTRID_INVALID Macro

File

sdp.h

C

```
#define SDP_ATTRID_INVALID 0xFFFF
```

Description

This is macro SDP_ATTRID_INVALID.

SDP_ATTRID_LanguageBaseAttributeIDList Macro

File

sdp.h

C

```
#define SDP_ATTRID_LanguageBaseAttributeIDList 0x0006
```

Description

This is macro SDP_ATTRID_LanguageBaseAttributeIDList.

SDP_ATTRID_OFFSET_ProviderName Macro

File

sdp.h

C

```
#define SDP_ATTRID_OFFSET_ProviderName 0x0002
```

Description

This is macro SDP_ATTRID_OFFSET_ProviderName.

SDP_ATTRID_OFFSET_ServiceDescription Macro

File

sdp.h

C

```
#define SDP_ATTRID_OFFSET_ServiceDescription 0x0001
```

Description

This is macro SDP_ATTRID_OFFSET_ServiceDescription.

SDP_ATTRID_OFFSET_ServiceName Macro

File

sdp.h

C

```
#define SDP_ATTRID_OFFSET_ServiceName 0x0000
```

Description

This is macro SDP_ATTRID_OFFSET_ServiceName.

SDP_ATTRID_PrimaryLanguageBaseId Macro

File

sdp.h

C

```
#define SDP_ATTRID_PrimaryLanguageBaseId 0x0100
```

Description

This is macro SDP_ATTRID_PrimaryLanguageBaseId.

SDP_ATTRID_ProtocolDescriptorList Macro

File

sdp.h

C

```
#define SDP_ATTRID_ProtocolDescriptorList 0x0004
```

Description

This is macro SDP_ATTRID_ProtocolDescriptorList.

SDP_ATTRID_ServiceAvailability Macro

File

sdp.h

C

```
#define SDP_ATTRID_ServiceAvailability 0x0008
```

Description

This is macro SDP_ATTRID_ServiceAvailability.

SDP_ATTRID_ServiceClassIDList Macro

File

sdp.h

C

```
#define SDP_ATTRID_ServiceClassIDList 0x0001
```

Description

This is macro SDP_ATTRID_ServiceClassIDList.

SDP_ATTRID_ServiceDatabaseState Macro

File

sdp.h

C

```
#define SDP_ATTRID_ServiceDatabaseState 0x0201
```

Description

This is macro SDP_ATTRID_ServiceDatabaseState.

SDP_ATTRID_ServiceID Macro

File

sdp.h

C

```
#define SDP_ATTRID_ServiceID 0x0003
```

Description

This is macro SDP_ATTRID_ServiceID.

SDP_ATTRID_ServiceInfoTimeToLive Macro

File

sdp.h

C

```
#define SDP_ATTRID_ServiceInfoTimeToLive 0x0007
```

Description

This is macro SDP_ATTRID_ServiceInfoTimeToLive.

SDP_ATTRID_ServiceRecordHandle Macro

File

sdp.h

C

```
#define SDP_ATTRID_ServiceRecordHandle 0x0000
```

Description

This is macro SDP_ATTRID_ServiceRecordHandle.

SDP_ATTRID_ServiceRecordState Macro

File

sdp.h

C

```
#define SDP_ATTRID_ServiceRecordState 0x0002
```

Description

This is macro SDP_ATTRID_ServiceRecordState.

SDP_ATTRID_SupportedFeatures Macro

File

sdp.h

C

```
#define SDP_ATTRID_SupportedFeatures 0x311
```

Description

This is macro SDP_ATTRID_SupportedFeatures.

SDP_ATTRID_VersionNumberList Macro

File

sdp.h

C

```
#define SDP_ATTRID_VersionNumberList 0x0200
```

Description

This is macro SDP_ATTRID_VersionNumberList.

SDP_CLIENT_EVT_CONNECTED Macro

File

sdp_client.h

C

```
#define SDP_CLIENT_EVT_CONNECTED 1
```

Description

This is macro SDP_CLIENT_EVT_CONNECTED.

SDP_CLIENT_EVT_DISCONNECTED Macro

File

sdp_client.h

C

```
#define SDP_CLIENT_EVT_DISCONNECTED 2
```

Description

This is macro SDP_CLIENT_EVT_DISCONNECTED.

SDP_CLIENT_EVT_NULL Macro

File

sdp_client.h

C

```
#define SDP_CLIENT_EVT_NULL 0
```

Description

This is macro SDP_CLIENT_EVT_NULL.

SDP_CLSID_ADVANCED_AUDIO_DISTRIBUTION Macro

File

sdp.h

C

```
#define SDP_CLSID_ADVANCED_AUDIO_DISTRIBUTION 0x110D
```

Description

This is macro SDP_CLSID_ADVANCED_AUDIO_DISTRIBUTION.

SDP_CLSID_AUDIO_SINK Macro

File

sdp.h

C

```
#define SDP_CLSID_AUDIO_SINK 0x110B
```

Description

This is macro SDP_CLSID_AUDIO_SINK.

SDP_CLSID_AUDIO_SOURCE Macro

File

sdp.h

C

```
#define SDP_CLSID_AUDIO_SOURCE 0x110A
```

Description

This is macro SDP_CLSID_AUDIO_SOURCE.

SDP_CLSID_AV_REMOTE_CONTROL Macro

File

sdp.h

C

```
#define SDP_CLSID_AV_REMOTE_CONTROL 0x110E
```

Description

This is macro SDP_CLSID_AV_REMOTE_CONTROL.

SDP_CLSID_AV_REMOTE_CONTROL_CONTROLLER Macro

File

sdp.h

C

```
#define SDP_CLSID_AV_REMOTE_CONTROL_CONTROLLER 0x110F
```

Description

This is macro SDP_CLSID_AV_REMOTE_CONTROL_CONTROLLER.

SDP_CLSID_AV_REMOTE_CONTROL_PROFILE_ID Macro

File

sdp.h

C

```
#define SDP_CLSID_AV_REMOTE_CONTROL_PROFILE_ID 0x110E
```

Description

This is macro SDP_CLSID_AV_REMOTE_CONTROL_PROFILE_ID.

SDP_CLSID_AV_REMOTE_CONTROL_TARGET Macro

File

sdp.h

C

```
#define SDP_CLSID_AV_REMOTE_CONTROL_TARGET 0x110C
```

Description

This is macro SDP_CLSID_AV_REMOTE_CONTROL_TARGET.

SDP_CLSID_AVCTP Macro

File

sdp.h

C

```
#define SDP_CLSID_AVCTP 0x17
```

Description

This is macro SDP_CLSID_AVCTP.

SDP_CLSID_AVDTP Macro

File

sdp.h

C

```
#define SDP_CLSID_AVDTP 0x19
```

Description

This is macro SDP_CLSID_AVDTP.

SDP_CLSID_BrowseGroupDescriptorServiceClassID Macro

File

sdp.h

C

```
#define SDP_CLSID_BrowseGroupDescriptorServiceClassID 0x1001
```

Description

This is macro SDP_CLSID_BrowseGroupDescriptorServiceClassID.

SDP_CLSID_DialupNetworking Macro

File

sdp.h

C

```
#define SDP_CLSID_DialupNetworking 0x1103
```

Description

This is macro SDP_CLSID_DialupNetworking.

SDP_CLSID_GENERIC_AUDIO Macro

File

sdp.h

C

```
#define SDP_CLSID_GENERIC_AUDIO 0x1203
```

Description

This is macro SDP_CLSID_GENERIC_AUDIO.

SDP_CLSID_HDP Macro

File

sdp.h

C

```
#define SDP_CLSID_HDP 0x1400
```

Description

This is macro SDP_CLSID_HDP.

SDP_CLSID_HDP_SINK Macro

File

sdp.h

C

```
#define SDP_CLSID_HDP_SINK 0x1402
```

Description

This is macro SDP_CLSID_HDP_SINK.

SDP_CLSID_HDP_SOURCE Macro

File

sdp.h

C

```
#define SDP_CLSID_HDP_SOURCE 0x1401
```

Description

This is macro SDP_CLSID_HDP_SOURCE.

SDP_CLSID_HFP Macro

File

sdp.h

C

```
#define SDP_CLSID_HFP 0x111E
```

Description

This is macro SDP_CLSID_HFP.

SDP_CLSID_HFP_AG Macro

File

sdp.h

C

```
#define SDP_CLSID_HFP_AG 0x111F
```

Description

This is macro SDP_CLSID_HFP_AG.

SDP_CLSID_HID Macro

File

sdp.h

C

```
#define SDP_CLSID_HID 0x1124
```

Description

This is macro SDP_CLSID_HID.

SDP_CLSID_HIDProtocol Macro

File

sdp.h

C

```
#define SDP_CLSID_HIDProtocol 0x11
```

Description

This is macro SDP_CLSID_HIDProtocol.

SDP_CLSID_HSP Macro

File

sdp.h

C

```
#define SDP_CLSID_HSP 0x1108
```

Description

This is macro SDP_CLSID_HSP.

SDP_CLSID_HSP_AG Macro

File

sdp.h

C

```
#define SDP_CLSID_HSP_AG 0x1112
```

Description

This is macro SDP_CLSID_HSP_AG.

SDP_CLSID_HSP_HS Macro

File

sdp.h

C

```
#define SDP_CLSID_HSP_HS 0x1131
```

Description

This is macro SDP_CLSID_HSP_HS.

SDP_CLSID_L2CAP Macro

File

sdp.h

C

```
#define SDP_CLSID_L2CAP 0x100
```

Description

This is macro SDP_CLSID_L2CAP.

SDP_CLSID_MCAP_CONTROL Macro

File

sdp.h

C

```
#define SDP_CLSID_MCAP_CONTROL 0x001E
```

Description

This is macro SDP_CLSID_MCAP_CONTROL.

SDP_CLSID_MCAP_DATA Macro

File

sdp.h

C

```
#define SDP_CLSID_MCAP_DATA 0x001F
```

Description

This is macro SDP_CLSID_MCAP_DATA.

SDP_CLSID_OBEXFileTransfer Macro

File

sdp.h

C

```
#define SDP_CLSID_OBEXFileTransfer 0x1106
```

Description

This is macro SDP_CLSID_OBEXFileTransfer.

SDP_CLSID_OBEXObjectPush Macro

File

sdp.h

C

```
#define SDP_CLSID_OBEXObjectPush 0x1105
```

Description

This is macro SDP_CLSID_OBEXObjectPush.

SDP_CLSID_PBAP_PCE Macro

File

sdp.h

C

```
#define SDP_CLSID_PBAP_PCE 0x112E
```

Description

This is macro SDP_CLSID_PBAP_PCE.

SDP_CLSID_PBAP_PSE Macro

File

sdp.h

C

```
#define SDP_CLSID_PBAP_PSE 0x112F
```

Description

This is macro SDP_CLSID_PBAP_PSE.

SDP_CLSID_PNPInformation Macro

File

sdp.h

C

```
#define SDP_CLSID_PNPInformation 0x1200
```

Description

This is macro SDP_CLSID_PNPInformation.

SDP_CLSID_PublicBrowseGroup Macro

File

sdp.h

C

```
#define SDP_CLSID_PublicBrowseGroup 0x1002
```

Description

This is macro SDP_CLSID_PublicBrowseGroup.

SDP_CLSID_RFCOMM Macro

File

sdp.h

C

```
#define SDP_CLSID_RFCOMM 0x3
```

Description

This is macro SDP_CLSID_RFCOMM.

SDP_CLSID_SerialPort Macro

File

sdp.h

C

```
#define SDP_CLSID_SerialPort 0x1101
```

Description

This is macro SDP_CLSID_SerialPort.

SDP_CLSID_ServiceDiscoveryServerServiceClassID Macro

File

sdp.h

C

```
#define SDP_CLSID_ServiceDiscoveryServerServiceClassID 0x1000
```

Description

defgroup sdp SDP

This module describe functions and data structures used to start the SDP server and perform SDP queries.

SDP_DATA_TYPE_ALTERNATIVE Macro

File

sdp.h

C

```
#define SDP_DATA_TYPE_ALTERNATIVE 7
```

Description

This is macro SDP_DATA_TYPE_ALTERNATIVE.

SDP_DATA_TYPE_BOOL Macro

File

sdp.h

C

```
#define SDP_DATA_TYPE_BOOL 5
```

Description

This is macro SDP_DATA_TYPE_BOOL.

SDP_DATA_TYPE_INT Macro

File

sdp.h

C

```
#define SDP_DATA_TYPE_INT 2
```

Description

This is macro SDP_DATA_TYPE_INT.

SDP_DATA_TYPE_INT128 Macro

File

sdp.h

C

```
#define SDP_DATA_TYPE_INT128 0x82
```

Description

This is macro SDP_DATA_TYPE_INT128.

SDP_DATA_TYPE_INT16 Macro

File

sdp.h

C

```
#define SDP_DATA_TYPE_INT16 0x12
```

Description

This is macro SDP_DATA_TYPE_INT16.

SDP_DATA_TYPE_INT32 Macro

File

sdp.h

C

```
#define SDP_DATA_TYPE_INT32 0x22
```

Description

This is macro SDP_DATA_TYPE_INT32.

SDP_DATA_TYPE_INT64 Macro

File

sdp.h

C

```
#define SDP_DATA_TYPE_INT64 0x42
```

Description

This is macro SDP_DATA_TYPE_INT64.

SDP_DATA_TYPE_INT8 Macro

File

sdp.h

C

```
#define SDP_DATA_TYPE_INT8 SDP_DATA_TYPE_INT
```

Description

This is macro SDP_DATA_TYPE_INT8.

SDP_DATA_TYPE_NIL Macro

File

sdp.h

C

```
#define SDP_DATA_TYPE_NIL 0
```

Description

This is macro SDP_DATA_TYPE_NIL.

SDP_DATA_TYPE_SEQUENCE Macro

File

sdp.h

C

```
#define SDP_DATA_TYPE_SEQUENCE 6
```

Description

This is macro SDP_DATA_TYPE_SEQUENCE.

SDP_DATA_TYPE_STRING Macro

File

sdp.h

C

```
#define SDP_DATA_TYPE_STRING 4
```

Description

This is macro SDP_DATA_TYPE_STRING.

SDP_DATA_TYPE_UINT Macro

File

sdp.h

C

```
#define SDP_DATA_TYPE_UINT 1
```

Description

This is macro SDP_DATA_TYPE_UINT.

SDP_DATA_TYPE_UINT128 Macro

File

sdp.h

C

```
#define SDP_DATA_TYPE_UINT128 0x81
```

Description

This is macro SDP_DATA_TYPE_UINT128.

SDP_DATA_TYPE_UINT16 Macro

File

sdp.h

C

```
#define SDP_DATA_TYPE_UINT16 0x11
```

Description

This is macro SDP_DATA_TYPE_UINT16.

SDP_DATA_TYPE_UINT32 Macro

File

sdp.h

C

```
#define SDP_DATA_TYPE_UINT32 0x21
```

Description

This is macro SDP_DATA_TYPE_UINT32.

SDP_DATA_TYPE_UINT64 Macro

File

sdp.h

C

```
#define SDP_DATA_TYPE_UINT64 0x41
```

Description

This is macro SDP_DATA_TYPE_UINT64.

SDP_DATA_TYPE_UINT8 Macro

File

sdp.h

C

```
#define SDP_DATA_TYPE_UINT8 SDP_DATA_TYPE_UINT
```

Description

This is macro SDP_DATA_TYPE_UINT8.

SDP_DATA_TYPE_URL Macro

File

sdp.h

C

```
#define SDP_DATA_TYPE_URL 8
```

Description

This is macro SDP_DATA_TYPE_URL.

SDP_DATA_TYPE_UUID Macro

File

sdp.h

C

```
#define SDP_DATA_TYPE_UUID 3
```

Description

This is macro SDP_DATA_TYPE_UUID.

SDP_DATA_TYPE_UUID128 Macro

File

sdp.h

C

```
#define SDP_DATA_TYPE_UUID128 0x23
```

Description

This is macro SDP_DATA_TYPE_UUID128.

SDP_DATA_TYPE_UUID16 Macro

File

sdp.h

C

```
#define SDP_DATA_TYPE_UUID16 SDP_DATA_TYPE_UUID
```

Description

This is macro SDP_DATA_TYPE_UUID16.

SDP_DATA_TYPE_UUID32 Macro

File

sdp.h

C

```
#define SDP_DATA_TYPE_UUID32 0x13
```

Description

This is macro SDP_DATA_TYPE_UUID32.

SDP_ERROR_INSUFFICIENT_RESOURCE Macro

File

sdp.h

C

```
#define SDP_ERROR_INSUFFICIENT_RESOURCE 0x06
```

Description

This is macro SDP_ERROR_INSUFFICIENT_RESOURCE.

SDP_ERROR_INVALID_CONTINUATION_STATE Macro

File

sdp.h

C

```
#define SDP_ERROR_INVALID_CONTINUATION_STATE 0x05
```

Description

This is macro SDP_ERROR_INVALID_CONTINUATION_STATE.

SDP_ERROR_INVALID_PDU_SIZE Macro

File

sdp.h

C

```
#define SDP_ERROR_INVALID_PDU_SIZE 0x04
```

Description

This is macro SDP_ERROR_INVALID_PDU_SIZE.

SDP_ERROR_INVALID_REQUEST_SYNTAX Macro

File

sdp.h

C

```
#define SDP_ERROR_INVALID_REQUEST_SYNTAX 0x03
```

Description

This is macro SDP_ERROR_INVALID_REQUEST_SYNTAX.

SDP_ERROR_INVALID_SDP_VERSION Macro

File

sdp.h

C

```
#define SDP_ERROR_INVALID_SDP_VERSION 0x01
```

Description

This is macro SDP_ERROR_INVALID_SDP_VERSION.

SDP_ERROR_INVALID_SR_HANDLE Macro

File

sdp.h

C

```
#define SDP_ERROR_INVALID_SR_HANDLE 0x02
```

Description

This is macro SDP_ERROR_INVALID_SR_HANDLE.

SDP_ERROR_RESERVED Macro

File

sdp.h

C

```
#define SDP_ERROR_RESERVED 0x00
```

Description

This is macro SDP_ERROR_RESERVED.

SDP_ErrorResponse Macro

File

sdp.h

C

```
#define SDP_ErrorResponse 0x01
```

Description

This is macro SDP_ErrorResponse.

SDP_FTP_SERVICE_ID Macro

File

sdp.h

C

```
#define SDP_FTP_SERVICE_ID 0x1237
```

Description

This is macro SDP_FTP_SERVICE_ID.

SDP_HID_SERVICE_ID Macro

File

sdp.h

C

```
#define SDP_HID_SERVICE_ID 0x1235
```

Description

This is macro SDP_HID_SERVICE_ID.

SDP_HSP_AG_SERVICE_ID Macro

File

sdp.h

C

```
#define SDP_HSP_AG_SERVICE_ID 0x1236
```

Description

This is macro SDP_HSP_AG_SERVICE_ID.

SDP_HSP_HS_SERVICE_ID Macro

File

sdp.h

C

```
#define SDP_HSP_HS_SERVICE_ID 0x1236
```

Description

This is macro SDP_HSP_HS_SERVICE_ID.

SDP_MAX_ATTRIBUTE_PATTERN_LEN Macro

File

sdp.h

C

```
#define SDP_MAX_ATTRIBUTE_PATTERN_LEN 12
```

Description

This is macro SDP_MAX_ATTRIBUTE_PATTERN_LEN.

SDP_MAX_DATA_ELEMENT_LEN Macro

File

sdp.h

C

```
#define SDP_MAX_DATA_ELEMENT_LEN 256
```

Description

This is macro SDP_MAX_DATA_ELEMENT_LEN.

SDP_MAX_DATA_ELEMENTS Macro

File

sdp.h

C

```
#define SDP_MAX_DATA_ELEMENTS 10
```

Description

This is macro SDP_MAX_DATA_ELEMENTS.

SDP_MAX_SEARCH_PATTERN_LEN Macro

File

sdp.h

C

```
#define SDP_MAX_SEARCH_PATTERN_LEN 12
```

Description

This is macro SDP_MAX_SEARCH_PATTERN_LEN.

SDP_MAX_TRANSACTIONS Macro

File

sdp.h

C

```
#define SDP_MAX_TRANSACTIONS 2
```

Description

typedef struct _sdp_db { bt_int count; bt_sdp_service_record_p *records; } bt_sdp_db, *bt_sdp_db_p;

SDP_PDU_HEADER_LEN Macro

File

sdp.h

C

```
#define SDP_PDU_HEADER_LEN 5
```

Description

This is macro SDP_PDU_HEADER_LEN.

SDP_RFCOMM_SERVICE_ID Macro

File

sdp.h

C

```
#define SDP_RFCOMM_SERVICE_ID 0x1234
```

Description

This is macro SDP_RFCOMM_SERVICE_ID.

SDP_ServiceAttributeRequest Macro

File

sdp.h

C

```
#define SDP_ServiceAttributeRequest 0x04
```

Description

This is macro SDP_ServiceAttributeRequest.

SDP_ServiceAttributeResponse Macro

File

sdp.h

C

```
#define SDP_ServiceAttributeResponse 0x05
```

Description

This is macro SDP_ServiceAttributeResponse.

SDP_ServiceSearchAttributeRequest Macro

File

sdp.h

C

```
#define SDP_ServiceSearchAttributeRequest 0x06
```

Description

This is macro SDP_ServiceSearchAttributeRequest.

SDP_ServiceSearchAttributeResponse Macro

File

sdp.h

C

```
#define SDP_ServiceSearchAttributeResponse 0x07
```

Description

This is macro SDP_ServiceSearchAttributeResponse.

SDP_ServiceSearchRequest Macro

File

sdp.h

C

```
#define SDP_ServiceSearchRequest 0x02
```

Description

This is macro SDP_ServiceSearchRequest.

SDP_ServiceSearchResponse Macro

File

sdp.h

C

```
#define SDP_ServiceSearchResponse 0x03
```

Description

This is macro SDP_ServiceSearchResponse.

SDP_SR_HANDLE_HDP_SINK Macro

File

sdp.h

C

```
#define SDP_SR_HANDLE_HDP_SINK 0x10009
```

Description

This is macro SDP_SR_HANDLE_HDP_SINK.

SDP_SR_HANDLE_HDP_SOURCE Macro

File

sdp.h

C

```
#define SDP_SR_HANDLE_HDP_SOURCE 0x10008
```

Description

This is macro SDP_SR_HANDLE_HDP_SOURCE.

SDP_SR_HANDLE_HFP_HF Macro

File

sdp.h

C

```
#define SDP_SR_HANDLE_HFP_HF 0x1000A
```

Description

This is macro SDP_SR_HANDLE_HFP_HF.

SDP_SR_HANDLE_HID Macro

File

sdp.h

C

```
#define SDP_SR_HANDLE_HID 0x10001
```

Description

This is macro SDP_SR_HANDLE_HID.

SDP_SR_HANDLE_HID_KEYBOARD Macro

File

sdp.h

C

```
#define SDP_SR_HANDLE_HID_KEYBOARD 0x10005
```

Description

This is macro SDP_SR_HANDLE_HID_KEYBOARD.

SDP_SR_HANDLE_HSP_HS Macro

File

sdp.h

C

```
#define SDP_SR_HANDLE_HSP_HS 0x10004
```

Description

This is macro SDP_SR_HANDLE_HSP_HS.

SDP_SR_HANDLE_OBEXFileTransfer Macro

File

sdp.h

C

```
#define SDP_SR_HANDLE_OBEXFileTransfer 0x10006
```

Description

This is macro SDP_SR_HANDLE_OBEXFileTransfer.

SDP_SR_HANDLE_OBEXObjectPush Macro

File

sdp.h

C

```
#define SDP_SR_HANDLE_OBEXObjectPush 0x10007
```

Description

This is macro SDP_SR_HANDLE_OBEXObjectPush.

SDP_SR_HANDLE_PNPINFORMATION Macro

File

sdp.h

C

```
#define SDP_SR_HANDLE_PNPINFORMATION 0x10002
```

Description

This is macro SDP_SR_HANDLE_PNPINFORMATION.

SDP_SR_HANDLE_RFCOMM Macro

File

sdp.h

C

```
#define SDP_SR_HANDLE_RFCOMM 0x10000
```

Description

This is macro SDP_SR_HANDLE_RFCOMM.

SDP_SR_HANDLE_SERVER Macro

File

sdp.h

C

```
#define SDP_SR_HANDLE_SERVER 0x0
```

Description

This is macro SDP_SR_HANDLE_SERVER.

SDP_SR_HANDLE_TEST Macro

File

sdp.h

C

```
#define SDP_SR_HANDLE_TEST 0x10003 // for simulating service search using continuation state
```

Description

for simulating service search using continuation state

bt_sdp_client_evt_connected_t Structure

File

sdp_client.h

C

```
struct bt_sdp_client_evt_connected_t {
  bt_l2cap_channel_t* channel;
};
```

Description

This is record bt_sdp_client_evt_connected_t.

bt_sdp_client_evt_disconnected_t Structure

File

sdp_client.h

C

```
struct bt_sdp_client_evt_disconnected_t {
  bt_l2cap_channel_t* channel;
};
```

Description

This is record bt_sdp_client_evt_disconnected_t.

bt_sdp_client_callback_fp Type

File

sdp_client.h

C

```
typedef void (* bt_sdp_client_callback_fp)(bt_byte evt, void* evt_param, void* param);
```

Description

This is type bt_sdp_client_callback_fp.

bt_sdp_data_element_cp Type

File

sdp.h

C

```
typedef const bt_sdp_data_element_t* bt_sdp_data_element_cp;
```

Description

This is type bt_sdp_data_element_cp.

bt_sdp_data_element_p Type

File

sdp.h

C

```
typedef bt_sdp_data_element_t* bt_sdp_data_element_p;
```

Description

This is type bt_sdp_data_element_p.

bt_sdp_data_element_t Structure

File

sdp.h

C

```
typedef struct _bt_sdp_data_element_t {
  bt_int type;
  bt_int bytecount;
  union {
    bt_ulong init;
    void* pdata;
    char* pstr;
    char* purl;
    struct _bt_sdp_sequence_t* pseq;
    bt_byte b;
    bt_int i;
    bt_uint ui;
    bt_long l;
    bt_ulong ul;
    bt_uuid16 uuid16;
    bt_uuid32 uuid32;
    bt_uuid_p uuid128;
  } data;
} bt_sdp_data_element_t;
```

Description

This is type bt_sdp_data_element_t.

bt_sdp_found_attr_list_t Structure

File

sdp.h

C

```
typedef struct _bt_sdp_found_attr_list_t {
  bt_int found_attr_count;
  const bt_byte** found_attr_list;
} bt_sdp_found_attr_list_t;
```

Members

Members	Description
const bt_byte** found_attr_list;	SDP_MAX_ATTRIBUTE_RESULT_LEN];

Description

This is type bt_sdp_found_attr_list_t.

bt_sdp_packet_t Structure

File

sdp_packet.h

C

```
typedef struct _bt_sdp_packet_t {
  bt_packet_t header;
  bt_byte sdu_type;
  union {
    union {
      struct _bt_sdp_service_transaction_t* service;
      struct _bt_sdp_transaction_t* attribute;
    } transaction;
    struct {
      bt_int trans_id;
      bt_int err_code;
    } error;
    struct {
      bt_int trans_id;
      struct _bt_sdp_data_element_t* pattern;
      bt_int uuid_index;
      bt_int uuid_pos;
    } service_search;
    struct {
      bt_int trans_id;
      struct _bt_sdp_data_element_t* pattern;
      bt_long sr;
    } attribute_search;
  } data;
} bt_sdp_packet_t;
```

Description

This is type bt_sdp_packet_t.

bt_sdp_read_de_callback_ftp Type

File

sdp_client.h

C

```
typedef bt_bool (* bt_sdp_read_de_callback_ftp)(bt_sdp_data_element_p pde, bt_bool seq_start);
```

Description

This is type bt_sdp_read_de_callback_ftp.

bt_sdp_sequence_cp Type

File

sdp.h

C

```
typedef const bt_sdp_sequence_t* bt_sdp_sequence_cp;
```

Description

This is type bt_sdp_sequence_cp.

bt_sdp_sequence_p Type

File

sdp.h

C

```
typedef bt_sdp_sequence_t* bt_sdp_sequence_p;
```

Description

This is type bt_sdp_sequence_p.

bt_sdp_sequence_t Structure

File

sdp.h

C

```
typedef struct _bt_sdp_sequence_t {
  bt_int count;
  bt_sdp_data_element_p elements;
} bt_sdp_sequence_t;
```

Description

This is type bt_sdp_sequence_t.

bt_sdp_serialization_state_p Structure

File

sdp.h

C

```
typedef struct _bt_sdp_serialization_state_t {
  bt_int listIndex;
  bt_int attrIndex;
  bt_byte attrPart;
  bt_int attrPartPos;
  bt_int attr_bytes;
  bt_int first_list;
  bt_int last_list;
  bt_int first_attr;
  bt_int last_attr;
  bt_uint attr_lists_seq_len;
  bt_bool write_seq_len;
} bt_sdp_serialization_state_t, * bt_sdp_serialization_state_p;
```

Description

This is type bt_sdp_serialization_state_p.

bt_sdp_serialization_state_t Structure

File

sdp.h

C

```
typedef struct _bt_sdp_serialization_state_t {
  bt_int listIndex;
  bt_int attrIndex;
  bt_byte attrPart;
  bt_int attrPartPos;
  bt_int attr_bytes;
  bt_int first_list;
  bt_int last_list;
  bt_int first_attr;
  bt_int last_attr;
  bt_uint attr_lists_seq_len;
  bt_bool write_seq_len;
} bt_sdp_serialization_state_t, * bt_sdp_serialization_state_p;
```

Description

This is type bt_sdp_serialization_state_t.

bt_sdp_server_attribute_t Structure

File

sdp_private.h

C

```
typedef struct _bt_sdp_server_attribute_t {
  bt_uint attr_id;
  bt_uint data_len;
  const bt_byte* data;
  bt_sdp_server_data_element_t attr_value;
} bt_sdp_server_attribute_t;
```

Description

This is type bt_sdp_server_attribute_t.

bt_sdp_server_data_element_t Structure

File

sdp_private.h

C

```
typedef struct _bt_sdp_server_data_element_t {
  bt_byte type;
  bt_uint data_len;
  const bt_byte* data;
} bt_sdp_server_data_element_t;
```

Description

Private types

bt_sdp_server_record_t Structure

File

sdp_private.h

C

```
typedef struct _bt_sdp_server_record_t {
  bt_sr_handle_t h;
```

```
  bt_uint data_len;
  const bt_byte* data;
} bt_sdp_server_record_t;
```

Description

This is type bt_sdp_server_record_t.

bt_sdp_service_attribute_callback_fp Type

File

sdp_client.h

C

```
typedef bt_bool (* bt_sdp_service_attribute_callback_fp)(bt_uint attrid, bt_sdp_data_element_p pde, void*
param);
```

Description

This is type bt_sdp_service_attribute_callback_fp.

bt_sdp_service_search_callback_fp Type

File

sdp_client.h

C

```
typedef void (* bt_sdp_service_search_callback_fp)(bt_sr_handle_p shlist, bt_int count, void* param);
```

Description

This is type bt_sdp_service_search_callback_fp.

bt_sdp_service_transaction_p Structure

File

sdp.h

C

```
typedef struct _bt_sdp_service_transaction_t {
  bt_int id;
  bt_int found_sr_count;
  bt_sr_handle_t* found_sr_list;
  bt_uint max_sr_count;
  bt_int next_tran_index;
} bt_sdp_service_transaction_t, * bt_sdp_service_transaction_p;
```

Members

Members	Description
bt_sr_handle_t* found_sr_list;	SDP_MAX_SEARCH_RESULT_LEN];

Description

This is type bt_sdp_service_transaction_p.

bt_sdp_service_transaction_t Structure

File

sdp.h

C

```
typedef struct _bt_sdp_service_transaction_t {
  bt_int id;
  bt_int found_sr_count;
  bt_sr_handle_t* found_sr_list;
```

```
  bt_uint max_sr_count;
  bt_int next_tran_index;
} bt_sdp_service_transaction_t, * bt_sdp_service_transaction_p;
```

Members

Members	Description
bt_sr_handle_t* found_sr_list;	SDP_MAX_SEARCH_RESULT_LEN];

Description

This is type bt_sdp_service_transaction_t.

bt_sdp_transaction_t Structure

File

sdp.h

C

```
typedef struct _bt_sdp_transaction_t {
  bt_int id;
  bt_int found_list_count;
  bt_sdp_found_attr_list_t* found_attr_lists;
  bt_uint max_bytecount;
  bt_bool complete;
  bt_bool continuation;
  bt_sdp_serialization_state_t sr_state;
} bt_sdp_transaction_t;
```

Members

Members	Description
bt_sdp_found_attr_list_t* found_attr_lists;	SDP_MAX_SEARCH_RESULT_LEN];

Description

This is type bt_sdp_transaction_t.

bt_sr_handle_p Type

File

sdp.h

C

```
typedef bt_long * bt_sr_handle_p;
```

Description

This is type bt_sr_handle_p.

bt_sr_handle_t Type

File

sdp.h

C

```
typedef bt_long bt_sr_handle_t;
```

Description

This is type bt_sr_handle_t.

SDP_CLIENT_EVT_CONNECTION_FAILED Macro

File

sdp_client.h

C

```
#define SDP_CLIENT_EVT_CONNECTION_FAILED 3
```

Description

This is macro SDP_CLIENT_EVT_CONNECTION_FAILED.

SDP_ATTRID_HCRP_1284ID Macro

File

sdp.h

C

```
#define SDP_ATTRID_HCRP_1284ID 0x300
```

Description

This is macro SDP_ATTRID_HCRP_1284ID.

SDP_ATTRID_HCRP_DeviceLocation Macro

File

sdp.h

C

```
#define SDP_ATTRID_HCRP_DeviceLocation 0x306
```

Description

This is macro SDP_ATTRID_HCRP_DeviceLocation.

SDP_ATTRID_HCRP_DeviceName Macro

File

sdp.h

C

```
#define SDP_ATTRID_HCRP_DeviceName 0x302
```

Description

This is macro SDP_ATTRID_HCRP_DeviceName.

SDP_ATTRID_HCRP_FriendlyName Macro

File

sdp.h

C

```
#define SDP_ATTRID_HCRP_FriendlyName 0x304
```

Description

This is macro SDP_ATTRID_HCRP_FriendlyName.

SDP_CLSID_HARD_COPY_CABLE_REPLACEMENT Macro

File

sdp.h

C

```
#define SDP_CLSID_HARD_COPY_CABLE_REPLACEMENT 0x1125
```

Description

This is macro SDP_CLSID_HARD_COPY_CABLE_REPLACEMENT.

SDP_CLSID_HARD_COPY_CONTROL_CHANNEL Macro

File

sdp.h

C

```
#define SDP_CLSID_HARD_COPY_CONTROL_CHANNEL 0x0012
```

Description

This is macro SDP_CLSID_HARD_COPY_CONTROL_CHANNEL.

SDP_CLSID_HARD_COPY_DATA_CHANNEL Macro

File

sdp.h

C

```
#define SDP_CLSID_HARD_COPY_DATA_CHANNEL 0x0014
```

Description

This is macro SDP_CLSID_HARD_COPY_DATA_CHANNEL.

SDP_CLSID_HARD_COPY_NOTIFICATION Macro

File

sdp.h

C

```
#define SDP_CLSID_HARD_COPY_NOTIFICATION 0x0016
```

Description

This is macro SDP_CLSID_HARD_COPY_NOTIFICATION.

SDP_CLSID_HCR_PRINT Macro

File

sdp.h

C

```
#define SDP_CLSID_HCR_PRINT 0x1126
```

Description

This is macro SDP_CLSID_HCR_PRINT.

SDP_CLSID_HCR_SCAN Macro

File

sdp.h

C

```
#define SDP_CLSID_HCR_SCAN 0x1127
```

Description

This is macro SDP_CLSID_HCR_SCAN.

SPP Functions

bt_spp_allocate Function

File

spp.h

C

```
bt_spp_port_t* bt_spp_allocate(bt_l2cap_mgr_t* l2cap_mgr, bt_spp_state_callback_fp callback, void*
callback_param);
```

Description

- Allocate a serial port.

The returned serial port is initially in the ::SPP_PORT_STATE_DISCONNECTED state. To establish a connection with a remote device, call bt_spp_connect(). To listen for incoming connections from other devices, call bt_spp_listen(). The p callback parameter must specify a callback function that will be used to notify about serial port events and state changes. When the port is not needed any more it must be deallocated by bt_spp_deallocate(). The maximum number of serial ports that can be allocated simultaneously is specified by the ::SPP_MAX_PORTS configuration parameter.

@param l2cap_mgr L2CAP manager.

@param callback Pointer to a callback function used to notify about serial port events. Cannot be NULL.

@param callback_param An arbitrary pointer that is passed as a parameter to the callback function.

@return A pointer to the ::bt_spp_port_t structure. Returns NULL if the maximum number of ports has been already allocated or the c callback parameter is NULL.

bt_spp_cancel_receive Function

File

spp.h

C

```
void bt_spp_cancel_receive(bt_spp_port_t* port);
```

Description

- Cancel receive data.

If a receive operation is currently in progress this function will cancel it. After calling this function the receive callback specified in bt_spp_receive() will not be called.

If there is no receive operation in progress calling this function has no effect.

@param port Serial port.

bt_spp_cancel_send Function

File

spp.h

C

```
void bt_spp_cancel_send(bt_spp_port_t* port);
```

Description

- Cancel send data.

If a send operation is currently in progress this function will try to cancel it. When the operation is canceled the send callback function will be called with the ::SPP_SEND_STATUS_CANCELED status.

If this function is called but the active send operation completes successfuly before the stack can actually cancel it the call back function will still be called with the ::SPP_SEND_STATUS_CANCELED status.

If there is no send operation in progess calling this function has no effect.

@param port Serial port.

bt_spp_clr_port_options Function

File

spp.h

C

```
void bt_spp_clr_port_options(bt_spp_port_t* port, bt_uint options);
```

Description

This is function bt_spp_clr_port_options.

bt_spp_connect Function

File

spp.h

C

```
bt_bool bt_spp_connect(bt_spp_port_t* port, bt_bdaddr_p remote_addr, bt_byte channel);
```

Description

- Connect to a remote device.

This function initiates a connection to a remote device. When the connection is successfully established the port's callback is called with the ::SPP_PORT_EVENT_CONNECTED event. If connection fails the callback is called with the ::SPP_PORT_EVENT_CONNECTION_FAILED event.

The port must be in ::SPP_PORT_STATE_DISCONNECTED state. Otherwise, the function will fail.

@param port Serial port. @param remote_addr Bluetooth address of the remote device. @param channel RFCOMM server channel on which the connection is to be established.

@return c TRUE if successful, c FALSE otherwise.

bt_spp_deallocate Function

File

spp.h

C

```
bt_bool bt_spp_deallocate(bt_spp_port_t* port);
```

Description

- Deallocate serial port.

This function deallocates the specified port structure and other resources associated with it.

The port must be in ::SPP_PORT_STATE_DISCONNECTED state. Otherwise, the function will fail.

If the function completes successfully the application must not try to access any fields in the structure and must not use it with any other SPP functions. Also, it becomes available for subsequent allocation by bt_spp_port_allocate().

@param port Serial port structure to deallocate.

@return c TRUE if successful, c FALSE otherwise.

bt_spp_disconnect Function

File

spp.h

C

```
void bt_spp_disconnect(bt_spp_port_t* port);
```

Description

- Disconnect from the remote device.

This function initiates the disconnection process. When it is complete the the port's callback is called with the SPP_PORT_EVENT_DISCONNECTED event.

If the port is already in the disconnected state the function does nothing and the callback is not called.

@param port Serial port.

bt_spp_find_server Function

File

spp.h

C

```
bt_bool bt_spp_find_server(bt_bdaddr_t* deviceAddress, const bt_uuid_t* service_class_id,
bt_spp_find_server_callback_fp callback, void* callback_param);
```

Description

This is function bt_spp_find_server.

bt_spp_get_frame_length Function

File

spp.h

C

```
bt_int bt_spp_get_frame_length(bt_spp_port_t* port);
```

Description

- Get frame length.

This function returns the RFCOMM frame length used by the RFCOMM protocol. The frame length depends on configuration of DotStack and configuration of the Bluetooth stack running on the remote device. In order to achieve maximum throughput over the serial port connection the application should send and receive data in chunks that are multiple of this frame length.

@return RFCOMM frame length in bytes.

bt_spp_get_hci_connection Function

File

spp.h

C

```
bt_hci_conn_state_t* bt_spp_get_hci_connection(const bt_spp_port_t* port);
```

Description

- Get SPP port's ACL connection.

This function returns a pinter to the structure that describes the ACL connection this port is on.

@return Pointer to ACL connection description if the port is connected, NULL otherwise.

bt_spp_get_local_modem_status Function

File

spp.h

C

```
bt_byte bt_spp_get_local_modem_status(const bt_spp_port_t* port);
```

Description

- Get local device's TS 07.10 control signals.

This function returns current state of the local device's TS 07.10 controls signals. The signals are defined as a mask of the following constants: SPP_RS232_DSR SPP_RS232_RTS SPP_RS232_RI SPP_RS232_DCD

@param port Serial port.

@return local device's TS 07.10 control signals.

bt_spp_get_remote_address Function

File

spp.h

C

```
bt_bdaddr_t* bt_spp_get_remote_address(const bt_spp_port_t* port);
```

Description

brief Get the address of the remote device this device is connected to. ingroup spp

param port Serial port.

return li c A pointer to bt_bdaddr structure that contains the address of the remote device.

bt_spp_get_remote_modem_status Function

File

spp.h

C

```
bt_byte bt_spp_get_remote_modem_status(const bt_spp_port_t* port);
```

Description

- Get remote device's TS 07.10 control signals.

This function returns current state of the remote device's V.24 controls signals. The signals are defined as a mask of the following constants: SPP_RS232_DTR SPP_RS232_CTS SPP_RS232_RI SPP_RS232_DCD

@param port Serial port.

@return remote device's TS 07.10 control signals.

bt_spp_init Function

File

spp.h

C

```
void bt_spp_init();
```

Description

- Initialize the SPP module.

This function initializes all internal variables of the SPP module. It must be called prior to using any other functions in this module.

bt_spp_listen Function

File

spp.h

C

```
bt_bool bt_spp_listen(bt_spp_port_t* port, bt_byte channel);
```

Description

- Listen for incoming connections.

This function registers the port to accept incoming connections from remote devices on a particular RFCOMM server channel. The specified server channel should be listed in the SDP database. Otherwise, remote devices will not be able to find out which server channel to use.

When a remote device successfully establishes a connection on the specified port the port's callback is called with the ::SPP_PORT_EVENT_CONNECTED event.

The port must be in ::SPP_PORT_STATE_DISCONNECTED state. Otherwise, the function will fail.

@param port Serial port. @param channel The RFCOMM server channel on which to listen for connections.

@return c TRUE if successful, c FALSE otherwise.

bt_spp_receive Function

File

spp.h

C

```
bt_bool bt_spp_receive(bt_spp_port_t* port, void* buffer, bt_int buffer_len, bt_spp_receive_callback_fp callback);
```

Description

- Receive data.

This function receives data from the serial port connection. The caller must provide a buffer and a callback function. Whenever the port receives data they are copied to the provided buffer and the callback function is called. The callback function is passed the length of received data and the same callback parameter that was specified when the port was allocated with bt_spp_allocate(). This function does not wait until the buffer is filled out completely. Any amount of received data will complete the operation.

The port must be in ::SPP_PORT_STATE_CONNECTED state. Otherwise, the function will fail. Also, the function will fail if a previously started receive operation is still in progress.

@param port Serial port. @param data Pointer to the data to be sent. @param data_len Length of the data. @param callback Send callback function.

@return c TRUE if successful, c FALSE otherwise.

bt_spp_send Function

File

spp.h

C

```
bt_bool bt_spp_send(bt_spp_port_t* port, const void* data, bt_ulong data_len, bt_spp_send_callback_fp callback);
```

Description

- Send data.

This function starts sending data over the serial port connection. Along with the data the caller must provide a callback function that is called when all data has been sent. Also, during execution of this operation the port's state callback function is called periodically with the ::SPP_PORT_EVENT_SEND_PROGRESS event.

The port must be in ::SPP_PORT_STATE_CONNECTED state. Otherwise, the function will fail. Also, the function will fail if a previously started send operation is still in progress.

The callback function is passed the same callback parameter that was specified when the port was allocated with bt_spp_allocate().

@param port Serial port. @param data Pointer to the data to be sent. @param data_len Length of the data. @param callback Send callback function.

@return c TRUE if successful, c FALSE otherwise.

bt_spp_set_dtr Function

File

spp.h

C

```
bt_byte bt_spp_set_dtr(bt_spp_port_t* port, bt_bool on);
```

Description

- Set local device's RS-232 DTR signal.

Changes the state of local device's RS-232 DTR signal and notifies the remote device of the change. This signal corresponds to the TS 07.10's RFCOMM_MODEM_STATUS_RTC signal.

If there are resources to send a command to the remote device this command will return TRUE.

If the remote party has been successfully notified SPP_PORT_EVENT_LOCAL_MODEM_STATUS_CHANGED event will be reported. If the command could not be send to the remote party for any reason other than lack of resources SPP_PORT_EVENT_LOCAL_MODEM_STATUS_CHANGE_FAILED will be reported.

If the status cannot be changed because there is no resources to send a command to the remote device this function will return FALSE and no events will be reported.

@param port Serial port. @param on Value indicating weather to set or clear the signal.

@return c TRUE if successful, c FALSE otherwise.

bt_spp_set_local_modem_status Function

File

spp.h

C

```
bt_byte bt_spp_set_local_modem_status(bt_spp_port_t* port, bt_byte ms);
```

Description

- Set local device's TS 07.10 control signals.

Changes the state of local device's TS 07.10 control signals and notifies the remote device of the change.

If there are resources to send a command to the remote device this command will return TRUE.

If the remote party has been successfully notified SPP_PORT_EVENT_LOCAL_MODEM_STATUS_CHANGED event will be reported. If the command could not be send to the remote party for any reason other than lack of resources SPP_PORT_EVENT_LOCAL_MODEM_STATUS_CHANGE_FAILED will be reported.

If the status cannot be changed because there is no resources to send a command to the remote device this function will return FALSE and no events will be reported.

@param port Serial port. @param ms Signals mask.

@return c TRUE if successful, c FALSE otherwise.

bt_spp_set_port_options Function

File

spp.h

C

```
void bt_spp_set_port_options(bt_spp_port_t* port, bt_uint options);
```

Description

This is function bt_spp_set_port_options.

bt_spp_set_rts Function

File

spp.h

C

```
bt_byte bt_spp_set_rts(bt_spp_port_t* port, bt_bool on);
```

Description

- Set local device's RS-232 RTS signal.

Changes the state of local device's RS-232 DTR signal and notifies the remote device of the change. This signal corresponds to the TS 07.10's RFCOMM_MODEM_STATUS_RTR signal.

If there are resources to send a command to the remote device this command will return TRUE.

If the remote party has been successfully notified SPP_PORT_EVENT_LOCAL_MODEM_STATUS_CHANGED event will be reported. If the command could not be send to the remote party for any reason other than lack of resources SPP_PORT_EVENT_LOCAL_MODEM_STATUS_CHANGE_FAILED will be reported.

If the status cannot be changed because there is no resources to send a command to the remote device this function will return FALSE and no events will be reported.

@param port Serial port. @param on Value indicating weather to set or clear the signal.

@return c TRUE if successful, c FALSE otherwise.

_bt_spp_find_port Function

File

spp_private.h

C

```
bt_spp_port_t* _bt_spp_find_port(bt_rfcomm_dlc_t* dlc);
```

Description

This is function _bt_spp_find_port.

_bt_spp_handle_rx Function

File

spp_private.h

C

```
void _bt_spp_handle_rx(bt_spp_port_t* port);
```

Description

This is function _bt_spp_handle_rx.

_bt_spp_handle_tx Function

File

spp_private.h

C

```
void _bt_spp_handle_tx(bt_spp_port_t* port);
```

Description

This is function _bt_spp_handle_tx.

_bt_spp_rfcomm_read_data_callback Function

File

spp_private.h

C

```
void _bt_spp_rfcomm_read_data_callback(bt_rfcomm_dlc_p pdlc, bt_byte_p pdata, bt_int len);
```

Description

This is function _bt_spp_rfcomm_read_data_callback.

_bt_spp_rfcomm_send_data_callback Function

File

spp_private.h

C

```
void _bt_spp_rfcomm_send_data_callback(bt_rfcomm_dlc_t* dlc, bt_byte* data, bt_int len, bt_int status);
```

Description

This is function _bt_spp_rfcomm_send_data_callback.

SPP Data Types and Constants

SPP_PORT_OPTION_ENCRYPTED Macro

File

spp.h

C

```
#define SPP_PORT_OPTION_ENCRYPTED 0x0002
```

Description

This is macro SPP_PORT_OPTION_ENCRYPTED.

SPP_PORT_OPTION_MASTER Macro

File

spp.h

C

```
#define SPP_PORT_OPTION_MASTER 0x0004
```

Description

This is macro SPP_PORT_OPTION_MASTER.

SPP_PORT_OPTION_SECURE Macro

File

spp.h

C

```
#define SPP_PORT_OPTION_SECURE 0x0001
```

Description

This is macro SPP_PORT_OPTION_SECURE.

SPP_PORT_TYPE_INCOMING Macro

File

spp.h

C

```
#define SPP_PORT_TYPE_INCOMING 0
```

Description

This is macro SPP_PORT_TYPE_INCOMING.

SPP_PORT_TYPE_OUTGOING Macro

File

spp.h

C

```
#define SPP_PORT_TYPE_OUTGOING 1
```

Description

This is macro SPP_PORT_TYPE_OUTGOING.

SPP_RS232_CTS Macro

File

spp.h

C

```
#define SPP_RS232_CTS RFCOMM_MODEM_STATUS_RTR
```

Description

This is macro SPP_RS232_CTS.

SPP_RS232_DCD Macro

File

spp.h

C

```
#define SPP_RS232_DCD RFCOMM_MODEM_STATUS_DV
```

Description

This is macro SPP_RS232_DCD.

SPP_RS232_DSR Macro

File

spp.h

C

```
#define SPP_RS232_DSR RFCOMM_MODEM_STATUS_RTC
```

Description

- @defgroup spp Serial Port Profile (SPP)

The DotStack SPP API is a simple API for communicating over a Bluetooth link using the Bluetooth Serial Port Profile.

Here are the steps for using this API:

- Call the bt_spp_init() function.
- Allocate a serial port structure with bt_spp_allocate(). One of the parameters to

this function is a pointer to a callback function. That callback function will be called by the stack whenever the state of the serial port changes.

- To connect to a remote device call bt_spp_connect(). The stack will notify when the connection is established by calling the state callback function.
- To wait for a connection from a remote device call bt_spp_listen(). The stack will notify when the connection is established by calling the state callback function.
- When the port is connected you can send data with bt_spp_send() and receive data with bt_spp_receive().
- To terminate the connection call bt_spp_disconnect().
- When you are finished using the port deallocate it with bt_spp_deallocate().

SPP_RS232_DTR Macro

File

spp.h

C

```
#define SPP_RS232_DTR RFCOMM_MODEM_STATUS_RTC
```

Description

This is macro SPP_RS232_DTR.

SPP_RS232_RI Macro

File

spp.h

C

```
#define SPP_RS232_RI RFCOMM_MODEM_STATUS_IC
```

Description

This is macro SPP_RS232_RI.

SPP_RS232_RTS Macro

File

spp.h

C

```
#define SPP_RS232_RTS RFCOMM_MODEM_STATUS_RTR
```

Description

This is macro SPP_RS232_RTS.

bt_spp_find_server_callback_fp Type

File

spp.h

C

```
typedef void (* bt_spp_find_server_callback_fp)(bt_byte server_channel, bt_bool found, void* param);
```

Description

This is type bt_spp_find_server_callback_fp.

bt_spp_port_event_e Enumeration

File

spp.h

C

```
typedef enum _bt_spp_port_event_e {
  SPP_PORT_EVENT_CONNECTED = 1,
  SPP_PORT_EVENT_DISCONNECTED,
  SPP_PORT_EVENT_CONNECTION_FAILED,
  SPP_PORT_EVENT_SEND_PROGRESS,
  SPP_PORT_EVENT_REMOTE_MODEM_STATUS_CHANGED,
  SPP_PORT_EVENT_LOCAL_MODEM_STATUS_CHANGED,
  SPP_PORT_EVENT_LOCAL_MODEM_STATUS_CHANGE_FAILED
} bt_spp_port_event_e;
```

Members

Members	Description
SPP_PORT_EVENT_CONNECTED = 1	• Connection with a remote device was successfully established. This event is reported independent of which side (local or remote) initiated the connection.
SPP_PORT_EVENT_DISCONNECTED	• Connection with the remote device was terminated. This event is reported independent of which side (local or remote) initiated termination of the connection.
SPP_PORT_EVENT_CONNECTION_FAILED	• Connection to a remote device failed. This event is reported when an attempt to establish a connection using bt_spp_connect() has failed.
SPP_PORT_EVENT_SEND_PROGRESS	• Send operation progress. This event is reported periodically during sending data over the serial port connection. It can be used by the application to track send operation progress.
SPP_PORT_EVENT_REMOTE_MODEM_STATUS_CHANGED	• Remote modem status. This event is reported when a remote device notifies the local device about the status of its V.24 control signals. The actual value of the signals can be obtained by calling bt_spp_get_remote_modem_status(const bt_spp_port_t* port)

SPP_PORT_EVENT_LOCAL_MODEM_STATUS_CHANGED	• Local modem status. This event is reported when the local device has successfully sent the state of its V.24 signals to the remote party.
SPP_PORT_EVENT_LOCAL_MODEM_STATUS_CHANGE_FAILED	• Local modem status. This event is reported when the local device failed to send the state of its V.24 signals to the remote party.

Description

- Serial port event.

Values of this enumeration represent serial port events. Events are reported to the application through a callback function. The callback function is specified when the port is allocated using bt_spp_allocate().

bt_spp_port_state_e Enumeration

File

spp.h

C

```
typedef enum _bt_spp_port_state_e {
  SPP_PORT_STATE_FREE,
  SPP_PORT_STATE_DISCONNECTED,
  SPP_PORT_STATE_DISCONNECTING,
  SPP_PORT_STATE_CONNECTING,
  SPP_PORT_STATE_CONNECTED
} bt_spp_port_state_e;
```

Members

Members	Description
SPP_PORT_STATE_FREE	*< Used internally to mark a port structure as available for allocation. The application should never encounter a port in this state.
SPP_PORT_STATE_DISCONNECTED	< Port is not connected.
SPP_PORT_STATE_DISCONNECTING	< Disconnecting from remote device.
SPP_PORT_STATE_CONNECTING	< Connecting to a remote device.
SPP_PORT_STATE_CONNECTED	< Port is connected to a remote device.

Description

- Serial port state.

Values of this enumeration represent states of the serial port.

bt_spp_port_t Type

File

spp.h

C

```
typedef struct _bt_spp_port_t bt_spp_port_t;
```

Description

This is type bt_spp_port_t.

bt_spp_receive_callback_fp Type

File

spp.h

C

```
typedef void (* bt_spp_receive_callback_fp)(bt_spp_port_t* port, bt_int bytes_received, void* param);
```

Description

- Serial port receive callback.

This callback function is called when a receive operation initiated by bt_spp_receive() completes.

@param port Serial port on which the receive operation completed.

@param bytes_received Number of received bytes.

@param param Callback parameter that was specified when bt_spp_allocate() was called.

bt_spp_send_callback_fp Type

File

spp.h

C

```
typedef void (* bt_spp_send_callback_fp)(bt_spp_port_t* port, bt_ulong bytes_sent, bt_spp_send_status_e
status, void* param);
```

Description

- Serial port send callback.

This callback function is called when a send operation initiated by bt_spp_send() completes.

@param port Serial port on which the send operation completed.

@param bytes_sent Number of bytes sent. This parameter is just a convenience as it always specifies the same number of bytes that was passed to bt_spp_send();

@param status Completion status. It is one of the values defined in the ::bt_spp_send_status_e enumeration.

@param param Callback parameter that was specified when bt_spp_allocate() was called.

bt_spp_send_status_e Enumeration

File

spp.h

C

```
typedef enum _bt_spp_send_status_e {
  SPP_SEND_STATUS_SUCCESS = 0,
  SPP_SEND_STATUS_TIMEOUT,
  SPP_SEND_STATUS_NOT_ENOUGH_RESOURCES,
  SPP_SEND_STATUS_CANCELED,
  SPP_SEND_STATUS_INTERRUPTED
} bt_spp_send_status_e;
```

Members

Members	Description
SPP_SEND_STATUS_SUCCESS = 0	< The operation completed successfully.
SPP_SEND_STATUS_TIMEOUT	< The operation timed out.
SPP_SEND_STATUS_NOT_ENOUGH_RESOURCES	< There was not enough resources to complete the operation.
SPP_SEND_STATUS_CANCELED	< The operation was canceled.
SPP_SEND_STATUS_INTERRUPTED	< The operation was interrupted because the connection closed.

Description

Send operation status.

bt_spp_state_callback_fp Type

File

spp.h

C

```
typedef void (* bt_spp_state_callback_fp)(bt_spp_port_t* port, bt_spp_port_event_e evt, void* param);
```

Description

- Serial port state callback.

This callback function is called whenever the state of a serial port is changed.

@param port Serial port which state has changed.

@param evt Event describing the nature of state change. It is one of the values defined in the ::bt_spp_port_event_e enumeration.

@param param Callback parameter that was specified when bt_spp_allocate() was called.

SSP Functions

bt_ssp_evt_handler Function

File

ssp.h

C

```
void bt_ssp_evt_handler(bt_hci_event_t* evt);
```

Description

This is function bt_ssp_evt_handler.

bt_ssp_init Function

File

ssp.h

C

```
void bt_ssp_init();
```

Description

This is function bt_ssp_init.

bt_ssp_read_local_oob_data Function

File

ssp.h

C

```
bt_bool bt_ssp_read_local_oob_data(bt_spp_read_local_oob_data_callback_fp callback);
```

Description

This is function bt_ssp_read_local_oob_data.

bt_ssp_send_keypress_notification Function

File

ssp.h

C

```
bt_bool bt_ssp_send_keypress_notification(bt_bdaddr_p bdaddr_remote, SSP_KEYPRESS_NOTIFICATION_TYPE
keypress_type);
```

Description

This is function bt_ssp_send_keypress_notification.

bt_ssp_send_user_confirmation Function

File

ssp.h

C

```
bt_bool bt_ssp_send_user_confirmation(bt_byte status, bt_ssp_user_confirmation_request* ucr,
bt_hci_cmd_callback_fp cb);
```

Description

This is function bt_ssp_send_user_confirmation.

bt_ssp_send_user_passkey Function

File

ssp.h

C

```
bt_bool bt_ssp_send_user_passkey(bt_byte status, bt_ssp_user_passkey_request* upkr, bt_hci_cmd_callback_fp
cb);
```

Description

This is function bt_ssp_send_user_passkey.

bt_ssp_set_io_capabilities Function

File

ssp.h

C

```
bt_bool bt_ssp_set_io_capabilities(bt_byte status, bt_ssp_io_capability* io_caps, bt_hci_cmd_callback_fp
cb);
```

Description

This is function bt_ssp_set_io_capabilities.

bt_ssp_set_mode Function

File

ssp.h

C

```
bt_bool bt_ssp_set_mode(SSP_MODE mode, bt_hci_cmd_callback_fp callback);
```

Description

This is function bt_ssp_set_mode.

bt_ssp_set_oob_data Function

File

ssp.h

C

```
bt_bool bt_ssp_set_oob_data(bt_byte status, bt_ssp_oob_data* oob_data, bt_hci_cmd_callback_fp cb);
```

Description

This is function bt_ssp_set_oob_data.

ssp_evt_io_capability_request Function

File

ssp_event_handler.h

C

```
void ssp_evt_io_capability_request(bt_hci_event_t* evt);
```

Description

This is function ssp_evt_io_capability_request.

ssp_evt_io_capability_response Function

File

ssp_event_handler.h

C

```
void ssp_evt_io_capability_response(bt_hci_event_t* evt);
```

Description

This is function ssp_evt_io_capability_response.

ssp_evt_keypress_notification Function

File

ssp_event_handler.h

C

```
void ssp_evt_keypress_notification(bt_hci_event_t* evt);
```

Description

This is function ssp_evt_keypress_notification.

ssp_evt_oob_data_request Function

File

ssp_event_handler.h

C

```
void ssp_evt_oob_data_request(bt_hci_event_t* evt);
```

Description

This is function ssp_evt_oob_data_request.

ssp_evt_ssp_complete Function

File

ssp_event_handler.h

C

```
void ssp_evt_ssp_complete(bt_hci_event_t* evt);
```

Description

This is function ssp_evt_ssp_complete.

ssp_evt_user_confirmation_request Function

File

ssp_event_handler.h

C

```
void ssp_evt_user_confirmation_request(bt_hci_event_t* evt);
```

Description

This is function ssp_evt_user_confirmation_request.

ssp_evt_user_passkey_notification Function

File

ssp_event_handler.h

C

```
void ssp_evt_user_passkey_notification(bt_hci_event_t* evt);
```

Description

This is function ssp_evt_user_passkey_notification.

ssp_evt_user_passkey_request Function

File

ssp_event_handler.h

C

```
void ssp_evt_user_passkey_request(bt_hci_event_t* evt);
```

Description

This is function ssp_evt_user_passkey_request.

_bt_spp_client_init Function

File

spp_private.h

C

```
void _bt_spp_client_init();
```

Description

This is function _bt_spp_client_init.

bt_spp_cancel_listen Function

File

spp.h

C

```
bt_bool bt_spp_cancel_listen(bt_spp_port_t* port, bt_byte server_channel);
```

Description

- Stop listening for incoming connections.

This function stops the port to accept incoming connections on a given server channel.

@param port Serial port. @param channel The RFCOMM server channel to accept connections on.

@return c TRUE if successful, c FALSE otherwise.

bt_spp_find_server_ex Function

File

spp.h

C

```
bt_bool bt_spp_find_server_ex(bt_bdaddr_t* deviceAddress, const bt_uuid_t* service_class_id, bt_bool
force_hci_disconnect, bt_spp_find_server_callback_fp callback, bt_sdp_client_callback_fp client_callback,
void* callback_param);
```

Description

This is function bt_spp_find_server_ex.

SSP Data Types and Constants

OOB_DATA_HASH_LENGTH Macro

File

ssp.h

C

```
#define OOB_DATA_HASH_LENGTH 16
```

Description

Remote OOB Data Request Event

OOB_DATA_RANDOMIZER_LENGTH Macro

File

ssp.h

C

```
#define OOB_DATA_RANDOMIZER_LENGTH 16
```

Description

This is macro OOB_DATA_RANDOMIZER_LENGTH.

SSP_MAX_MANAGERS Macro

File

ssp.h

C

```
#define SSP_MAX_MANAGERS 1
```

Description

This is macro SSP_MAX_MANAGERS.

_bt_spp_port_t Structure

File

spp.h

C

```
struct _bt_spp_port_t {
  bt_byte state;
  bt_byte type;
```

```
  bt_byte server_channel;
  bt_rfcomm_dlc_t* rfcomm_dlc;
  bt_byte cur_local_ms;
  bt_ulong listening_server_channels;
  struct _bt_spp_port_flags_t {
    unsigned int sig_tx : 1;
    unsigned int sig_rx : 1;
    unsigned int sig_state : 1;
    unsigned int sig_credit : 1;
    unsigned int open : 1;
    unsigned int listening : 1;
    unsigned int secure : 1;
    unsigned int encrypted : 1;
    unsigned int master : 1;
  } flags;
  bt_signal_t signal;
  bt_spp_state_callback_fp state_cb;
  bt_spp_receive_callback_fp receive_cb;
  bt_spp_send_callback_fp send_cb;
  void* cb_param;
  bt_byte* rx_buffer;
  bt_int rx_buffer_len;
  bt_byte* tx_data;
  bt_ulong tx_data_len;
  bt_ulong tx_remaining_len;
  bt_bool tx_cancel;
  bt_byte remaining_connect_attemtps;
  bt_bdaddr_t connect_address;
  bt_rfcomm_session_listener_t rfcomm_session_listener;
  bt_byte* frame_buffers;
  bt_int* frame_len;
  bt_byte read_frame_index;
  bt_byte write_frame_index;
  bt_int read_frame_pos;
};
```

Members

Members	Description
bt_byte state;	• Port state. The field is set to one of the values defined in the ::bt_spp_port_state_e enumeration. This field must never be modified by the application.

Description

- Serial port structure.

This structure represents a Bluetooth serial port. Application code may only use those fields that are documented. The rest of the fields are private to the SPP implementation.

bt_spp_read_local_oob_data_callback_fp Type

File

ssp.h

C

```
typedef void (* bt_spp_read_local_oob_data_callback_fp)(bt_byte status, bt_ssp_oob_data* oob_data, void*
init_param);
```

Description

This is type bt_spp_read_local_oob_data_callback_fp.

bt_ssp_io_capability Structure

File

ssp.h

C

```
typedef struct _bt_ssp_io_capability {
  bt_bdaddr_t bdaddr_remote;
```

```
  SSP_IO_CAPABILITY io_capability;
  SSP_OOB_DATA_PRESENT oob_data_present;
  SSP_AUTHENTICATION_REQUIREMENTS authentication_requirements;
} bt_ssp_io_capability;
```

Description

This is type bt_ssp_io_capability.

bt_ssp_keypress_notification Structure

File

ssp.h

C

```
typedef struct _bt_ssp_keypress_notification {
  bt_bdaddr_t bdaddr_remote;
  SSP_KEYPRESS_NOTIFICATION_TYPE type;
} bt_ssp_keypress_notification;
```

Description

Keypress Notification Event

bt_ssp_oob_data Structure

File

ssp.h

C

```
typedef struct _bt_ssp_oob_data {
  bt_bdaddr_t bdaddr;
  bt_byte hash[OOB_DATA_HASH_LENGTH];
  bt_byte randomizer[OOB_DATA_RANDOMIZER_LENGTH];
} bt_ssp_oob_data;
```

Description

This is type bt_ssp_oob_data.

bt_ssp_simple_pairing_complete Structure

File

ssp.h

C

```
typedef struct _bt_ssp_simple_pairing_complete {
  bt_byte status;
  bt_bdaddr_t bdaddr_remote;
} bt_ssp_simple_pairing_complete;
```

Description

Simple Pairing Complete Event

bt_ssp_user_confirmation_request Structure

File

ssp.h

C

```
typedef struct _bt_ssp_user_confirmation_request {
  bt_bdaddr_t bdaddr_remote;
  bt_ulong numeric_value;
} bt_ssp_user_confirmation_request;
```

Description

User Confirmation Request Event

bt_ssp_user_passkey_notification Structure

File

ssp.h

C

```
typedef struct _bt_ssp_user_passkey_notification {
  bt_bdaddr_t bdaddr_remote;
  bt_ulong passkey;
} bt_ssp_user_passkey_notification;
```

Description

User Passkey Notification Event

bt_ssp_user_passkey_request Structure

File

ssp.h

C

```
typedef struct _bt_ssp_user_passkey_request {
  bt_bdaddr_t bdaddr_remote;
  bt_ulong passkey;
} bt_ssp_user_passkey_request;
```

Description

User Passkey Request Event

SSP_AUTHENTICATION_REQUIREMENTS Enumeration

File

ssp.h

C

```
typedef enum _SSP_AUTHENTICATION_REQUIREMENTS {
  SSP_MITM_NOT_REQUIRED_NO_BONDING = 0,
  SSP_MITM_REQUIRED_NO_BONDING,
  SSP_MITM_NOT_REQUIRED_DEDICATED_BONDING,
  SSP_MITM_REQUIRED_DEDICATED_BONDING,
  SSP_MITM_NOT_REQUIRED_GENERAL_BONDING,
  SSP_MITM_REQUIRED_GENERAL_BONDING
} SSP_AUTHENTICATION_REQUIREMENTS;
```

Description

This is type SSP_AUTHENTICATION_REQUIREMENTS.

SSP_IO_CAPABILITY Enumeration

File

ssp.h

C

```
typedef enum _SSP_IO_CAPABILITY {
  SSP_IO_CAPABILITY_DISPLAY_ONLY = 0,
  SSP_IO_CAPABILITY_DISPLAY_YESNO,
  SSP_IO_CAPABILITY_KEYBOARD_ONLY,
  SSP_IO_CAPABILITY_NO_INPUT_NO_OUTPUT
} SSP_IO_CAPABILITY;
```

Description

This is type SSP_IO_CAPABILITY.

SSP_KEYPRESS_NOTIFICATION_TYPE Enumeration

File

ssp.h

C

```
typedef enum _SSP_KEYPRESS_NOTIFICATION_TYPE {
  PASSKEY_ENTRY_STYARTED,
  PASSKEY_DIGIT_ENTERED,
  PASSKEY_DIGIT_ERASED,
  PASSKEY_CLEARED,
  PASSKEY_ENTRY_COMPLETED
} SSP_KEYPRESS_NOTIFICATION_TYPE;
```

Description

This is type SSP_KEYPRESS_NOTIFICATION_TYPE.

SSP_MODE Enumeration

File

ssp.h

C

```
typedef enum _SSP_MODE {
  SSP_MODE_DISABLED = 0,
  SSP_MODE_ENABLED
} SSP_MODE;
```

Description

This is type SSP_MODE.

SSP_OOB_DATA_PRESENT Enumeration

File

ssp.h

C

```
typedef enum _SSP_OOB_DATA_PRESENT {
  SSP_OOB_DATA_NOT_PRESENT = 0,
  SSP_OOB_DATA_REMOTE_DEVICE_DATA_PRESENT
} SSP_OOB_DATA_PRESENT;
```

Description

This is type SSP_OOB_DATA_PRESENT.

SSP_EVENT Enumeration

File

ssp_event.h

C

```
typedef enum _SSP_EVENT {
  SSP_EVENT_IO_CAPABILITY_RESPONSE,
  SSP_EVENT_KEYPRESS_NOTIFICATION,
  SSP_EVENT_USER_PASSKEY_REQUEST,
  SSP_EVENT_USER_PASSKEY_NOTIFICATION,
  SSP_EVENT_USER_CONFIRMATION_REQUEST,
  SSP_EVENT_OOB_DATA_REQUEST,
  SSP_EVENT_IO_CAPABILITY_REQUEST,
```

```
    SSP_EVENT_SIMPLE_PAIRING_COMPLETE
} SSP_EVENT;
```

Description

This is type SSP_EVENT.

Misc. Functions

btx_csr_alloc_bccmd_getreq Function

File

csr.h

C

```
bt_hci_command_t* btx_csr_alloc_bccmd_getreq(bt_uint var_id, bt_uint data_word_count,
bt_hci_cmd_callback_fp callback, void* callback_param);
```

Description

This is function btx_csr_alloc_bccmd_getreq.

btx_csr_alloc_bccmd_setreq Function

File

csr.h

C

```
bt_hci_command_t* btx_csr_alloc_bccmd_setreq(bt_uint var_id, bt_uint data_word_count,
bt_hci_cmd_callback_fp callback, void* callback_param);
```

Description

This is function btx_csr_alloc_bccmd_setreq.

btx_csr_autobaud Function

File

csr.h

C

```
void btx_csr_autobaud(btx_csr_autobaud_buffer_t* buffer, btx_csr_autobaud_callback_fp callback, void*
callback_param);
```

Description

brief Configure controller's UART speed. ingroup btx_csr

details This function makes the controller auto-configure its UART speed. The host transport must be set to H4. This function works only with BC6 controllers.

btx_csr_bc7_sel_host_interface_h4 Function

File

csr.h

C

```
void btx_csr_bc7_sel_host_interface_h4(btx_csr_autobaud_buffer_t* buffer, bt_byte interval,
btx_csr_autobaud_callback_fp callback, void* callback_param);
```

Description

brief Configure controller's UART speed and host interface. ingroup btx_csr

details This function makes the controller auto-configure its UART speed and select H4 as host interface. PS_KEY_HOST_INTERFACE must not be set. PS_KEY_UART_BITRATE must be set to 0. This function works only with BC7 controllers.

btx_csr_enable_tx Function

File

csr.h

C

```
bt_bool btx_csr_enable_tx(bt_bool enable, bt_hci_cmd_callback_fp callback, void* callback_param);
```

Description

brief Enable/disable transmitter ingroup btx_csr

param enable Specifies whether the transmitter should be enable or disabled. param callback The callback function that will be called after the command has completed. param callback_param A pointer to arbitrary data to be passed to the c callback callback.

btx_csr_exec_script Function

File

csr.h

C

```
void btx_csr_exec_script(const btx_csr_script_t* script, btx_csr_exec_script_buffer_t* buffer,
btx_csr_exec_script_callback_fp callback, void* callback_param);
```

Description

brief Patch controller's firmware ingroup btx_csr

details This function executes a script that patches the controller's firmware. The c script must point to a structure that contain a complete patch script for a particular controller model and revision. If the revision specified in the script and revision read from the controller are the same btx_csr_patch_controller() loads the script to the controller and calls the c callback with the first parameter TRUE. Otherwise the c callback is called with the first parameter FALSE.

If support for multiple firmware revisions is neede use btx_csr_patch_controller().

param script Array of patch scripts. param buffer A buffer for storing temporary data needed for script execution. param callback The callback function that will be called when the script has been executed. param callback_param A pointer to arbitrary data to be passed to the c callback callback..

btx_csr_get_cached_temperature Function

File

csr.h

C

```
bt_bool btx_csr_get_cached_temperature(btx_csr_get_var_callback_fp callback, void* callback_param);
```

Description

brief Get chip's cached temperature ingroup btx_csr

param callback The callback function that will be called after the command has completed. param callback_param A pointer to arbitrary data to be passed to the c callback callback.

btx_csr_get_ps_var Function

File

csr.h

C

```
bt_bool btx_csr_get_ps_var(bt_uint ps_key, bt_uint value_word_count, btx_csr_get_ps_var_callback_fp
callback, void* callback_param);
```

Description

This is function btx_csr_get_ps_var.

btx_csr_get_ps_var_ex Function

File

csr.h

C

```
bt_bool btx_csr_get_ps_var_ex(bt_uint ps_key, bt_uint value_word_count, bt_uint store,
btx_csr_get_ps_var_callback_fp callback, void* callback_param);
```

Description

This is function btx_csr_get_ps_var_ex.

btx_csr_get_rssi_acl Function

File

csr.h

C

```
bt_bool btx_csr_get_rssi_acl(bt_hci_hconn_t hconn, btx_csr_get_var_callback_fp callback, void*
callback_param);
```

Description

brief Get RSSI ingroup btx_csr

details This function retrieves the RSSI for a given HCI ACL handle.

param hconn ACL connection handle. param callback The callback function that will be called after the command has completed. param callback_param A pointer to arbitrary data to be passed to the c callback callback.

btx_csr_get_script__PB_27_R20_BC6ROM_A04 Function

File

csr.h

C

```
const btx_csr_script_t* btx_csr_get_script__PB_27_R20_BC6ROM_A04();
```

Description

brief Return script for patching BlueCore 6 ingroup btx_csr

btx_csr_get_script__PB_90_REV6 Function

File

csr.h

C

```
const btx_csr_script_t* btx_csr_get_script__PB_90_REV6();
```

Description

brief Return script for patching CSR8810 (BlueCore 7) ingroup btx_csr

btx_csr_get_var Function

File

csr.h

C

```
bt_bool btx_csr_get_var(bt_uint var_id, btx_csr_get_var_callback_fp callback, void* callback_param);
```

Description

This is function btx_csr_get_var.

btx_csr_set_ps_var Function

File

csr.h

C

```
bt_bool btx_csr_set_ps_var(bt_uint ps_key, const bt_uint* value, bt_uint value_word_count,
bt_hci_cmd_callback_fp callback);
```

Description

This is function btx_csr_set_ps_var.

btx_csr_set_ps_var_ex Function

File

csr.h

C

```
bt_bool btx_csr_set_ps_var_ex(bt_uint ps_key, const bt_uint* value, bt_uint value_word_count, bt_uint
store, bt_hci_cmd_callback_fp callback);
```

Description

This is function btx_csr_set_ps_var_ex.

btx_csr_set_ps_vars Function

File

csr.h

C

```
void btx_csr_set_ps_vars(const bt_uint* ps_vars, btx_csr_set_ps_vars_buffer_t* buffer,
btx_csr_set_ps_vars_callback_fp callback, void* callback_param);
```

Description

brief Write PS variables ingroup btx_csr

details

param ps_vars PS values param buffer A buffer for storing temporary data during function execution. param callback The callback function that will be called when all PS values have been sent to the controller or error occurred. param callback_param A pointer to arbitrary data to be passed to the c callback callback..

btx_csr_set_ps_vars_ex Function

File

csr.h

C

```
void btx_csr_set_ps_vars_ex(const bt_uint* ps_vars, btx_csr_set_ps_vars_buffer_t* buffer, bt_uint store,
btx_csr_set_ps_vars_callback_fp callback, void* callback_param);
```

Description

brief Write PS variables ingroup btx_csr

param ps_vars PS values param buffer A buffer for storing temporary data during function execution. param store param callback The callback function that will be called when all PS values have been sent to the controller or error occurred. param callback_param A pointer to arbitrary data to be passed to the c callback callback..

btx_csr_set_var Function

File

csr.h

C

```
bt_bool btx_csr_set_var(bt_uint var_id, const bt_uint* value, bt_uint value_word_count,
btx_csr_set_var_callback_fp callback, void* callback_param);
```

Description

This is function btx_csr_set_var.

btx_csr_warm_reset Function

File

csr.h

C

```
bt_bool btx_csr_warm_reset();
```

Description

brief Warm reset ingroup btx_csr

details This function performs warm reset of the controller. All patches and configuration parameters sent to the controller before warm reset are kept intact.

btx_csr_warm_reset_ex Function

File

csr.h

C

```
bt_bool btx_csr_warm_reset_ex(bt_hci_cmd_callback_fp callback, void* callback_param);
```

Description

brief Warm reset ingroup btx_csr

details This function performs warm reset of the controller. All patches and configuration parameters sent to the controller before warm reset are kept intact. Since the controller does not respond to the warm reset command as it starts resetting immediately upon receiving the command, the c callback is called right after the command packet has been transmitted to the controller.

param callback The callback function that will be called after the warm reset command has been sent to the controller. param callback_param A pointer to arbitrary data to be passed to the c callback callback.

btx_ti_drpb_enable_rf_calibration Function

File

ti.h

C

```
bt_bool btx_ti_drpb_enable_rf_calibration(bt_byte periodic_mode, bt_ulong calibration_procedure, bt_byte
temp_condition, bt_hci_cmd_callback_fp callback);
```

Description

This is function btx_ti_drpb_enable_rf_calibration.

btx_ti_drpb_set_power_vector Function

File

ti.h

C

```
bt_bool btx_ti_drpb_set_power_vector(bt_byte modulation_type, const bt_byte* power_vector, bt_byte
epc_max_level_threshold, bt_uint external_pa_mode, bt_hci_cmd_callback_fp callback);
```

Description

This is function btx_ti_drpb_set_power_vector.

btx_ti_drpb_tester_con_tx Function

File

ti.h

C

```
bt_bool btx_ti_drpb_tester_con_tx(bt_byte modulation_scheme, bt_byte test_pattern, bt_byte
frequency_channel, bt_byte power_level, bt_ulong generator_initialization_value, bt_ulong
edr_generator_mask, bt_hci_cmd_callback_fp callback);
```

Description

This is function btx_ti_drpb_tester_con_tx.

btx_ti_enable_deep_sleep Function

File

ti.h

C

```
bt_bool btx_ti_enable_deep_sleep(bt_bool enable, bt_hci_cmd_callback_fp callback);
```

Description

This is function btx_ti_enable_deep_sleep.

btx_ti_enable_fast_clock_crystal Function

File

ti.h

C

```
bt_bool btx_ti_enable_fast_clock_crystal(bt_hci_cmd_callback_fp callback);
```

Description

This is function btx_ti_enable_fast_clock_crystal.

btx_ti_enable_low_power_scan Function

File

ti.h

C

```
bt_bool btx_ti_enable_low_power_scan(bt_bool enable, bt_hci_cmd_callback_fp callback);
```

Description

This is function btx_ti_enable_low_power_scan.

btx_ti_enable_low_power_scan_default Function

File

ti.h

C

```
bt_bool btx_ti_enable_low_power_scan_default(bt_bool enable, bt_hci_cmd_callback_fp callback);
```

Description

This is function btx_ti_enable_low_power_scan_default.

btx_ti_exec_script Function

File

ti.h

C

```
void btx_ti_exec_script(const btx_ti_script_t* script, btx_ti_exec_script_buffer_t* buffer,
btx_ti_completion_callback_fp callback, void* callback_param);
```

Description

This is function btx_ti_exec_script.

btx_ti_exec_script_oem Function

File

ti.h

C

```
void btx_ti_exec_script_oem(const btx_ti_script_t* script, btx_ti_exec_script_oem_buffer_t* buffer,
btx_ti_exec_script_oem_callback_fp callback, void* callback_param);
```

Description

This is function btx_ti_exec_script_oem.

btx_ti_get_script__BL6450_2_0_BT_Service_Pack_2_36 Function

File

ti.h

C

```
const btx_ti_script_t* btx_ti_get_script__BL6450_2_0_BT_Service_Pack_2_36();
```

Description

CC2560 Scripts (Service pack 2.36)

btx_ti_get_script__BL6450_2_0_BT_Service_Pack_2_44 Function

File

ti.h

C

```
const btx_ti_script_t* btx_ti_get_script__BL6450_2_0_BT_Service_Pack_2_44();
```

Description

CC2560 Scripts (Service pack 2.44)

btx_ti_get_script__BL6450L_BT_Service_Pack_2_10 Function

File

ti.h

C

```
const btx_ti_script_t* btx_ti_get_script__BL6450L_BT_Service_Pack_2_10();
```

Description

CC2564 Scripts (Service pack 2.10)

btx_ti_get_script__BL6450L_BT_Service_Pack_2_10_BLE_AddOn Function

File

ti.h

C

```
const btx_ti_script_t* btx_ti_get_script__BL6450L_BT_Service_Pack_2_10_BLE_AddOn();
```

Description

This is function btx_ti_get_script__BL6450L_BT_Service_Pack_2_10_BLE_AddOn.

btx_ti_get_script__BL6450L_BT_Service_Pack_2_12 Function

File

ti.h

C

```
const btx_ti_script_t* btx_ti_get_script__BL6450L_BT_Service_Pack_2_12();
```

Description

CC2564 Scripts (Service pack 2.12)

btx_ti_get_script__BL6450L_BT_Service_Pack_2_12_BLE_AddOn Function

File

ti.h

C

```
const btx_ti_script_t* btx_ti_get_script__BL6450L_BT_Service_Pack_2_12_BLE_AddOn();
```

Description

This is function btx_ti_get_script__BL6450L_BT_Service_Pack_2_12_BLE_AddOn.

btx_ti_get_script__BL6450L_BT_Service_Pack_2_7 Function

File

ti.h

C

```
const btx_ti_script_t* btx_ti_get_script__BL6450L_BT_Service_Pack_2_7();
```

Description

CC2564 Scripts (Service pack 2.7)

btx_ti_get_script__BL6450L_BT_Service_Pack_2_8_AVPR_AddOn Function

File

ti.h

C

```
const btx_ti_script_t* btx_ti_get_script__BL6450L_BT_Service_Pack_2_8_AVPR_AddOn();
```

Description

This is function btx_ti_get_script__BL6450L_BT_Service_Pack_2_8_AVPR_AddOn.

btx_ti_get_script__BL6450L_BT_Service_Pack_2_8_BLE_AddOn Function

File

ti.h

C

```
const btx_ti_script_t* btx_ti_get_script__BL6450L_BT_Service_Pack_2_8_BLE_AddOn();
```

Description

This is function btx_ti_get_script__BL6450L_BT_Service_Pack_2_8_BLE_AddOn.

btx_ti_get_script__BL6450L_BT_Service_Pack_2_8_Short Function

File

ti.h

C

```
const btx_ti_script_t* btx_ti_get_script__BL6450L_BT_Service_Pack_2_8_Short();
```

Description

CC2564 Scripts (Service pack 2.8)

btx_ti_get_script__BL6450x_BT_Service_Pack_2_7_AVPR_AddOn Function

File

ti.h

C

```
const btx_ti_script_t* btx_ti_get_script__BL6450x_BT_Service_Pack_2_7_AVPR_AddOn();
```

Description

This is function btx_ti_get_script__BL6450x_BT_Service_Pack_2_7_AVPR_AddOn.

btx_ti_get_script__BL6450x_BT_Service_Pack_2_7_BLE_AddOn Function

File

ti.h

C

```
const btx_ti_script_t* btx_ti_get_script__BL6450x_BT_Service_Pack_2_7_BLE_AddOn();
```

Description

This is function btx_ti_get_script__BL6450x_BT_Service_Pack_2_7_BLE_AddOn.

btx_ti_get_script__CC2564B_BT_Service_Pack_0_1 Function

File

ti.h

C

```
const btx_ti_script_t* btx_ti_get_script__CC2564B_BT_Service_Pack_0_1();
```

Description

CC2564B Scripts (Service pack 0.1)

btx_ti_get_script__CC2564B_BT_Service_Pack_0_1_BLE_AddOn Function

File

ti.h

C

```
const btx_ti_script_t* btx_ti_get_script__CC2564B_BT_Service_Pack_0_1_BLE_AddOn();
```

Description

This is function btx_ti_get_script__CC2564B_BT_Service_Pack_0_1_BLE_AddOn.

btx_ti_get_script__CC2564B_BT_Service_Pack_0_2 Function

File

ti.h

C

```
const btx_ti_script_t* btx_ti_get_script__CC2564B_BT_Service_Pack_0_2();
```

Description

CC2564B Scripts (Service pack 0.2)

btx_ti_get_script__CC2564B_BT_Service_Pack_0_2_BLE_AddOn Function

File

ti.h

C

```
const btx_ti_script_t* btx_ti_get_script__CC2564B_BT_Service_Pack_0_2_BLE_AddOn();
```

Description

This is function btx_ti_get_script__CC2564B_BT_Service_Pack_0_2_BLE_AddOn.

btx_ti_get_script__WL127xL_BT_Service_Pack_2_3 Function

File

ti.h

C

```
const btx_ti_script_t* btx_ti_get_script__WL127xL_BT_Service_Pack_2_3();
```

Description

CC2564 Scripts (Service pack 2.3)

btx_ti_get_script__WL127xL_BT_Service_Pack_2_3_AVPR_AddOn Function

File

ti.h

C

```
const btx_ti_script_t* btx_ti_get_script__WL127xL_BT_Service_Pack_2_3_AVPR_AddOn();
```

Description

This is function btx_ti_get_script__WL127xL_BT_Service_Pack_2_3_AVPR_AddOn.

btx_ti_get_script__WL127xL_BT_Service_Pack_2_3_BLE_AddOn Function

File

ti.h

C

```
const btx_ti_script_t* btx_ti_get_script__WL127xL_BT_Service_Pack_2_3_BLE_AddOn();
```

Description

This is function btx_ti_get_script__WL127xL_BT_Service_Pack_2_3_BLE_AddOn.

btx_ti_get_script__WL127xL_BT_Service_Pack_2_3_DC2DC_AddOn Function

File

ti.h

C

```
const btx_ti_script_t* btx_ti_get_script__WL127xL_BT_Service_Pack_2_3_DC2DC_AddOn();
```

Description

This is function btx_ti_get_script__WL127xL_BT_Service_Pack_2_3_DC2DC_AddOn.

btx_ti_get_script__WL127xL_BT_Service_Pack_2_4 Function

File

ti.h

C

```
const btx_ti_script_t* btx_ti_get_script__WL127xL_BT_Service_Pack_2_4();
```

Description

CC2564 Scripts (Service pack 2.4)

btx_ti_get_script__WL127xL_BT_Service_Pack_2_4_AVPR_AddOn Function

File

ti.h

C

```
const btx_ti_script_t* btx_ti_get_script__WL127xL_BT_Service_Pack_2_4_AVPR_AddOn();
```

Description

This is function btx_ti_get_script__WL127xL_BT_Service_Pack_2_4_AVPR_AddOn.

btx_ti_get_script__WL127xL_BT_Service_Pack_2_4_BLE_AddOn Function

File

ti.h

C

```
const btx_ti_script_t* btx_ti_get_script__WL127xL_BT_Service_Pack_2_4_BLE_AddOn();
```

Description

This is function btx_ti_get_script__WL127xL_BT_Service_Pack_2_4_BLE_AddOn.

btx_ti_get_script__WL127xL_BT_Service_Pack_2_4_DC2DC_AddOn Function

File

ti.h

C

```
const btx_ti_script_t* btx_ti_get_script__WL127xL_BT_Service_Pack_2_4_DC2DC_AddOn();
```

Description

This is function btx_ti_get_script__WL127xL_BT_Service_Pack_2_4_DC2DC_AddOn.

btx_ti_get_script__XWL1271L1_BT_ServicePack_1_3 Function

File

ti.h

C

```
const btx_ti_script_t* btx_ti_get_script__XWL1271L1_BT_ServicePack_1_3();
```

Description

CC2564 Scripts (Service pack 1.3)

btx_ti_get_script__XWL1271L1_BT_ServicePack_1_3_BLE_Init Function

File

ti.h

C

```
const btx_ti_script_t* btx_ti_get_script__XWL1271L1_BT_ServicePack_1_3_BLE_Init();
```

Description

This is function btx_ti_get_script__XWL1271L1_BT_ServicePack_1_3_BLE_Init.

btx_ti_init_ble_controller Function

File

ti.h

C

```
void btx_ti_init_ble_controller(btx_ti_exec_script_buffer_t* buffer, btx_ti_completion_callback_fp
callback, void* callback_param);
```

Description

This is function btx_ti_init_ble_controller.

btx_ti_init_controller Function

File

ti.h

C

```
void btx_ti_init_controller(btx_ti_exec_script_buffer_t* buffer, btx_ti_completion_callback_fp callback,
void* callback_param);
```

Description

This is function btx_ti_init_controller.

btx_ti_set_uart_baud_rate Function

File

ti.h

C

```
bt_bool btx_ti_set_uart_baud_rate(bt_ulong baud_rate, bt_hci_cmd_callback_fp callback);
```

Description

This is function btx_ti_set_uart_baud_rate.

btx_ti_write_bdaddr Function

File

ti.h

C

```
bt_bool btx_ti_write_bdaddr(bt_bdaddr_t* bdaddr, bt_hci_cmd_callback_fp callback);
```

Description

This is function btx_ti_write_bdaddr.

btx_ti_write_hardware_register Function

File

ti.h

C

```
bt_bool btx_ti_write_hardware_register(bt_ulong address, bt_uint value, bt_hci_cmd_callback_fp callback);
```

Description

This is function btx_ti_write_hardware_register.

btx_csr_get_script__dsp_script__PB_109_DSP_rev8 Function

File

csr.h

C

```
const btx_csr_script_t* btx_csr_get_script__dsp_script__PB_109_DSP_rev8();
```

Description

brief Return script for patching DSP in CSR8x11 A08 (BlueCore 7) ingroup btx_csr

btx_csr_get_script__PB_101_CSR8811_CSP28_UART Function

File

csr.h

C

```
const btx_csr_script_t* btx_csr_get_script__PB_101_CSR8811_CSP28_UART();
```

Description

brief Return script for patching CSR8x11 A06 (BlueCore 7) ingroup btx_csr

btx_csr_get_script__PB_109_CSR8811_REV16 Function

File

csr.h

C

```
const btx_csr_script_t* btx_csr_get_script__PB_109_CSR8811_REV16();
```

Description

brief Return script for patching CSR8x11 A08 (BlueCore 7) ingroup btx_csr

btx_csr_get_script__PB_173_CSR8X11_REV1 Function

File

csr.h

C

```
const btx_csr_script_t* btx_csr_get_script__PB_173_CSR8X11_REV1();
```

Description

brief Return script for patching CSR8x11 A12 (BlueCore 7) ingroup btx_csr

btx_csr_init Function

File

csr.h

C

```
void btx_csr_init();
```

Description

brief Initialize CSR support layer. ingroup btx_csr

details This function initializes all internal variables of the CSR support layer. CSR controllers use vendor specific event (0xFF) to carry the BCCMD protocol. They also do not report number of completed packets for BCCMD commands. This function installs a vendor specific event handler that makes sure that callback are called on corresponding vendor specific commands and the number of free command buffers in the controller is kept correct.

btx_csr_init_hq_script Function

File

csr.h

C

```
void btx_csr_init_hq_script(const btx_csr_script_t* script, btx_csr_exec_hq_script_buffer_t* buffer);
```

Description

This is function btx_csr_init_hq_script.

btx_csr_patch_controller Function

File

csr.h

C

```
void btx_csr_patch_controller(const btx_csr_get_script_fp* scripts, bt_int script_count,
btx_csr_exec_script_buffer_t* buffer, btx_csr_exec_script_callback_fp callback, void* callback_param);
```

Description

brief Patch controller's firmware ingroup btx_csr

details This function executes a script that patches the controller's firmware. Each entry of the c scripts array must be a complete patch script for a particular controller model and revision. btx_csr_patch_controller() reads the revision number from the controller then tries to find the corresponding script in the c scripts. If there is a matching script it is loaded to the controller and c callback is called with the first parameter TRUE. If no suitable script found c callback is called with the first parameter FALSE.

param scripts Array of patch scripts. param script_count The number of scripts in c scripts. param buffer A buffer for storing temporary data needed for script execution. param callback The callback function that will be called when the script has been executed. param callback_param A pointer to arbitrary data to be passed to the c callback callback..

btx_csr_register_bccmd_listener Function

File

csr.h

C

```
bt_bool btx_csr_register_bccmd_listener(btx_csr_bccmd_listener_t* listener);
```

Description

This is function btx_csr_register_bccmd_listener.

btx_csr_send_dsp_config_data Function

File

csr.h

C

```
bt_bool btx_csr_send_dsp_config_data(bt_uint total_config_blocks, bt_uint seq_no, bt_uint status, const
bt_uint* data, bt_uint data_word_count, bt_hci_cmd_callback_fp callback, void* callback_param);
```

Description

This is function btx_csr_send_dsp_config_data.

btx_csr_send_next_hq_script_packet Function

File

csr.h

C

```
bt_bool btx_csr_send_next_hq_script_packet(bt_uint seq_no, btx_csr_exec_hq_script_buffer_t* buffer);
```

Description

This is function btx_csr_send_next_hq_script_packet.

btx_csr_unregister_bccmd_listener Function

File

csr.h

C

```
void btx_csr_unregister_bccmd_listener(btx_csr_bccmd_listener_t* listener);
```

Description

This is function btx_csr_unregister_bccmd_listener.

btx_ti_a3dp_sink_close_stream Function

File

ti.h

C

```
bt_bool btx_ti_a3dp_sink_close_stream(bt_hci_cmd_callback_fp callback);
```

Description

This is function btx_ti_a3dp_sink_close_stream.

btx_ti_a3dp_sink_codec_config Function

File

ti.h

C

```
bt_bool btx_ti_a3dp_sink_codec_config(bt_byte channels, bt_byte sample_frequency, bt_byte channel_mode,
bt_byte blocks, bt_byte sub_bands, bt_byte allocation_method, bt_hci_cmd_callback_fp callback);
```

Description

This is function btx_ti_a3dp_sink_codec_config.

btx_ti_a3dp_sink_open_stream Function

File

ti.h

C

```
bt_bool btx_ti_a3dp_sink_open_stream(bt_byte acl_handle, bt_uint l2cap_cid, bt_hci_cmd_callback_fp
callback);
```

Description

This is function btx_ti_a3dp_sink_open_stream.

btx_ti_a3dp_sink_start_stream Function

File

ti.h

C

```
bt_bool btx_ti_a3dp_sink_start_stream(bt_hci_cmd_callback_fp callback);
```

Description

This is function btx_ti_a3dp_sink_start_stream.

btx_ti_a3dp_sink_stop_stream Function

File

ti.h

C

```
bt_bool btx_ti_a3dp_sink_stop_stream(bt_hci_cmd_callback_fp callback);
```

Description

This is function btx_ti_a3dp_sink_stop_stream.

btx_ti_avpr_debug Function

File

ti.h

C

```
bt_bool btx_ti_avpr_debug(bt_hci_cmd_callback_fp callback);
```

Description

This is function btx_ti_avpr_debug.

btx_ti_avpr_enable Function

File

ti.h

C

```
bt_bool btx_ti_avpr_enable(bt_byte enable, bt_byte load_a3dp_code, bt_byte a3dp_role,
bt_hci_cmd_callback_fp callback);
```

Description

This is function btx_ti_avpr_enable.

btx_ti_get_script__BL6450L_BT_Service_Pack_2_14 Function

File

ti.h

C

```
const btx_ti_script_t* btx_ti_get_script__BL6450L_BT_Service_Pack_2_14();
```

Description

CC2564 Scripts (Service pack 2.14)

btx_ti_get_script__BL6450L_BT_Service_Pack_2_14_BLE_AddOn Function

File

ti.h

C

```
const btx_ti_script_t* btx_ti_get_script__BL6450L_BT_Service_Pack_2_14_BLE_AddOn();
```

Description

This is function btx_ti_get_script__BL6450L_BT_Service_Pack_2_14_BLE_AddOn.

btx_ti_get_script__CC2564B_BT_Service_Pack_1_0 Function

File

ti.h

C

```
const btx_ti_script_t* btx_ti_get_script__CC2564B_BT_Service_Pack_1_0();
```

Description

CC2564B Scripts (Service pack 1.0)

btx_ti_get_script__CC2564B_BT_Service_Pack_1_0_BLE_AddOn Function

File

ti.h

C

```
const btx_ti_script_t* btx_ti_get_script__CC2564B_BT_Service_Pack_1_0_BLE_AddOn();
```

Description

This is function btx_ti_get_script__CC2564B_BT_Service_Pack_1_0_BLE_AddOn.

btx_ti_get_script__CC2564B_BT_Service_Pack_1_1 Function

File

ti.h

C

```
const btx_ti_script_t* btx_ti_get_script__CC2564B_BT_Service_Pack_1_1();
```

Description

CC2564B Scripts (Service pack 1.1)

btx_ti_get_script__CC2564B_BT_Service_Pack_1_1_BLE_AddOn Function

File

ti.h

C

```
const btx_ti_script_t* btx_ti_get_script__CC2564B_BT_Service_Pack_1_1_BLE_AddOn();
```

Description

This is function btx_ti_get_script__CC2564B_BT_Service_Pack_1_1_BLE_AddOn.

btx_ti_get_script__CC2564B_BT_Service_Pack_1_2 Function

File

ti.h

C

```
const btx_ti_script_t* btx_ti_get_script__CC2564B_BT_Service_Pack_1_2();
```

Description

CC2564B Scripts (Service pack 1.2)

btx_ti_get_script__CC2564B_BT_Service_Pack_1_2_AVPR_AddOn Function

File

ti.h

C

```
const btx_ti_script_t* btx_ti_get_script__CC2564B_BT_Service_Pack_1_2_AVPR_AddOn();
```

Description

This is function btx_ti_get_script__CC2564B_BT_Service_Pack_1_2_AVPR_AddOn.

btx_ti_get_script__CC2564B_BT_Service_Pack_1_2_BLE_AddOn Function

File

ti.h

C

```
const btx_ti_script_t* btx_ti_get_script__CC2564B_BT_Service_Pack_1_2_BLE_AddOn();
```

Description

This is function btx_ti_get_script__CC2564B_BT_Service_Pack_1_2_BLE_AddOn.

btx_ti_le_enable Function

File

ti.h

C

```
bt_bool btx_ti_le_enable(bt_byte enable, bt_byte load_le_code, bt_hci_cmd_callback_fp callback);
```

Description

This is function btx_ti_le_enable.

btx_ti_set_afh_mode Function

File

ti.h

C

```
bt_bool btx_ti_set_afh_mode(bt_hci_hconn_t hconn, bt_bool enable, bt_hci_cmd_callback_fp callback);
```

Description

This is function btx_ti_set_afh_mode.

btx_ti_write_codec_config Function

File

ti.h

C

```
bt_bool btx_ti_write_codec_config(const btx_ti_codec_config_t* codec_config, bt_hci_cmd_callback_fp
callback);
```

Description

This is function btx_ti_write_codec_config.

btx_csr_exec_hq_script Function

File

csr.h

C

```
void btx_csr_exec_hq_script(const btx_csr_script_t* script, btx_csr_exec_hq_script_buffer_t* buffer,
btx_csr_exec_hq_script_callback_fp callback, void* callback_param);
```

Description

This is function btx_csr_exec_hq_script.

Misc. Data Types and Constants

CSR_SNK_ADC Macro

File

csr.h

C

```
#define CSR_SNK_ADC 3
```

Description

This is macro CSR_SNK_ADC.

CSR_SNK_FASTPIPE Macro

File

csr.h

C

```
#define CSR_SNK_FASTPIPE 8
```

Description

This is macro CSR_SNK_FASTPIPE.

CSR_SNK_FM Macro

File

csr.h

C

```
#define CSR_SNK_FM 4
```

Description

This is macro CSR_SNK_FM.

CSR_SNK_I2S Macro

File

csr.h

C

```
#define CSR_SNK_I2S 2
```

Description

This is macro CSR_SNK_I2S.

CSR_SNK_L2CAP Macro

File

csr.h

C

```
#define CSR_SNK_L2CAP 7
```

Description

This is macro CSR_SNK_L2CAP.

CSR_SNK_PCM Macro

File

csr.h

C

```
#define CSR_SNK_PCM 1
```

Description

This is macro CSR_SNK_PCM.

CSR_SNK_SCO Macro

File

csr.h

C

```
#define CSR_SNK_SCO 9
```

Description

This is macro CSR_SNK_SCO.

CSR_SNK_SPDIF Macro

File

csr.h

C

```
#define CSR_SNK_SPDIF 5
```

Description

This is macro CSR_SNK_SPDIF.

CSR_SRC_ADC Macro

File

csr.h

C

```
#define CSR_SRC_ADC 3
```

Description

This is macro CSR_SRC_ADC.

CSR_SRC_FASTPIPE Macro

File

csr.h

C

```
#define CSR_SRC_FASTPIPE 8
```

Description

This is macro CSR_SRC_FASTPIPE.

CSR_SRC_FM Macro

File

csr.h

C

```
#define CSR_SRC_FM 4
```

Description

This is macro CSR_SRC_FM.

CSR_SRC_I2S Macro

File

csr.h

C

```
#define CSR_SRC_I2S 2
```

Description

This is macro CSR_SRC_I2S.

CSR_SRC_L2CAP Macro

File

csr.h

C

```
#define CSR_SRC_L2CAP 7
```

Description

This is macro CSR_SRC_L2CAP.

CSR_SRC_MIC Macro

File

csr.h

C

```
#define CSR_SRC_MIC 6
```

Description

This is macro CSR_SRC_MIC.

CSR_SRC_PCM Macro

File

csr.h

C

```
#define CSR_SRC_PCM 1
```

Description

This is macro CSR_SRC_PCM.

CSR_SRC_SCO Macro

File

csr.h

C

```
#define CSR_SRC_SCO 9
```

Description

This is macro CSR_SRC_SCO.

CSR_SRC_SPDIF Macro

File

csr.h

C

```
#define CSR_SRC_SPDIF 5
```

Description

This is macro CSR_SRC_SPDIF.

CSR_VARID_CACHED_TEMPERATURE Macro

File

csr.h

C

```
#define CSR_VARID_CACHED_TEMPERATURE 0x2872
```

Description

This is macro CSR_VARID_CACHED_TEMPERATURE.

CSR_VARID_ENABLE_SCO_STREAMS Macro

File

csr.h

C

```
#define CSR_VARID_ENABLE_SCO_STREAMS 0x4876
```

Description

This is macro CSR_VARID_ENABLE_SCO_STREAMS.

CSR_VARID_MAP_SCO_AUDIO Macro

File

csr.h

C

```
#define CSR_VARID_MAP_SCO_AUDIO 0x506a
```

Description

This is macro CSR_VARID_MAP_SCO_AUDIO.

CSR_VARID_PIO Macro

File

csr.h

C

```
#define CSR_VARID_PIO 0x681f
```

Description

This is macro CSR_VARID_PIO.

CSR_VARID_PIO_DIRECTION_MASK Macro

File

csr.h

C

```
#define CSR_VARID_PIO_DIRECTION_MASK 0x681e
```

Description

This is macro CSR_VARID_PIO_DIRECTION_MASK.

CSR_VARID_PIO_PROTECT_MASK Macro

File

csr.h

C

```
#define CSR_VARID_PIO_PROTECT_MASK 0x2823
```

Description

This is macro CSR_VARID_PIO_PROTECT_MASK.

CSR_VARID_RSSI_ACL Macro

File

csr.h

C

```
#define CSR_VARID_RSSI_ACL 0x301d
```

Description

This is macro CSR_VARID_RSSI_ACL.

CSR_VARID_STREAM_CLOSE_SINK Macro

File

csr.h

C

```
#define CSR_VARID_STREAM_CLOSE_SINK 0x486c
```

Description

This is macro CSR_VARID_STREAM_CLOSE_SINK.

CSR_VARID_STREAM_CLOSE_SOURCE Macro

File

csr.h

C

```
#define CSR_VARID_STREAM_CLOSE_SOURCE 0x486b
```

Description

This is macro CSR_VARID_STREAM_CLOSE_SOURCE.

CSR_VARID_STREAM_CONFIGURE Macro

File

csr.h

C

```
#define CSR_VARID_STREAM_CONFIGURE 0x505c
```

Description

This is macro CSR_VARID_STREAM_CONFIGURE.

CSR_VARID_STREAM_CONNECT Macro

File

csr.h

C

```
#define CSR_VARID_STREAM_CONNECT 0x505e
```

Description

This is macro CSR_VARID_STREAM_CONNECT.

CSR_VARID_STREAM_GET_SINK Macro

File

csr.h

C

```
#define CSR_VARID_STREAM_GET_SINK 0x505b
```

Description

This is macro CSR_VARID_STREAM_GET_SINK.

CSR_VARID_STREAM_GET_SOURCE Macro

File

csr.h

C

```
#define CSR_VARID_STREAM_GET_SOURCE 0x505a
```

Description

This is macro CSR_VARID_STREAM_GET_SOURCE.

GETRESP_BAD_REQ Macro

File

csr.h

C

```
#define GETRESP_BAD_REQ 0x0004
```

Description

This is macro GETRESP_BAD_REQ.

GETRESP_ERROR Macro

File

csr.h

C

```
#define GETRESP_ERROR 0x0008
```

Description

This is macro GETRESP_ERROR.

GETRESP_NO_ACCESS Macro

File

csr.h

C

```
#define GETRESP_NO_ACCESS 0x0005
```

Description

This is macro GETRESP_NO_ACCESS.

GETRESP_NO_SUCH_VARID Macro

File

csr.h

C

```
#define GETRESP_NO_SUCH_VARID 0x0001
```

Description

This is macro GETRESP_NO_SUCH_VARID.

GETRESP_NO_VALUE Macro

File

csr.h

C

```
#define GETRESP_NO_VALUE 0x0003
```

Description

This is macro GETRESP_NO_VALUE.

GETRESP_OK Macro

File

csr.h

C

```
#define GETRESP_OK 0x0000
```

Description

BCCMD GETRESP status codes

GETRESP_PERMISSION_DENIED Macro

File

csr.h

C

```
#define GETRESP_PERMISSION_DENIED 0x0009
```

Description

This is macro GETRESP_PERMISSION_DENIED.

GETRESP_READ_ONLY Macro

File

csr.h

C

```
#define GETRESP_READ_ONLY 0x0006
```

Description

This is macro GETRESP_READ_ONLY.

GETRESP_TOO_BIG Macro

File

csr.h

C

```
#define GETRESP_TOO_BIG 0x0002
```

Description

This is macro GETRESP_TOO_BIG.

GETRESP_WRITE_ONLY Macro

File

csr.h

C

```
#define GETRESP_WRITE_ONLY 0x0007
```

Description

This is macro GETRESP_WRITE_ONLY.

H Macro

File

patch.h

C

```
#define H(ps_key, val_len, seq) \
    0x01,          /* HCI command packet indicator.*/ \
    0x00, 0xfc,    /* HCI opcode { OGF=0x3f OCF=0x00 } */ \
                   /*     OGF=0x3f implies a manufacturer-sepcific command. */ \
                   /* Parameter length. */ \
    1 + (5 + 3 + val_len) * 2, \
    0xc2,          /* CSR payload descriptor: */ \
                   /*    bit 7: Last fragment = 1 */ \
                   /*    bit 6: First fragment = 1 */ \
                   /*    bits 5-0 Channel ID = 2 (BCCMD) */ \
    W(0x0002),     /* SETREQ message */ \
    W(5+3+val_len), /* length*/ \
    W(seq),        /* sequence number */ \
    W(0x7003),     /* Variable ID of PS command */ \
    W(0x0000),     /* Status (must be 0 in SETREQ) */ \
    \
    W(ps_key),     /* PS key */ \
    W(val_len),    /* Value length in 16-bit words */ \
    W(0)           /* PS key store (0 = use default) */
```

Description

PS key store (0 = use default)

PS_DEFAULT Macro

File

csr.h

C

```
#define PS_DEFAULT 0x0000
```

Description

This is macro PS_DEFAULT.

PS_F Macro

File

csr.h

C

```
#define PS_F 0x0002
```

Description

This is macro PS_F.

PS_I Macro

File

csr.h

C

```
#define PS_I 0x0001
```

Description

This is macro PS_I.

PS_RAM Macro

File

csr.h

C

```
#define PS_RAM 0x0008
```

Description

This is macro PS_RAM.

PS_ROM Macro

File

csr.h

C

```
#define PS_ROM 0x0004
```

Description

This is macro PS_ROM.

PSKEY_ANA_FREQ Macro

File

csr.h

C

```
#define PSKEY_ANA_FREQ 0x01fe
```

Description

This is macro PSKEY_ANA_FREQ.

PSKEY_BDADDR Macro

File

csr.h

C

```
#define PSKEY_BDADDR 0x0001
```

Description

Select PS key definitions

PSKEY_CLOCK_REQUEST_ENABLE Macro

File

csr.h

C

```
#define PSKEY_CLOCK_REQUEST_ENABLE 0x0246
```

Description

This is macro PSKEY_CLOCK_REQUEST_ENABLE.

PSKEY_DEEP_SLEEP_CLEAR_RTS Macro

File

csr.h

C

```
#define PSKEY_DEEP_SLEEP_CLEAR_RTS 0x0252
```

Description

This is macro PSKEY_DEEP_SLEEP_CLEAR_RTS.

PSKEY_DEEP_SLEEP_STATE Macro

File

csr.h

C

```
#define PSKEY_DEEP_SLEEP_STATE 0x0229
```

Description

This is macro PSKEY_DEEP_SLEEP_STATE.

PSKEY_DEEP_SLEEP_USE_EXTERNAL_CLOCK Macro

File

csr.h

C

```
#define PSKEY_DEEP_SLEEP_USE_EXTERNAL_CLOCK 0x03c3
```

Description

This is macro PSKEY_DEEP_SLEEP_USE_EXTERNAL_CLOCK.

PSKEY_DEEP_SLEEP_WAKE_CTS Macro

File

csr.h

C

```
#define PSKEY_DEEP_SLEEP_WAKE_CTS 0x023c
```

Description

This is macro PSKEY_DEEP_SLEEP_WAKE_CTS.

PSKEY_DIGITAL_AUDIO_BITS_PER_SAMPLE Macro

File

csr.h

C

```
#define PSKEY_DIGITAL_AUDIO_BITS_PER_SAMPLE 0x01db
```

Description

This is macro PSKEY_DIGITAL_AUDIO_BITS_PER_SAMPLE.

PSKEY_DIGITAL_AUDIO_CONFIG Macro

File

csr.h

C

```
#define PSKEY_DIGITAL_AUDIO_CONFIG 0x01d9
```

Description

This is macro PSKEY_DIGITAL_AUDIO_CONFIG.

PSKEY_HCI_NOP_DISABLE Macro

File

csr.h

C

```
#define PSKEY_HCI_NOP_DISABLE 0x00f2
```

Description

This is macro PSKEY_HCI_NOP_DISABLE.

PSKEY_HOST_INTERFACE Macro

File

csr.h

C

```
#define PSKEY_HOST_INTERFACE 0x01f9
```

Description

This is macro PSKEY_HOST_INTERFACE.

PSKEY_HOSTIO_MAP_SCO_PCM Macro

File

csr.h

C

```
#define PSKEY_HOSTIO_MAP_SCO_PCM 0x01ab
```

Description

This is macro PSKEY_HOSTIO_MAP_SCO_PCM.

PSKEY_HOSTIO_UART_RESET_TIMEOUT Macro

File

csr.h

C

```
#define PSKEY_HOSTIO_UART_RESET_TIMEOUT 0x01a4
```

Description

This is macro PSKEY_HOSTIO_UART_RESET_TIMEOUT.

PSKEY_PCM_PULL_CONTROL Macro

File

csr.h

C

```
#define PSKEY_PCM_PULL_CONTROL 0x01e2
```

Description

This is macro PSKEY_PCM_PULL_CONTROL.

PSKEY_PIO_DEEP_SLEEP_EITHER_LEVEL Macro

File

csr.h

C

```
#define PSKEY_PIO_DEEP_SLEEP_EITHER_LEVEL 0x21bd
```

Description

This is macro PSKEY_PIO_DEEP_SLEEP_EITHER_LEVEL.

PSKEY_UART_BAUDRATE Macro

File

csr.h

C

```
#define PSKEY_UART_BAUDRATE 0x01be
```

Description

This is macro PSKEY_UART_BAUDRATE.

PSKEY_UART_BITRATE Macro

File

csr.h

C

```
#define PSKEY_UART_BITRATE 0x01ea
```

Description

This is macro PSKEY_UART_BITRATE.

PSKEY_UART_CONFIG_BCSP Macro

File

csr.h

C

```
#define PSKEY_UART_CONFIG_BCSP 0x01bf
```

Description

This is macro PSKEY_UART_CONFIG_BCSP.

PSKEY_UART_CONFIG_H4 Macro

File

csr.h

C

```
#define PSKEY_UART_CONFIG_H4 0x01c0
```

Description

This is macro PSKEY_UART_CONFIG_H4.

PSKEY_UART_CONFIG_H5 Macro

File

csr.h

C

```
#define PSKEY_UART_CONFIG_H5 0x01c1
```

Description

This is macro PSKEY_UART_CONFIG_H5.

PSKEY_UART_TX_WINDOW_SIZE Macro

File

csr.h

C

```
#define PSKEY_UART_TX_WINDOW_SIZE 0x01c6
```

Description

This is macro PSKEY_UART_TX_WINDOW_SIZE.

PSKEY_VM_DISABLE Macro

File

csr.h

C

```
#define PSKEY_VM_DISABLE 0x025d
```

Description

This is macro PSKEY_VM_DISABLE.

SET_PS_VALUE_BDADDR Macro

File

csr.h

C

```
#define SET_PS_VALUE_BDADDR(key, m, l) key, 4, (uint16_t)((((uint32_t)l) >> 16) & 0xFF), (uint16_t)(l
&0xFFFF), (uint16_t)((((uint32_t)l) >> 24) & 0xFF), (uint16_t)(m &0xFFFF)
```

Description

This is macro SET_PS_VALUE_BDADDR.

SET_PS_VALUE_UINT16 Macro

File

csr.h

C

```
#define SET_PS_VALUE_UINT16(key, value) key, 1, value
```

Description

Macros for defining lists of PS values for use with btx_csr_set_ps_vars().

SET_PS_VALUE_UINT32 Macro

File

csr.h

C

```
#define SET_PS_VALUE_UINT32(key, value) key, 2, (uint16_t)((((uint32_t)value) >> 16) & 0xFFFF),
(uint16_t)(value &0xFFFF)
```

Description

This is macro SET_PS_VALUE_UINT32.

W Macro

File

patch.h

C

```
#define W(w) _W((bt_uint)(w))
```

Description

This is macro W.

BTX_TI_EXEC_SCRIPT_OEM_RX_BUFFER_SIZE Macro

File

ti.h

C

```
#define BTX_TI_EXEC_SCRIPT_OEM_RX_BUFFER_SIZE 10
```

Description

This is macro BTX_TI_EXEC_SCRIPT_OEM_RX_BUFFER_SIZE.

btx_ti_get_script__CC2560_ServicePack Macro

File

ti.h

C

```
#define btx_ti_get_script__CC2560_ServicePack btx_ti_get_script__BL6450_2_0_BT_Service_Pack_2_44
```

Description

Alias for the latest script

btx_ti_get_script__CC2564_BLE_Init Macro

File

ti.h

C

```
#define btx_ti_get_script__CC2564_BLE_Init btx_ti_get_script__BL6450L_BT_Service_Pack_2_14_BLE_AddOn
```

Description

This is macro btx_ti_get_script__CC2564_BLE_Init.

btx_ti_get_script__CC2564_ServicePack Macro

File

ti.h

C

```
#define btx_ti_get_script__CC2564_ServicePack btx_ti_get_script__BL6450L_BT_Service_Pack_2_14
```

Description

Aliases for latest scripts

btx_ti_get_script__CC2564B_BLE_Init Macro

File

ti.h

C

```
#define btx_ti_get_script__CC2564B_BLE_Init btx_ti_get_script__CC2564B_BT_Service_Pack_1_2_BLE_AddOn
```

Description

This is macro btx_ti_get_script__CC2564B_BLE_Init.

btx_ti_get_script__CC2564B_ServicePack Macro

File

ti.h

C

```
#define btx_ti_get_script__CC2564B_ServicePack btx_ti_get_script__CC2564B_BT_Service_Pack_1_2
```

Description

Aliases for latest scripts

BTX_TI_MODULATION_SCHEME_8_DPSK Macro

File

ti.h

C

```
#define BTX_TI_MODULATION_SCHEME_8_DPSK 3
```

Description

This is macro BTX_TI_MODULATION_SCHEME_8_DPSK.

BTX_TI_MODULATION_SCHEME_CW Macro

File

ti.h

C

```
#define BTX_TI_MODULATION_SCHEME_CW 0
```

Description

This is macro BTX_TI_MODULATION_SCHEME_CW.

BTX_TI_MODULATION_SCHEME_GFSK Macro

File

ti.h

C

```
#define BTX_TI_MODULATION_SCHEME_GFSK 1
```

Description

This is macro BTX_TI_MODULATION_SCHEME_GFSK.

BTX_TI_MODULATION_SCHEME_P4_DPSK Macro

File

ti.h

C

```
#define BTX_TI_MODULATION_SCHEME_P4_DPSK 2
```

Description

This is macro BTX_TI_MODULATION_SCHEME_P4_DPSK.

BTX_TI_MODULATION_TYPE_EDR2 Macro

File

ti.h

C

```
#define BTX_TI_MODULATION_TYPE_EDR2 1
```

Description

This is macro BTX_TI_MODULATION_TYPE_EDR2.

BTX_TI_MODULATION_TYPE_EDR3 Macro

File

ti.h

C

```
#define BTX_TI_MODULATION_TYPE_EDR3 2
```

Description

This is macro BTX_TI_MODULATION_TYPE_EDR3.

BTX_TI_MODULATION_TYPE_GFSK Macro

File

ti.h

C

```
#define BTX_TI_MODULATION_TYPE_GFSK 0
```

Description

This is macro BTX_TI_MODULATION_TYPE_GFSK.

BTX_TI_TEST_PATTERN_ALL_0 Macro

File

ti.h

C

```
#define BTX_TI_TEST_PATTERN_ALL_0 4
```

Description

This is macro BTX_TI_TEST_PATTERN_ALL_0.

BTX_TI_TEST_PATTERN_ALL_1 Macro

File

ti.h

C

```
#define BTX_TI_TEST_PATTERN_ALL_1 3
```

Description

This is macro BTX_TI_TEST_PATTERN_ALL_1.

BTX_TI_TEST_PATTERN_F0F0 Macro

File

ti.h

C

```
#define BTX_TI_TEST_PATTERN_F0F0 5
```

Description

This is macro BTX_TI_TEST_PATTERN_F0F0.

BTX_TI_TEST_PATTERN_FF00 Macro

File

ti.h

C

```
#define BTX_TI_TEST_PATTERN_FF00 6
```

Description

This is macro BTX_TI_TEST_PATTERN_FF00.

BTX_TI_TEST_PATTERN_PN15 Macro

File

ti.h

C

```
#define BTX_TI_TEST_PATTERN_PN15 1
```

Description

This is macro BTX_TI_TEST_PATTERN_PN15.

BTX_TI_TEST_PATTERN_PN9 Macro

File

ti.h

C

```
#define BTX_TI_TEST_PATTERN_PN9 0
```

Description

This is macro BTX_TI_TEST_PATTERN_PN9.

BTX_TI_TEST_PATTERN_USER Macro

File

ti.h

C

```
#define BTX_TI_TEST_PATTERN_USER 7
```

Description

This is macro BTX_TI_TEST_PATTERN_USER.

BTX_TI_TEST_PATTERN_Z0Z0 Macro

File

ti.h

C

```
#define BTX_TI_TEST_PATTERN_Z0Z0 2
```

Description

This is macro BTX_TI_TEST_PATTERN_Z0Z0.

_W Macro

File

patch.h

C

```
#define _W(w) (bt_byte)((w) & 0xFF), (bt_byte)((w) >> 8)
```

Description

This is macro _W.

_btx_csr_autobaud_buffer_t Structure

File

csr.h

C

```
struct _btx_csr_autobaud_buffer_t {
  bt_byte* recv_buffer;
  bt_uint recv_buffer_len;
  btx_csr_autobaud_callback_fp callback;
  void* callback_param;
};
```

Description

This is record _btx_csr_autobaud_buffer_t.

_btx_csr_exec_script_buffer_t Structure

File

csr.h

C

```
struct _btx_csr_exec_script_buffer_t {
  const btx_csr_script_t* script;
  btx_csr_exec_script_callback_fp callback;
  void* callback_param;
  bt_int current_packet;
};
```

Description

This is record _btx_csr_exec_script_buffer_t.

_btx_csr_set_ps_vars_buffer_t Structure

File

csr.h

C

```
struct _btx_csr_set_ps_vars_buffer_t {
  const bt_uint* ps_vars;
  btx_csr_set_ps_vars_callback_fp callback;
  void* callback_param;
  bt_uint current_var;
};
```

Description

This is record _btx_csr_set_ps_vars_buffer_t.

_btx_ti_exec_script_buffer_t Structure

File

ti.h

C

```
struct _btx_ti_exec_script_buffer_t {
  const btx_ti_script_t* script;
  btx_ti_completion_callback_fp callback;
  void* callback_param;
  bt_int current_packet;
};
```

Description

This is record _btx_ti_exec_script_buffer_t.

_btx_ti_exec_script_oem_buffer_t Structure

File

ti.h

C

```
struct _btx_ti_exec_script_oem_buffer_t {
  const btx_ti_script_t* script;
  btx_ti_exec_script_oem_callback_fp callback;
  void* callback_param;
  bt_int current_packet;
  bt_int state;
  bt_byte rx_buffer[BTX_TI_EXEC_SCRIPT_OEM_RX_BUFFER_SIZE];
};
```

Description

This is record _btx_ti_exec_script_oem_buffer_t.

btx_csr_autobaud_buffer_t Type

File

csr.h

C

```
typedef struct _btx_csr_autobaud_buffer_t btx_csr_autobaud_buffer_t;
```

Description

This is type btx_csr_autobaud_buffer_t.

btx_csr_autobaud_callback_fp Type

File

csr.h

C

```
typedef void (* btx_csr_autobaud_callback_fp)(bt_bool success, btx_csr_autobaud_buffer_t* buffer);
```

Description

This is type btx_csr_autobaud_callback_fp.

btx_csr_bccmd_header_t Structure

File

csr.h

C

```
typedef struct _btx_csr_bccmd_header_s {
  bt_uint type;
  bt_uint len;
  bt_uint seq_no;
  bt_uint var_id;
  bt_uint status;
  bt_byte* payload;
} btx_csr_bccmd_header_t;
```

Description

This is type btx_csr_bccmd_header_t.

btx_csr_cached_temperature_t Structure

File

csr.h

C

```
typedef struct _btx_csr_cached_temperature_s {
  btx_csr_bccmd_header_t message;
  bt_uint temperature;
} btx_csr_cached_temperature_t;
```

Description

This is type btx_csr_cached_temperature_t.

btx_csr_exec_script_buffer_t Type

File

csr.h

C

```
typedef struct _btx_csr_exec_script_buffer_t btx_csr_exec_script_buffer_t;
```

Description

This is type btx_csr_exec_script_buffer_t.

btx_csr_exec_script_callback_fp Type

File

csr.h

C

```
typedef void (* btx_csr_exec_script_callback_fp)(bt_bool success, btx_csr_exec_script_buffer_t* buffer);
```

Description

This is type btx_csr_exec_script_callback_fp.

btx_csr_get_ps_var_callback_fp Type

File

csr.h

C

```
typedef void (* btx_csr_get_ps_var_callback_fp)(bt_uint success, const bt_byte* value, bt_uint len, void*
callback_param);
```

Description

This is type btx_csr_get_ps_var_callback_fp.

btx_csr_get_var_callback_fp Type

File

csr.h

C

```
typedef void (* btx_csr_get_var_callback_fp)(bt_uint status, bt_uint var_id, btx_csr_var_t* var_value,
void* callback_param);
```

Description

This is type btx_csr_get_var_callback_fp.

btx_csr_pio_direction_mask_t Structure

File

csr.h

C

```
typedef struct _btx_csr_pio_direction_mask_s {
  btx_csr_bccmd_header_t message;
  bt_uint direction;
} btx_csr_pio_direction_mask_t;
```

Description

This is type btx_csr_pio_direction_mask_t.

btx_csr_pio_protection_mask_t Structure

File

csr.h

C

```
typedef struct _btx_csr_pio_protection_mask_s {
  btx_csr_bccmd_header_t message;
  bt_uint protection;
} btx_csr_pio_protection_mask_t;
```

Description

This is type btx_csr_pio_protection_mask_t.

btx_csr_pio_t Structure

File

csr.h

C

```
typedef struct _btx_csr_pio_s {
  btx_csr_bccmd_header_t message;
  bt_uint pio;
} btx_csr_pio_t;
```

Description

This is type btx_csr_pio_t.

btx_csr_rssi_acl_t Structure

File

csr.h

C

```
typedef struct _btx_csr_rssi_acl_s {
  btx_csr_bccmd_header_t message;
  bt_hci_hconn_t hconn;
  bt_uint rssi;
} btx_csr_rssi_acl_t;
```

Description

This is type btx_csr_rssi_acl_t.

btx_csr_script_t Structure

File

csr.h

C

```
typedef struct _btx_csr_script_t {
  const bt_byte* const * packets;
  bt_int packet_count;
  bt_uint revision;
} btx_csr_script_t;
```

Description

This is type btx_csr_script_t.

btx_csr_set_ps_vars_buffer_t Type

File

csr.h

C

```
typedef struct _btx_csr_set_ps_vars_buffer_t btx_csr_set_ps_vars_buffer_t;
```

Description

This is type btx_csr_set_ps_vars_buffer_t.

btx_csr_set_ps_vars_callback_fp Type

File

csr.h

C

```
typedef void (* btx_csr_set_ps_vars_callback_fp)(bt_bool success, btx_csr_set_ps_vars_buffer_t* buffer);
```

Description

This is type btx_csr_set_ps_vars_callback_fp.

btx_csr_set_var_callback_fp Type

File

csr.h

C

```
typedef void (* btx_csr_set_var_callback_fp)(bt_uint status, bt_uint var_id, btx_csr_var_t* var_value,
void* callback_param);
```

Description

This is type btx_csr_set_var_callback_fp.

btx_csr_strm_get_sink_t Structure

File

csr.h

C

```
typedef struct _btx_csr_strm_get_sink_s {
  btx_csr_bccmd_header_t message;
  bt_uint sink_id;
} btx_csr_strm_get_sink_t;
```

Description

This is type btx_csr_strm_get_sink_t.

btx_csr_strm_get_source_t Structure

File

csr.h

C

```
typedef struct _btx_csr_strm_get_source_s {
  btx_csr_bccmd_header_t message;
  bt_uint source_id;
} btx_csr_strm_get_source_t;
```

Description

This is type btx_csr_strm_get_source_t.

btx_csr_var_t Union

File

csr.h

C

```
typedef union _btx_csr_var_u {
  btx_csr_bccmd_header_t message;
  btx_csr_cached_temperature_t cached_temperature;
  btx_csr_rssi_acl_t rssi_acl;
  btx_csr_pio_t pio;
  btx_csr_pio_direction_mask_t pio_direction;
  btx_csr_pio_protection_mask_t pio_protection;
  btx_csr_strm_get_sink_t strm_get_sink;
  btx_csr_strm_get_source_t strm_get_source;
  btx_csr_create_operator_c_t create_operator;
  btx_csr_strm_connect_t strm_connect;
} btx_csr_var_t;
```

Description

This is type btx_csr_var_t.

btx_ti_completion_callback_fp Type

File

ti.h

C

```
typedef void (* btx_ti_completion_callback_fp)(bt_bool success, btx_ti_exec_script_buffer_t* buffer, void*
callback_param);
```

Description

This is type btx_ti_completion_callback_fp.

btx_ti_exec_script_buffer_t Type

File

ti.h

C

```
typedef struct _btx_ti_exec_script_buffer_t btx_ti_exec_script_buffer_t;
```

Description

This is type btx_ti_exec_script_buffer_t.

btx_ti_exec_script_oem_buffer_t Type

File

ti.h

C

```
typedef struct _btx_ti_exec_script_oem_buffer_t btx_ti_exec_script_oem_buffer_t;
```

Description

This is type btx_ti_exec_script_oem_buffer_t.

btx_ti_exec_script_oem_callback_fp Type

File

ti.h

C

```
typedef void (* btx_ti_exec_script_oem_callback_fp)(bt_bool success, btx_ti_exec_script_oem_buffer_t*
buffer, void* callback_param);
```

Description

This is type btx_ti_exec_script_oem_callback_fp.

btx_ti_script_t Structure

File

ti.h

C

```
typedef struct _btx_ti_script_t {
  const bt_byte* const * packets;
  bt_int packet_count;
  bt_byte fw_version_x;
  bt_byte fw_version_z;
} btx_ti_script_t;
```

Description

This is type btx_ti_script_t.

BTX_TI_A3DP_ROLE_SINK Macro

File

ti.h

C

```
#define BTX_TI_A3DP_ROLE_SINK 1
```

Description

This is macro BTX_TI_A3DP_ROLE_SINK.

BTX_TI_A3DP_ROLE_SOURCE Macro

File

ti.h

C

```
#define BTX_TI_A3DP_ROLE_SOURCE 0
```

Description

This is macro BTX_TI_A3DP_ROLE_SOURCE.

BTX_TI_AVPR_DISABLE Macro

File

ti.h

C

```
#define BTX_TI_AVPR_DISABLE 0
```

Description

This is macro BTX_TI_AVPR_DISABLE.

BTX_TI_AVPR_DO_NOT_LOAD_A3DP_CODE Macro

File

ti.h

C

```
#define BTX_TI_AVPR_DO_NOT_LOAD_A3DP_CODE 0
```

Description

This is macro BTX_TI_AVPR_DO_NOT_LOAD_A3DP_CODE.

BTX_TI_AVPR_ENABLE Macro

File

ti.h

C

```
#define BTX_TI_AVPR_ENABLE 1
```

Description

This is macro BTX_TI_AVPR_ENABLE.

BTX_TI_AVPR_LOAD_A3DP_CODE Macro

File

ti.h

C

```
#define BTX_TI_AVPR_LOAD_A3DP_CODE 1
```

Description

This is macro BTX_TI_AVPR_LOAD_A3DP_CODE.

btx_ti_get_script__CC2564B_AVPR_Init Macro

File

ti.h

C

```
#define btx_ti_get_script__CC2564B_AVPR_Init btx_ti_get_script__CC2564B_BT_Service_Pack_1_2_AVPR_AddOn
```

Description

This is macro btx_ti_get_script__CC2564B_AVPR_Init.

BTX_TI_SBC_ALLOCATION_METHOD_LOUDNESS Macro

File

ti.h

C

```
#define BTX_TI_SBC_ALLOCATION_METHOD_LOUDNESS 0
```

Description

This is macro BTX_TI_SBC_ALLOCATION_METHOD_LOUDNESS.

BTX_TI_SBC_ALLOCATION_METHOD_SNR Macro

File

ti.h

C

```
#define BTX_TI_SBC_ALLOCATION_METHOD_SNR 1
```

Description

This is macro BTX_TI_SBC_ALLOCATION_METHOD_SNR.

BTX_TI_SBC_BLOCKS_12 Macro

File

ti.h

C

```
#define BTX_TI_SBC_BLOCKS_12 12
```

Description

This is macro BTX_TI_SBC_BLOCKS_12.

BTX_TI_SBC_BLOCKS_16 Macro

File

ti.h

C

```
#define BTX_TI_SBC_BLOCKS_16 16
```

Description

This is macro BTX_TI_SBC_BLOCKS_16.

BTX_TI_SBC_BLOCKS_4 Macro

File

ti.h

C

```
#define BTX_TI_SBC_BLOCKS_4 4
```

Description

This is macro BTX_TI_SBC_BLOCKS_4.

BTX_TI_SBC_BLOCKS_8 Macro

File

ti.h

C

```
#define BTX_TI_SBC_BLOCKS_8 8
```

Description

This is macro BTX_TI_SBC_BLOCKS_8.

BTX_TI_SBC_CHANNEL_MODE_DUAL_CHANNEL Macro

File

ti.h

C

```
#define BTX_TI_SBC_CHANNEL_MODE_DUAL_CHANNEL 1
```

Description

This is macro BTX_TI_SBC_CHANNEL_MODE_DUAL_CHANNEL.

BTX_TI_SBC_CHANNEL_MODE_JOINT_STEREO Macro

File

ti.h

C

```
#define BTX_TI_SBC_CHANNEL_MODE_JOINT_STEREO 3
```

Description

This is macro BTX_TI_SBC_CHANNEL_MODE_JOINT_STEREO.

BTX_TI_SBC_CHANNEL_MODE_MONO Macro

File

ti.h

C

```
#define BTX_TI_SBC_CHANNEL_MODE_MONO 0
```

Description

This is macro BTX_TI_SBC_CHANNEL_MODE_MONO.

BTX_TI_SBC_CHANNEL_MODE_STEREO Macro

File

ti.h

C

```
#define BTX_TI_SBC_CHANNEL_MODE_STEREO 2
```

Description

This is macro BTX_TI_SBC_CHANNEL_MODE_STEREO.

BTX_TI_SBC_SAMPLE_FREQUENCY_16000 Macro

File

ti.h

C

```
#define BTX_TI_SBC_SAMPLE_FREQUENCY_16000 0
```

Description

This is macro BTX_TI_SBC_SAMPLE_FREQUENCY_16000.

BTX_TI_SBC_SAMPLE_FREQUENCY_32000 Macro

File

ti.h

C

```
#define BTX_TI_SBC_SAMPLE_FREQUENCY_32000 1
```

Description

This is macro BTX_TI_SBC_SAMPLE_FREQUENCY_32000.

BTX_TI_SBC_SAMPLE_FREQUENCY_44100 Macro

File

ti.h

C

```
#define BTX_TI_SBC_SAMPLE_FREQUENCY_44100 2
```

Description

This is macro BTX_TI_SBC_SAMPLE_FREQUENCY_44100.

BTX_TI_SBC_SAMPLE_FREQUENCY_48000 Macro

File

ti.h

C

```
#define BTX_TI_SBC_SAMPLE_FREQUENCY_48000 3
```

Description

This is macro BTX_TI_SBC_SAMPLE_FREQUENCY_48000.

CSR_I2S_STREAM_CFG_KEY_AUDIO_ATTEN Macro

File

csr.h

C

```
#define CSR_I2S_STREAM_CFG_KEY_AUDIO_ATTEN 0x207
```

Description

This is macro CSR_I2S_STREAM_CFG_KEY_AUDIO_ATTEN.

CSR_I2S_STREAM_CFG_KEY_AUDIO_ATTEN_ENABLE Macro

File

csr.h

C

```
#define CSR_I2S_STREAM_CFG_KEY_AUDIO_ATTEN_ENABLE 0x206
```

Description

This is macro CSR_I2S_STREAM_CFG_KEY_AUDIO_ATTEN_ENABLE.

CSR_I2S_STREAM_CFG_KEY_BITS_PER_SAMPLE Macro

File

csr.h

C

```
#define CSR_I2S_STREAM_CFG_KEY_BITS_PER_SAMPLE 0x20A
```

Description

This is macro CSR_I2S_STREAM_CFG_KEY_BITS_PER_SAMPLE.

CSR_I2S_STREAM_CFG_KEY_CROP_ENABLE Macro

File

csr.h

C

```
#define CSR_I2S_STREAM_CFG_KEY_CROP_ENABLE 0x209
```

Description

This is macro CSR_I2S_STREAM_CFG_KEY_CROP_ENABLE.

CSR_I2S_STREAM_CFG_KEY_JUSTIFY_DELAY Macro

File

csr.h

C

```
#define CSR_I2S_STREAM_CFG_KEY_JUSTIFY_DELAY 0x204
```

Description

This is macro CSR_I2S_STREAM_CFG_KEY_JUSTIFY_DELAY.

CSR_I2S_STREAM_CFG_KEY_JUSTIFY_FORMAT Macro

File

csr.h

C

```
#define CSR_I2S_STREAM_CFG_KEY_JUSTIFY_FORMAT 0x203
```

Description

This is macro CSR_I2S_STREAM_CFG_KEY_JUSTIFY_FORMAT.

CSR_I2S_STREAM_CFG_KEY_JUSTIFY_RESOLUTION Macro

File

csr.h

C

```
#define CSR_I2S_STREAM_CFG_KEY_JUSTIFY_RESOLUTION 0x208
```

Description

This is macro CSR_I2S_STREAM_CFG_KEY_JUSTIFY_RESOLUTION.

CSR_I2S_STREAM_CFG_KEY_MASTER Macro

File

csr.h

C

```
#define CSR_I2S_STREAM_CFG_KEY_MASTER 0x202
```

Description

This is macro CSR_I2S_STREAM_CFG_KEY_MASTER.

CSR_I2S_STREAM_CFG_KEY_MCLK Macro

File

csr.h

C

```
#define CSR_I2S_STREAM_CFG_KEY_MCLK 0x201
```

Description

This is macro CSR_I2S_STREAM_CFG_KEY_MCLK.

CSR_I2S_STREAM_CFG_KEY_POLARITY Macro

File

csr.h

C

```
#define CSR_I2S_STREAM_CFG_KEY_POLARITY 0x205
```

Description

This is macro CSR_I2S_STREAM_CFG_KEY_POLARITY.

CSR_I2S_STREAM_CFG_KEY_RX_START_SAMPLE Macro

File

csr.h

C

```
#define CSR_I2S_STREAM_CFG_KEY_RX_START_SAMPLE 0x20C
```

Description

This is macro CSR_I2S_STREAM_CFG_KEY_RX_START_SAMPLE.

CSR_I2S_STREAM_CFG_KEY_SYNC_RATE Macro

File

csr.h

C

```
#define CSR_I2S_STREAM_CFG_KEY_SYNC_RATE 0x200
```

Description

This is macro CSR_I2S_STREAM_CFG_KEY_SYNC_RATE.

CSR_I2S_STREAM_CFG_KEY_TX_START_SAMPLE Macro

File

csr.h

C

```
#define CSR_I2S_STREAM_CFG_KEY_TX_START_SAMPLE 0x20B
```

Description

This is macro CSR_I2S_STREAM_CFG_KEY_TX_START_SAMPLE.

CSR_MAX_HQ_PACKET_LEN Macro

File

csr.h

C

```
#define CSR_MAX_HQ_PACKET_LEN 129
```

Description

This is macro CSR_MAX_HQ_PACKET_LEN.

CSR_MSG_TYPE_GETREQ Macro

File

csr.h

C

```
#define CSR_MSG_TYPE_GETREQ 0
```

Description

This is macro CSR_MSG_TYPE_GETREQ.

CSR_MSG_TYPE_GETRESP Macro

File

csr.h

C

```
#define CSR_MSG_TYPE_GETRESP 1
```

Description

This is macro CSR_MSG_TYPE_GETRESP.

CSR_MSG_TYPE_SETREQ Macro

File

csr.h

C

```
#define CSR_MSG_TYPE_SETREQ 2
```

Description

This is macro CSR_MSG_TYPE_SETREQ.

CSR_PCM_STREAM_CFG_KEY_LSB_FIRST Macro

File

csr.h

C

```
#define CSR_PCM_STREAM_CFG_KEY_LSB_FIRST 0x108
```

Description

This is macro CSR_PCM_STREAM_CFG_KEY_LSB_FIRST.

CSR_PCM_STREAM_CFG_KEY_MANCHESTER Macro

File

csr.h

C

```
#define CSR_PCM_STREAM_CFG_KEY_MANCHESTER 0x104
```

Description

This is macro CSR_PCM_STREAM_CFG_KEY_MANCHESTER.

CSR_PCM_STREAM_CFG_KEY_MANCHESTER_SLAVE Macro

File

csr.h

C

```
#define CSR_PCM_STREAM_CFG_KEY_MANCHESTER_SLAVE 0x106
```

Description

This is macro CSR_PCM_STREAM_CFG_KEY_MANCHESTER_SLAVE.

CSR_PCM_STREAM_CFG_KEY_MASTER Macro

File

csr.h

C

```
#define CSR_PCM_STREAM_CFG_KEY_MASTER 0x102
```

Description

This is macro CSR_PCM_STREAM_CFG_KEY_MASTER.

CSR_PCM_STREAM_CFG_KEY_MCLK Macro

File

csr.h

C

```
#define CSR_PCM_STREAM_CFG_KEY_MCLK 0x101
```

Description

This is macro CSR_PCM_STREAM_CFG_KEY_MCLK.

CSR_PCM_STREAM_CFG_KEY_SHORT_SYNC Macro

File

csr.h

C

```
#define CSR_PCM_STREAM_CFG_KEY_SHORT_SYNC 0x105
```

Description

This is macro CSR_PCM_STREAM_CFG_KEY_SHORT_SYNC.

CSR_PCM_STREAM_CFG_KEY_SIGN_EXTEND Macro

File

csr.h

C

```
#define CSR_PCM_STREAM_CFG_KEY_SIGN_EXTEND 0x107
```

Description

This is macro CSR_PCM_STREAM_CFG_KEY_SIGN_EXTEND.

CSR_PCM_STREAM_CFG_KEY_SLOT_COUNT Macro

File

csr.h

C

```
#define CSR_PCM_STREAM_CFG_KEY_SLOT_COUNT 0x103
```

Description

This is macro CSR_PCM_STREAM_CFG_KEY_SLOT_COUNT.

CSR_PCM_STREAM_CFG_KEY_SYNC_RATE Macro

File

csr.h

C

```
#define CSR_PCM_STREAM_CFG_KEY_SYNC_RATE 0x100
```

Description

This is macro CSR_PCM_STREAM_CFG_KEY_SYNC_RATE.

CSR_PCM_STREAM_CFG_KEY_TX_TRISTATE Macro

File

csr.h

C

```
#define CSR_PCM_STREAM_CFG_KEY_TX_TRISTATE 0x109
```

Description

This is macro CSR_PCM_STREAM_CFG_KEY_TX_TRISTATE.

CSR_VARID_CAPABILITY_DOWNLOAD_INDICATION Macro

File

csr.h

C

```
#define CSR_VARID_CAPABILITY_DOWNLOAD_INDICATION 0x101b
```

Description

This is macro CSR_VARID_CAPABILITY_DOWNLOAD_INDICATION.

CSR_VARID_CREATE_OPERATOR_C Macro

File

csr.h

C

```
#define CSR_VARID_CREATE_OPERATOR_C 0x5075
```

Description

This is macro CSR_VARID_CREATE_OPERATOR_C.

CSR_VARID_CREATE_OPERATOR_P Macro

File

csr.h

C

```
#define CSR_VARID_CREATE_OPERATOR_P 0x5076
```

Description

This is macro CSR_VARID_CREATE_OPERATOR_P.

CSR_VARID_DESTROY_OPERATORS Macro

File

csr.h

C

```
#define CSR_VARID_DESTROY_OPERATORS 0x5073
```

Description

This is macro CSR_VARID_DESTROY_OPERATORS.

CSR_VARID_DSPMANAGER_CONFIG_REQUEST Macro

File

csr.h

C

```
#define CSR_VARID_DSPMANAGER_CONFIG_REQUEST 0x101c
```

Description

This is macro CSR_VARID_DSPMANAGER_CONFIG_REQUEST.

CSR_VARID_MESSAGE_FROM_OPERATOR Macro

File

csr.h

C

```
#define CSR_VARID_MESSAGE_FROM_OPERATOR 0x101a
```

Description

Select Variable definitions

CSR_VARID_START_OPERATORS Macro

File

csr.h

C

```
#define CSR_VARID_START_OPERATORS 0x5070
```

Description

This is macro CSR_VARID_START_OPERATORS.

CSR_VARID_STOP_OPERATORS Macro

File

csr.h

C

```
#define CSR_VARID_STOP_OPERATORS 0x5071
```

Description

This is macro CSR_VARID_STOP_OPERATORS.

CSR_VARID_STREAM_DRAINED_NOTIFICATION Macro

File

csr.h

C

```
#define CSR_VARID_STREAM_DRAINED_NOTIFICATION 0x101d
```

Description

This is macro CSR_VARID_STREAM_DRAINED_NOTIFICATION.

CSR_VARID_STREAM_TRANSFORM_DISCONNECT Macro

File

csr.h

C

```
#define CSR_VARID_STREAM_TRANSFORM_DISCONNECT 0x486d
```

Description

This is macro CSR_VARID_STREAM_TRANSFORM_DISCONNECT.

PSKEY_BLE_DEFAULT_TX_POWER Macro

File

csr.h

C

```
#define PSKEY_BLE_DEFAULT_TX_POWER 0x22c8
```

Description

This is macro PSKEY_BLE_DEFAULT_TX_POWER.

PSKEY_DEEP_SLEEP_EXTERNAL_CLOCK_SOURCE_PIO Macro

File

csr.h

C

```
#define PSKEY_DEEP_SLEEP_EXTERNAL_CLOCK_SOURCE_PIO 0x2579
```

Description

This is macro PSKEY_DEEP_SLEEP_EXTERNAL_CLOCK_SOURCE_PIO.

PSKEY_H_HC_FC_MAX_SCO_PKT_LEN Macro

File

csr.h

C

```
#define PSKEY_H_HC_FC_MAX_SCO_PKT_LEN 0x0012
```

Description

This is macro PSKEY_H_HC_FC_MAX_SCO_PKT_LEN.

PSKEY_H_HC_FC_MAX_SCO_PKTS Macro

File

csr.h

C

```
#define PSKEY_H_HC_FC_MAX_SCO_PKTS 0x0014
```

Description

This is macro PSKEY_H_HC_FC_MAX_SCO_PKTS.

PSKEY_PCM_CLOCK_RATE Macro

File

csr.h

C

```
#define PSKEY_PCM_CLOCK_RATE 0x23bc
```

Description

This is macro PSKEY_PCM_CLOCK_RATE.

PSKEY_PCM_CONFIG32 Macro

File

csr.h

C

```
#define PSKEY_PCM_CONFIG32 0x01b3
```

Description

This is macro PSKEY_PCM_CONFIG32.

PSKEY_PCM_FORMAT Macro

File

csr.h

C

```
#define PSKEY_PCM_FORMAT 0x01b6
```

Description

This is macro PSKEY_PCM_FORMAT.

PSKEY_PCM_SYNC_RATE Macro

File

csr.h

C

```
#define PSKEY_PCM_SYNC_RATE 0x23c3
```

Description

This is macro PSKEY_PCM_SYNC_RATE.

PSKEY_PCM_USE_LOW_JITTER_MODE Macro

File

csr.h

C

```
#define PSKEY_PCM_USE_LOW_JITTER_MODE 0x23c9
```

Description

This is macro PSKEY_PCM_USE_LOW_JITTER_MODE.

PSKEY_UART_CONFIG_H4DS Macro

File

csr.h

C

```
#define PSKEY_UART_CONFIG_H4DS 0x01cb
```

Description

This is macro PSKEY_UART_CONFIG_H4DS.

BT_LOG_LEVEL_A2DP_PAKCET Macro

File

bt_log.h

C

```
#define BT_LOG_LEVEL_A2DP_PAKCET 10
```

Description

This is macro BT_LOG_LEVEL_A2DP_PAKCET.

DCRH Macro

File

patch.h

C

```
#define DCRH(block_count, val_len, seq) \
    0x01,         /* HCI command packet indicator.*/ \
    0x00, 0xfc,   /* HCI opcode { OGF=0x3f OCF=0x00 } */ \
    /*     OGF=0x3f implies a manufacturer-sepcific command. */ \
    /* Parameter length. */ \
    1 + (5 + 1 + val_len) * 2, \
    0xc3,         /* CSR payload descriptor: */ \
    /*    bit 7: Last fragment = 1 */ \
    /*    bit 6: First fragment = 1 */ \
    /*    bits 5-0 Channel ID = 2 (BCCMD) */ \
    W(0x0001),    /* GETRESP message */ \
    W(5+1+val_len), /* length*/ \
    W(seq),       /* sequence number */ \
    W(0x101c),    /* Variable ID of DSP manager config request */ \
    W(0x0000),    /* Status (must be 0 in SETREQ) */ \
    \
    W(block_count)     /* Number of configuration blocks */
```

Description

Number of configuration blocks

btx_csr_bccmd_callback_fp Type

File

csr.h

C

```
typedef void (* btx_csr_bccmd_callback_fp)(btx_csr_bccmd_header_t* message, void* cb_param);
```

Description

This is type btx_csr_bccmd_callback_fp.

btx_csr_bccmd_listener_t Type

File

csr.h

C

```
typedef struct _btx_csr_bccmd_listener_t btx_csr_bccmd_listener_t;
```

Description

This is type btx_csr_bccmd_listener_t.

btx_csr_create_operator_c_t Structure

File

csr.h

C

```
typedef struct _btx_csr_create_operator_c_s {
  btx_csr_bccmd_header_t message;
  bt_uint operator_id;
  bt_uint item_count;
  bt_uint skip_count;
  bt_uint skip_flag;
} btx_csr_create_operator_c_t;
```

Description

This is type btx_csr_create_operator_c_t.

btx_csr_exec_hq_script_buffer_t Type

File

csr.h

C

```
typedef struct _btx_csr_exec_hq_script_buffer_t btx_csr_exec_hq_script_buffer_t;
```

Description

This is type btx_csr_exec_hq_script_buffer_t.

btx_csr_exec_hq_script_callback_fp Type

File

csr.h

C

```
typedef void (* btx_csr_exec_hq_script_callback_fp)(bt_bool success, btx_csr_exec_hq_script_buffer_t*
buffer);
```

Description

This is type btx_csr_exec_hq_script_callback_fp.

btx_csr_get_script_fp Type

File

csr.h

C

```
typedef const btx_csr_script_t* (* btx_csr_get_script_fp)(void);
```

Description

This is type btx_csr_get_script_fp.

btx_csr_strm_connect_t Structure

File

csr.h

C

```
typedef struct _btx_csr_strm_connect_s {
  btx_csr_bccmd_header_t message;
  bt_uint transform_id;
} btx_csr_strm_connect_t;
```

Description

This is type btx_csr_strm_connect_t.

btx_ti_codec_config_t Structure

File

ti.h

C

```
typedef struct _btx_ti_codec_config_s {
  bt_uint clock_rate;
  bt_byte clock_direction;
  bt_ulong frame_sync_freq;
  bt_uint frame_sync_duty_cycle;
  bt_byte frame_sync_edge;
  bt_byte frame_sync_polarity;
  bt_byte frame_sync_multiplier;
  bt_uint ch1_data_out_size;
  bt_uint ch1_data_out_offset;
  bt_byte ch1_data_out_edge;
  bt_uint ch1_data_in_size;
  bt_uint ch1_data_in_offset;
  bt_byte ch1_data_in_edge;
  bt_uint ch2_data_out_size;
  bt_uint ch2_data_out_offset;
  bt_byte ch2_data_out_edge;
  bt_uint ch2_data_in_size;
  bt_uint ch2_data_in_offset;
  bt_byte ch2_data_in_edge;
} btx_ti_codec_config_t;
```

Description

This is type btx_ti_codec_config_t.

_btx_csr_bccmd_listener_t Structure

File

csr.h

C

```
struct _btx_csr_bccmd_listener_t {
  btx_csr_bccmd_listener_t* next_listener;
  btx_csr_bccmd_callback_fp callback;
  void* callback_param;
};
```

Description

This is record _btx_csr_bccmd_listener_t.

_btx_csr_exec_hq_script_buffer_t Structure

File

csr.h

C

```
struct _btx_csr_exec_hq_script_buffer_t {
  const btx_csr_script_t* script;
  bt_int current_packet;
  bt_byte packet[CSR_MAX_HQ_PACKET_LEN];
  bt_bool success;
  btx_csr_exec_hq_script_callback_fp callback;
  void* callback_param;
};
```

Description

This is record _btx_csr_exec_hq_script_buffer_t.

PSKEY_LC_DEFAULT_TX_POWER Macro

File

csr.h

C

```
#define PSKEY_LC_DEFAULT_TX_POWER 0x0021
```

Description

This is macro PSKEY_LC_DEFAULT_TX_POWER.

PSKEY_LC_MAX_TX_POWER Macro

File

csr.h

C

```
#define PSKEY_LC_MAX_TX_POWER 0x0017
```

Description

This is macro PSKEY_LC_MAX_TX_POWER.

PSKEY_LC_MAX_TX_POWER_NO_RSSI Macro

File

csr.h

C

```
#define PSKEY_LC_MAX_TX_POWER_NO_RSSI 0x002d
```

Description

This is macro PSKEY_LC_MAX_TX_POWER_NO_RSSI.

Utility Functions

bt_alloc_buffer Function

File

buffer.h

C

```
void* bt_alloc_buffer(bt_buffer_mgr_p pmgr);
```

Description

This is function bt_alloc_buffer.

bt_at_parse_fragment Function

File

at_parser.h

C

```
void bt_at_parse_fragment(bt_at_parser_t* parser, bt_byte* data, bt_uint len);
```

Description

This is function bt_at_parse_fragment.

bt_at_parser_reset Function

File

at_parser.h

C

```
void bt_at_parser_reset(bt_at_parser_t* parser);
```

Description

This is function bt_at_parser_reset.

bt_free_buffer Function

File

buffer.h

C

```
void bt_free_buffer(bt_buffer_mgr_p pmgr, void* data);
```

Description

This is function bt_free_buffer.

bt_get_buffer Function

File

buffer.h

C

```
void* bt_get_buffer(bt_buffer_mgr_p pmgr, bt_int index);
```

Description

This is function bt_get_buffer.

bt_get_buffer_header Function

File

buffer.h

C

```
bt_buffer_header_t* bt_get_buffer_header(bt_buffer_mgr_p pmgr, bt_int index);
```

Description

This is function bt_get_buffer_header.

bt_get_buffer_index Function

File

buffer.h

C

```
bt_int bt_get_buffer_index(bt_buffer_mgr_p pmgr, void* data);
```

Description

This is function bt_get_buffer_index.

bt_init_buffer_mgr Function

File

buffer.h

C

```
bt_bool bt_init_buffer_mgr(bt_buffer_mgr_p pmgr, bt_int num_buffers, bt_int buffer_size,
bt_buffer_header_t* buffer_headers, void* buffer_data);
```

Description

This is function bt_init_buffer_mgr.

bt_q_add Function

File

queue.h

C

```
void bt_q_add(bt_queue_element_t** phead, void* element);
```

Description

This is function bt_q_add.

bt_q_get Function

File

queue.h

C

```
void* bt_q_get(bt_queue_element_t* head, bt_int index);
```

Description

This is function bt_q_get.

bt_q_get_head Function

File

queue.h

C

```
void* bt_q_get_head(bt_queue_element_t** phead, bt_bool remove);
```

Description

This is function bt_q_get_head.

bt_q_get_length Function

File

queue.h

C

```
bt_int bt_q_get_length(bt_queue_element_t* phead);
```

Description

This is function bt_q_get_length.

bt_q_get_next Function

File

queue.h

C

```
void* bt_q_get_next(void* element);
```

Description

This is function bt_q_get_next.

bt_q_push Function

File

queue.h

C

```
void bt_q_push(bt_queue_element_t** phead, void* element);
```

Description

This is function bt_q_push.

bt_q_remove Function

File

queue.h

C

```
void* bt_q_remove(bt_queue_element_t** phead, void* pdata);
```

Description

This is function bt_q_remove.

bt_q_remove_by_idx Function

File

queue.h

C

```
void* bt_q_remove_by_idx(bt_queue_element_t* head, bt_int index);
```

Description

This is function bt_q_remove_by_idx.

bt_vcard_parse_fragment Function

File

vcard_parser.h

C

```
void bt_vcard_parse_fragment(bt_vcard_parser_t* parser, bt_byte* data, bt_uint len);
```

Description

This is function bt_vcard_parse_fragment.

bt_vcard_parser_reset Function

File

vcard_parser.h

C

```
void bt_vcard_parser_reset(bt_vcard_parser_t* parser);
```

Description

This is function bt_vcard_parser_reset.

bt_xml_parse_fragment Function

File

xml_parser.h

C

```
void bt_xml_parse_fragment(bt_xml_parser_t* parser, bt_byte_cp data, bt_uint len);
```

Description

This is function bt_xml_parse_fragment.

bt_xml_parser_reset Function

File

xml_parser.h

C

```
void bt_xml_parser_reset(bt_xml_parser_t* parser);
```

Description

This is function bt_xml_parser_reset.

_compact_buffer Function

File

bufferutils.h

C

```
void _compact_buffer(bt_byte_p pdst, bt_uint offset, bt_byte_cp psrc, bt_uint len);
```

Description

This is function _compact_buffer.

_expand_buffer Function

File

bufferutils.h

C

```
void _expand_buffer(bt_byte_p pdst, bt_byte_cp psrc, bt_uint len, bt_uint offset);
```

Description

This is function _expand_buffer.

_read_bdaddr Function

File

bufferutils.h

C

```
void _read_bdaddr(bt_bdaddr_p pbdaddr, bt_byte_cp buffer);
```

Description

This is function _read_bdaddr.

_readb Function

File

bufferutils.h

C

```
bt_bool _readb(bt_byte_p pvalue, bt_byte_cp pb, bt_int buf_len, bt_int_p offset);
```

Description

This is function _readb.

_readi Function

File

bufferutils.h

C

```
bt_bool _readi(bt_int_p pvalue, bt_byte_cp pb, bt_int buf_len, bt_int_p offset);
```

Description

This is function _readi.

_readin Function

File

bufferutils.h

C

```
bt_bool _readin(bt_int_p pvalue, bt_byte_cp pb, bt_int buf_len, bt_int_p offset);
```

Description

This is function _readin.

_readl Function

File

bufferutils.h

C

```
bt_bool _readl(bt_long_p pvalue, bt_byte_cp pb, bt_int buf_len, bt_int_p offset);
```

Description

This is function _readl.

_readln Function

File

bufferutils.h

C

```
bt_bool _readln(bt_long_p pvalue, bt_byte_cp pb, bt_int buf_len, bt_int_p offset);
```

Description

This is function _readln.

_readui Function

File

bufferutils.h

C

```
bt_bool _readui(bt_uint_ptr pvalue, bt_byte_cp pb, bt_int buf_len, bt_int_p offset);
```

Description

This is function _readui.

_readuin Function

File

bufferutils.h

C

```
bt_bool _readuin(bt_uint_ptr pvalue, bt_byte_cp pb, bt_int buf_len, bt_int_p offset);
```

Description

This is function _readuin.

_readul Function

File

bufferutils.h

C

```
bt_bool _readul(bt_ulong_ptr pvalue, bt_byte_cp pb, bt_int buf_len, bt_int_p offset);
```

Description

This is function _readul.

_readuln Function

File

bufferutils.h

C

```
bt_bool _readuln(bt_ulong_ptr pvalue, bt_byte_cp pb, bt_int buf_len, bt_int_p offset);
```

Description

This is function _readuln.

_str2ulong_dec Function

File

bufferutils.h

C

```
bt_ulong _str2ulong_dec(const char* s);
```

Description

This is function _str2ulong_dec.

_to_lower_case Function

File

bufferutils.h

C

```
void _to_lower_case(bt_byte* str, bt_uint len);
```

Description

This is function _to_lower_case.

_ulong2str Function

File

bufferutils.h

C

```
const char* _ulong2str(bt_ulong data);
```

Description

This is function _ulong2str.

_ulong2str_dec Function

File

bufferutils.h

C

```
const char* _ulong2str_dec(bt_ulong data);
```

Description

This is function _ulong2str_dec.

_write_bdaddr Function

File

bufferutils.h

C

```
void _write_bdaddr(bt_bdaddr_cp pbdaddr, bt_byte_p buffer);
```

Description

This is function _write_bdaddr.

_writeb Function

File

bufferutils.h

C

```
bt_bool _writeb(bt_byte value, bt_byte_p pb, bt_int buf_len, bt_int_p offset);
```

Description

This is function _writeb.

_writei Function

File

bufferutils.h

C

```
bt_bool _writei(bt_int value, bt_byte_p pb, bt_int buf_len, bt_int_p offset);
```

Description

This is function _writei.

_writein Function

File

bufferutils.h

C

```
bt_bool _writein(bt_int value, bt_byte_p pb, bt_int buf_len, bt_int_p offset);
```

Description

This is function _writein.

_writel Function

File

bufferutils.h

C

```
bt_bool _writel(bt_long value, bt_byte_p pb, bt_int buf_len, bt_int_p offset);
```

Description

This is function _writel.

_writeln Function

File

bufferutils.h

C

```
bt_bool _writeln(bt_long value, bt_byte_p pb, bt_int buf_len, bt_int_p offset);
```

Description

This is function _writeln.

_writes Function

File

bufferutils.h

C

```
bt_bool _writes(const char* pstr, bt_byte_p pb, bt_int buf_len, bt_int_p offset);
```

Description

This is function _writes.

_writesx Function

File

bufferutils.h

C

```
bt_bool _writesx(const char* pstr, bt_int str_len, bt_byte_p pb, bt_int buf_len, bt_int_p offset);
```

Description

This is function _writesx.

_zero_memory Function

File

bufferutils.h

C

```
void _zero_memory(void* ptr, size_t len);
```

Description

This is function _zero_memory.

_bt_memcpy Function

File

bufferutils.h

C

```
void _bt_memcpy(bt_byte_p pdst, bt_uint dst_offset, bt_byte_cp psrc, bt_uint len, bt_uint offset);
```

Description

This is function _bt_memcpy.

_is_empty_str Function

File

bufferutils.h

C

```
bt_bool _is_empty_str(const char* s);
```

Description

This is function _is_empty_str.

_str2ulong Function

File

bufferutils.h

C

```
bt_ulong _str2ulong(const char* s);
```

Description

This is function _str2ulong.

Utility Data Types and Constants

AT_EVT_CMD_CODE Macro

File

at_parser.h

C

```
#define AT_EVT_CMD_CODE 3
```

Description

This is macro AT_EVT_CMD_CODE.

AT_EVT_CMD_COMPLETED Macro

File

at_parser.h

C

```
#define AT_EVT_CMD_COMPLETED 6
```

Description

This is macro AT_EVT_CMD_COMPLETED.

AT_EVT_CMD_PARAM Macro

File

at_parser.h

C

```
#define AT_EVT_CMD_PARAM 5
```

Description

This is macro AT_EVT_CMD_PARAM.

AT_EVT_CMD_READ_CODE Macro

File

at_parser.h

C

```
#define AT_EVT_CMD_READ_CODE 4
```

Description

This is macro AT_EVT_CMD_READ_CODE.

AT_EVT_ERROR Macro

File

at_parser.h

C

```
#define AT_EVT_ERROR 2
```

Description

This is macro AT_EVT_ERROR.

AT_EVT_OK Macro

File

at_parser.h

C

```
#define AT_EVT_OK 1
```

Description

This is macro AT_EVT_OK.

AT_EVT_RING Macro

File

at_parser.h

C

```
#define AT_EVT_RING 0
```

Description

This is macro AT_EVT_RING.

ATCMD_BUFFER_LEN Macro

File

at_parser.h

C

```
#define ATCMD_BUFFER_LEN 20
```

Description

This is macro ATCMD_BUFFER_LEN.

bt_max Macro

File

bufferutils.h

C

```
#define bt_max(a,b) (((a) > (b)) ? (a) : (b))
```

Description

This is macro bt_max.

bt_min Macro

File

bufferutils.h

C

```
#define bt_min(a,b) (((a) < (b)) ? (a) : (b))
```

Description

This is macro bt_min.

BUFFER_HDR_LEN Macro

File

buffer.h

C

```
#define BUFFER_HDR_LEN offsetof(bt_buffer, data)
```

Description

This is macro BUFFER_HDR_LEN.

BUFFER_STATE_FREE Macro

File

buffer.h

C

```
#define BUFFER_STATE_FREE 0
```

Description

for offsetof

BUFFER_STATE_USED Macro

File

buffer.h

C

```
#define BUFFER_STATE_USED 1
```

Description

This is macro BUFFER_STATE_USED.

max Macro

File

bufferutils.h

C

```
#define max(a,b) (using_max_is_dangerous = 0, use_bt_max_instead)
```

Description

This is macro max.

min Macro

File

bufferutils.h

C

```
#define min(a,b) (using_min_is_dangerous = 0, use_bt_min_instead)
```

Description

This is macro min.

VCARD_CALL_TYPE_DIALED Macro

File

vcard_parser.h

C

```
#define VCARD_CALL_TYPE_DIALED 0
```

Description

This is macro VCARD_CALL_TYPE_DIALED.

VCARD_CALL_TYPE_MISSED Macro

File

vcard_parser.h

C

```
#define VCARD_CALL_TYPE_MISSED 2
```

Description

This is macro VCARD_CALL_TYPE_MISSED.

VCARD_CALL_TYPE_RECEIVED Macro

File

vcard_parser.h

C

```
#define VCARD_CALL_TYPE_RECEIVED 1
```

Description

This is macro VCARD_CALL_TYPE_RECEIVED.

VCARD_EVT_PROPERTY_PARAM Macro

File

vcard_parser.h

C

```
#define VCARD_EVT_PROPERTY_PARAM 3
```

Description

This is macro VCARD_EVT_PROPERTY_PARAM.

VCARD_EVT_PROPERTY_STARTED Macro

File

vcard_parser.h

C

```
#define VCARD_EVT_PROPERTY_STARTED 2
```

Description

This is macro VCARD_EVT_PROPERTY_STARTED.

VCARD_EVT_PROPERTY_VALUE Macro

File

vcard_parser.h

C

```
#define VCARD_EVT_PROPERTY_VALUE 4
```

Description

This is macro VCARD_EVT_PROPERTY_VALUE.

VCARD_EVT_VCARD_ENDED Macro

File

vcard_parser.h

C

```
#define VCARD_EVT_VCARD_ENDED 1
```

Description

This is macro VCARD_EVT_VCARD_ENDED.

VCARD_EVT_VCARD_STARTED Macro

File

vcard_parser.h

C

```
#define VCARD_EVT_VCARD_STARTED 0
```

Description

This is macro VCARD_EVT_VCARD_STARTED.

VCARD_PARAM_CALL_TYPE Macro

File

vcard_parser.h

C

```
#define VCARD_PARAM_CALL_TYPE 0x80
```

Description

This is macro VCARD_PARAM_CALL_TYPE.

VCARD_PARAM_CELL Macro

File

vcard_parser.h

C

```
#define VCARD_PARAM_CELL 7
```

Description

This is macro VCARD_PARAM_CELL.

VCARD_PARAM_DIALED Macro

File

vcard_parser.h

C

```
#define VCARD_PARAM_DIALED 4
```

Description

This is macro VCARD_PARAM_DIALED.

VCARD_PARAM_ENCODING Macro

File

vcard_parser.h

C

```
#define VCARD_PARAM_ENCODING 1
```

Description

This is macro VCARD_PARAM_ENCODING.

VCARD_PARAM_HOME Macro

File

vcard_parser.h

C

```
#define VCARD_PARAM_HOME 8
```

Description

This is macro VCARD_PARAM_HOME.

VCARD_PARAM_LANGUAGE Macro

File

vcard_parser.h

C

```
#define VCARD_PARAM_LANGUAGE 3
```

Description

This is macro VCARD_PARAM_LANGUAGE.

VCARD_PARAM_MISSED Macro

File

vcard_parser.h

C

```
#define VCARD_PARAM_MISSED 6
```

Description

This is macro VCARD_PARAM_MISSED.

VCARD_PARAM_RECEIVED Macro

File

vcard_parser.h

C

```
#define VCARD_PARAM_RECEIVED 5
```

Description

This is macro VCARD_PARAM_RECEIVED.

VCARD_PARAM_TYPE Macro

File

vcard_parser.h

C

```
#define VCARD_PARAM_TYPE 0
```

Description

This is macro VCARD_PARAM_TYPE.

VCARD_PARAM_VALUE Macro

File

vcard_parser.h

C

```
#define VCARD_PARAM_VALUE 2
```

Description

This is macro VCARD_PARAM_VALUE.

VCARD_PARAM_WORK Macro

File

vcard_parser.h

C

```
#define VCARD_PARAM_WORK 9
```

Description

This is macro VCARD_PARAM_WORK.

VCARD_PARSER_STATE_READ_PARAM_NAME Macro

File

vcard_parser.h

C

```
#define VCARD_PARSER_STATE_READ_PARAM_NAME 1
```

Description

This is macro VCARD_PARSER_STATE_READ_PARAM_NAME.

VCARD_PARSER_STATE_READ_PARAM_VALUE Macro

File

vcard_parser.h

C

```
#define VCARD_PARSER_STATE_READ_PARAM_VALUE 3
```

Description

This is macro VCARD_PARSER_STATE_READ_PARAM_VALUE.

VCARD_PARSER_STATE_READ_TYPE_NAME Macro

File

vcard_parser.h

C

```
#define VCARD_PARSER_STATE_READ_TYPE_NAME 0
```

Description

This is macro VCARD_PARSER_STATE_READ_TYPE_NAME.

VCARD_PARSER_STATE_READ_TYPE_VALUE Macro

File

vcard_parser.h

C

```
#define VCARD_PARSER_STATE_READ_TYPE_VALUE 2
```

Description

This is macro VCARD_PARSER_STATE_READ_TYPE_VALUE.

VCARD_PARSER_STATE_SKIP_PARAM Macro

File

vcard_parser.h

C

```
#define VCARD_PARSER_STATE_SKIP_PARAM 4
```

Description

This is macro VCARD_PARSER_STATE_SKIP_PARAM.

VCARD_PARSER_STATE_SKIP_TYPE Macro

File

vcard_parser.h

C

```
#define VCARD_PARSER_STATE_SKIP_TYPE 5
```

Description

This is macro VCARD_PARSER_STATE_SKIP_TYPE.

VCARD_PROPERTY_ADR Macro

File

vcard_parser.h

C

```
#define VCARD_PROPERTY_ADR 7 // Delivery Address
```

Description

Delivery Address

VCARD_PROPERTY_AGENT Macro

File

vcard_parser.h

C

```
#define VCARD_PROPERTY_AGENT 17 // vCard of Person Representing
```

Description

vCard of Person Representing

VCARD_PROPERTY_BDAY Macro

File

vcard_parser.h

C

```
#define VCARD_PROPERTY_BDAY 6 // Birthday
```

Description

Birthday

VCARD_PROPERTY_BEGIN Macro

File

vcard_parser.h

C

```
#define VCARD_PROPERTY_BEGIN 0
```

Description

This is macro VCARD_PROPERTY_BEGIN.

VCARD_PROPERTY_CATEGORIES Macro

File

vcard_parser.h

C

```
#define VCARD_PROPERTY_CATEGORIES 26 // Categories
```

Description

Categories

VCARD_PROPERTY_CLASS Macro

File

vcard_parser.h

C

```
#define VCARD_PROPERTY_CLASS 28 // Class information
```

Description

Class information

VCARD_PROPERTY_EMAIL Macro

File

vcard_parser.h

C

```
#define VCARD_PROPERTY_EMAIL 10 // Electronic Mail Address
```

Description

Electronic Mail Address

VCARD_PROPERTY_END Macro

File

vcard_parser.h

C

```
#define VCARD_PROPERTY_END 1
```

Description

This is macro VCARD_PROPERTY_END.

VCARD_PROPERTY_FN Macro

File

vcard_parser.h

C

```
#define VCARD_PROPERTY_FN 3 // Formatted Name
```

Description

Formatted Name

VCARD_PROPERTY_GEO Macro

File

vcard_parser.h

C

```
#define VCARD_PROPERTY_GEO 13 // Geographic Position
```

Description

Geographic Position

VCARD_PROPERTY_KEY Macro

File

vcard_parser.h

C

```
#define VCARD_PROPERTY_KEY 24 // Public Encryption Key
```

Description

Public Encryption Key

VCARD_PROPERTY_LABEL Macro

File

vcard_parser.h

C

```
#define VCARD_PROPERTY_LABEL 8 // Delivery
```

Description

Delivery

VCARD_PROPERTY_LOGO Macro

File

vcard_parser.h

C

```
#define VCARD_PROPERTY_LOGO 16 // Organization Logo
```

Description

Organization Logo

VCARD_PROPERTY_MAILER Macro

File

vcard_parser.h

C

```
#define VCARD_PROPERTY_MAILER 11 // Electronic Mail
```

Description

Electronic Mail

VCARD_PROPERTY_N Macro

File

vcard_parser.h

C

```
#define VCARD_PROPERTY_N 4 // Structured Presentation of Name
```

Description

Structured Presentation of Name

VCARD_PROPERTY_NICKNAME Macro

File

vcard_parser.h

C

```
#define VCARD_PROPERTY_NICKNAME 25 // Nickname
```

Description

Nickname

VCARD_PROPERTY_NOTE Macro

File

vcard_parser.h

C

```
#define VCARD_PROPERTY_NOTE 19 // Comments
```

Description

Comments

VCARD_PROPERTY_ORG Macro

File

vcard_parser.h

C

```
#define VCARD_PROPERTY_ORG 18 // Name of Organization
```

Description

Name of Organization

VCARD_PROPERTY_PHOTO Macro

File

vcard_parser.h

C

```
#define VCARD_PROPERTY_PHOTO 5 // Associated Image or Photo
```

Description

Associated Image or Photo

VCARD_PROPERTY_PROID Macro

File

vcard_parser.h

C

```
#define VCARD_PROPERTY_PROID 27 // Product ID
```

Description

Product ID

VCARD_PROPERTY_REV Macro

File

vcard_parser.h

C

```
#define VCARD_PROPERTY_REV 20 // Revision
```

Description

Revision

VCARD_PROPERTY_ROLE Macro

File

vcard_parser.h

C

```
#define VCARD_PROPERTY_ROLE 15 // Role within the Organization
```

Description

Role within the Organization

VCARD_PROPERTY_SORT_STRING Macro

File

vcard_parser.h

C

```
#define VCARD_PROPERTY_SORT_STRING 29 // String used for sorting operations
```

Description

String used for sorting operations

VCARD_PROPERTY_SOUND Macro

File

vcard_parser.h

C

```
#define VCARD_PROPERTY_SOUND 21 // Pronunciation of Name
```

Description

Pronunciation of Name

VCARD_PROPERTY_TEL Macro

File

vcard_parser.h

C

```
#define VCARD_PROPERTY_TEL 9 // Telephone Number
```

Description

Telephone Number

VCARD_PROPERTY_TITLE Macro

File

vcard_parser.h

C

```
#define VCARD_PROPERTY_TITLE 14 // Job
```

Description

Job

VCARD_PROPERTY_TZ Macro

File

vcard_parser.h

C

```
#define VCARD_PROPERTY_TZ 12 // Time Zone
```

Description

Time Zone

VCARD_PROPERTY_UID Macro

File

vcard_parser.h

C

```
#define VCARD_PROPERTY_UID 23 // Unique ID
```

Description

Unique ID

VCARD_PROPERTY_URL Macro

File

vcard_parser.h

C

```
#define VCARD_PROPERTY_URL 22 // Uniform Resource Locator
```

Description

Uniform Resource Locator

VCARD_PROPERTY_VERSION Macro

File

vcard_parser.h

C

```
#define VCARD_PROPERTY_VERSION 2 // vCard Version
```

Description

vCard Version

VCARD_PROPERTY_X_IRMC_CALL_DATETIME Macro

File

vcard_parser.h

C

```
#define VCARD_PROPERTY_X_IRMC_CALL_DATETIME 30 // Time stamp
```

Description

Time stamp

XML_EVT_ATTR_NAME Macro

File

xml_parser.h

C

```
#define XML_EVT_ATTR_NAME 2
```

Description

This is macro XML_EVT_ATTR_NAME.

XML_EVT_ATTR_VALUE Macro

File

xml_parser.h

C

```
#define XML_EVT_ATTR_VALUE 3
```

Description

This is macro XML_EVT_ATTR_VALUE.

XML_EVT_TAG_ENDED Macro

File

xml_parser.h

C

```
#define XML_EVT_TAG_ENDED 1
```

Description

This is macro XML_EVT_TAG_ENDED.

XML_EVT_TAG_STARTED Macro

File

xml_parser.h

C

```
#define XML_EVT_TAG_STARTED 0
```

Description

This is macro XML_EVT_TAG_STARTED.

XML_PARSER_STATE_READ_ATTR_NAME Macro

File

xml_parser.h

C

```
#define XML_PARSER_STATE_READ_ATTR_NAME 5
```

Description

This is macro XML_PARSER_STATE_READ_ATTR_NAME.

XML_PARSER_STATE_READ_ATTR_VALUE Macro

File

xml_parser.h

C

```
#define XML_PARSER_STATE_READ_ATTR_VALUE 8
```

Description

This is macro XML_PARSER_STATE_READ_ATTR_VALUE.

XML_PARSER_STATE_READ_TAG_NAME Macro

File

xml_parser.h

C

```
#define XML_PARSER_STATE_READ_TAG_NAME 1
```

Description

This is macro XML_PARSER_STATE_READ_TAG_NAME.

XML_PARSER_STATE_READ_TAG_VALUE Macro

File

xml_parser.h

C

```
#define XML_PARSER_STATE_READ_TAG_VALUE 7
```

Description

This is macro XML_PARSER_STATE_READ_TAG_VALUE.

XML_PARSER_STATE_SKIP_ATTR Macro

File

xml_parser.h

C

```
#define XML_PARSER_STATE_SKIP_ATTR 9
```

Description

This is macro XML_PARSER_STATE_SKIP_ATTR.

XML_PARSER_STATE_SKIP_COMMENTS Macro

File

xml_parser.h

C

```
#define XML_PARSER_STATE_SKIP_COMMENTS 11
```

Description

This is macro XML_PARSER_STATE_SKIP_COMMENTS.

XML_PARSER_STATE_SKIP_PROC_INST Macro

File

xml_parser.h

C

```
#define XML_PARSER_STATE_SKIP_PROC_INST 10
```

Description

This is macro XML_PARSER_STATE_SKIP_PROC_INST.

XML_PARSER_STATE_SKIP_TAG Macro

File

xml_parser.h

C

```
#define XML_PARSER_STATE_SKIP_TAG 3
```

Description

This is macro XML_PARSER_STATE_SKIP_TAG.

XML_PARSER_STATE_WAIT_ATTR_NAME Macro

File

xml_parser.h

C

```
#define XML_PARSER_STATE_WAIT_ATTR_NAME 4
```

Description

This is macro XML_PARSER_STATE_WAIT_ATTR_NAME.

XML_PARSER_STATE_WAIT_ATTR_VALUE Macro

File

xml_parser.h

C

```
#define XML_PARSER_STATE_WAIT_ATTR_VALUE 6
```

Description

This is macro XML_PARSER_STATE_WAIT_ATTR_VALUE.

XML_PARSER_STATE_WAIT_TAG_CLOSE Macro

File

xml_parser.h

C

```
#define XML_PARSER_STATE_WAIT_TAG_CLOSE 2
```

Description

This is macro XML_PARSER_STATE_WAIT_TAG_CLOSE.

XML_PARSER_STATE_WAIT_TAG_NAME Macro

File

xml_parser.h

C

```
#define XML_PARSER_STATE_WAIT_TAG_NAME 0
```

Description

This is macro XML_PARSER_STATE_WAIT_TAG_NAME.

_readbn Macro

File

bufferutils.h

C

```
#define _readbn _readb
```

Description

This is macro _readbn.

_readui Macro

File

bufferutils.h

C

```
#define _readui(pvalue, pb, buf_len, offset) _readi((bt_int_p)pvalue, pb, buf_len, offset)
```

Description

This is macro _readui.

_readuin Macro

File

bufferutils.h

C

```
#define _readuin(pvalue, pb, buf_len, offset) _readin((bt_int_p)pvalue, pb, buf_len, offset)
```

Description

This is macro _readuin.

_readul Macro

File

bufferutils.h

C

```
#define _readul(pvalue, pb, buf_len, offset) _readl((bt_long_p)pvalue, pb, buf_len, offset)
```

Description

This is macro _readul.

_readuln Macro

File

bufferutils.h

C

```
#define _readuln(pvalue, pb, buf_len, offset) _readln((bt_long_p)pvalue, pb, buf_len, offset)
```

Description

This is macro _readuln.

_writebn Macro

File

bufferutils.h

C

```
#define _writebn _writeb
```

Description

This is macro _writebn.

bt_at_parser_callback_pf Type

File

at_parser.h

C

```
typedef void (* bt_at_parser_callback_pf)(struct _bt_at_parser_t* parser, bt_byte evt, void* evt_param,
void* callback_param);
```

Description

This is type bt_at_parser_callback_pf.

bt_at_parser_t Structure

File

at_parser.h

C

```
typedef struct _bt_at_parser_t {
  bt_byte state;
  bt_byte buffer[ATCMD_BUFFER_LEN + 1];
  bt_byte write_pos;
  bt_byte param_read_state;
  bt_byte param_read_nesting_level;
  bt_at_parser_callback_pf callback;
  void* callback_param;
} bt_at_parser_t;
```

Description

This is type bt_at_parser_t.

bt_bt_buffer_header_p Structure

File

buffer.h

C

```
typedef struct _bt_buffer_header_t {
  bt_int state;
} bt_buffer_header_t, * bt_bt_buffer_header_p;
```

Description

This is type bt_bt_buffer_header_p.

bt_buffer_header_t Structure

File

buffer.h

C

```
typedef struct _bt_buffer_header_t {
  bt_int state;
} bt_buffer_header_t, * bt_bt_buffer_header_p;
```

Description

This is type bt_buffer_header_t.

bt_buffer_mgr_p Structure

File

buffer.h

C

```
typedef struct _bt_buffer_mgr_t {
  bt_int max_buffers;
  bt_int buffer_size;
  bt_buffer_header_t* headers;
  bt_byte_p buffers;
} bt_buffer_mgr_t, * bt_buffer_mgr_p;
```

Description

This is type bt_buffer_mgr_p.

bt_buffer_mgr_t Structure

File

buffer.h

C

```
typedef struct _bt_buffer_mgr_t {
  bt_int max_buffers;
  bt_int buffer_size;
  bt_buffer_header_t* headers;
  bt_byte_p buffers;
} bt_buffer_mgr_t, * bt_buffer_mgr_p;
```

Description

This is type bt_buffer_mgr_t.

bt_packet_assembler_fp Type

File

packet.h

C

```
typedef bt_int (* bt_packet_assembler_fp)(struct _bt_packet_t* packet, bt_byte* buffer, bt_int buffer_len);
```

Description

This is type bt_packet_assembler_fp.

bt_packet_t Structure

File

packet.h

C

```
typedef struct _bt_packet_t {
  bt_packet_assembler_fp packet_assembler;
  const bt_byte* data;
  bt_int len;
  bt_byte state;
  bt_int data_pos;
  void* param;
} bt_packet_t;
```

Description

This is type bt_packet_t.

bt_queue_element_t Structure

File

queue.h

C

```
typedef struct _bt_queue_element_t {
  struct _bt_queue_element_t* next;
} bt_queue_element_t;
```

Description

This is type bt_queue_element_t.

bt_vcard_evt_prop_param_t Structure

File

vcard_parser.h

C

```
typedef struct _bt_vcard_evt_prop_param_t {
  bt_byte param_id;
  bt_byte* param_value;
  bt_byte param_len;
} bt_vcard_evt_prop_param_t;
```

Description

This is type bt_vcard_evt_prop_param_t.

bt_vcard_evt_prop_t Structure

File

vcard_parser.h

C

```
typedef struct _bt_vcard_evt_prop_t {
  bt_byte prop_id;
  bt_byte* prop_value;
  bt_byte value_len;
  bt_bool final;
} bt_vcard_evt_prop_t;
```

Description

This is type bt_vcard_evt_prop_t.

bt_vcard_parser_callback_fp Type

File

vcard_parser.h

C

```
typedef void (* bt_vcard_parser_callback_fp)(struct _bt_vcard_parser_t* parser, bt_byte evt, void*
evt_param, void* callback_param);
```

Description

This is type bt_vcard_parser_callback_fp.

bt_vcard_parser_t Structure

File

vcard_parser.h

C

```
typedef struct _bt_vcard_parser_t {
  bt_byte state;
  bt_byte* buffer;
  bt_uint buffer_size;
  bt_byte write_pos;
  bt_byte prev_c;
  bt_byte cur_prop_id;
  bt_byte cur_param_id;
  bt_bool vcard_started;
  bt_vcard_parser_callback_fp callback;
  void* callback_param;
} bt_vcard_parser_t;
```

Description

This is type bt_vcard_parser_t.

bt_xml_evt_attribute_t Structure

File

xml_parser.h

C

```
typedef struct _bt_xml_evt_attribute_t {
  bt_byte* name;
  bt_byte len;
} bt_xml_evt_attribute_t;
```

Description

This is type bt_xml_evt_attribute_t.

bt_xml_evt_attribute_value_t Structure

File

xml_parser.h

C

```
typedef struct _bt_xml_evt_attribute_value_t {
  bt_byte* value;
  bt_byte len;
} bt_xml_evt_attribute_value_t;
```

Description

This is type bt_xml_evt_attribute_value_t.

bt_xml_evt_tag_started_t Structure

File

xml_parser.h

C

```
typedef struct _bt_xml_evt_tag_started_t {
  bt_byte* name;
  bt_byte len;
} bt_xml_evt_tag_started_t;
```

Description

This is type bt_xml_evt_tag_started_t.

bt_xml_parser_callback_pf Type

File

xml_parser.h

C

```
typedef void (* bt_xml_parser_callback_pf)(struct _bt_xml_parser_t* parser, bt_byte evt, void* evt_param,
void* callback_param);
```

Description

This is type bt_xml_parser_callback_pf.

bt_xml_parser_t Structure

File

xml_parser.h

C

```
typedef struct _bt_xml_parser_t {
  bt_byte state;
  bt_byte* buffer;
  bt_uint buffer_size;
  bt_byte write_pos;
  bt_byte prev_c;
  bt_byte quote_c;
  bt_xml_parser_callback_pf callback;
  void* callback_param;
} bt_xml_parser_t;
```

Description

This is type bt_xml_parser_t.

Files

Files

Name	Description
a2dp.h	Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.
a2dp_codec_aac.h	Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.
a2dp_codec_mpeg.h	Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.
a2dp_codec_sbc.h	Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.
a2dp_private.h	Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.
avctp.h	Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.
avctp_config.h	Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.
avctp_packet.h	Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.
avctp_private.h	Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.
avdtp.h	Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.
avdtp_config.h	Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.
avdtp_control.h	Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.
avdtp_private.h	Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.
avrcp.h	Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.
avrcp_command.h	Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.
avrcp_config.h	Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.
avrcp_config_event_handlers.h	Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved. SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements. Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.
avrcp_private.h	Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.

at_parser.h	Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved. SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements. Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.
bt_bdaddr.h	Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.
baseband.h	Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved. SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements. Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.
bt_config.h	Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved. SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements. Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.
bt_hcitr.h	Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved. SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements. Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.
bt_log.h	Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved. SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements. Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.
bt_oem.h	Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved. SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements. Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.
bt_oem_config.h	Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved. SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements. Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.
bt_private.h	Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved. SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements. Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.

bt_signal.h	Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved. SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements. Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.
bt_std.h	Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved. SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements. Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.
bt_storage.h	Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved. SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements. Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.
bt_system.h	Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved. SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements. Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.
bt_timer.h	Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved. SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements. Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.
bt_types.h	Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved. SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements. Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.
buffer.h	Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved. SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements. Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.
bufferutils.h	Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved. SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements. Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.

channel.h	Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved. SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements. Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.
chmanager.h	Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved. SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements. Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.
cmdbuffer.h	Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved. SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements. Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.
command.h	Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved. SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements. Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.
config.h	Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.
csr.h	Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.
frame_buffer.h	Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved. SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements. Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.
gap.h	Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.
hci.h	Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved. SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements. Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.
hci_cmd_buffer.h	Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved. SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements. Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.

hci_cmd_queue.h	Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved. SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements. Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.
hci_config.h	Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved. SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements. Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.
hci_config_event_handlers.h	Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved. SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements. Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.
hci_conn_state.h	Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved. SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements. Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.
hci_ctrl_state.h	Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved. SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements. Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.
hci_data_buffer.h	Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved. SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements. Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.
hci_data_queue.h	Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved. SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements. Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.
hci_eir.h	Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved. SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements. Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.

hci_errors.h	Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved. SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements. Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.
hci_evt_handlers.h	Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved. SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements. Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.
hci_le.h	Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.
hci_linkkey_buffer.h	Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved. SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements. Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.
hci_private.h	Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved. SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements. Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.
hci_signal.h	Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved. SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements. Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.
hci_transport.h	Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved. SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements. Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.
hci_utils.h	Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved. SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements. Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.
hcitr_3wire.h	Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved. SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements. Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.

hcitr_bcsp.h	Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved. SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements. Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.
hcitr_packet.h	Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved. SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements. Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.
hcitr_tih4.h	Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved. SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements. Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.
hcitr_uart.h	Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved. SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements. Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.
lm.h	Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved. SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements. Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.
l2cap.h	Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved. SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements. Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.
l2cap_command_queue.h	Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved. SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements. Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.
l2cap_config.h	Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved. SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements. Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.

l2cap_config_handlers.h	Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved. SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements. Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.
l2cap_eretr.h	Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved. SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements. Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.
l2cap_fixed_channel.h	Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.
l2cap_packet.h	Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.
l2cap_private.h	Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved. SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements. Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.
l2cap_psm.h	Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved. SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements. Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.
l2cap_signal.h	Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved. SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements. Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.
l2cap_test.h	Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved. SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements. Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.
l2cap_timer.h	Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved. SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements. Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.
packet.h	Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.
patch.h	Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.

queue.h	Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved. SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements. Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.
rfcomm.h	Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved. SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements. Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.
rfcomm_cmd_queue.h	Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved. SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements. Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.
rfcomm_config.h	Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved. SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements. Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.
rfcomm_mgr.h	Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved. SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements. Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.
rfcomm_mx.h	Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved. SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements. Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.
rfcomm_private.h	Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved. SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements. Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.
rfcomm_signal.h	Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved. SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements. Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.

rfcomm_timer.h	Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved. SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements. Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.
sbc.h	Provides SBC Decoder Library interface services.
sdp.h	Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved. SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements. Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.
sdp_client.h	Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved. SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements. Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.
sdp_config.h	Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved. SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements. Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.
sdp_packet.h	Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.
sdp_private.h	Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved. SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements. Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.
sdp_utils.h	Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved. SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements. Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.
spp.h	Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved. SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements. Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.
spp_config.h	Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved. SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements. Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.

spp_private.h	Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved. SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements. Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.
ssp.h	Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved. SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements. Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.
ssp_event.h	Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved. SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements. Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.
ssp_event_handler.h	Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved. SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements. Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.
ti.h	Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.
ti_private.h	Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.
types.h	Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.
vcard_parser.h	Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved. SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements. Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.
xml_parser.h	Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved. SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements. Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.

Description

a2dp.h

Functions

	Name	Description
	bt_a2dp_find_server	This is function bt_a2dp_find_server.

	bt_a2dp_find_sink	brief Find sink ingroup a2dp details This function looks for a sink on a remote device specified by c deviceAddress and, if found, returns features supported by the sink. param deviceAddress The address of a remote device. param callback The callback function that will be called when search has completed. param client_callback The optional callback function that an application can set if it wants to be notified of state changes of the SDP client. The c evt parameter of the callback can be one of the following values: li SDP_CLIENT_STATE_IDLE li SDP_CLIENT_STATE_CONNECTING li SDP_CLIENT_STATE_DISCONNECTING li SDP_CLIENT_STATE_CONNECTED param callback_param A pointer... more
	bt_a2dp_find_source	brief Find source ingroup a2dp details This function looks for a source on a remote device specified by c deviceAddress and, if found, returns features supported by the source. param deviceAddress The address of a remote device. param callback The callback function that will be called when search has completed. param client_callback The optional callback function that an application can set if it wants to be notified of state changes of the SDP client. The c evt parameter of the callback can be one of the following values: li SDP_CLIENT_STATE_IDLE li SDP_CLIENT_STATE_CONNECTING li SDP_CLIENT_STATE_DISCONNECTING li SDP_CLIENT_STATE_CONNECTED param callback_param A pointer... more
	bt_a2dp_get_mgr	brief Return a pointer to an instance of the A2DP manager. ingroup a2dp details This function returns a pointer to an instance of the A@DP manager. There is only one instance of the manager allocated by the stack. The pointer is passed as the first parameter to all A2DP functions.
	bt_a2dp_init	brief Initialize the A2DP layer. ingroup a2dp details This function initializes the A2DP layer of the stack. It must be called prior to any other A2DP function can be called.
	bt_a2dp_open_and_start_stream	brief Open & start a stream ingroup a2dp details Opening a stream involves sending 3 requests to a remote device - "set configuration", "open stream" and "start stream". Each event generates its own event which must be handled and acted accordingly by the application. To make the use of API easier dotstack combines all these requests in one request called "open & start stream". dotstack sends necessary requests in a proper sequence, handles responses and generates only one event (A2DP_EVT_OPEN_AND_START_STREAM_COMPLETED) at the end. If any of the individual requests has failed the event's parameter bt_a2dp_event_t::open_and_start_stream_completed is populated with... more
	bt_a2dp_register_aac_codec	brief Register default AAC codec ingroup a2dp details This function adds AAC codec implemented by dotstack to the list of known codecs. For more information about codecs see description of ::bt_avdtp_register_codec. The only codec A2DP is mandatory to support is SBC. All other codecs are optional. If an application wants to use AAC codec it must call this function when it is initializing. note dotstack codecs do not do actual encoding/decoding. their function is to parse and serialize codec's configuration. param mgr A2DP manager.
	bt_a2dp_register_callback	brief Register a A2DP application callback. ingroup a2dp details In order to be notified of various events a consumer of the A2DP layer has to register a callback function. The stack will call this function whenever a new event has been generated passing the code of the event as the second parameter. The event can be one of the following values: @arg A2DP_EVT_CTRL_CHANNEL_CONNECTED: Control channel connected. @arg A2DP_EVT_CTRL_CHANNEL_DISCONNECTED: Control channel disconnected. @arg A2DP_EVT_CTRL_CONNECTION_FAILED: Control channel connection failed (generated only if control connection has been initiated by the local device). @arg A2DP_EVT_DISCOVER_COMPLETED: Local device completed discovering remote SEPs.... more
	bt_a2dp_register_mpeg_codec	brief Register default MPEG codec ingroup a2dp details This function adds MPEG codec implemented by dotstack to the list of known codecs. For more information about codecs see description of ::bt_avdtp_register_codec. The only codec A2DP is mandatory to support is SBC. All other codecs are optional. If an application wants to use MPEG-1,2 codec it must call this function when it is initializing. note dotstack codecs do not do actual encoding/decoding. their function is to parse and serialize codec's configuration. param mgr A2DP manager.
	bt_a2dp_set_configuration	This is function bt_a2dp_set_configuration.

	bt_a2dp_start	brief Start the A2DP layer. ingroup a2dp details This function makes the A2DP layer ready to accept connection requests from remote device. To make an outgoing connection calling this function is not required. param mgr AVDTP manager. return li c TRUE if the function succeeds. li c FALSE otherwise.

Macros

	Name	Description
	__A2DP_H	This is macro __A2DP_H.
	A2DP_EVT_ABORT_STREAM_COMPLETED	< This event is generated when a local device received a response (either positive or negative) to a "abort stream" request.
	A2DP_EVT_ABORT_STREAM_REQUESTED	< This event is generated when a local device received "abort stream" request.
	A2DP_EVT_CLOSE_STREAM_COMPLETED	< This event is generated when a local device received a response (either positive or negative) to a "close stream" request.
	A2DP_EVT_CLOSE_STREAM_REQUESTED	< This event is generated when a local device received "close stream" request.
	A2DP_EVT_CTRL_CHANNEL_CONNECTED	< This event is generated when a control channel between two AVDTP entities has been established.
	A2DP_EVT_CTRL_CHANNEL_DISCONNECTED	< This event is generated when a control channel between two AVDTP entities has been terminated.
	A2DP_EVT_CTRL_CONNECTION_FAILED	< This event is generated when a local device failed to create a control channel between two AVDTP entities.
	A2DP_EVT_DISCOVER_SEP_COMPLETED	< This event is generated when a local device received a response (either positive or negative) to a "discover" request.
	A2DP_EVT_GET_SEP_CAPABILITIES_COMPLETED	< This event is generated when a local device received a response (either positive or negative) to a "get SEP capabilities" request.
	A2DP_EVT_GET_STREAM_CONFIGURATION_COMPLETED	< This event is generated when a local device received a response (either positive or negative) to a "get stream configuration" request.
	A2DP_EVT_MEDIA_PACKET_RECEIVED	< This event is generated when a local device received a media packet.
	A2DP_EVT_MEDIA_PACKET_SEND_FAILED	< This event is generated when a local device failed to send a media packet.
	A2DP_EVT_MEDIA_PACKET_SENT	< This event is generated when a local device sent a media packet.
	A2DP_EVT_NOTHING	This is macro A2DP_EVT_NOTHING.
	A2DP_EVT_OPEN_AND_START_STREAM_COMPLETED	< This event is generated when a local device completed "open and start" request.
	A2DP_EVT_OPEN_STREAM_COMPLETED	< This event is generated when a local device received a response (either positive or negative) to a "open stream" request.
	A2DP_EVT_OPEN_STREAM_REQUESTED	< This event is generated when a local device received "open stream" request.
	A2DP_EVT_RECONFIGURE_STREAM_COMPLETED	< This event is generated when a local device received a response (either positive or negative) to a "change stream configuration" request.
	A2DP_EVT_RECONFIGURE_STREAM_REQUESTED	< This event is generated when a local device received "change stream configuration" request.
	A2DP_EVT_SEP_CAPABILITIES_RECEIVED	< This event is generated when a local device received a positive response to a "get SEP capabilities" request.
	A2DP_EVT_SEP_INFO_RECEIVED	< This event is generated for each SEP contained in a positive response to a "discover" request.
	A2DP_EVT_SET_STREAM_CONFIGURATION	This is macro A2DP_EVT_SET_STREAM_CONFIGURATION.
	A2DP_EVT_SET_STREAM_CONFIGURATION_COMPLETED	< This event is generated when a local device received a response (either positive or negative) to a "set stream configuration" request.
	A2DP_EVT_SET_STREAM_CONFIGURATION_REQUESTED	< This event is generated when a local device received "set stream configuration" request.
	A2DP_EVT_START_STREAM_COMPLETED	< This event is generated when a local device received a response (either positive or negative) to a "start stream" request.
	A2DP_EVT_START_STREAM_REQUESTED	< This event is generated when a local device received "start stream" request.

	A2DP_EVT_STREAM_ABORTED	< This event is generated when a local device has successfully aborted a stream. < This event follows the A2DP_EVT_ABORT_STREAM_REQUESTED if the upper layer has accepted it. < This event is not generated if stream abortion was initiated by the local device.
	A2DP_EVT_STREAM_CLOSED	< This event is generated when a local device has successfully closed a stream. < This event follows the A2DP_EVT_CLOSE_STREAM_REQUESTED if the upper layer has accepted it. < This event is not generated if stream closing was initiated by the local device.
	A2DP_EVT_STREAM_CONFIGURATION_RECEIVED	< This event is generated when a local device received a positive response to a "get stream configuration" request.
	A2DP_EVT_STREAM_CONFIGURED	< This event is generated when a local device has successfully configured a stream. < This event follows the A2DP_EVT_SET_STREAM_CONFIGURATION_REQUESTED if the upper layer has accepted it. < This event is not generated if stream configuration was initiated by the local device.
	A2DP_EVT_STREAM_OPENED	< This event is generated when a local device has successfully opened a stream. < This event follows the A2DP_EVT_OPEN_STREAM_REQUESTED if the upper layer has accepted it. < This event is not generated if stream opening was initiated by the local device.
	A2DP_EVT_STREAM_RECONFIGURED	< This event is generated when a local device has successfully reconfigured a stream. < This event follows the A2DP_EVT_RECONFIGURE_STREAM_REQUESTED if the upper layer has accepted it. < This event is not generated if stream reconfiguration was initiated by the local device.
	A2DP_EVT_STREAM_SECURITY_CONTROL_COMPLETED	< This event is generated when a local device received a response (either positive or negative) to a "exchange content protection control data" request.
	A2DP_EVT_STREAM_STARTED	< This event is generated when a local device has successfully started a stream. < This event follows the A2DP_EVT_START_STREAM_REQUESTED if the upper layer has accepted it. < This event is not generated if stream starting was initiated by the local device.
	A2DP_EVT_STREAM_SUSPENDED	< This event is generated when a local device has successfully suspended a stream. < This event follows the A2DP_EVT_SUSPEND_STREAM_REQUESTED if the upper layer has accepted it. < This event is not generated if stream suspension was initiated by the local device.
	A2DP_EVT_SUSPEND_STREAM_COMPLETED	< This event is generated when a local device received a response (either positive or negative) to a "suspend stream" request.
	A2DP_EVT_SUSPEND_STREAM_REQUESTED	< This event is generated when a local device received "suspend stream" request.
	A2DP_MANAGER_STATE_CONNECTING	This is macro A2DP_MANAGER_STATE_CONNECTING.
	A2DP_MANAGER_STATE_IDLE	This is macro A2DP_MANAGER_STATE_IDLE.
	A2DP_SINK_FEATURE_AMPLIFIER	< Amplifier
	A2DP_SINK_FEATURE_HEADPHONE	< Headphone
	A2DP_SINK_FEATURE_RECORDER	< Recorder
	A2DP_SINK_FEATURE_SPEAKER	< Speaker
	A2DP_SOURCE_FEATURE_MICROPHONE	< Mic
	A2DP_SOURCE_FEATURE_MIXER	< Mixer
	A2DP_SOURCE_FEATURE_PLAYER	< Player
	A2DP_SOURCE_FEATURE_TUNER	< Tuner

	bt_a2dp_abort_stream	brief Suspend a stream. ingroup a2dp details This function tries to suspend a stream by sending a request to the remote party. The stream can be in any state state except AVDTP_STREAM_STATE_IDLE. As a result of this operation the A2DP_EVT_ABORT_STREAM_COMPLETED event will be generated. This operation cannot be rejected. The p evt_param.abort_stream_requested.err_code is always == AVDTP_ERROR_SUCCESS. param mgr A2DP manager. param strm_handle Stream handle. return li c TRUE if the function succeeds, i.e. the actual request has been sent to the remote party. li c FALSE otherwise. No events will be generated.
	bt_a2dp_add_media_rx_buffer	brief Add a media packet buffer to a receive queue ingroup a2dp details The consumer of A2DP is responsible for allocating and supplying A2DP with buffers used to store received packets. A2DP itself only has a queue for storing pointers to buffers supplied by the consumer. When a packet comes in A2DP finds the first buffer large enough to hold the received packet, copies the packet to the buffer and generates a A2DP_EVT_MEDIA_PACKET_RECEIVED event. The consumer then has to process the data in the buffer and return it back to the queue. If there is no buffers in the queue... more
	bt_a2dp_add_media_tx_buffer	brief Add a media packet buffer to a send queue ingroup a2dp details When the consumer of A2DP wants to send a packet to a remote device it calls this function. The function adds the packet to a queue and tells A2DP that it has something to send. The packet will be send as soon as the stream goes to A2DP_STREAM_STATE_STREAMING state. When the packet has been successfully sent a A2DP_EVT_MEDIA_PACKET_SENT is generated. Otherwise a A2DP_EVT_MEDIA_PACKET_SEND_FAILED is generated. Regardless of the event generated the consumer can re-use the buffer as A2DP has removed it from the queue and gave up... more
	bt_a2dp_call_codec	This is macro bt_a2dp_call_codec.
	bt_a2dp_cancel_listen	brief Cancel listening for incoming connections. ingroup a2dp details This function removes a SEP from a list of SEPS which a stream can use for incoming requests. param mgr A2DP manager. param strm_handle Stream handle. param sep_id Local SEP ID. return li c TRUE if the function succeeds. li c FALSE otherwise.
	bt_a2dp_clear_media_tx_queue	brief Clear send queue ingroup a2dp details When the consumer of A2DP wants to send a packet to a remote device it calls bt_avdtp_add_media_tx_buffer function. The function adds the packet to a queue and tells A2DP that it has something to send. The packet will be send as soon as the stream goes to A2DP_STREAM_STATE_STREAMING state. The consumer can remove all packets from the queue before they have been sent to a remote device by calling ::bt_a2dp_clear_media_tx_queue. param mgr A2DP manager. param strm_handle Stream handle. return li c TRUE if the function succeeds. li c FALSE otherwise. The function fails... more
	bt_a2dp_close_stream	brief Close a stream. ingroup a2dp details This function tries to close a stream by sending a request to the remote party. The stream has to be in AVDTP_STREAM_STATE_OPEN or AVDTP_STREAM_STATE_STREAMING state. As a result of this operation the A2DP_EVT_CLOSE_STREAM_COMPLETED event will be generated. If the stream has been closed the p evt_param.bt_avdtp_evt_close_stream_completed_t.err_code == AVDTP_ERROR_SUCCESS. Otherwise, if the remote device for any reason cannot or does not wish to close the stream, the p evt_param.bt_avdtp_evt_close_stream_completed_t.err_code == the error code sent by the remote. param mgr A2DP manager. param strm_handle Stream handle. return li c TRUE if the function succeeds,... more

	bt_a2dp_connect	brief Connect to a remote device. ingroup a2dp details This function opens a control channel connection to a remote device specified by the p remote_addr. If connection cannot be initiated for some reason, for example, there is not enough resources, it returns FALSE and not events are generated. Otherwise the result of an attempt to connect to the remote device is reported via the AVDTP callback. The events generated will either be A2DP_EVT_CTRL_CHANNEL_CONNECTED or A2DP_EVT_CTRL_CHANNEL_CONNECTION_FAILED. param mgr A2DP manager. param remote_addr The address of a remote device. return li c TRUE if connection establishment has been started. li c FALSE... more
	bt_a2dp_connect_ex	brief Connect to a remote device. ingroup a2dp details This function opens a control channel connection to a remote device specified by the p remote_addr. If connection cannot be initiated for some reason, for example, there is not enough resources, it returns FALSE and not events are generated. Otherwise the result of an attempt to connect to the remote device is reported via the AVDTP callback. The events generated will either be A2DP_EVT_CTRL_CHANNEL_CONNECTED or A2DP_EVT_CTRL_CHANNEL_CONNECTION_FAILED. param mgr A2DP manager. param remote_addr The address of a remote device. param acl_config ACL link configuration. This can be a combination of the following... more
	bt_a2dp_create_stream	brief Create a stream. ingroup a2dp details This function allocates memory for storing stream's data and assigns a stream handle. The stream handle is used to manipulate the stream - open, close, configure, suspend, abort. param mgr A2DP manager. return li c Stream handle if the function succeeds. li c 0 otherwise.
	bt_a2dp_destroy_stream	brief Destroy a stream. ingroup a2dp details This function frees memory used by the stream. The stream has to exist and be in the "idle" state for this function to succeed. I.e. the stream has to be closed or aborted before this function can be called. param mgr A2DP manager. param strm_handle Stream handle. return li c TRUE if the function succeeds. li c FALSE otherwise.
	bt_a2dp_disconnect	brief Disconnect from a remote device. ingroup a2dp details This function closes a control and transport channels on all streams associated with the remote device specified by the p remote_addr. As a result of this operation the following events will be generated: @arg A2DP_EVT_MEDIA_PACKET_RECEIVED: if a stream's receive queue is not empty this event is generated for each buffer with bt_media_packet_t::data_len set to 0 @arg A2DP_EVT_MEDIA_PACKET_SENT: if a stream's send queue is not empty this event is generated for each buffer with bt_media_packet_t::data_len set to 0 @arg A2DP_EVT_STREAM_CLOSED: this event is generate if a stream is in... more
	bt_a2dp_discover	brief Discover SEPs on a remote device. ingroup a2dp details This function asks the remote device to send a list of all available SEPs. As a result of this operation the following events will be generated: @arg A2DP_EVT_SEP_INFO_RECEIVED: this event is generated for every SEP received from the remote device. the p evt_param.sep_info_received contains SEP information. @arg A2DP_EVT_DISCOVER_COMPLETED: this event is generated after the last A2DP_EVT_SEP_INFO_RECEIVED if the remote accepted the request and the p evt_param.discover_completed.err_code == AVDTP_ERROR_SUCCESS. if the remote rejected the request the p evt_param.discover_completed.err_code == the error code sent by the remote. param mgr A2DP... more

	bt_a2dp_find_codec	brief Find a codec ingroup a2dp details A2DP in theory can support any type of codec. Each codec uses its own format for exchanging capabilities and configuration information. In order to make our implementation do not care about these formats we use a simple way of telling AVDTP how to parse and serialize codec's configuration. The consumer of A2DP has to register a callback function (one per codec type) for each codec it wishes to support. That callback has to perform two function. The first one is to read the configuration received from the remote device and store it in... more
	bt_a2dp_get_all_capabilities	brief Get remote SEP capabilities. ingroup a2dp details This function asks the remote device to send capabilities of a SEP specified by the p seid_acp. As a result of this operation the following events will be generated: @arg A2DP_EVT_SEP_CAPABILITIES_RECEIVED: this event is generated if the remote device accepted the request. the p evt_param.sep_capabilities_received contains SEP capabilities. @arg A2DP_EVT_GET_SEP_CAPABILITIES_COMPLETED: this event is generated right after A2DP_EVT_SEP_CAPABILITIES_RECEIVED if the remote accepted the request the p evt_param.get_sep_capabilities_completed.err_code == AVDTP_ERROR_SUCCESS. if the remote rejected the request the p evt_param.get_sep_capabilities_completed.err_code == the error code sent by the remote. param mgr A2DP manager.... more
	bt_a2dp_get_capabilities	brief Get remote SEP capabilities. ingroup a2dp details This function asks the remote device to send capabilities of a SEP specified by the p seid_acp. As a result of this operation the following events will be generated: @arg A2DP_EVT_SEP_CAPABILITIES_RECEIVED: this event is generated if the remote device accepted the request. the p evt_param.sep_capabilities_received contains SEP capabilities. @arg A2DP_EVT_GET_SEP_CAPABILITIES_COMPLETED: this event is generated right after A2DP_EVT_SEP_CAPABILITIES_RECEIVED if the remote accepted the request the p evt_param.get_sep_capabilities_completed.err_code == AVDTP_ERROR_SUCCESS. if the remote rejected the request the p evt_param.get_sep_capabilities_completed.err_code == the error code sent by the remote. param mgr A2DP manager.... more
	bt_a2dp_get_configuration	This is macro bt_a2dp_get_configuration.
	bt_a2dp_get_hci_connection	brief Get HCI connection for a stream ingroup a2dp details This function returns a pointer to a structure that describes an HCI connection a stream is open on. The return value can be used to call various function from the HCI layer. For example, if an app wants to force disconnection from a remote device it can call ::bt_hci_disconnect. param mgr A2DP manager. param strm_handle Stream handle. return li c Pointer to a structure that describes an HCI connection if the function succeeds. li c NULL otherwise. The function fails only if a stream specified by the p... more
	bt_a2dp_get_stream_codec_config	brief Get the configuration of the codec currently used with the stream. ingroup a2dp details This function returns a pointer to a structure that contains configuration of the codec currently used with the stream. The structure returned depends on the codec. The dotstack defines structures only for SBC, MPEG-1,2 and MPEG-2,4 AAC codecs: @arg SBC: bt_a2dp_sbc_config_t (defined in a2dp_sbc_codec.h) @arg MPEG-1,2: bt_a2dp_mpeg_config_t (defined in a2dp_mpeg_codec.h) @arg MPEG-2,4 AAC: bt_a2dp_aac_config_t (defined in a2dp_aac_codec.h) param mgr A2DP manager. param strm_handle Stream handle. return li The codec's configuration if strm_handle specifies a valid stream and the stream is in one of the following... more

	bt_a2dp_get_stream_codec_type	brief Get the type of the codec currently used with the stream. ingroup a2dp details This function returns the type of the codec currently used with the stream. param mgr A2DP manager. param strm_handle Stream handle. return @arg The type of the codec if strm_handle specifies a valid stream and the stream is in one of the following states: AVDTP_STREAM_STATE_CONFIGURED AVDTP_STREAM_STATE_OPEN AVDTP_STREAM_STATE_STREAMING @arg The result will be one of the following values: AVDTP_CODEC_TYPE_SBC: SBC AVDTP_CODEC_TYPE_MPEG1_2_AUDIO: MPEG-1,2 (used in MP3 files) AVDTP_CODEC_TYPE_MPEG2_4_AAC: MPEG-2,4 AAC (used in Apple products) AVDTP_CODEC_TYPE_ATRAC: ATRAC (used in Sony products) AVDTP_CODEC_TYPE_NON_A2DP: Non-A2DP... more
	bt_a2dp_get_stream_config	brief Get stream's configuration. ingroup a2dp details This function returns a pointer to a structure holding current configuration of the stream. param mgr A2DP manager. param strm_handle Stream handle. return li The stream's configuration if strm_handle specifies a valid stream and the stream is in one of the following state: AVDTP_STREAM_STATE_CONFIGURED AVDTP_STREAM_STATE_OPEN AVDTP_STREAM_STATE_STREAMING li NULL otherwise.
	bt_a2dp_get_stream_direction	This is macro bt_a2dp_get_stream_direction.
	bt_a2dp_get_stream_local_sep_id	brief Get stream's local SEP ID. ingroup a2dp details This function returns the ID of the local SEP associated with the stream. param mgr A2DP manager. param strm_handle Stream handle. return li The ID of the local SEP if strm_handle specifies a valid stream. li 0 otherwise.
	bt_a2dp_get_stream_remote_address	brief Get stream's remote BT address. ingroup a2dp details This function returns the address of the remote device associated with the stream. param mgr A2DP manager. param strm_handle Stream handle. return li The address of the remote device if strm_handle specifies a valid stream. li NULL otherwise.
	bt_a2dp_get_stream_remote_sep_id	brief Get stream's remote SEP ID. ingroup a2dp details This function returns the ID of the remote SEP associated with the stream. param mgr A2DP manager. param strm_handle Stream handle. return li The ID of the remote SEP if strm_handle specifies a valid stream. li 0 otherwise.
	bt_a2dp_get_stream_state	brief Get local stream state. ingroup a2dp details This function returns local state of a stream specified by the p strm_handle. No request is sent to the remote party. param mgr A2DP manager. param strm_handle Stream handle. return The state of the stream. The result will be one of the following values: @arg AVDTP_STREAM_STATE_IDLE: The stream is idle. This can mean two things. The stream specified by p strm_handle does not exist or the stream is closed. @arg AVDTP_STREAM_OPENING_TRANSPORT_CHANNELS: The stream is opening transport channels. @arg AVDTP_STREAM_CLOSING_TRANSPORT_CHANNELS: The stream is closing transport channels. @arg AVDTP_STREAM_STATE_CONFIGURED: The... more

	bt_a2dp_listen	brief Listen for incoming connections. ingroup a2dp details This function tells a stream that it can use a particular SEP to accept incoming requests to open it. The SEP can be associated with multiple streams but used with only one. The stream has to be closed before the SEP can be used with another stream. For outgoing connections this is not needed. Any SEP can be used with any stream given that the SEP is not already in use by another stream. param mgr A2DP manager. param strm_handle Stream handle. param sep_id Local SEP ID. return li c TRUE if... more
	bt_a2dp_open_stream	This is macro bt_a2dp_open_stream.
	bt_a2dp_reconfigure_stream	brief Reconfigure stream. ingroup a2dp details This function tries to change the stream's configuration. For this function to succeed the stream has to be open. As a result of this operation the A2DP_EVT_STREAM_RECONFIGURE_COMPLETED event will be generated. If reconfiguration was a success the p evt_param.stream_reconfigure_completed.err_code == AVDTP_ERROR_SUCCESS. Otherwise the p evt_param.stream_reconfigure_completed.err_code == the error code sent by the remote. param mgr A2DP manager. param strm_handle Stream handle. param caps New stream configuration. return li c TRUE if the function succeeds, i.e. the actual request has been sent to the remote party. li c FALSE otherwise. No events will be... more
	bt_a2dp_register_sink	brief Register a Sink SEP with the local A2DP manager. ingroup a2dp details This function is used to add a sink SEP to a list of SEPs supported by the local A2DP entity. param mgr A2DP manager. param caps The capabilities of a SEP. return li c ID of a SEP if the function succeeds. li c FALSE otherwise.
	bt_a2dp_register_source	brief Register a Source SEP with the local A2DP manager. ingroup a2dp details This function is used to add a source SEP to a list of SEPs supported by the local A2DP entity. param mgr A2DP manager. param caps The capabilities of a SEP. return li c ID of a SEP if the function succeeds. li c FALSE otherwise.
	bt_a2dp_remove_media_rx_buffer	brief Remove a media packet buffer from a receive queue ingroup a2dp details The consumer of A2DP is responsible for allocating and supplying A2DP with buffers used to store received packets. A2DP itself only has a queue for storing pointers to buffers supplied by the consumer. When a packet comes in A2DP finds the first buffer large enough to hold the received packet, copies the packet to the buffer and generates a A2DP_EVT_MEDIA_PACKET_RECEIVED event. The consumer then has to process the data in the buffer and return it back to the queue. If there is no buffers in the queue... more
	bt_a2dp_remove_media_tx_buffer	brief Remove a media packet buffer from a send queue ingroup a2dp details When the consumer of A2DP wants to send a packet to a remote device it calls bt_avdtp_add_media_tx_buffer function. The function adds the packet to a queue and tells A2DP that it has something to send. The packet will be send as soon as the stream goes to A2DP_STREAM_STATE_STREAMING state. The consumer has a chance to remove a packet from the queue before it has been sent to a remote device by calling ::bt_a2dp_remove_media_tx_buffer. param mgr A2DP manager. param strm_handle Stream handle. param buffer Pointer to a structure... more
	bt_a2dp_report_delay	This is macro bt_a2dp_report_delay.

	bt_a2dp_set_media_tx_queue_limit	brief Set limit on the send queue ingroup a2dp details When the consumer of A2DP wants to send a packet to a remote device it calls bt_avdtp_add_media_tx_buffer function. The function adds the packet to a queue and tells A2DP that it has something to send. The packet will be send as soon as the stream goes to A2DP_STREAM_STATE_STREAMING state. By default the send queue can contain unlimited number of packets. The consumer can set a limit on how many packets are held in the queue. In this case when new packet is added to the queue and the length of... more
	bt_a2dp_start_stream	brief Start a stream. ingroup a2dp details This function tries to start a stream by sending a request to the remote party. The stream has to be in AVDTP_STREAM_STATE_OPEN state. The stream goes to this state as a result of successful configuration or suspension (both can be initiated by either party). As a result of this operation the A2DP_EVT_START_STREAM_COMPLETED event will be generated. If the stream has been open the p evt_param.start_stream_requested.err_code == AVDTP_ERROR_SUCCESS. Otherwise, if the remote device for any reason cannot or does not wish to start the stream, the p evt_param.start_stream_requested.err_code == the error code sent... more
	bt_a2dp_suspend_stream	brief Suspend a stream. ingroup a2dp details This function tries to suspend a stream by sending a request to the remote party. The stream has to be in AVDTP_STREAM_STATE_STREAMING state. As a result of this operation the A2DP_EVT_SUSPEND_STREAM_COMPLETED event will be generated. If the stream has been suspended the p evt_param.bt_avdtp_evt_suspend_stream_requested_t.err_code == AVDTP_ERROR_SUCCESS. Otherwise, if the remote device for any reason cannot or does not wish to suspend the stream, the p evt_param.bt_avdtp_evt_suspend_stream_requested_t.err_code == the error code sent by the remote. param mgr A2DP manager. param strm_handle Stream handle. return li c TRUE if the function succeeds, i.e. the... more

Structures

	Name	Description
	_bt_a2dp_evt_open_and_start_stream_completed_s	brief Parameter to A2DP_EVT_OPEN_AND_START_STREAM_COMPLETED event ingroup a2dp details A pointer to this structure is passed to the A2DP application callback as a valid member of the bt_a2dp_event_t union - bt_a2dp_event_t::open_and_start_stream_completed - when A2DP completed a "open & start stream" request.
	_bt_a2dp_mgr_t	brief A2DP manager. ingroup a2dp details A structure that glues all pieces together. There is only one instance of this structure allocated by dotstack. A pointer to the instance can be get with ::bt_a2dp_get_mgr.
	bt_a2dp_evt_open_and_start_stream_completed_t	brief Parameter to A2DP_EVT_OPEN_AND_START_STREAM_COMPLETED event ingroup a2dp details A pointer to this structure is passed to the A2DP application callback as a valid member of the bt_a2dp_event_t union - bt_a2dp_event_t::open_and_start_stream_completed - when A2DP completed a "open & start stream" request.

Types

	Name	Description
	bt_a2dp_event_t	This is type bt_a2dp_event_t.

	bt_a2dp_find_server_callback_fp	brief Notify the application of the result of searching for a remote A2DP entity (sourse or sink) ingroup a2dp details This function is called by the A2DP layer when searching for an A2DP entity on a remote device has completed. param supported_features Features supported by a remote A2DP entity. param found c TRUE if an A2DP entity has been found on the remote device. c FALSE otherwise. param param pointer to arbitrary data passed to the bt_a2dp_find_source() or bt_a2dp_find_sink function through its c callback_param parameter.
	bt_a2dp_mgr_callback_fp	brief A2DP application callback. ingroup a2dp details In order to be notified of various events a consumer of the A2DP layer has to register a callback function. The stack will call that function whenever a new event has been generated. param mgr A2DP manager. param evt A2DP event. The event can be one of the following values: @arg A2DP_EVT_CTRL_CHANNEL_CONNECTED: Control channel connected. @arg A2DP_EVT_CTRL_CHANNEL_DISCONNECTED: Control channel disconnected. @arg A2DP_EVT_CTRL_CONNECTION_FAILED: Control channel connection failed (generated only if control connection has been initiated by the local device). @arg A2DP_EVT_DISCOVER_COMPLETED: Local device completed discovering remote SEPs. @arg A2DP_EVT_GET_SEP_CAPABILITIES_COMPLETED: Local... more
	bt_a2dp_mgr_t	This is type bt_a2dp_mgr_t.

Unions

	Name	Description
	_bt_a2dp_event_u	brief Parameter to an application callback. ingroup a2dp details This union is used to pass event specific data to the A2DP consumer. Which member of the union points to a valid structure depends on the event reported to the consumer. In general, each event has a corresponding member in the union.

Description

Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.

a2dp_codec_aac.h

Functions

	Name	Description
	bt_a2dp_aac_codec_handler	This is function bt_a2dp_aac_codec_handler.

Macros

	Name	Description
	__A2DP_CODEC_AAC_H	This is macro __A2DP_CODEC_AAC_H.
	AAC_CHANNELS_1	This is macro AAC_CHANNELS_1.
	AAC_CHANNELS_2	This is macro AAC_CHANNELS_2.
	AAC_CHANNELS_ALL	This is macro AAC_CHANNELS_ALL.
	AAC_OBJECT_TYPE_MPEG_2_LC	This is macro AAC_OBJECT_TYPE_MPEG_2_LC.
	AAC_OBJECT_TYPE_MPEG_4_LC	This is macro AAC_OBJECT_TYPE_MPEG_4_LC.
	AAC_OBJECT_TYPE_MPEG_4_LTP	This is macro AAC_OBJECT_TYPE_MPEG_4_LTP.
	AAC_OBJECT_TYPE_MPEG_4_SCALABLE	This is macro AAC_OBJECT_TYPE_MPEG_4_SCALABLE.
	AAC_SAMPLING_FREQUENCY_11025	This is macro AAC_SAMPLING_FREQUENCY_11025.
	AAC_SAMPLING_FREQUENCY_12000	This is macro AAC_SAMPLING_FREQUENCY_12000.
	AAC_SAMPLING_FREQUENCY_16000	This is macro AAC_SAMPLING_FREQUENCY_16000.
	AAC_SAMPLING_FREQUENCY_22050	This is macro AAC_SAMPLING_FREQUENCY_22050.
	AAC_SAMPLING_FREQUENCY_24000	This is macro AAC_SAMPLING_FREQUENCY_24000.
	AAC_SAMPLING_FREQUENCY_32000	This is macro AAC_SAMPLING_FREQUENCY_32000.
	AAC_SAMPLING_FREQUENCY_44100	This is macro AAC_SAMPLING_FREQUENCY_44100.
	AAC_SAMPLING_FREQUENCY_48000	This is macro AAC_SAMPLING_FREQUENCY_48000.
	AAC_SAMPLING_FREQUENCY_64000	This is macro AAC_SAMPLING_FREQUENCY_64000.
	AAC_SAMPLING_FREQUENCY_8000	This is macro AAC_SAMPLING_FREQUENCY_8000.
	AAC_SAMPLING_FREQUENCY_88200	This is macro AAC_SAMPLING_FREQUENCY_88200.
	AAC_SAMPLING_FREQUENCY_96000	This is macro AAC_SAMPLING_FREQUENCY_96000.

	AAC_SAMPLING_FREQUENCY_ALL	This is macro AAC_SAMPLING_FREQUENCY_ALL.
	AAC_VBR_NOT_SUPPORTED	This is macro AAC_VBR_NOT_SUPPORTED.
	AAC_VBR_SUPPORTED	This is macro AAC_VBR_SUPPORTED.

Structures

	Name	Description
	_bt_a2dp_aac_config_t	This is type bt_a2dp_aac_config_t.
	bt_a2dp_aac_config_t	This is type bt_a2dp_aac_config_t.

Description

Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.

a2dp_codec_mpeg.h

Functions

	Name	Description
	bt_a2dp_mpeg_codec_handler	This is function bt_a2dp_mpeg_codec_handler.

Macros

	Name	Description
	__A2DP_CODEC_MPEG_H	This is macro __A2DP_CODEC_MPEG_H.
	MPEG_BITRATE_0000	This is macro MPEG_BITRATE_0000.
	MPEG_BITRATE_0001	This is macro MPEG_BITRATE_0001.
	MPEG_BITRATE_0010	This is macro MPEG_BITRATE_0010.
	MPEG_BITRATE_0011	This is macro MPEG_BITRATE_0011.
	MPEG_BITRATE_0100	This is macro MPEG_BITRATE_0100.
	MPEG_BITRATE_0101	This is macro MPEG_BITRATE_0101.
	MPEG_BITRATE_0110	This is macro MPEG_BITRATE_0110.
	MPEG_BITRATE_0111	This is macro MPEG_BITRATE_0111.
	MPEG_BITRATE_1000	This is macro MPEG_BITRATE_1000.
	MPEG_BITRATE_1001	This is macro MPEG_BITRATE_1001.
	MPEG_BITRATE_1010	This is macro MPEG_BITRATE_1010.
	MPEG_BITRATE_1011	This is macro MPEG_BITRATE_1011.
	MPEG_BITRATE_1100	This is macro MPEG_BITRATE_1100.
	MPEG_BITRATE_1101	This is macro MPEG_BITRATE_1101.
	MPEG_BITRATE_1110	This is macro MPEG_BITRATE_1110.
	MPEG_BITRATE_ALL	This is macro MPEG_BITRATE_ALL.
	MPEG_CHANNEL_MODE_ALL	This is macro MPEG_CHANNEL_MODE_ALL.
	MPEG_CHANNEL_MODE_DUAL_CHANNEL	This is macro MPEG_CHANNEL_MODE_DUAL_CHANNEL.
	MPEG_CHANNEL_MODE_JOINT_STEREO	This is macro MPEG_CHANNEL_MODE_JOINT_STEREO.
	MPEG_CHANNEL_MODE_MONO	This is macro MPEG_CHANNEL_MODE_MONO.
	MPEG_CHANNEL_MODE_STEREO	This is macro MPEG_CHANNEL_MODE_STEREO.
	MPEG_CRC_PROTECTION_NOT_SUPPORTED	This is macro MPEG_CRC_PROTECTION_NOT_SUPPORTED.
	MPEG_CRC_PROTECTION_SUPPORTED	This is macro MPEG_CRC_PROTECTION_SUPPORTED.
	MPEG_LAYER_1	This is macro MPEG_LAYER_1.
	MPEG_LAYER_2	This is macro MPEG_LAYER_2.
	MPEG_LAYER_3	This is macro MPEG_LAYER_3.
	MPEG_LAYER_ALL	This is macro MPEG_LAYER_ALL.
	MPEG_MPF_1	This is macro MPEG_MPF_1.
	MPEG_MPF_2	This is macro MPEG_MPF_2.
	MPEG_SAMPLING_FREQUENCY_16000	This is macro MPEG_SAMPLING_FREQUENCY_16000.
	MPEG_SAMPLING_FREQUENCY_22050	This is macro MPEG_SAMPLING_FREQUENCY_22050.
	MPEG_SAMPLING_FREQUENCY_24000	This is macro MPEG_SAMPLING_FREQUENCY_24000.
	MPEG_SAMPLING_FREQUENCY_32000	This is macro MPEG_SAMPLING_FREQUENCY_32000.

	MPEG_SAMPLING_FREQUENCY_44100	This is macro MPEG_SAMPLING_FREQUENCY_44100.
	MPEG_SAMPLING_FREQUENCY_48000	This is macro MPEG_SAMPLING_FREQUENCY_48000.
	MPEG_SAMPLING_FREQUENCY_ALL	This is macro MPEG_SAMPLING_FREQUENCY_ALL.
	MPEG_VBR_NOT_SUPPORTED	This is macro MPEG_VBR_NOT_SUPPORTED.
	MPEG_VBR_SUPPORTED	This is macro MPEG_VBR_SUPPORTED.

Structures

	Name	Description
	_bt_a2dp_mpeg_config_t	This is type bt_a2dp_mpeg_config_t.
	bt_a2dp_mpeg_config_t	This is type bt_a2dp_mpeg_config_t.

Description

Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.

a2dp_codec_sbc.h

Functions

	Name	Description
	bt_a2dp_sbc_codec_handler	This is function bt_a2dp_sbc_codec_handler.

Macros

	Name	Description
	__A2DP_CODEC_SBC_H	This is macro __A2DP_CODEC_SBC_H.
	SBC_ALLOCATION_METHOD_ALL	This is macro SBC_ALLOCATION_METHOD_ALL.
	SBC_ALLOCATION_METHOD_LOUDNESS	This is macro SBC_ALLOCATION_METHOD_LOUDNESS.
	SBC_ALLOCATION_METHOD_SNR	This is macro SBC_ALLOCATION_METHOD_SNR.
	SBC_BLOCK_LENGTH_12	This is macro SBC_BLOCK_LENGTH_12.
	SBC_BLOCK_LENGTH_16	This is macro SBC_BLOCK_LENGTH_16.
	SBC_BLOCK_LENGTH_4	This is macro SBC_BLOCK_LENGTH_4.
	SBC_BLOCK_LENGTH_8	This is macro SBC_BLOCK_LENGTH_8.
	SBC_BLOCK_LENGTH_ALL	This is macro SBC_BLOCK_LENGTH_ALL.
	SBC_CHANNEL_MODE_ALL	This is macro SBC_CHANNEL_MODE_ALL.
	SBC_CHANNEL_MODE_DUAL_CHANNEL	This is macro SBC_CHANNEL_MODE_DUAL_CHANNEL.
	SBC_CHANNEL_MODE_JOINT_STEREO	This is macro SBC_CHANNEL_MODE_JOINT_STEREO.
	SBC_CHANNEL_MODE_MONO	This is macro SBC_CHANNEL_MODE_MONO.
	SBC_CHANNEL_MODE_STEREO	This is macro SBC_CHANNEL_MODE_STEREO.
	SBC_SAMPLING_FREQUENCY_16000	This is macro SBC_SAMPLING_FREQUENCY_16000.
	SBC_SAMPLING_FREQUENCY_32000	This is macro SBC_SAMPLING_FREQUENCY_32000.
	SBC_SAMPLING_FREQUENCY_44100	This is macro SBC_SAMPLING_FREQUENCY_44100.
	SBC_SAMPLING_FREQUENCY_48000	This is macro SBC_SAMPLING_FREQUENCY_48000.
	SBC_SAMPLING_FREQUENCY_ALL	This is macro SBC_SAMPLING_FREQUENCY_ALL.
	SBC_SUBBANDS_4	This is macro SBC_SUBBANDS_4.
	SBC_SUBBANDS_8	This is macro SBC_SUBBANDS_8.
	SBC_SUBBANDS_ALL	This is macro SBC_SUBBANDS_ALL.

Structures

	Name	Description
	_bt_a2dp_sbc_config_t	This is type bt_a2dp_sbc_config_t.
	_bt_a2dp_sbc_packet_info_t	This is type bt_a2dp_sbc_packet_info_t.
	bt_a2dp_sbc_config_t	This is type bt_a2dp_sbc_config_t.
	bt_a2dp_sbc_packet_info_t	This is type bt_a2dp_sbc_packet_info_t.

Description

Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.

a2dp_private.h

Functions

	Name	Description
	_bt_a2dp_avdtp_mgr_callback	This is function _bt_a2dp_avdtp_mgr_callback.

Macros

	Name	Description
	__A2DP_PRIVATE_H	This is macro __A2DP_PRIVATE_H.

Description

Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.

avctp.h

Functions

	Name	Description
	bt_avctp_cancel_command	brief Cancel a command message. ingroup avctp details If a message has not yet been sent to the remote device, it can be canceled (i.e. removed from send queue) by calling this function. param channel AVCTP channel. param tran_id Transaction Id. This value is obtained by calling bt_avctp_send_command. return li c TRUE if the function succeeds. li c FALSE otherwise. No events will be generated.
	bt_avctp_cancel_listen	brief Cancel listening for incoming connections. ingroup avctp details This function stops listening for incoming connections on the specified channel. param channel AVCTP channel. return li c TRUE if the function succeeds. li c FALSE otherwise.
	bt_avctp_cancel_response	brief Cancel a response message. ingroup avctp details If a message has not yet been sent to the remote device, it can be canceled (i.e. removed from send queue) by calling this function. param channel AVCTP channel. param tran_id Transaction Id. This value is obtained by calling bt_avctp_send_command. return li c TRUE if the function succeeds. li c FALSE otherwise. No events will be generated.
	bt_avctp_connect	brief Connect to a remote device. ingroup avctp details This function establishes a connection to a remote device specified by the p remote_address. If connection cannot be initiated for some reason, for example, there is not enough resources, it returns FALSE and no events are generated. Otherwise the result of an attempt to connect to the remote device is reported via the AVCTP callback. The events generated will either be AVCTP_EVT_CHANNEL_CONNECTED or AVCTP_EVT_CONNECTION_FAILED. param channel AVCTP channel. param remote_address The address of a remote device. param acl_config ACL link configuration. This can be a combination of the following values:... more
	bt_avctp_create_channel	brief Allocate AVCTP channel ingroup avctp details This function allocates a new incoming AVCTP channel. The channel is intended to be used to accept a connection from a remote device. There can be only one channel for each combination of c profile_id and c psm. param mgr AVCTP manager. param profile_id Profile Id param psm The PSM on which the underlying L2CAP channel will listen and accept incoming connections. param l2cap_mode Underlying L2CAP channel mode. This currently can only be CMODE_BASIC. return li A pointer to the new AVCTP channel if the function succeeds. li c NULL otherwise.

	bt_avctp_create_outgoing_channel	brief Allocate AVCTP channel ingroup avctp details This function allocates a new outgoing AVCTP channel. The channel is intended to be used to create a connection to a remote device. There can be multiple channels with the same c profile_id and c psm. param mgr AVCTP manager. param profile_id Profile Id param psm The PSM on which the underlying L2CAP channel will listen and accept incoming connections. param l2cap_mode Underlying L2CAP channel mode. This currently can only be CMODE_BASIC. return li A pointer to the new AVCTP channel if the function succeeds. li c NULL otherwise.
	bt_avctp_destroy_channel	brief Destroy AVCTP channel. ingroup avctp details This function frees memory used by the channel. The channel has to exist and be in the "idle" state for this function to succeed. I.e. the channel has to be disconnected before this function can be called. param channel AVCTP channel. return li c TRUE if the function succeeds. li c FALSE otherwise.
	bt_avctp_disconnect	brief Disconnect from a remote device. ingroup avctp details This function closes a connection to a remote device. param channel AVCTP channel. return li c TRUE if disconnection has been started. li c FALSE otherwise. No events will be generated.
	bt_avctp_get_channel_remote_address	brief Get channel's remote BT address. ingroup avctp details This function returns the address of the remote device associated with the channel. param channel AVCTP channel. return li The address of the remote device if channel is connected. li NULL otherwise.
	bt_avctp_get_channel_state	brief Get channel state ingroup avctp details This function return current state of the specified channel param channel AVCTP channel. return li AVCTP_CHANNEL_STATE_FREE li AVCTP_CHANNEL_STATE_IDLE li AVCTP_CHANNEL_STATE_CONNECTING li AVCTP_CHANNEL_STATE_CONNECTED li AVCTP_CHANNEL_STATE_DISCONNECTING
	bt_avctp_get_hci_connection	brief Get HCI connection for a channel ingroup avctp details This function returns a pointer to a structure that describes an HCI connection a channel is open on. The return value can be used to call various function from the HCI layer. For example, if an app wants to force disconnection from a remote device it can call ::bt_hci_disconnect. param channel AVCTP channel. return li c Pointer to a structure that describes an HCI connection if the function succeeds. li c NULL otherwise. The function fails only if a channel specified by the p channel parameter li does... more
	bt_avctp_get_mgr	brief Return a pointer to an instance of the AVCTP manager. ingroup avctp details This function returns a pointer to an instance of the AVCTP manager. There is only one instance of the manager allocated by the stack.
	bt_avctp_init	brief Initialize the AVCTP layer. ingroup avctp details This function initializes the AVCTP layer of the stack. It must be called prior to any other AVCTP function can be called.
	bt_avctp_listen	brief Listen for incoming connections. ingroup avctp details This function enables incoming connections on the specified AVCTP channel. param channel AVCTP channel. return li c TRUE if the function succeeds. li c FALSE otherwise.
	bt_avctp_send_command	brief Send a command message to a remote device. ingroup avctp details This function sends a command message to a remote device. param channel AVCTP channel. param data Message body. param data_len Message body length. param tran_id Pointer to a bt_byte where AVRCP will write transaction id assigned to the message. return li c TRUE if the function succeeds. li c FALSE otherwise. No events will be generated.
	bt_avctp_send_response	brief Send a response message to a remote device. ingroup avctp details This function sends a response message to a remote device. param channel AVCTP channel. param tran_id Transaction Id. This value is obtained by calling bt_avctp_send_command. param data Message body. param data_len Message body length. return li c TRUE if the function succeeds. li c FALSE otherwise. No events will be generated.

	bt_avctp_set_callback	brief Register a AVCTP application callback. ingroup avctp details In order to be notified of various events a consumer of the AVCTP layer has to register a callback function. The stack will call this function whenever a new event has been generated passing the code of the event as the second parameter. The event can be one of the following values: @arg AVCTP_EVT_CHANNEL_CONNECTED Channel connected. @arg AVCTP_EVT_CHANNEL_DISCONNECTED Channel disconnected. @arg AVCTP_EVT_CONNECTION_FAILED Channel connection failed (generated only if connection has been initiated by the local device). @arg AVCTP_EVT_COMMAND_RECEIVED Command received. @arg AVCTP_EVT_RESPONE_RECEIVED Response received. @arg AVCTP_EVT_COMMAND_SENT Command sent. @arg AVCTP_EVT_RESPONSE_SENT Response... more
	bt_avctp_start	This is function bt_avctp_start.

Macros

	Name	Description
	__AVCTP_H	This is macro __AVCTP_H.
	AVCTP_CHANNEL_FLAG_LISTENING	This is macro AVCTP_CHANNEL_FLAG_LISTENING.
	AVCTP_CHANNEL_FLAG_SENDING	This is macro AVCTP_CHANNEL_FLAG_SENDING.
	AVCTP_CHANNEL_STATE_CONNECTED	This is macro AVCTP_CHANNEL_STATE_CONNECTED.
	AVCTP_CHANNEL_STATE_CONNECTING	This is macro AVCTP_CHANNEL_STATE_CONNECTING.
	AVCTP_CHANNEL_STATE_DISCONNECTING	This is macro AVCTP_CHANNEL_STATE_DISCONNECTING.
	AVCTP_CHANNEL_STATE_FREE	This is macro AVCTP_CHANNEL_STATE_FREE.
	AVCTP_CHANNEL_STATE_IDLE	This is macro AVCTP_CHANNEL_STATE_IDLE.
	AVCTP_EVT_CHANNEL_CONNECTED	< This event is generated when a channel between two AVCTP entities has been established.
	AVCTP_EVT_CHANNEL_DISCONNECTED	< This event is generated when a channel between two AVCTP entities has been terminated.
	AVCTP_EVT_COMMAND_CANCELLED	< This event is generated when a command has been canceled.
	AVCTP_EVT_COMMAND_RECEIVED	< This event is generated when a local device received a command.
	AVCTP_EVT_COMMAND_SENT	< This event is generated when a local device finished sending a command.
	AVCTP_EVT_CONNECTION_FAILED	< This event is generated when a local device failed to create a channel between two AVCTP entities.
	AVCTP_EVT_NOTHING	addtogroup avrcp @{ @name Events details The following is a list of events AVRCP layer generates and can report to the upper layer when it completes executing an operation initiated by either local or remote device.
	AVCTP_EVT_RESPONE_RECEIVED	< This event is generated when a local device received a response.
	AVCTP_EVT_RESPONSE_CANCELLED	< This event is generated when a response has been canceled.
	AVCTP_EVT_RESPONSE_SENT	< This event is generated when a local device finished sending a response.
	AVCTP_MANAGER_STATE_IDLE	This is macro AVCTP_MANAGER_STATE_IDLE.
	AVCTP_TRANSPORT_FLAG_RX_MESSAGE_STARTED	This is macro AVCTP_TRANSPORT_FLAG_RX_MESSAGE_STARTED.
	AVCTP_TRANSPORT_FLAG_SENDING	This is macro AVCTP_TRANSPORT_FLAG_SENDING.
	AVCTP_TRANSPORT_STATE_CONNECTED	This is macro AVCTP_TRANSPORT_STATE_CONNECTED.
	AVCTP_TRANSPORT_STATE_CONNECTING	This is macro AVCTP_TRANSPORT_STATE_CONNECTING.
	AVCTP_TRANSPORT_STATE_DISCONNECTING	This is macro AVCTP_TRANSPORT_STATE_DISCONNECTING.
	AVCTP_TRANSPORT_STATE_FREE	This is macro AVCTP_TRANSPORT_STATE_FREE.
	AVCTP_TRANSPORT_STATE_IDLE	This is macro AVCTP_TRANSPORT_STATE_IDLE.

Structures

	Name	Description
	_bt_avctp_channel_t	brief AVCTP channel description ingroup avctp details This structure is used to hold information about an AVCTP channel.
	_bt_avctp_evt_channel_connected_t	brief Parameter to AVCTP_EVT_CHANNEL_CONNECTED event ingroup avctp details A pointer to this structure is passed to the AVCTP application callback as a valid member of the bt_avctp_event_t union - bt_avctp_event_t::channel_connected - when a channel between two devices has been established.

	_bt_avctp_evt_channel_disconnected_t	brief Parameter to AVCTP_EVT_CHANNEL_DISCONNECTED event ingroup avctp details A pointer to this structure is passed to the AVCTP application callback as a valid member of the bt_avctp_event_t union - bt_avctp_event_t::channel_disconnected - when a channel between two devices has been terminated.
	_bt_avctp_evt_command_cancelled_t	brief Parameter to AVCTP_EVT_COMMAND_CANCELLED event ingroup avctp details A pointer to this structure is passed to the AVCTP application callback as a valid member of the bt_avctp_event_t union - bt_avctp_event_t::command_cancelled - when sending a command message has been canceled.
	_bt_avctp_evt_command_received_t	brief Parameter to AVCTP_EVT_COMMAND_RECEIVED event ingroup avctp details A pointer to this structure is passed to the AVCTP application callback as a valid member of the bt_avctp_event_t union - bt_avctp_event_t::command_received - when a local device received a command message.
	_bt_avctp_evt_command_sent_t	brief Parameter to AVCTP_EVT_COMMAND_SENT event ingroup avctp details A pointer to this structure is passed to the AVCTP application callback as a valid member of the bt_avctp_event_t union - bt_avctp_event_t::command_sent - when a local device finished sending a command message.
	_bt_avctp_evt_connection_failed_t	brief Parameter to AVCTP_EVT_CONNECTION_FAILED event ingroup avctp details A pointer to this structure is passed to the AVCTP application callback as a valid member of the bt_avctp_event_t union - bt_avctp_event_t::connection_failed - when a channel between two devices could not be established.
	_bt_avctp_evt_response_cancelled_t	brief Parameter to AVCTP_EVT_RESPONSE_CANCELLED event ingroup avctp details A pointer to this structure is passed to the AVCTP application callback as a valid member of the bt_avctp_event_t union - bt_avctp_event_t::response_cancelled - when sending a response message has been canceled.
	_bt_avctp_evt_response_received_t	brief Parameter to AVCTP_EVT_RESPONSE_RECEIVED event ingroup avctp details A pointer to this structure is passed to the AVCTP application callback as a valid member of the bt_avctp_event_t union - bt_avctp_event_t::response_received - when a local device received a response message.
	_bt_avctp_evt_response_sent_t	brief Parameter to AVCTP_EVT_RESPONSE_SENT event ingroup avctp details A pointer to this structure is passed to the AVCTP application callback as a valid member of the bt_avctp_event_t union - bt_avctp_event_t::response_sent - when a local device finished sending a response message.
	_bt_avctp_message_t	brief AVCTP message description ingroup avctp details This structure is used to hold information about an AVCTP message.
	_bt_avctp_mgr_t	brief AVCTP manager. ingroup avctp details A structure that glues all pieces together. There is only one instance of this structure allocated by dotstack. A pointer to the instance can be get with c bt_avctp_get_mgr().
	_bt_avctp_transport_t	brief AVCTP transport description ingroup avctp details This structure is used to hold information about an AVCTP transport.
	bt_avctp_evt_channel_connected_t	brief Parameter to AVCTP_EVT_CHANNEL_CONNECTED event ingroup avctp details A pointer to this structure is passed to the AVCTP application callback as a valid member of the bt_avctp_event_t union - bt_avctp_event_t::channel_connected - when a channel between two devices has been established.
	bt_avctp_evt_channel_disconnected_t	brief Parameter to AVCTP_EVT_CHANNEL_DISCONNECTED event ingroup avctp details A pointer to this structure is passed to the AVCTP application callback as a valid member of the bt_avctp_event_t union - bt_avctp_event_t::channel_disconnected - when a channel between two devices has been terminated.
	bt_avctp_evt_command_cancelled_t	brief Parameter to AVCTP_EVT_COMMAND_CANCELLED event ingroup avctp details A pointer to this structure is passed to the AVCTP application callback as a valid member of the bt_avctp_event_t union - bt_avctp_event_t::command_cancelled - when sending a command message has been canceled.
	bt_avctp_evt_command_received_t	brief Parameter to AVCTP_EVT_COMMAND_RECEIVED event ingroup avctp details A pointer to this structure is passed to the AVCTP application callback as a valid member of the bt_avctp_event_t union - bt_avctp_event_t::command_received - when a local device received a command message.
	bt_avctp_evt_command_sent_t	brief Parameter to AVCTP_EVT_COMMAND_SENT event ingroup avctp details A pointer to this structure is passed to the AVCTP application callback as a valid member of the bt_avctp_event_t union - bt_avctp_event_t::command_sent - when a local device finished sending a command message.
	bt_avctp_evt_connection_failed_t	brief Parameter to AVCTP_EVT_CONNECTION_FAILED event ingroup avctp details A pointer to this structure is passed to the AVCTP application callback as a valid member of the bt_avctp_event_t union - bt_avctp_event_t::connection_failed - when a channel between two devices could not be established.

	bt_avctp_evt_response_cancelled_t	brief Parameter to AVCTP_EVT_RESPONSE_CANCELLED event ingroup avctp details A pointer to this structure is passed to the AVCTP application callback as a valid member of the bt_avctp_event_t union - bt_avctp_event_t::response_cancelled - when sending a response message has been canceled.
	bt_avctp_evt_response_received_t	brief Parameter to AVCTP_EVT_RESPONSE_RECEIVED event ingroup avctp details A pointer to this structure is passed to the AVCTP application callback as a valid member of the bt_avctp_event_t union - bt_avctp_event_t::response_received - when a local device received a response message.
	bt_avctp_evt_response_sent_t	brief Parameter to AVCTP_EVT_RESPONSE_SENT event ingroup avctp details A pointer to this structure is passed to the AVCTP application callback as a valid member of the bt_avctp_event_t union - bt_avctp_event_t::response_sent - when a local device finished sending a response message.

Types

	Name	Description
	bt_avctp_channel_t	This is type bt_avctp_channel_t.
	bt_avctp_message_t	This is type bt_avctp_message_t.
	bt_avctp_mgr_callback_fp	brief AVCTP application callback. ingroup avctp details In order to be notified of various events a consumer of the AVCTP layer has to register a callback function (done with bt_avctp_set_callback()). The stack will call that function whenever a new event has been generated. param mgr AVCTP manager. param evt AVCTP event. The event can be one of the following values: @arg AVCTP_EVT_CHANNEL_CONNECTED Channel connected. @arg AVCTP_EVT_CHANNEL_DISCONNECTED Channel disconnected. @arg AVCTP_EVT_CONNECTION_FAILED Channel connection failed (generated only if connection has been initiated by the local device). @arg AVCTP_EVT_COMMAND_RECEIVED Command received. @arg AVCTP_EVT_RESPONE_RECEIVED Response received. @arg AVCTP_EVT_COMMAND_SENT Command sent. @arg AVCTP_EVT_RESPONSE_SENT Response... more
	bt_avctp_mgr_t	This is type bt_avctp_mgr_t.
	bt_avctp_transport_t	This is type bt_avctp_transport_t.

Unions

	Name	Description
	_bt_avctp_event_u	brief Parameter to an application callback. ingroup avctp details This union is used to pass event specific data to the AVCTP consumer. Which member of the union points to a valid structure depends on the event reported to the consumer. In general, each event has a corresponding member in the union.
	bt_avctp_event_t	brief Parameter to an application callback. ingroup avctp details This union is used to pass event specific data to the AVCTP consumer. Which member of the union points to a valid structure depends on the event reported to the consumer. In general, each event has a corresponding member in the union.

Description

Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.

avctp_config.h

Macros

	Name	Description
	__AVCTP_CONFIG_H	This is macro __AVCTP_CONFIG_H.

	AVCTP_ALLOCATE_BUFFERS_RAM_SIZE_VAR	• defgroup avctp_config Configuration
		• ingroup avctp
		*
		• This module describes parameters used to configure AVCTP layer.
		*
		• dotstack is customized using a configuration file. The configuration file tailors the dotstack to the application being built. It has to have the structure shown below.
		* code #include "cdbt/bt/bt_std.h"
		// HCI, L2CAP and SDP must always be present
		// HCI configuration parameters #define HCI_MAX_CMD_BUFFERS ... #define HCI_MAX_DATA_BUFFERS ... #define HCI_MAX_HCI_CONNECTIONS ... #define HCI_RX_BUFFER_LEN ... #define HCI_TX_BUFFER_LEN ... #define HCI_L2CAP_BUFFER_LEN ... #define HCI_MAX_CMD_PARAM_LEN ...
		// L2CAP configuration parameters #define L2CAP_MAX_CMD_BUFFERS ... #define L2CAP_MAX_FRAME_BUFFERS ... #define L2CAP_MAX_PSMS ... #define L2CAP_MAX_CHANNELS ...
		// SDP... more

	AVCTP_ALLOCATE_BUFFERS_VARS	brief Maximum number of message buffers ingroup avctp_config details This parameter defines the maximum number of buffer available for sending message.
	AVCTP_MAX_CHANNELS	brief Maximum number of AVCTP channels ingroup avctp_config details This parameter defines the maximum number of channels a local device can have with remote devices.
	AVCTP_MAX_TRANSPORT_CHANNELS	brief Maximum number of AVCTP transports ingroup avctp_config details This parameter defines the maximum number of transports a local device can have with remote devices. This value should not exceed AVCTP_MAX_CHANNELS.

Description

Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.

avctp_packet.h

Functions

	Name	Description
	_bt_avctp_packet_assembler	This is function _bt_avctp_packet_assembler.

Macros

	Name	Description
	__AVCTP_PACKET_H	This is macro __AVCTP_PACKET_H.

Structures

	Name	Description
	_bt_avctp_packet_t	This is type bt_avctp_packet_t.
	bt_avctp_packet_t	This is type bt_avctp_packet_t.

Description

Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.

avctp_private.h

Functions

	Name	Description
	_bt_avctp_allocate_channel	This is function _bt_avctp_allocate_channel.
	_bt_avctp_allocate_message	This is function _bt_avctp_allocate_message.

	_bt_avctp_allocate_transport	This is function _bt_avctp_allocate_transport.
	_bt_avctp_find_channel	This is function _bt_avctp_find_channel.
	_bt_avctp_find_transport	This is function _bt_avctp_find_transport.
	_bt_avctp_free_channel	This is function _bt_avctp_free_channel.
	_bt_avctp_free_message	This is function _bt_avctp_free_message.
	_bt_avctp_free_transport	This is function _bt_avctp_free_transport.
	_bt_avctp_init_message_buffers	This is function _bt_avctp_init_message_buffers.
	_bt_avctp_init_signal	This is function _bt_avctp_init_signal.
	_bt_avctp_l2cap_read_data_callback	This is function _bt_avctp_l2cap_read_data_callback.
	_bt_avctp_send_ipid	This is function _bt_avctp_send_ipid.
	_bt_avctp_set_signal	This is function _bt_avctp_set_signal.
	_bt_avrcp_init_cmd_buffers	This is function _bt_avrcp_init_cmd_buffers.

Macros

	Name	Description
	__AVCTP_PRIVATE_H	This is macro __AVCTP_PRIVATE_H.

Description

Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.

avdtp.h

Functions

	Name	Description
	bt_avdtp_abort_stream	brief Suspend a stream. ingroup avdtp details This function tries to suspend a stream by sending a request to the remote party. The stream can be in any state state except AVDTP_STREAM_STATE_IDLE. As a result of this operation the AVDTP_EVT_ABORT_STREAM_COMPLETED event will be generated. This operation cannot be rejected. The p evt_param.abort_stream_requested.err_code is always == AVDTP_ERROR_SUCCESS. param mgr AVDTP manager. param strm_handle Stream handle. return li c TRUE if the function succeeds, i.e. the actual request has been sent to the remote party. li c FALSE otherwise. No events will be generated.
	bt_avdtp_add_media_rx_buffer	brief Add a media packet buffer to a receive queue ingroup avdtp details The consumer of AVDTP is responsible for allocating and supplying AVDTP with buffers used to store received packets. AVDTP itself only has a queue for storing pointers to buffers supplied by the consumer. When a packet comes in AVDTP finds the first buffer large enough to hold the received packet, copies the packet to the buffer and generates a AVDTP_EVT_MEDIA_PACKET_RECEIVED event. The consumer then has to process the data in the buffer and return it back to the queue. If there is no buffers in the queue... more
	bt_avdtp_add_media_tx_buffer	brief Add a media packet buffer to a send queue ingroup avdtp details When the consumer of AVDTP wants to send a packet to a remote device it calls this function. The function adds the packet to a queue and tells AVDTP that it has something to send. The packet will be send as soon as the stream goes to AVDTP_STREAM_STATE_STREAMING state. When the packet has been successfully sent a AVDTP_EVT_MEDIA_PACKET_SENT is generated. Otherwise a AVDTP_EVT_MEDIA_PACKET_SEND_FAILED is generated. Regardless of the event generated the consumer can re-use the buffer as AVDTP has removed it from the queue and gave up... more
	bt_avdtp_cancel_listen	brief Cancel listening for incoming connections. ingroup avdtp details This function removes a SEP from a list of SEPS which a stream can use for incoming requests. param mgr AVDTP manager. param strm_handle Stream handle. param sep_id Local SEP ID. return li c TRUE if the function succeeds. li c FALSE otherwise.

	bt_avdtp_clear_media_tx_queue	brief Clear send queue ingroup avdtp details When the consumer of AVDTP wants to send a packet to a remote device it calls bt_avdtp_add_media_tx_buffer function. The function adds the packet to a queue and tells AVDTP that it has something to send. The packet will be send as soon as the stream goes to AVDTP_STREAM_STATE_STREAMING state. The consumer can remove all packets from the queue before they have been sent to a remote device by calling ::bt_avdtp_clear_media_tx_queue. param mgr AVDTP manager. param strm_handle Stream handle. return li c TRUE if the function succeeds. li c FALSE otherwise. The function fails... more
	bt_avdtp_close_stream	brief Close a stream. ingroup avdtp details This function tries to close a stream by sending a request to the remote party. The stream has to be in AVDTP_STREAM_STATE_OPEN or AVDTP_STREAM_STATE_STREAMING state. As a result of this operation the AVDTP_EVT_CLOSE_STREAM_COMPLETED event will be generated. If the stream has been closed the p evt_param.bt_avdtp_evt_close_stream_completed_t.err_code == AVDTP_ERROR_SUCCESS. Otherwise, if the remote device for any reason cannot or does not wish to close the stream, the p evt_param.bt_avdtp_evt_close_stream_completed_t.err_code == the error code sent by the remote. param mgr AVDTP manager. param strm_handle Stream handle. return li c TRUE if the function succeeds,... more
	bt_avdtp_create_stream	brief Create a stream. ingroup avdtp details This function allocates memory for storing stream's data and assigns a stream handle. The stream handle is used to manipulate the stream - open, close, configure, suspend, abort. param mgr AVDTP manager. return li c Stream handle if the function succeeds. li c 0 otherwise.
	bt_avdtp_destroy_stream	brief Destroy a stream. ingroup avdtp details This function frees memory used by the stream. The stream has to exist and be in the "idle" state for this function to succeed. I.e. the stream has to be closed or aborted before this function can be called. param mgr AVDTP manager. param strm_handle Stream handle. return li c TRUE if the function succeeds. li c FALSE otherwise.
	bt_avdtp_disconnect	brief Disconnect from a remote device. ingroup avdtp details This function closes a control and transport channels on all streams associated with the remote device specified by the p remote_addr. As a result of this operation the following events will be generated: @arg AVDTP_EVT_MEDIA_PACKET_RECEIVED: if a stream's receive queue is not empty this event is generated for each buffer with bt_media_packet_t::data_len set to 0 @arg AVDTP_EVT_MEDIA_PACKET_SENT: if a stream's send queue is not empty this event is generated for each buffer with bt_media_packet_t::data_len set to 0 @arg AVDTP_EVT_STREAM_CLOSED: this event is generate if a stream is in... more
	bt_avdtp_discover	brief Discover SEPs on a remote device. ingroup avdtp details This function asks the remote device to send a list of all available SEPs. As a result of this operation the following events will be generated: @arg AVDTP_EVT_SEP_INFO_RECEIVED: this event is generated for every SEP received from the remote device. the p evt_param.sep_info_received contains SEP information. @arg AVDTP_EVT_DISCOVER_COMPLETED: this event is generated after last AVDTP_EVT_SEP_INFO_RECEIVED if the remote accepted the request and the p evt_param.discover_completed.err_code == AVDTP_ERROR_SUCCESS. if the remote rejected the request the p evt_param.discover_completed.err_code == the error code sent by the remote. param mgr AVDTP... more
	bt_avdtp_find_codec	brief Find a codec ingroup avdtp details AVDTP in theory can support any type of codec. Each codec uses its own format for exchanging capabilities and configuration information. In order to make out implementation do not care about these formats we use a simple way of telling AVDTP how to parse and serialize codec's configuration. The consumer of AVDTP (e.g. A2DP) for each codec it wishes to support has to register a callback function (one per codec type). That callback has to perform two function. The first one is to read the configuration received from the remote device and store... more

	bt_avdtp_get_all_capabilities	brief Get remote SEP capabilities. ingroup avdtp details This function asks the remote device to send capabilities of a SEP specified by the p seid_acp. As a result of this operation the following events will be generated: @arg AVDTP_EVT_SEP_CAPABILITIES_RECEIVED: this event is generated if the remote device accepted the request. the p evt_param.sep_capabilities_received contains SEP capabilities. @arg AVDTP_EVT_GET_SEP_CAPABILITIES_COMPLETED: this event is generated right after AVDTP_EVT_SEP_CAPABILITIES_RECEIVED if the remote accepted the request the p evt_param.get_sep_capabilities_completed.err_code == AVDTP_ERROR_SUCCESS. if the remote rejected the request the p evt_param.get_sep_capabilities_completed.err_code == the error code sent by the remote. param mgr AVDTP manager.... more
	bt_avdtp_get_capabilities	brief Get remote SEP capabilities. ingroup avdtp details This function asks the remote device to send capabilities of a SEP specified by the p seid_acp. As a result of this operation the following events will be generated: @arg AVDTP_EVT_SEP_CAPABILITIES_RECEIVED: this event is generated if the remote device accepted the request. the p evt_param.sep_capabilities_received contains SEP capabilities. @arg AVDTP_EVT_GET_SEP_CAPABILITIES_COMPLETED: this event is generated right after AVDTP_EVT_SEP_CAPABILITIES_RECEIVED if the remote accepted the request the p evt_param.get_sep_capabilities_completed.err_code == AVDTP_ERROR_SUCCESS. if the remote rejected the request the p evt_param.get_sep_capabilities_completed.err_code == the error code sent by the remote. param mgr AVDTP manager.... more
	bt_avdtp_get_configuration	brief Get stream configuration. ingroup avdtp details This function requests stream configuration from a remote device. As a result of this operation the following events will be generated: @arg AVDTP_EVT_STREAM_CONFIGURATION_RECEIVED: this event is generated if the remote accepted the request. the p ebt_para.sep_capabilities_received.caps will contain current stream configuration. @arg AVDTP_EVT_GET_STREAM_CONFIGURATION_COMPLETED: If the remote accepted the request the p evt_param.get_stream_configuration_completed.err_code == AVDTP_ERROR_SUCCESS. if the remote rejected the request the p evt_param.get_stream_configuration_completed.err_code == the error code sent by the remote. param mgr AVDTP manager. param strm_handle Stream handle. return li c TRUE if the function succeeds, i.e. the actual... more
	bt_avdtp_get_hci_connection	brief Get HCI connection for a stream ingroup avdtp details This function returns a pointer to a structure that describes an HCI connection a stream is open on. The return value can be used to call various function from the HCI layer. For example, if an app wants to force disconnection from a remote device it can call bt_hci_disconnect. param mgr AVDTP manager. param strm_handle Stream handle. return li c Pointer to a structure that describes an HCI connection if the function succeeds. li c NULL otherwise. The function fails only if a stream specified by the p strm_handle... more
	bt_avdtp_get_l2cap_channel	This is function bt_avdtp_get_l2cap_channel.
	bt_avdtp_get_mgr	brief Return a pointer to an instance of the AVDTP manager. ingroup avdtp details This function returns a pointer to an instance of the AVDTP manager. There is only one instance of the manager allocated by the stack. The pointer is passed as the first parameter to all AVDTP functions.
	bt_avdtp_get_sep	brief Get a SEP info by its ID. ingroup avdtp details This function returns a pointer to bt_avdtp_sep_t structure that describes a SEP previously registered with bt_avdtp_register_sep. param mgr AVDTP manager. param sep_id The ID of a SEP. return li c Pointer to bt_avdtp_sep_t if the SEP is in the list of registered SEPs. li c NULL otherwise.
	bt_avdtp_get_stream_codec_config	brief Get the configuration of the codec currently used with the stream. ingroup avdtp details This function returns a pointer to a structure that contains configuration of the codec currently used with the stream. The structure returned depends on the codec. The dotstack defines structures only for SBC, MPEG-1,2 and MPEG-2,4 AAC codecs: @arg SBC: bt_a2dp_sbc_config_t (defined in a2dp_sbc_codec.h) @arg MPEG-1,2: bt_a2dp_mpeg_config_t (defined in a2dp_mpeg_codec.h) @arg MPEG-2,4 AAC: bt_a2dp_aac_config_t (defined in a2dp_aac_codec.h) param mgr AVDTP manager. param strm_handle Stream handle. return li The codec's configuration if strm_handle specifies a valid stream and the stream is in one of the following... more

	bt_avdtp_get_stream_codec_type	brief Get the type of the codec currently used with the stream. ingroup avdtp details This function returns the type of the codec currently used with the stream. param mgr AVDTP manager. param strm_handle Stream handle. return @arg The type of the codec if strm_handle specifies a valid stream and the stream is in one of the following states: AVDTP_STREAM_STATE_CONFIGURED AVDTP_STREAM_STATE_OPEN AVDTP_STREAM_STATE_STREAMING @arg The result will be one of the following values: AVDTP_CODEC_TYPE_SBC: SBC AVDTP_CODEC_TYPE_MPEG1_2_AUDIO: MPEG-1,2 (used in MP3 files) AVDTP_CODEC_TYPE_MPEG2_4_AAC: MPEG-2,4 AAC (used in Apple products) AVDTP_CODEC_TYPE_ATRAC: ATRAC (used in Sony products) AVDTP_CODEC_TYPE_NON_A2DP: Non-A2DP... more
	bt_avdtp_get_stream_config	brief Get stream's configuration. ingroup avdtp details This function returns a pointer to a structure holding the current configuration of stream. param mgr AVDTP manager. param strm_handle Stream handle. return li The stream's configuration if strm_handle specifies a valid stream and the stream is in one of the following state: AVDTP_STREAM_STATE_CONFIGURED AVDTP_STREAM_STATE_OPEN AVDTP_STREAM_STATE_STREAMING li NULL otherwise.
	bt_avdtp_get_stream_direction	This is function bt_avdtp_get_stream_direction.
	bt_avdtp_get_stream_local_sep_id	brief Get stream's local SEP ID. ingroup avdtp details This function returns the ID of the local SEP associated with the stream. param mgr AVDTP manager. param strm_handle Stream handle. return li The ID of the local SEP if strm_handle specifies a valid stream. li 0 otherwise.
	bt_avdtp_get_stream_remote_address	brief Get stream's remote BT address. ingroup avdtp details This function returns the address of the remote device associated with the stream. param mgr AVDTP manager. param strm_handle Stream handle. return li The address of the remote device if strm_handle specifies a valid stream. li NULL otherwise.
	bt_avdtp_get_stream_remote_sep_id	brief Get stream's remote SEP ID. ingroup avdtp details This function returns the ID of the remote SEP associated with the stream. param mgr AVDTP manager. param strm_handle Stream handle. return li The ID of the remote SEP if strm_handle specifies a valid stream. li 0 otherwise.
	bt_avdtp_get_stream_state	brief Get local stream state. ingroup avdtp details This function returns local state of a stream specified by the p strm_handle. No request is sent to the remote party. param mgr AVDTP manager. param strm_handle Stream handle. return The state of the stream. The result will be one of the following values: @arg AVDTP_STREAM_STATE_IDLE: The stream is idle. This can mean two things. The stream specified by p strm_handle does not exist or the stream is closed. @arg AVDTP_STREAM_OPENING_TRANSPORT_CHANNELS: The stream is opening transport channels. @arg AVDTP_STREAM_CLOSING_TRANSPORT_CHANNELS: The stream is closing transport channels. @arg AVDTP_STREAM_STATE_CONFIGURED: The... more
	bt_avdtp_init	brief Initialize the AVDTP layer. ingroup avdtp details This function initializes the AVDTP layer of the stack. It must be called prior to any other AVDTP function can be called.
	bt_avdtp_listen	brief Listen for incoming connections. ingroup avdtp details This function tells a stream that it can use a particular SEP to accept incoming requests to open it. The SEP can be associated with multiple streams but used with only one. The stream has to be closed before the SEP can be used with another stream. For outgoing connections this is not needed. Any SEP can be used with any stream given that the SEP is not already in use by another stream. param mgr AVDTP manager. param strm_handle Stream handle. param sep_id Local SEP ID. return li c TRUE if... more

	bt_avdtp_open_stream	brief Open a stream. ingroup avdtp details This function tries to open a stream by sending a request to the remote party. The stream has to be already configured with a bt_avdtp_set_configuration call. As a result of this operation the AVDTP_EVT_OPEN_STREAM_COMPLETED event will be generated. If the stream has been open the p evt_param.open_stream_completed.err_code == AVDTP_ERROR_SUCCESS. Otherwise, if the remote device for any reason cannot or does not wish to open the stream, the p evt_param.open_stream_completed.err_code == the error code sent by the remote. param mgr AVDTP manager. param strm_handle Stream handle. return li c TRUE if the function... more
	bt_avdtp_reconfigure_stream	brief Reconfigure stream. ingroup avdtp details This function tries to change the stream's configuration. For this function to succeed the stream has to be open. As a result of this operation the AVDTP_EVT_STREAM_RECONFIGURE_COMPLETED event will be generated. If reconfiguration was a success the p evt_param.stream_reconfigure_completed.err_code == AVDTP_ERROR_SUCCESS. Otherwise the p evt_param.stream_reconfigure_completed.err_code == the error code sent by the remote. param mgr AVDTP manager. param strm_handle Stream handle. param caps New stream configuration. return li c TRUE if the function succeeds, i.e. the actual request has been sent to the remote party. li c FALSE otherwise. No events will be... more
	bt_avdtp_register_callback	brief Register a AVDTP application callback. ingroup avdtp details In order to be notified of various events a consumer of the AVDTP layer has to register a callback function. The stack will call this function whenever a new event has been generated passing the code of the event as the second parameter. The event can be one of the following values: @arg AVDTP_EVT_CTRL_CHANNEL_CONNECTED: Control channel connected. @arg AVDTP_EVT_CTRL_CHANNEL_DISCONNECTED: Control channel disconnected. @arg AVDTP_EVT_CTRL_CONNECTION_FAILED: Control channel connection failed (generated only if control connection has been initiated by the local device). @arg AVDTP_EVT_DISCOVER_COMPLETED: Local device completed discovering remote... more
	bt_avdtp_register_codec	brief Register a codec ingroup avdtp details AVDTP in theory can support any type of codec. Each codec uses its own format for exchanging capabilities and configuration information. In order to make out implementation do not care about these formats we use a simple way of telling AVDTP how to parse and serialize codec's configuration. The consumer of AVDTP (e.g. A2DP) for each codec it wishes to support has to register a callback function (one per codec type). That callback has to perform two function. The first one is to read the configuration received from the remote device and store... more
	bt_avdtp_register_sep	brief Register a SEP with the local AVDTP manager. ingroup avdtp details This function is used to make a list of SEPs supported by the local ADVTP entity. param mgr AVDTP manager. param type The type of a SEP. The type can be one of the following values: @arg AVDTP_SEP_TYPE_SOURCE: The SEP is a source. @arg AVDTP_SEP_TYPE_SINK: The SEP is a sink. param caps The capabilities of a SEP. return li c ID of a SEP if the function succeeds. li c FALSE otherwise.
	bt_avdtp_remove_media_rx_buffer	brief Remove a media packet buffer from a receive queue ingroup avdtp details The consumer of AVDTP is responsible for allocating and supplying AVDTP with buffers used to store received packets. AVDTP itself only has a queue for storing pointers to buffers supplied by the consumer. When a packet comes in AVDTP finds the first buffer large enough to hold the received packet, copies the packet to the buffer and generates a AVDTP_EVT_MEDIA_PACKET_RECEIVED event. The consumer then has to process the data in the buffer and return it back to the queue. If there is no buffers in the queue... more
	bt_avdtp_remove_media_tx_buffer	brief Remove a media packet buffer from a send queue ingroup avdtp details When the consumer of AVDTP wants to send a packet to a remote device it calls bt_avdtp_add_media_tx_buffer function. The function adds the packet to a queue and tells AVDTP that it has something to send. The packet will be send as soon as the stream goes to AVDTP_STREAM_STATE_STREAMING state. The consumer has a chance to remove a packet from the queue before it has been sent to a remote device by calling bt_avdtp_remove_media_tx_buffer. param mgr AVDTP manager. param strm_handle Stream handle. param buffer Pointer to a structure... more

	bt_avdtp_report_delay	brief Report delay value of a Sink to a Source. ingroup avdtp details This function sends the delay value of a Sink to a Source. This enables synchronous playback of audio and video. Delay reports are always sent from the Sink to the Source. If the Sink's delay report has been accepted by the Source the p evt_param.delay_report_completed.err_code == AVDTP_ERROR_SUCCESS. Otherwise the p evt_param.delay_report_completed.err_code == the error code sent by the Source. param mgr AVDTP manager. param strm_handle Stream handle. param delay The delay value in 1/10 milliseconds. return li c TRUE if the function succeeds, i.e. the actual... more
	bt_avdtp_security_control	brief Exchange content protection control data. ingroup avdtp details This function tries to establish content protection by sending a request to the remote party. The stream can be in any state state except AVDTP_STREAM_STATE_IDLE, AVDTP_STREAM_STATE_CLOSING, AVDTP_STREAM_STATE_ABORTING. As a result of this operation the AVDTP_EVT_STREAM_SECURITY_CONTROL_COMPLETED event will be generated. If the stream's content protection data has been accepted by the remote party the p evt_param.security_control_completed.err_code == AVDTP_ERROR_SUCCESS. Otherwise the p evt_param.security_control_completed.err_code == the error code sent by the remote. note The dotstack does not support content protection. Although the request can be sent it will not affect... more
	bt_avdtp_set_configuration	brief Set stream configuration. ingroup avdtp details This function tries to configure a stream before opening it. As a result of this operation the AVDTP_EVT_SET_STREAM_CONFIGURATION_COMPLETED event will be generated. If configuration was a success the p evt_param.set_stream_configuration_completed.err_code == AVDTP_ERROR_SUCCESS. Otherwise the p evt_param.set_stream_configuration_completed.err_code == the error code sent by the remote and p evt_param.set_stream_configuration_completed.svc_category == the value of the first Service Category to fail. param mgr AVDTP manager. param strm_handle Stream handle. param remote_addr The address of a remote device. param seid_int Local SEP ID. param seid_acp Remote SEP ID. param caps Stream configuration. return li c TRUE if... more
	bt_avdtp_set_media_tx_queue_limit	brief Set limit on the send queue ingroup avdtp details When the consumer of AVDTP wants to send a packet to a remote device it calls bt_avdtp_add_media_tx_buffer function. The function adds the packet to a queue and tells AVDTP that it has something to send. The packet will be send as soon as the stream goes to AVDTP_STREAM_STATE_STREAMING state. By default the send queue can contain unlimited number of packets. The consumer can set a limit on how many packets are held in the queue. In this case when new packet is added to the queue and the length of... more
	bt_avdtp_start	brief Start the AVDTP layer. ingroup avdtp details This function makes the AVDTP layer ready to accept connection requests from remote device. To make an outgoing connection calling this function is not required. param mgr AVDTP manager. return li c TRUE if the function succeeds. li c FALSE otherwise.
	bt_avdtp_start_stream	brief Start a stream. ingroup avdtp details This function tries to start a stream by sending a request to the remote party. The stream has to be in AVDTP_STREAM_STATE_OPEN state. The stream goes to this state as a result of successful configuration or suspension (both can be initiated by either party). As a result of this operation the AVDTP_EVT_START_STREAM_COMPLETED event will be generated. If the stream has been open the p evt_param.start_stream_requested.err_code == AVDTP_ERROR_SUCCESS. Otherwise, if the remote device for any reason cannot or does not wish to start the stream, the p evt_param.start_stream_requested.err_code == the error code sent... more
	bt_avdtp_suspend_stream	brief Suspend a stream. ingroup avdtp details This function tries to suspend a stream by sending a request to the remote party. The stream has to be in AVDTP_STREAM_STATE_STREAMING state. As a result of this operation the AVDTP_EVT_SUSPEND_STREAM_COMPLETED event will be generated. If the stream has been suspended the p evt_param.bt_avdtp_evt_suspend_stream_requested_t.err_code == AVDTP_ERROR_SUCCESS. Otherwise, if the remote device for any reason cannot or does not wish to suspend the stream, the p evt_param.bt_avdtp_evt_suspend_stream_requested_t.err_code == the error code sent by the remote. param mgr AVDTP manager. param strm_handle Stream handle. return li c TRUE if the function succeeds, i.e. the... more

	bt_avdtp_unregister_codec	brief Unregister a codec ingroup avdtp details AVDTP in theory can support any type of codec. Each codec uses its own format for exchanging capabilities and configuration information. In order to make out implementation do not care about these formats we use a simple way of telling AVDTP how to parse and serialize codec's configuration. The consumer of AVDTP (e.g. A2DP) for each codec it wishes to support has to register a callback function (one per codec type). That callback has to perform two function. The first one is to read the configuration received from the remote device and store... more

Macros

	Name	Description
	__AVDTP_H	This is macro __AVDTP_H.
	AVDTP_CODEC_OPCODE_PARSE_CONFIG	< Parse codec configuration.
	AVDTP_CODEC_OPCODE_PARSE_PACKET	This is macro AVDTP_CODEC_OPCODE_PARSE_PACKET.
	AVDTP_CODEC_OPCODE_SERIALIZE_CONFIG	< Serialize codec configuration.
	AVDTP_CODEC_TYPE_ATRAC	< ATRAC (proprietary codec owned by Sony Corporation).
	AVDTP_CODEC_TYPE_MPEG1_2_AUDIO	< MPEG-1,2 (optional).
	AVDTP_CODEC_TYPE_MPEG2_4_AAC	< MPEG-2,4 AAC (optional, used in Apple's products).
	AVDTP_CODEC_TYPE_NON_A2DP	< Vendor specific.
	AVDTP_CODEC_TYPE_SBC	< SBC (mandatory to support in A2DP profile).
	AVDTP_CONTENT_PROTECTION_METHOD_SCMS_T	This is macro AVDTP_CONTENT_PROTECTION_METHOD_SCMS_T.
	AVDTP_ERROR_BAD_ACP_SEID	< The requested command indicates an invalid ACP SEP ID (not addressable)
	AVDTP_ERROR_BAD_CP_FORMAT	< The format of Content Protection Service Capability is not correct.
	AVDTP_ERROR_BAD_HEADER_FORMAT	< The request packet header format is invalid.
	AVDTP_ERROR_BAD_LENGTH	< The request packet length is not match the assumed length.
	AVDTP_ERROR_BAD_MEDIA_TRANSPORT_FORMAT	< The format of Media Transport Capability is not correct.
	AVDTP_ERROR_BAD_MULTIPLEXING_FORMAT	< The format of Multiplexing Service Capability is not correct.
	AVDTP_ERROR_BAD_PAYLOAD_FORMAT	< The requested command has an incorrect payload format.
	AVDTP_ERROR_BAD_RECOVERY_FORMAT	< The format of Recovery Service Capability is not correct.
	AVDTP_ERROR_BAD_RECOVERY_TYPE	< The requested Recovery Type is not defined in AVDTP.
	AVDTP_ERROR_BAD_ROHC_FORMAT	< The format of Header Compression Service Capability is not correct.
	AVDTP_ERROR_BAD_SERV_CATEGORY	< The value of Service Category in the request packet is not defined in AVDTP.
	AVDTP_ERROR_BAD_STATE	< The stream is in state that does not permit executing commands.
	AVDTP_ERROR_FAILED_TO_CONNECT_CONTROL	< An attempt to establish a control channel has failed.
	AVDTP_ERROR_FAILED_TO_CONNECT_TRANSPORT	< An attempt to establish a transport channel has failed.
	AVDTP_ERROR_INVALID_CAPABILITIES	< The reconfigure command is an attempt to reconfigure a transport service capabilities of the SEP. Reconfigure is only permitted for application service capabilities.
	AVDTP_ERROR_NOT_SUPPORTED_COMMAND	< The requested command is not supported by the device.
	AVDTP_ERROR_SEP_IN_USE	< The SEP is in use.
	AVDTP_ERROR_SEP_NOT_IN_USE	< The SEP is not in use.
	AVDTP_ERROR_SUCCESS	< The operation completed with no errors.
	AVDTP_ERROR_UNSUPPORTED_CONFIGURAION	< Configuration not supported.
	AVDTP_EVT_ABORT_STREAM_COMPLETED	< This event is generated when a local device received a response (either positive or negative) to a "abort stream" request.
	AVDTP_EVT_ABORT_STREAM_REQUESTED	< This event is generated when a local device received "abort stream" request.
	AVDTP_EVT_CLOSE_STREAM_COMPLETED	< This event is generated when a local device received a response (either positive or negative) to a "close stream" request.
	AVDTP_EVT_CLOSE_STREAM_REQUESTED	< This event is generated when a local device received "close stream" request.
	AVDTP_EVT_CTRL_CHANNEL_CONNECTED	< This event is generated when a control channel between two AVDTP entities has been established.

	AVDTP_EVT_CTRL_CHANNEL_DISCONNECTED	< This event is generated when a control channel between two AVDTP entities has been terminated.
	AVDTP_EVT_CTRL_CONNECTION_FAILED	< This event is generated when a local device failed to create a control channel between two AVDTP entities.
	AVDTP_EVT_DELAYREPORT_COMPLETED	< This event is generated when a local device received a response (either positive or negative) to a "delay report" request.
	AVDTP_EVT_DISCOVER_COMPLETED	< This event is generated when a local device received a response (either positive or negative) to a "discover" request.
	AVDTP_EVT_GET_SEP_CAPABILITIES_COMPLETED	< This event is generated when a local device received a response (either positive or negative) to a "get SEP capabilities" request.
	AVDTP_EVT_GET_STREAM_CONFIGURATION_COMPLETED	< This event is generated when a local device received a response (either positive or negative) to a "get stream configuration" request.
	AVDTP_EVT_LAST	This is macro AVDTP_EVT_LAST.
	AVDTP_EVT_MEDIA_PACKET_RECEIVED	< This event is generated when a local device received a media packet.
	AVDTP_EVT_MEDIA_PACKET_SEND_FAILED	< This event is generated when a local device failed to send a media packet.
	AVDTP_EVT_MEDIA_PACKET_SENT	< This event is generated when a local device sent a media packet.
	AVDTP_EVT_NULL	This is macro AVDTP_EVT_NULL.
	AVDTP_EVT_OPEN_STREAM_COMPLETED	< This event is generated when a local device received a response (either positive or negative) to a "open stream" request.
	AVDTP_EVT_OPEN_STREAM_REQUESTED	< This event is generated when a local device received "open stream" request.
	AVDTP_EVT_RECONFIGURE_STREAM_REQUESTED	< This event is generated when a local device received "change stream configuration" request.
	AVDTP_EVT_SEP_CAPABILITIES_RECEIVED	< This event is generated when a local device received a positive response to a "get SEP capabilities" request.
	AVDTP_EVT_SEP_INFO_RECEIVED	< This event is generated for each SEP contained in a positive response to a "discover" request.
	AVDTP_EVT_SET_STREAM_CONFIGURATION	This is macro AVDTP_EVT_SET_STREAM_CONFIGURATION.
	AVDTP_EVT_SET_STREAM_CONFIGURATION_COMPLETED	< This event is generated when a local device received a response (either positive or negative) to a "set stream configuration" request.
	AVDTP_EVT_SET_STREAM_CONFIGURATION_REQUESTED	< This event is generated when a local device received "set stream configuration" request.
	AVDTP_EVT_START_STREAM_COMPLETED	< This event is generated when a local device received a response (either positive or negative) to a "start stream" request.
	AVDTP_EVT_START_STREAM_REQUESTED	< This event is generated when a local device received "start stream" request.
	AVDTP_EVT_STREAM_ABORTED	< This event is generated when a local device has successfully aborted a stream. < This event follows the AVDTP_EVT_ABORT_STREAM_REQUESTED if the upper layer has accepted it. < This event is not generated if stream abortion was initiated by the local device.
	AVDTP_EVT_STREAM_CLOSED	< This event is generated when a local device has successfully closed a stream. < This event follows the AVDTP_EVT_CLOSE_STREAM_REQUESTED if the upper layer has accepted it. < This event is not generated if stream closing was initiated by the local device.
	AVDTP_EVT_STREAM_CONFIGURATION_RECEIVED	< This event is generated when a local device received a positive response to a "get stream configuration" request.
	AVDTP_EVT_STREAM_CONFIGURED	< This event is generated when a local device has successfully configured a stream. < This event follows the AVDTP_EVT_SET_STREAM_CONFIGURATION_REQUESTED if the upper layer has accepted it. < This event is not generated if stream configuration was initiated by the local device.

	AVDTP_EVT_STREAM_OPENED	< This event is generated when a local device has successfully opened a stream. < This event follows the AVDTP_EVT_OPEN_STREAM_REQUESTED if the upper layer has accepted it. < This event is not generated if stream opening was initiated by the local device.
	AVDTP_EVT_STREAM_RECONFIGURE_COMPLETED	< This event is generated when a local device received a response (either positive or negative) to a "change stream configuration" request.
	AVDTP_EVT_STREAM_RECONFIGURED	< This event is generated when a local device has successfully reconfigured a stream. < This event follows the AVDTP_EVT_RECONFIGURE_STREAM_REQUESTED if the upper layer has accepted it. < This event is not generated if stream reconfiguration was initiated by the local device.
	AVDTP_EVT_STREAM_SECURITY_CONTROL_COMPLETED	< This event is generated when a local device received a response (either positive or negative) to a "exchange content protection control data" request.
	AVDTP_EVT_STREAM_STARTED	< This event is generated when a local device has successfully started a stream. < This event follows the AVDTP_EVT_START_STREAM_REQUESTED if the upper layer has accepted it. < This event is not generated if stream starting was initiated by the local device.
	AVDTP_EVT_STREAM_SUSPENDED	< This event is generated when a local device has successfully suspended a stream. < This event follows the AVDTP_EVT_SUSPEND_STREAM_REQUESTED if the upper layer has accepted it. < This event is not generated if stream suspension was initiated by the local device.
	AVDTP_EVT_SUSPEND_STREAM_COMPLETED	< This event is generated when a local device received a response (either positive or negative) to a "suspend stream" request.
	AVDTP_EVT_SUSPEND_STREAM_REQUESTED	< This event is generated when a local device received "suspend stream" request.
	AVDTP_MANAGER_FLAG_SENDING_MEDIA_PACKET	This is macro AVDTP_MANAGER_FLAG_SENDING_MEDIA_PACKET.
	AVDTP_MANAGER_STATE_CONNECTING	This is macro AVDTP_MANAGER_STATE_CONNECTING.
	AVDTP_MANAGER_STATE_IDLE	This is macro AVDTP_MANAGER_STATE_IDLE.
	AVDTP_MAX_STREAM_TRANSPORT_SESSION	This is macro AVDTP_MAX_STREAM_TRANSPORT_SESSION.
	AVDTP_MEDIA_TYPE_AUDIO	< Audio.
	AVDTP_MEDIA_TYPE_MULTIMEDIA	< Both Audio & Video.
	AVDTP_MEDIA_TYPE_VIDEO	< Video.
	AVDTP_SCMS_T_CP_BIT	This is macro AVDTP_SCMS_T_CP_BIT.
	AVDTP_SCMS_T_L_BIT	This is macro AVDTP_SCMS_T_L_BIT.
	AVDTP_SEP_CAPABILITY_FLAG_CONTENT_PROTECTION	< Content Prortection. A SEP is capable of transferring content protection packets.
	AVDTP_SEP_CAPABILITY_FLAG_DELAY_REPORTING	< Delat reporitng.
	AVDTP_SEP_CAPABILITY_FLAG_HEADER_COMPRESSION	< Header Compression. A SEP can use header compression for transferring Media or Recovery packets.
	AVDTP_SEP_CAPABILITY_FLAG_MEDIA_CODEC	< Media Codec. Defines which codec a SEP supports. A SEP can support only one codec.
	AVDTP_SEP_CAPABILITY_FLAG_MEDIA_TRANSPORT	< Media. A SEP is capable of transferring media (audio, video or both) packets.
	AVDTP_SEP_CAPABILITY_FLAG_MULTIPLEXING	< Multiplexing. Multiple transport sessions, belonging to the same or to a different stream, can share a common transport (L2CAP) channel.
	AVDTP_SEP_CAPABILITY_FLAG_RECOVERY	< Recovery. A SEP is capable of transferring recovery packets.
	AVDTP_SEP_CAPABILITY_FLAG_REPORTING	< Reporting. A SEP is capable of transferring reporting packets.
	AVDTP_SEP_SERVICE_CAPABILITY_CONTENT_PROTECTION	< Content Prortection. A SEP is capable of transferring content protection packets.
	AVDTP_SEP_SERVICE_CAPABILITY_DELAY_REPORTING	< Delay Reporting.
	AVDTP_SEP_SERVICE_CAPABILITY_HEADER_COMPRESSION	< Header Compression. A SEP can use header compression for transferring Media or Recovery packets.

	AVDTP_SEP_SERVICE_CAPABILITY_MEDIA_CODEC	< Media Codec. Defines which codec a SEP supports. A SEP can support only one codec.
	AVDTP_SEP_SERVICE_CAPABILITY_MEDIA_TRANSPORT	< Media. A SEP is capable of transferring media (audio, video or both) packets.
	AVDTP_SEP_SERVICE_CAPABILITY_MULTIPLEXING	< Multiplexing. Multiple transport sessions, belonging to the same or to a different stream, can share a common transport (L2CAP) channel.
	AVDTP_SEP_SERVICE_CAPABILITY_RECOVERY	< Recovery. A SEP is capable of transferring recovery packets.
	AVDTP_SEP_SERVICE_CAPABILITY_REPORTING	< Reporting. A SEP is capable of transferring reporting packets.
	AVDTP_SEP_STATE_FREE	This is macro AVDTP_SEP_STATE_FREE.
	AVDTP_SEP_STATE_IDLE	This is macro AVDTP_SEP_STATE_IDLE.
	AVDTP_SEP_TYPE_SINK	< Sink (usually a device like a headphones or BMW).
	AVDTP_SEP_TYPE_SOURCE	< Source (usually a device like a phone, desktop or laptop).
	AVDTP_STREAM_CLOSING_TRANSPORT_CHANNELS	< The stream is closing transport channels.
	AVDTP_STREAM_FLAG_LISTENING	This is macro AVDTP_STREAM_FLAG_LISTENING.
	AVDTP_STREAM_OPENING_TRANSPORT_CHANNELS	< The stream is opening transport channels.
	AVDTP_STREAM_STATE_ABORTING	< The stream is aborting. This means that all transport channels associated with the stream are being closed. < After they have been closed the stream goes to AVDTP_STREAM_STATE_IDLE state.
	AVDTP_STREAM_STATE_CLOSING	< The stream is closing. This means that all transport channels associated with the stream are being closed. < After they have been closed the stream goes to AVDTP_STREAM_STATE_IDLE state.
	AVDTP_STREAM_STATE_CONFIGURED	< The stream has been configured.
	AVDTP_STREAM_STATE_IDLE	< The stream is idle. This can mean two things. The stream specified by strm_handle does not exist < or the stream is closed.
	AVDTP_STREAM_STATE_OPEN	< The stream has been opened.
	AVDTP_STREAM_STATE_STREAMING	< The stream has been started. Depending on the local SEP type (source or sink) it means < that the stream can send or receive media packets.
	AVDTP_TRANSPORT_CHANNEL_TYPE_DEDICATED	This is macro AVDTP_TRANSPORT_CHANNEL_TYPE_DEDICATED.
	AVDTP_TRANSPORT_CHANNEL_TYPE_SHARED	This is macro AVDTP_TRANSPORT_CHANNEL_TYPE_SHARED.
	AVDTP_TRANSPORT_SESSION_TYPE_MEDIA	< Media (audio or video).
	AVDTP_TRANSPORT_SESSION_TYPE_RECOVERY	< Recovery (currently not supported).
	AVDTP_TRANSPORT_SESSION_TYPE_REPORTING	< Reporting (currently not supported).
	bt_avdtp_connect	brief Connect to a remote device. ingroup avdtp details This function opens a control channel connection to a remote device specified by the p remote_addr. If connection cannot be initiated for some reason, for example, there is not enough resources, it returns FALSE and not events are generated. Otherwise the result of an attempt to connect to the remote device is reported via the AVDTP callback. The events generated will either be AVDTP_EVT_CTRL_CHANNEL_CONNECTED or AVDTP_EVT_CTRL_CHANNEL_CONNECTION_FAILED. param mgr AVDTP manager. param remote_addr The address of a remote device. return li c TRUE if connection establishment has been started. li c FALSE... more

	bt_avdtp_connect_ex	brief Connect to a remote device. ingroup avdtp details This function opens a control channel connection to a remote device specified by the p remote_addr. If connection cannot be initiated for some reason, for example, there is not enough resources, it returns FALSE and not events are generated. Otherwise the result of an attempt to connect to the remote device is reported via the AVDTP callback. The events generated will either be AVDTP_EVT_CTRL_CHANNEL_CONNECTED or AVDTP_EVT_CTRL_CHANNEL_CONNECTION_FAILED. param mgr AVDTP manager. param remote_addr The address of a remote device. param acl_config ACL link configuration. This can be a combination of the following... more

Structures

	Name	Description
	_bt_avdtp_codec_op_decode_t	There is currently no use for this structure.
	_bt_avdtp_codec_op_encode_t	There is currently no use for this structure.
	_bt_avdtp_codec_op_parse_config_t	brief Parameter to AVDTP_CODEC_OPCODE_PARSE_CONFIG operation. ingroup avdtp details A pointer to this structure is passed to the codec handler as a valid member of the bt_avdtp_codec_op_param_t union - bt_avdtp_codec_op_param_t::parse - when ADVDTP needs to parse codec's capabilities/configuration received from the remote device.
	_bt_avdtp_codec_op_parse_packet_t	There is currently no use for this structure.
	_bt_avdtp_codec_op_serialize_config_t	brief Parameter to AVDTP_CODEC_OPCODE_SERIALIZE_CONFIG operation. ingroup avdtp details A pointer to this structure is passed to the codec handler as a valid member of the bt_avdtp_codec_op_param_t union - bt_avdtp_codec_op_param_t::serialize - when ADVDTP needs to serialize codec's capabilities/configuration for sending to the remote device.
	_bt_avdtp_codec_t	brief Codec handler description. ingroup avdtp details This structure is used to register a codec handler for parsing/serializing codec capabilities and configuration. See description of the ::bt_avdtp_register_codec for more details.
	_bt_avdtp_evt_abort_stream_requested_t	brief Parameter to AVDTP_EVT_ABORT_STREAM_REQUESTED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::abort_stream_requested - when AVDTP received a "abort stream" request.
	_bt_avdtp_evt_close_stream_completed_t	brief Parameter to AVDTP_EVT_CLOSE_STREAM_COMPLETED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::close_stream_completed - when AVDTP received a response to a "close stream" request.
	_bt_avdtp_evt_close_stream_requested_t	brief Parameter to AVDTP_EVT_CLOSE_STREAM_REQUESTED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::close_stream_requested - when AVDTP received a "close stream" request.
	_bt_avdtp_evt_ctrl_channel_connected_s	brief Parameter to AVDTP_EVT_CTRL_CHANNEL_CONNECTED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::ctrl_channel_connected - when a control channel between two devices has been established.
	_bt_avdtp_evt_ctrl_channel_disconnected_s	brief Parameter to AVDTP_EVT_CTRL_CHANNEL_DISCONNECTED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::ctrl_channel_disconnected - when a control channel between two devices has been terminated.

	_bt_avdtp_evt_ctrl_connection_failed_s	brief Parameter to AVDTP_EVT_CTRL_CONNECTION_FAILED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::ctrl_connection_failed - when a control channel between two devices has been established.
	_bt_avdtp_evt_delay_report_completed_t	brief Parameter to AVDTP_EVT_DELAYREPORT_COMPLETED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::delay_report_completed - when AVDTP received a response to a "delay report" request.
	_bt_avdtp_evt_discover_completed_t	brief Parameter to AVDTP_EVT_DISCOVER_COMPLETED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::discover_completed - when AVDTP completed discovering SEPs available on a remote device.
	_bt_avdtp_evt_get_sep_capabilities_completed_t	brief Parameter to AVDTP_EVT_GET_SEP_CAPABILITIES_COMPLETED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::get_sep_capabilities_completed - when AVDTP received a response to a "get SEP capabilities" request. AVDTP_EVT_GET_SEP_CAPABILITIES_COMPLETED only informs the status of the request - success or failure. In case of success another event - AVDTP_EVT_SEP_CAPABILITIES_RECEIVED - is generate with a pointer to a structure that holds actual SEP's capabilities.
	_bt_avdtp_evt_get_stream_configuration_completed_t	brief Parameter to AVDTP_EVT_SET_STREAM_CONFIGURATION_COMPLETED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::get_stream_configuration_completed - when AVDTP received a response to a "get stream configuration" request. AVDTP_EVT_SET_STREAM_CONFIGURATION_COMPLETED only informs the status of the request - success or failure. In case of success another event - AVDTP_EVT_STREAM_CONFIGURATION_RECEIVED - is generate with a pointer to a structure that hold actual stream's configuration.
	_bt_avdtp_evt_media_packet_received_t	brief Parameter to AVDTP_EVT_MEDIA_PACKET_RECEIVED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::media_packet_received - when ADVDTP received a media packet from the remote device.
	_bt_avdtp_evt_media_packet_send_failed_t	brief Parameter to AVDTP_EVT_MEDIA_PACKET_SEND_FAILED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::media_packet_send_failed - when ADVDTP failed to send a media packet to the remote device.
	_bt_avdtp_evt_media_packet_sent_t	brief Parameter to AVDTP_EVT_MEDIA_PACKET_SENT event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::media_packet_sent - when ADVDTP sent a media packet to the remote device.
	_bt_avdtp_evt_open_stream_completed_t	brief Parameter to AVDTP_EVT_OPEN_STREAM_COMPLETED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::open_stream_completed - when AVDTP received a response to a "open stream" request.

	_bt_avdtp_evt_open_stream_requested_t	brief Parameter to AVDTP_EVT_OPEN_STREAM_REQUESTED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::open_stream_requested - when AVDTP received a "open stream" request.
	_bt_avdtp_evt_reconfigure_stream_requested_t	brief Parameter to AVDTP_EVT_RECONFIGURE_STREAM_REQUESTED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::reconfigure_stream_requested - when AVDTP received a "change stream configuration" request.
	_bt_avdtp_evt_sep_capabilities_received_t	brief Parameter to AVDTP_EVT_SEP_CAPABILITIES_RECEIVED and AVDTP_EVT_STREAM_CONFIGURATION_RECEIVED events ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::sep_capabilities_received - when AVDTP received a positive response to a "get SEP capabilities" or "get stream configuration" request.
	_bt_avdtp_evt_sep_info_received_t	brief Parameter to AVDTP_EVT_SEP_INFO_RECEIVED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::sep_info_received - when AVDTP received positive result to a "discover" request. AVDTP_EVT_SEP_INFO_RECEIVED is generated for every SEP received from the remote device.
	_bt_avdtp_evt_set_stream_configuration_completed_t	brief Parameter to AVDTP_EVT_SET_STREAM_CONFIGURATION_COMPLETED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::set_stream_configuration_completed - when AVDTP received a response to a "set stream configuration" request.
	_bt_avdtp_evt_set_stream_configuration_requested_t	brief Parameter to AVDTP_EVT_SET_STREAM_CONFIGURATION_REQUESTED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::set_stream_configuration_requested - when AVDTP received a "set stream configuration" request.
	_bt_avdtp_evt_set_stream_configuration_t	brief Parameter to AVDTP_EVT_SET_STREAM_CONFIGURATION event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::set_stream_configuration - when AVDTP received a "set stream configuration" request.
	_bt_avdtp_evt_start_stream_completed_t	brief Parameter to AVDTP_EVT_START_STREAM_COMPLETED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::start_stream_completed - when AVDTP received a response to a "start stream" request.
	_bt_avdtp_evt_start_stream_requested_t	brief Parameter to AVDTP_EVT_START_STREAM_REQUESTED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::start_stream_requested - when AVDTP received a "start stream" request.
	_bt_avdtp_evt_stream_aborted_t	brief Parameter to AVDTP_EVT_STREAM_ABORTED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::stream_closed - to notify the AVDTP consumer that a stream has been successfully aborted.

	_bt_avdtp_evt_stream_closed_t	brief Parameter to AVDTP_EVT_STREAM_CLOSED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::stream_closed - to notify the AVDTP consumer that a stream has been successfully closed.
	_bt_avdtp_evt_stream_configured_t	brief Parameter to AVDTP_EVT_STREAM_CONFIGURED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::stream_configured - to notify the AVDTP consumer that a stream configuration has been successfully completed.
	_bt_avdtp_evt_stream_opened_t	brief Parameter to AVDTP_EVT_STREAM_OPENED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::stream_opened - to notify the AVDTP consumer that a stream has been successfully opened.
	_bt_avdtp_evt_stream_reconfigure_completed_t	brief Parameter to AVDTP_EVT_STREAM_RECONFIGURE_COMPLETED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::stream_reconfigure_completed - when AVDTP received a response to a "change stream configuration" request.
	_bt_avdtp_evt_stream_reconfigured_t	brief Parameter to AVDTP_EVT_STREAM_RECONFIGURED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::stream_reconfigured - to notify the AVDTP consumer that a stream configuration has been successfully changed.
	_bt_avdtp_evt_stream_security_control_completed_t	brief Parameter to AVDTP_EVT_STREAM_SECURITY_CONTROL_COMPLETED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::security_control_completed - when AVDTP received a response to a "exchange content protection control data" request.
	_bt_avdtp_evt_stream_started_t	brief Parameter to AVDTP_EVT_STREAM_STARTED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::stream_started - to notify the AVDTP consumer that a stream has been successfully started.
	_bt_avdtp_evt_stream_suspended_t	brief Parameter to AVDTP_EVT_STREAM_SUSPENDED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::stream_suspended - to notify the AVDTP consumer that a stream has been successfully suspended.
	_bt_avdtp_evt_suspend_stream_completed_t	brief Parameter to AVDTP_EVT_SUSPEND_STREAM_COMPLETED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::suspend_stream_completed - when AVDTP received a response to a "suspend stream" request.
	_bt_avdtp_evt_suspend_stream_requested_t	brief Parameter to AVDTP_EVT_SUSPEND_STREAM_REQUESTED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::suspend_stream_requested - when AVDTP received a "suspend stream" request.
	_bt_avdtp_mgr_t	brief AVDTP manager. ingroup avdtp details A structure that glues all pieces together. There is only one instance of this structure allocated by dotstack. A pointer to the instance can be get with c bt_avdtp_get_mgr().
	_bt_avdtp_sep_capabilities_t	brief SEP capabilities ingroup avdtp details This structure is used to hold SEP capabilities.
	_bt_avdtp_sep_t	brief SEP description ingroup avdtp details This structure is used to hold information about SEPs available on a local device.

	_bt_avdtp_stream_t	brief Stream description ingroup avdtp details This structure is used to hold information about streams available on a local device.
	_bt_avdtp_transport_channel_t	brief Transport channel description ingroup avdtp details This structure is used to hold information about transport channels available on a local device.
	_bt_avdtp_transport_session_t	brief Transport session description ingroup avdtp details This structure is used to hold information about transport sessions available on a local device.
	_bt_cp_header_s	This is record _bt_cp_header_s.
	_bt_media_packet_t	brief Media packet buffer ingroup avdtp details This structure is used to receive and send media packet from/to the remote device. See more information about usage of this structure in descriptions of ::bt_avdtp_add_media_rx_buffer and ::bt_avdtp_add_media_tx_buffer.
	bt_avdtp_codec_op_decode_t	There is currently no use for this structure.
	bt_avdtp_codec_op_encode_t	There is currently no use for this structure.
	bt_avdtp_codec_op_parse_config_t	brief Parameter to AVDTP_CODEC_OPCODE_PARSE_CONFIG operation. ingroup avdtp details A pointer to this structure is passed to the codec handler as a valid member of the bt_avdtp_codec_op_param_t union - bt_avdtp_codec_op_param_t::parse - when ADVDTP needs to parse codec's capabilities/configuration received from the remote device.
	bt_avdtp_codec_op_parse_packet_t	There is currently no use for this structure.
	bt_avdtp_codec_op_serialize_config_t	brief Parameter to AVDTP_CODEC_OPCODE_SERIALIZE_CONFIG operation. ingroup avdtp details A pointer to this structure is passed to the codec handler as a valid member of the bt_avdtp_codec_op_param_t union - bt_avdtp_codec_op_param_t::serialize - when ADVDTP needs to serialize codec's capabilities/configuration for sending to the remote device.
	bt_avdtp_codec_t	brief Codec handler description. ingroup avdtp details This structure is used to register a codec handler for parsing/serializing codec capabilities and configuration. See description of the ::bt_avdtp_register_codec for more details.
	bt_avdtp_evt_abort_stream_requested_t	brief Parameter to AVDTP_EVT_ABORT_STREAM_REQUESTED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::abort_stream_requested - when AVDTP received a "abort stream" request.
	bt_avdtp_evt_close_stream_completed_t	brief Parameter to AVDTP_EVT_CLOSE_STREAM_COMPLETED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::close_stream_completed - when AVDTP received a response to a "close stream" request.
	bt_avdtp_evt_close_stream_requested_t	brief Parameter to AVDTP_EVT_CLOSE_STREAM_REQUESTED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::close_stream_requested - when AVDTP received a "close stream" request.
	bt_avdtp_evt_ctrl_channel_connected_t	brief Parameter to AVDTP_EVT_CTRL_CHANNEL_CONNECTED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::ctrl_channel_connected - when a control channel between two devices has been established.
	bt_avdtp_evt_ctrl_channel_disconnected_t	brief Parameter to AVDTP_EVT_CTRL_CHANNEL_DISCONNECTED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::ctrl_channel_disconnected - when a control channel between two devices has been terminated.

	bt_avdtp_evt_ctrl_connection_failed_t	brief Parameter to AVDTP_EVT_CTRL_CONNECTION_FAILED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::ctrl_connection_failed - when a control channel between two devices has been established.
	bt_avdtp_evt_delay_report_completed_t	brief Parameter to AVDTP_EVT_DELAYREPORT_COMPLETED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::delay_report_completed - when AVDTP received a response to a "delay report" request.
	bt_avdtp_evt_discover_completed_t	brief Parameter to AVDTP_EVT_DISCOVER_COMPLETED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::discover_completed - when AVDTP completed discovering SEPs available on a remote device.
	bt_avdtp_evt_get_sep_capabilities_completed_t	brief Parameter to AVDTP_EVT_GET_SEP_CAPABILITIES_COMPLETED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::get_sep_capabilities_completed - when AVDTP received a response to a "get SEP capabilities" request. AVDTP_EVT_GET_SEP_CAPABILITIES_COMPLETED only informs the status of the request - success or failure. In case of success another event - AVDTP_EVT_SEP_CAPABILITIES_RECEIVED - is generate with a pointer to a structure that holds actual SEP's capabilities.
	bt_avdtp_evt_get_stream_configuration_completed_t	brief Parameter to AVDTP_EVT_SET_STREAM_CONFIGURATION_COMPLETED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::get_stream_configuration_completed - when AVDTP received a response to a "get stream configuration" request. AVDTP_EVT_SET_STREAM_CONFIGURATION_COMPLETED only informs the status of the request - success or failure. In case of success another event - AVDTP_EVT_STREAM_CONFIGURATION_RECEIVED - is generate with a pointer to a structure that hold actual stream's configuration.
	bt_avdtp_evt_media_packet_received_t	brief Parameter to AVDTP_EVT_MEDIA_PACKET_RECEIVED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::media_packet_received - when ADVDTP received a media packet from the remote device.
	bt_avdtp_evt_media_packet_send_failed_t	brief Parameter to AVDTP_EVT_MEDIA_PACKET_SEND_FAILED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::media_packet_send_failed - when ADVDTP failed to send a media packet to the remote device.
	bt_avdtp_evt_media_packet_sent_t	brief Parameter to AVDTP_EVT_MEDIA_PACKET_SENT event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::media_packet_sent - when ADVDTP sent a media packet to the remote device.
	bt_avdtp_evt_open_stream_completed_t	brief Parameter to AVDTP_EVT_OPEN_STREAM_COMPLETED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::open_stream_completed - when AVDTP received a response to a "open stream" request.

	bt_avdtp_evt_open_stream_requested_t	brief Parameter to AVDTP_EVT_OPEN_STREAM_REQUESTED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::open_stream_requested - when AVDTP received a "open stream" request.
	bt_avdtp_evt_reconfigure_stream_requested_t	brief Parameter to AVDTP_EVT_RECONFIGURE_STREAM_REQUESTED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::reconfigure_stream_requested - when AVDTP received a "change stream configuration" request.
	bt_avdtp_evt_sep_capabilities_received_t	brief Parameter to AVDTP_EVT_SEP_CAPABILITIES_RECEIVED and AVDTP_EVT_STREAM_CONFIGURATION_RECEIVED events ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::sep_capabilities_received - when AVDTP received a positive response to a "get SEP capabilities" or "get stream configuration" request.
	bt_avdtp_evt_sep_info_received_t	brief Parameter to AVDTP_EVT_SEP_INFO_RECEIVED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::sep_info_received - when AVDTP received positive result to a "discover" request. AVDTP_EVT_SEP_INFO_RECEIVED is generated for every SEP received from the remote device.
	bt_avdtp_evt_set_stream_configuration_completed_t	brief Parameter to AVDTP_EVT_SET_STREAM_CONFIGURATION_COMPLETED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::set_stream_configuration_completed - when AVDTP received a response to a "set stream configuration" request.
	bt_avdtp_evt_set_stream_configuration_requested_t	brief Parameter to AVDTP_EVT_SET_STREAM_CONFIGURATION_REQUESTED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::set_stream_configuration_requested - when AVDTP received a "set stream configuration" request.
	bt_avdtp_evt_set_stream_configuration_t	brief Parameter to AVDTP_EVT_SET_STREAM_CONFIGURATION event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::set_stream_configuration - when AVDTP received a "set stream configuration" request.
	bt_avdtp_evt_start_stream_completed_t	brief Parameter to AVDTP_EVT_START_STREAM_COMPLETED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::start_stream_completed - when AVDTP received a response to a "start stream" request.
	bt_avdtp_evt_start_stream_requested_t	brief Parameter to AVDTP_EVT_START_STREAM_REQUESTED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::start_stream_requested - when AVDTP received a "start stream" request.
	bt_avdtp_evt_stream_aborted_t	brief Parameter to AVDTP_EVT_STREAM_ABORTED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::stream_closed - to notify the AVDTP consumer that a stream has been successfully aborted.

	bt_avdtp_evt_stream_closed_t	brief Parameter to AVDTP_EVT_STREAM_CLOSED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::stream_closed - to notify the AVDTP consumer that a stream has been successfully closed.
	bt_avdtp_evt_stream_configured_t	brief Parameter to AVDTP_EVT_STREAM_CONFIGURED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::stream_configured - to notify the AVDTP consumer that a stream configuration has been successfully completed.
	bt_avdtp_evt_stream_opened_t	brief Parameter to AVDTP_EVT_STREAM_OPENED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::stream_opened - to notify the AVDTP consumer that a stream has been successfully opened.
	bt_avdtp_evt_stream_reconfigure_completed_t	brief Parameter to AVDTP_EVT_STREAM_RECONFIGURE_COMPLETED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::stream_reconfigure_completed - when AVDTP received a response to a "change stream configuration" request.
	bt_avdtp_evt_stream_reconfigured_t	brief Parameter to AVDTP_EVT_STREAM_RECONFIGURED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::stream_reconfigured - to notify the AVDTP consumer that a stream configuration has been successfully changed.
	bt_avdtp_evt_stream_security_control_completed_t	brief Parameter to AVDTP_EVT_STREAM_SECURITY_CONTROL_COMPLETED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::security_control_completed - when AVDTP received a response to a "exchange content protection control data" request.
	bt_avdtp_evt_stream_started_t	brief Parameter to AVDTP_EVT_STREAM_STARTED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::stream_started - to notify the AVDTP consumer that a stream has been successfully started.
	bt_avdtp_evt_stream_suspended_t	brief Parameter to AVDTP_EVT_STREAM_SUSPENDED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::stream_suspended - to notify the AVDTP consumer that a stream has been successfully suspended.
	bt_avdtp_evt_suspend_stream_completed_t	brief Parameter to AVDTP_EVT_SUSPEND_STREAM_COMPLETED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::suspend_stream_completed - when AVDTP received a response to a "suspend stream" request.
	bt_avdtp_evt_suspend_stream_requested_t	brief Parameter to AVDTP_EVT_SUSPEND_STREAM_REQUESTED event ingroup avdtp details A pointer to this structure is passed to the AVDTP application callback as a valid member of the bt_avdtp_event_t union - bt_avdtp_event_t::suspend_stream_requested - when AVDTP received a "suspend stream" request.
	bt_avdtp_mgr_t	brief AVDTP manager. ingroup avdtp details A structure that glues all pieces together. There is only one instance of this structure allocated by dotstack. A pointer to the instance can be get with c bt_avdtp_get_mgr().
	bt_avdtp_sep_capabilities_t	brief SEP capabilities ingroup avdtp details This structure is used to hold SEP capabilities.
	bt_avdtp_stream_t	brief Stream description ingroup avdtp details This structure is used to hold information about streams available on a local device.

	bt_avdtp_transport_channel_t	brief Transport channel description ingroup avdtp details This structure is used to hold information about transport channels available on a local device.
	bt_avdtp_transport_session_t	brief Transport session description ingroup avdtp details This structure is used to hold information about transport sessions available on a local device.

Types

	Name	Description
	bt_avdtp_codec_handler_fp	brief Codec handler. ingroup avdtp details AVDTP in theory can support any type of codec. Each codec uses its own format for exchanging capabilities and configuration information. In order to make out implementation do not care about these formats we use a simple way of telling AVDTP how to parse and serialize codec's configuration. The consumer of AVDTP (e.g. A2DP) for each codec it wishes to support has to register a callback function (one per codec type). That callback has to perform two function. The first one is to read the configuration received from the remote device and store it... more
	bt_avdtp_codec_op_param_t	This is type bt_avdtp_codec_op_param_t.
	bt_avdtp_mgr_callback_fp	brief AVDTP application callback. ingroup avdtp details In order to be notified of various events a consumer of the AVDTP layer has to register a callback function. The stack will call that function whenever a new event has been generated. param mgr AVDTP manager. param evt AVDTP event. The event can be one of the following values: @arg AVDTP_EVT_CTRL_CHANNEL_CONNECTED: Control channel connected. @arg AVDTP_EVT_CTRL_CHANNEL_DISCONNECTED: Control channel disconnected. @arg AVDTP_EVT_CTRL_CONNECTION_FAILED: Control channel connection failed (generated only if control connection has been initiated by the local device). @arg AVDTP_EVT_DISCOVER_COMPLETED: Local device completed discovering remote SEPs. @arg AVDTP_EVT_GET_SEP_CAPABILITIES_COMPLETED:... more
	bt_avdtp_sep_t	This is type bt_avdtp_sep_t.
	bt_cp_header_t	This is type bt_cp_header_t.
	bt_media_packet_t	This is type bt_media_packet_t.

Unions

	Name	Description
	_bt_avdtp_codec_op_param_u	brief Parameter to a codec handler. ingroup avdtp details This union is used to pass operation specific data to a codec handler. Which member of the union points to a valid structure depends on the operation.
	_bt_avdtp_event_u	brief Parameter to an application callback. ingroup avdtp details This union is used to pass event specific data to the AVDTP consumer. Which member of the union points to a valid structure depends on the event reported to the consumer. In general, each event has a corresponding member in the union.
	bt_avdtp_event_t	brief Parameter to an application callback. ingroup avdtp details This union is used to pass event specific data to the AVDTP consumer. Which member of the union points to a valid structure depends on the event reported to the consumer. In general, each event has a corresponding member in the union.

Description

Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.

avdtp_config.h

Macros

	Name	Description
	__AVDTP_CONFIG_H	This is macro __AVDTP_CONFIG_H.
	AVDTP_ALLOCATE_BUFFERS_FUNCTION	This is macro AVDTP_ALLOCATE_BUFFERS_FUNCTION.

	AVDTP_ALLOCATE_BUFFERS_RAM_SIZE_VAR	• defgroup avdtp_config Configuration
		• ingroup avdtp
		*
		• This module describes parameters used to configure AVDTP layer.
		*
		• dotstack is customized using a configuration file. The configuration file tailors the dotstack to the application being built. It has to have the structure shown below.
		* code #include "cdbt/bt/bt_std.h"
		// HCI, L2CAP and SDP must always be present
		// HCI configuration parameters #define HCI_MAX_CMD_BUFFERS ... #define HCI_MAX_DATA_BUFFERS ... #define HCI_MAX_HCI_CONNECTIONS ... #define HCI_RX_BUFFER_LEN ... #define HCI_TX_BUFFER_LEN ... #define HCI_L2CAP_BUFFER_LEN ... #define HCI_MAX_CMD_PARAM_LEN ...
		// L2CAP configuration parameters #define L2CAP_MAX_CMD_BUFFERS ... #define L2CAP_MAX_FRAME_BUFFERS ... #define L2CAP_MAX_PSMS ... #define L2CAP_MAX_CHANNELS ...
		// SDP... more

	AVDTP_ALLOCATE_BUFFERS_VARS	This is macro AVDTP_ALLOCATE_BUFFERS_VARS.
	AVDTP_CODEC_CONFIG_BUFFER_LEN	brief Size of the buffer used to store codec specific configuration. ingroup avdtp_config details Each codec uses unique configuration which can take different amount of memory. This parameter defines the size of the buffer for storing codec's configuration. The value of 16 is sufficient for SBC, AAC and MPEG1,2. If vendor specific codec is to be used this value may need to increased.
	AVDTP_MAX_CMD_BUFFERS	brief Maximum number of command buffers. ingroup avdtp_config details This parameter defines the number of buffers reserved for sending commands to a remote device over its control channel. Each channel uses its own buffers so the total number of buffers is AVDTP_MAX_REMOTE_DEVICES * AVDTP_MAX_CMD_BUFFERS. The minimum value is 1. The maximum value is 255. 2 is usually sufficient.
	AVDTP_MAX_CMD_PARAM_LEN	brief Maximum length of control command parameters ingroup avdtp_config details This parameter defines the maximum length of all command parameters. The value should not exceed AVDTP_MAX_TX_BUFFER_LEN - 2 (command header).
	AVDTP_MAX_REMOTE_DEVICES	brief Maximum number of remote devices a local device can be connected to ingroup avdtp_config details This parameter defines the number of remote devices a local device can have simultaneous connections to (i.e. control channels). This value should not exceed AVDTP_MAX_STREAMS. For each remote device AVDTP creates one control channel regardless of the number of streams between the local and the remote devices. Assuming that the local devices wants to have only one channel with each remote device and if AVDTP_MAX_REMOTE_DEVICES > AVDTP_MAX_STREAMS all memory reserved for devices in excess of AVDTP_MAX_STREAMS will be wasted. The minimum value is... more
	AVDTP_MAX_SEP	brief Maximum number of SEPs that can be exposed by a local device ingroup avdtp_config details This parameter defines the number of SEPs an application can expose to remote devices. The minimum value is 1. The maximum value is 255.
	AVDTP_MAX_STREAMS	brief Maximum number of streams that can be exposed by a local device ingroup avdtp_config details This parameter defines the number of streams an application can open between local and remote devices. This value can be different from AVDTP_MAX_SEP but should not exceed it. Since each SEP can only be used once the local device can only have as much streams as there are SEPs. If AVDTP_MAX_STREAMS > AVDTP_MAX_SEP all memory reserved for streams in excess of AVDTP_MAX_SEP will be wasted. The minimum value is 1. The maximum value is 255.

	AVDTP_MAX_TRANSPORT_CHANNELS	brief Maximum number of transport channels. ingroup avdtp_config details Depending on the SEP capabilities (multiplexing, recovery, reporting) each stream may need up to 3 transport channels. E.g., if multiplexing is not supported and recovery and reporting are supported a stream will use 3 transport channels - 1 for media transport session, 1 for recovery transport session and 1 for reporting transport session. If multiplexing is not supported and only reporting is supported the stream will use 2 transport channels - 1 for media transport session and 1 for reporting transport session. If multiplexing is supported the stream will need only... more
	AVDTP_MAX_TX_BUFFER_LEN	brief Size of the transmit buffer. ingroup avdtp_config details This parameter defines the size of the buffer used to send AVDTP control commands to L2CAP layer. Each control channel has it own buffer so the total amount of memory allocated for these buffers is AVDTP_MAX_TX_BUFFER_LEN) * AVDTP_MAX_REMOTE_DEVICES. The minimum value is 1. The maximum value is 255. The value of 32 is usually sufficient.

Description

Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.

avdtp_control.h

Functions

	Name	Description
	_bt_avdtp_add_param_byte	This is function _bt_avdtp_add_param_byte.
	_bt_avdtp_add_param_uint	This is function _bt_avdtp_add_param_uint.
	_bt_avdtp_add_param_uintn	This is function _bt_avdtp_add_param_uintn.
	_bt_avdtp_control_channel_event_handler	This is function _bt_avdtp_control_channel_event_handler.
	_bt_avdtp_get_control_channel	This is function _bt_avdtp_get_control_channel.
	_bt_avdtp_open_control_channel_ex	This is function _bt_avdtp_open_control_channel_ex.

Macros

	Name	Description
	__AVDTP_CONTROL_H	This is macro __AVDTP_CONTROL_H.
	AVDTP_CMD_ABORT	This is macro AVDTP_CMD_ABORT.
	AVDTP_CMD_CLOSE	This is macro AVDTP_CMD_CLOSE.
	AVDTP_CMD_DELAYREPORT	This is macro AVDTP_CMD_DELAYREPORT.
	AVDTP_CMD_DISCOVER	This is macro AVDTP_CMD_DISCOVER.
	AVDTP_CMD_GET_ALL_CAPABILITIES	This is macro AVDTP_CMD_GET_ALL_CAPABILITIES.
	AVDTP_CMD_GET_CAPABILITIES	This is macro AVDTP_CMD_GET_CAPABILITIES.
	AVDTP_CMD_GET_CONFIGURATION	This is macro AVDTP_CMD_GET_CONFIGURATION.
	AVDTP_CMD_OPEN	This is macro AVDTP_CMD_OPEN.
	AVDTP_CMD_RECONFIGURE	This is macro AVDTP_CMD_RECONFIGURE.
	AVDTP_CMD_SECURITY_CONTROL	This is macro AVDTP_CMD_SECURITY_CONTROL.
	AVDTP_CMD_SET_CONFIGURATION	This is macro AVDTP_CMD_SET_CONFIGURATION.
	AVDTP_CMD_START	This is macro AVDTP_CMD_START.
	AVDTP_CMD_SUSPEND	This is macro AVDTP_CMD_SUSPEND.
	AVDTP_CTLR_CHANNEL_EVT_CONNECTED	This is macro AVDTP_CTLR_CHANNEL_EVT_CONNECTED.
	AVDTP_CTLR_CHANNEL_EVT_CONNECTION_FAILED	This is macro AVDTP_CTLR_CHANNEL_EVT_CONNECTION_FAILED.
	AVDTP_CTLR_CHANNEL_EVT_DISCONNECTED	This is macro AVDTP_CTLR_CHANNEL_EVT_DISCONNECTED.
	AVDTP_CTLR_CHANNEL_EVT_NOTHING	This is macro AVDTP_CTLR_CHANNEL_EVT_NOTHING.
	AVDTP_CTRL_CHANNEL_EVT_DATA_RECEIVED	This is macro AVDTP_CTRL_CHANNEL_EVT_DATA_RECEIVED.
	AVDTP_CTRL_CHANNEL_STATE_CONNECTED	This is macro AVDTP_CTRL_CHANNEL_STATE_CONNECTED.
	AVDTP_CTRL_CHANNEL_STATE_CONNECTING	This is macro AVDTP_CTRL_CHANNEL_STATE_CONNECTING.
	AVDTP_CTRL_CHANNEL_STATE_DISCONNECTED	This is macro AVDTP_CTRL_CHANNEL_STATE_DISCONNECTED.
	AVDTP_CTRL_MESSAGE_TYPE_ACCEPT	This is macro AVDTP_CTRL_MESSAGE_TYPE_ACCEPT.
	AVDTP_CTRL_MESSAGE_TYPE_COMMAND	This is macro AVDTP_CTRL_MESSAGE_TYPE_COMMAND.

	AVDTP_CTRL_MESSAGE_TYPE_FLD	This is macro AVDTP_CTRL_MESSAGE_TYPE_FLD.
	AVDTP_CTRL_MESSAGE_TYPE_GENERAL_REJECT	This is macro AVDTP_CTRL_MESSAGE_TYPE_GENERAL_REJECT.
	AVDTP_CTRL_MESSAGE_TYPE_REJECT	This is macro AVDTP_CTRL_MESSAGE_TYPE_REJECT.
	AVDTP_CTRL_PACKET_TYPE_CONTINUE	This is macro AVDTP_CTRL_PACKET_TYPE_CONTINUE.
	AVDTP_CTRL_PACKET_TYPE_END	This is macro AVDTP_CTRL_PACKET_TYPE_END.
	AVDTP_CTRL_PACKET_TYPE_FLD	This is macro AVDTP_CTRL_PACKET_TYPE_FLD.
	AVDTP_CTRL_PACKET_TYPE_SIGNLE	This is macro AVDTP_CTRL_PACKET_TYPE_SIGNLE.
	AVDTP_CTRL_PACKET_TYPE_START	This is macro AVDTP_CTRL_PACKET_TYPE_START.

Structures

	Name	Description
	_bt_avdtp_control_channel_t	This is type bt_avdtp_control_channel_t.
	_bt_avdtp_control_cmd_t	This is type bt_avdtp_control_cmd_t.
	_bt_avdtp_ctrl_evt_data_received_t	This is type bt_avdtp_ctrl_evt_data_received_t.
	bt_avdtp_control_channel_t	This is type bt_avdtp_control_channel_t.
	bt_avdtp_control_cmd_t	This is type bt_avdtp_control_cmd_t.
	bt_avdtp_ctrl_evt_data_received_t	This is type bt_avdtp_ctrl_evt_data_received_t.

Description

Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.

avdtp_private.h

Functions

	Name	Description
	_bt_avdtp_allocate_buffers	This is function _bt_avdtp_allocate_buffers.
	_bt_avdtp_allocate_cmd	This is function _bt_avdtp_allocate_cmd.
	_bt_avdtp_allocate_sep	This is function _bt_avdtp_allocate_sep.
	_bt_avdtp_allocate_sep_config	This is function _bt_avdtp_allocate_sep_config.
	_bt_avdtp_allocate_stream	This is function _bt_avdtp_allocate_stream.
	_bt_avdtp_allocate_transport_channel	This is function _bt_avdtp_allocate_transport_channel.
	_bt_avdtp_allocate_transport_session_id	This is function _bt_avdtp_allocate_transport_session_id.
	_bt_avdtp_begin_tc_channel_operation	This is function _bt_avdtp_begin_tc_channel_operation.
	_bt_avdtp_commit_tc_channel_operation	This is function _bt_avdtp_commit_tc_channel_operation.
	_bt_avdtp_control_channel_accept_handler	This is function _bt_avdtp_control_channel_accept_handler.
	_bt_avdtp_control_channel_cmd_handler	This is function _bt_avdtp_control_channel_cmd_handler.
	_bt_avdtp_control_channel_reject_handler	This is function _bt_avdtp_control_channel_reject_handler.
	_bt_avdtp_execute_tc_channel_operation	This is function _bt_avdtp_execute_tc_channel_operation.
	_bt_avdtp_find_listening_stream	This is function _bt_avdtp_find_listening_stream.
	_bt_avdtp_find_stream	This is function _bt_avdtp_find_stream.
	_bt_avdtp_find_stream_by_remote_sep_id	This is function _bt_avdtp_find_stream_by_remote_sep_id.
	_bt_avdtp_find_stream_by_sep_id	This is function _bt_avdtp_find_stream_by_sep_id.
	_bt_avdtp_free_cmd	This is function _bt_avdtp_free_cmd.
	_bt_avdtp_free_sep_config	This is function _bt_avdtp_free_sep_config.
	_bt_avdtp_free_stream	This is function _bt_avdtp_free_stream.
	_bt_avdtp_free_transport_channel	This is function _bt_avdtp_free_transport_channel.
	_bt_avdtp_init_cmd_buffers	This is function _bt_avdtp_init_cmd_buffers.
	_bt_avdtp_init_sep_config_buffers	This is function _bt_avdtp_init_sep_config_buffers.
	_bt_avdtp_init_signal	This is function _bt_avdtp_init_signal.
	_bt_avdtp_is_sep_inuse	This is function _bt_avdtp_is_sep_inuse.
	_bt_avdtp_open_control_channel_ex	This is function _bt_avdtp_open_control_channel_ex.
	_bt_avdtp_read_caps	This is function _bt_avdtp_read_caps.
	_bt_avdtp_register_transport_channel_for_operation	This is function _bt_avdtp_register_transport_channel_for_operation.
	_bt_avdtp_send_command	This is function _bt_avdtp_send_command.

	_bt_avdtp_send_media_packet	This is function _bt_avdtp_send_media_packet.
	_bt_avdtp_set_signal	This is function _bt_avdtp_set_signal.
	_bt_avdtp_transport_l2cap_read_data_callback	This is function _bt_avdtp_transport_l2cap_read_data_callback.
	_bt_avdtp_transport_l2cap_state_changed_callback	This is function _bt_avdtp_transport_l2cap_state_changed_callback.
	_bt_avdtp_write_caps	This is function _bt_avdtp_write_caps.

Macros

	Name	Description
	__AVDTP_PRIVATE_H	This is macro __AVDTP_PRIVATE_H.
	AVDTP_TC_OPCODE_CONNECT	This is macro AVDTP_TC_OPCODE_CONNECT.
	AVDTP_TC_OPCODE_DISCONNECT	This is macro AVDTP_TC_OPCODE_DISCONNECT.
	AVDTP_TC_STATUS_ERROR	This is macro AVDTP_TC_STATUS_ERROR.
	AVDTP_TC_STATUS_SUCCESS	This is macro AVDTP_TC_STATUS_SUCCESS.

Types

	Name	Description
	bt_avdtp_transport_op_callback_fp	This is type bt_avdtp_transport_op_callback_fp.

Description

Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.

avrcp.h

Functions

	Name	Description
	bt_avrcp_add_to_now_playing	brief Add to "now playing" list. ingroup avrcp details This function adds a media element specified by p element_id to the "now playing" list on the target. param channel AVRCP channel. param scope The scope in which the p element_id is valid. This value can be on the following values: li AVC_SCOPE_MEDIA_PLAYER_LIST li AVC_MEDIA_PLAYER_VIRTUAL_FILESYSTEM li AVC_SEARCH li AVC_NOW_PLAYING param element_id UID of the media element to be added to the "now playing" list. param counter UID counter. return li c TRUE if the function succeeds. li c FALSE otherwise.
	bt_avrcp_cancel_find	brief Cancel finding Targets ingroup avrcp details This function stops AVRCP layer looking for targets on nearby devices. As a result of this operation the AVRCP_EVT_SEARCH_COMPLETED event will be generated. return li c TRUE if the function succeeds. li c FALSE otherwise. the callback is not called in this case.
	bt_avrcp_cancel_listen	brief Cancel listening for incoming connections. ingroup avrcp details This function stops listening for incoming connections on a specified channel. param channel AVRCP channel. return li c TRUE if the function succeeds. li c FALSE otherwise.
	bt_avrcp_connect	brief Connect to a remote device. ingroup avrcp details This function establishes a connection to a remote device specified by the p remote_address. If connection cannot be initiated for some reason, for example, there is not enough resources, it returns FALSE and no events are generated. Otherwise the result of an attempt to connect to the remote device is reported via the AVRCP callback. The events generated will either be AVRCP_EVT_CONTROL_CHANNEL_CONNECTED or AVRCP_EVT_CONTROL_CONNECTION_FAILED. param channel AVRCP channel. param remote_address The address of a remote device. return li c TRUE if connection establishment has been started. li c FALSE otherwise.... more

	bt_avrcp_create_channel	brief Allocate AVRCP channel ingroup avrcp details This function allocates a new incoming AVRCP channel. The channel is intended to be used to accept a connection from a remote device. There can be only one incoming channel. param mgr AVRCP manager. param create_browsing_channel Defines weather a browsing channel will be created. return li A pointer to the new AVRCP channel if the function succeeds. li c NULL otherwise.
	bt_avrcp_create_outgoing_channel	brief Allocate AVRCP channel ingroup avrcp details This function allocates a new outgoing AVRCP channel. The channel is intended to be used to create a connection to a remote device. There can be multiple outgoing channels. param mgr AVRCP manager. param create_browsing_channel Defines weather a browsing channel will be created. return li A pointer to the new AVRCP channel if the function succeeds. li c NULL otherwise.
	bt_avrcp_destroy_channel	brief Destroy AVRCP channel. ingroup avrcp details This function frees memory used by the channel. The channel has to exist and be in the "idle" state for this function to succeed. I.e. the channel has to be disconnected before this function can be called. param channel AVRCP channel. return li c TRUE if the function succeeds. li c FALSE otherwise.
	bt_avrcp_disconnect	brief Disconnect from a remote device. ingroup avrcp details This function closes a connection to a remote device. param channel AVRCP channel. return li c TRUE if disconnection has been started. li c FALSE otherwise. No events will be generated.
	bt_avrcp_find_controller	brief Find Controller ingroup avrcp details This function looks for a controller on a remote device specified by p deviceAddress. param deviceAddress The address of a remote device. param callback The callback function that will be called when search has completed. param client_callback The optional callback function that an application can set if it wants to be notified of state changes of the SDP client. The c evt parameter of the callback can be one of the following values: li SDP_CLIENT_STATE_IDLE li SDP_CLIENT_STATE_CONNECTING li SDP_CLIENT_STATE_DISCONNECTING li SDP_CLIENT_STATE_CONNECTED param callback_param A pointer to arbitrary data to be passed to the p... more
	bt_avrcp_find_target	brief Find Target ingroup avrcp details This function looks for a target on a remote device specified by p deviceAddress. param deviceAddress The address of a remote device. param callback The callback function that will be called when search has completed. param client_callback The optional callback function that an application can set if it wants to be notified of state changes of the SDP client. The c evt parameter of the callback can be one of the following values: li SDP_CLIENT_STATE_IDLE li SDP_CLIENT_STATE_CONNECTING li SDP_CLIENT_STATE_DISCONNECTING li SDP_CLIENT_STATE_CONNECTED param callback_param A pointer to arbitrary data to be passed to the p... more
	bt_avrcp_find_targets	brief Find Targets ingroup avrcp details This function looks for targets on nearby devices. The AVRCP_EVT_SEARCH_COMPLETED event is generated when the search has completed. param search_length The amount of time the search will be performed for. return li c TRUE if the function succeeds. li c FALSE otherwise. the callback is not called in this case.
	bt_avrcp_get_browsing_channel_state	brief Get AVCTP browsing channel state. ingroup avrcp details This function returns status of the AVCTP browsing channel. param channel AVRCP channel. return Returns of the following values: li AVCTP_CHANNEL_STATE_FREE li AVCTP_CHANNEL_STATE_IDLE li AVCTP_CHANNEL_STATE_CONNECTING li AVCTP_CHANNEL_STATE_CONNECTED li AVCTP_CHANNEL_STATE_DISCONNECTING

	bt_avrcp_get_channel_remote_address	brief Get channel's remote BT address. ingroup avrcp details This function returns the address of the remote device associated with the channel. param channel AVRCP channel. return li The address of the remote device if channel is connected. li NULL otherwise.
	bt_avrcp_get_company_id_list	brief Get Company ID list. ingroup avrcp details This function requests a list of company id's supported by the remote device param channel AVRCP channel. return li c TRUE if the function succeeds. li c FALSE otherwise.
	bt_avrcp_get_control_channel_state	brief Get AVCTP control channel state. ingroup avrcp details This function returns status of the AVCTP control channel. param channel AVRCP channel. return Returns of the following values: li AVCTP_CHANNEL_STATE_FREE li AVCTP_CHANNEL_STATE_IDLE li AVCTP_CHANNEL_STATE_CONNECTING li AVCTP_CHANNEL_STATE_CONNECTED li AVCTP_CHANNEL_STATE_DISCONNECTING
	bt_avrcp_get_current_player_application_setting_value	brief Get current player setting values. ingroup avrcp details This function requests a list of current set values for the player application on the target. The list of attribute ids whose values have to be returned is passed via the p response_buffer parameter. The caller has to set bt_av_player_setting_current_values_t::setting_id_list to a list of player setting attribute ids, bt_av_player_setting_current_values_t::count to the number of entries in the list, bt_av_player_setting_current_values_t::setting_value_id_list to a buffer where returned values will be stored. param channel AVRCP channel. param response_buffer Pointer to bt_av_player_setting_current_values_t structure initialized as stated above. return li c TRUE if the function succeeds. li c... more
	bt_avrcp_get_element_attributes	brief Get media element attributes. ingroup avrcp details This function requests the attributes of the element specified with p element_id. note Currently p element_id is ignored. The AVRCP specification defines that only UID 0 can be used to return the attributes of the current track. param channel AVRCP channel. param element_id UID of the media element whose attributes are requested. param attr_mask Bitmask that defines which attributes are requested. This value can be a combination of the following values: li AVC_MEDIA_ATTR_FLAG_TITLE li AVC_MEDIA_ATTR_FLAG_ARTIST li AVC_MEDIA_ATTR_FLAG_ALBUM li AVC_MEDIA_ATTR_FLAG_NUMBER li AVC_MEDIA_ATTR_FLAG_TOTAL_NUMBER li AVC_MEDIA_ATTR_FLAG_GENRE li AVC_MEDIA_ATTR_FLAG_PLAYING_TIME li AVC_MEDIA_ATTR_FLAG_ALL return li c TRUE if... more
	bt_avrcp_get_hci_connection	brief Get HCI connection for a channel ingroup avrcp details This function returns a pointer to a structure that describes an HCI connection a channel is open on. The return value can be used to call various function from the HCI layer. For example, if an app wants to force disconnection from a remote device it can call ::bt_hci_disconnect. param channel AVRCP channel. return li c Pointer to a structure that describes an HCI connection if the function succeeds. li c NULL otherwise. The function fails only if a channel specified by the p channel parameter li does... more
	bt_avrcp_get_mgr	brief Return a pointer to an instance of the AVRCP manager. ingroup avrcp details This function returns a pointer to an instance of the AVRCP manager. There is only one instance of the manager allocated by the stack.

	bt_avrcp_get_play_status	brief Get playback status. ingroup avrcp details This function requests the status of the currently playing media at the target. param channel AVRCP channel. param repeat_interval Interval in milliseconds at which AVRCP polls the target for playback status. If 0 is passed polling is stopped. return li c TRUE if the function succeeds. li c FALSE otherwise.
	bt_avrcp_get_player_application_setting_attr_text	brief Get player setting attribute text. ingroup avrcp details This function requests the target device to provide supported player application setting attribute displayable text for the provided player application setting attributes. The list of attribute ids whose displayable text have to be returned is passed via the p response_buffer parameter. The caller has to set bt_av_player_settings_text_t::setting_id_list to a list of player setting attribute ids, bt_av_player_settings_text_t::count to the number of entries in the list, bt_av_player_settings_text_t::setting_text_list to a buffer where returned values will be stored. param channel AVRCP channel. param response_buffer Pointer to bt_av_player_settings_text_t structure initialized as stated above. return li c... more
	bt_avrcp_get_player_application_setting_value_text	brief Get player setting value text. ingroup avrcp details This function request the target device to provide target supported player application setting value displayable text for the provided player application setting attribute values. The list of attribute ids whose value displayable text have to be returned is passed via the p response_buffer parameter. The caller has to set bt_av_player_setting_values_text_t::setting_value_id_list to a list of player setting attribute value ids, bt_av_player_setting_values_text_t::count to the number of entries in the list, bt_av_player_setting_values_text_t::setting_value_text_list to a buffer where returned values will be stored. param channel AVRCP channel. param response_buffer Pointer to bt_av_player_setting_values_text_t structure initialized as stated... more
	bt_avrcp_get_supported_event_id_list	brief Get supported events. ingroup avrcp details This function requests a list of events supported by the remote device param channel AVRCP channel. return li c TRUE if the function succeeds. li c FALSE otherwise.
	bt_avrcp_inform_battery_status	brief Inform controller's battery status. ingroup avrcp details This function is used to inform the target about the controller's battery status. param channel AVRCP channel. param status Battery status. This can be one of the following values: li AVC_BATTERY_STATUS_NORMAL li AVC_BATTERY_STATUS_WARNING li AVC_BATTERY_STATUS_CRITICAL li AVC_BATTERY_STATUS_EXTERNAL li AVC_BATTERY_STATUS_FULL_CHARGE return li c TRUE if the function succeeds. li c FALSE otherwise.
	bt_avrcp_inform_displayable_character_set	brief Inform displayable character set. ingroup avrcp details This function informs the list of character set supported by the controller to the target. param channel AVRCP channel. param charset_list List of displayable character sets. param charset_count Number of entries in the list. return li c TRUE if the function succeeds. li c FALSE otherwise.
	bt_avrcp_init_controller	brief Initialize AVRCP to be used in controller mode. ingroup avrcp details This function initializes the AVRCP layer of the stack in controller mode. It must be called prior to any other AVRCP function can be called.

	bt_avrcp_init_target	brief Initialize AVRCP to be used in target mode. ingroup avrcp details This function initializes the AVRCP layer of the stack in target mode. It must be called prior to any other AVRCP function can be called. param company_id The 24-bit unique ID obtained from the IEEE Registration Authority Committee. If the vendor of a TG device does not have the unique ID, the value 0xFFFFFF may be used. param supported_events Bitmask that specifies events supported by the target. This value can be a combination of the following values: li AVC_EVENT_FLAG_PLAYBACK_STATUS_CHANGED li AVC_EVENT_FLAG_TRACK_CHANGED li AVC_EVENT_FLAG_TRACK_REACHED_END li AVC_EVENT_FLAG_TRACK_REACHED_START li AVC_EVENT_FLAG_PLAYBACK_POS_CHANGED li... more
	bt_avrcp_list_player_application_setting_attributes	brief Get supported player setting attributes. ingroup avrcp details This function request the target device to provide target supported player application setting attributes. The list of attribute ids is stored in the setting_id_list member of the p response_buffer parameter. The caller has to set bt_av_player_settings_t::setting_id_list to a buffer where returned values will be stored and bt_av_player_settings_t::count to the number of entries in the list. param channel AVRCP channel. param response_buffer Pointer to bt_av_player_settings_t structure initialized as stated above. return li c TRUE if the function succeeds. li c FALSE otherwise.
	bt_avrcp_list_player_application_setting_values	brief Get player setting attribute values. ingroup avrcp details This function requests the target device to list the set of possible values for the requested player application setting attribute. The list of attribute value ids is stored in the setting_value_id_list member of the p response_buffer parameter. The caller has to set bt_av_player_setting_values_t::setting_value_id_list to a buffer where returned values will be stored and bt_av_player_setting_values_t::count to the number of entries in the list. param channel AVRCP channel. param response_buffer Pointer to bt_av_player_setting_values_t structure initialized as stated above. return li c TRUE if the function succeeds. li c FALSE otherwise.
	bt_avrcp_listen	brief Listen for incoming connections. ingroup avrcp details This function enables incoming connections on the specified AVRCP channel. param channel AVRCP channel. return li c TRUE if the function succeeds. li c FALSE otherwise.
	bt_avrcp_play_item	brief Play media item. ingroup avrcp details This function starts playing an item indicated by the UID. param channel AVRCP channel. param scope The scope in which the p element_id is valid. This value can be on the following values: li AVC_SCOPE_MEDIA_PLAYER_LIST li AVC_MEDIA_PLAYER_VIRTUAL_FILESYSTEM li AVC_SEARCH li AVC_NOW_PLAYING param element_id UID of the media element to be played. param counter UID counter. return li c TRUE if the function succeeds. li c FALSE otherwise.
	bt_avrcp_register_notification	brief Register notification. ingroup avrcp details This function registers with the target to receive notifications asynchronously based on specific events occurring. param channel AVRCP channel. param event_id Event Id. This value can be one of the following values: li AVC_EVENT_PLAYBACK_STATUS_CHANGED li AVC_EVENT_TRACK_CHANGED li AVC_EVENT_TRACK_REACHED_END li AVC_EVENT_TRACK_REACHED_START li AVC_EVENT_PLAYBACK_POS_CHANGED li AVC_EVENT_BATT_STATUS_CHANGED li AVC_EVENT_SYSTEM_STATUS_CHANGED li AVC_EVENT_PLAYER_APPLICATION_SETTING_CHANGED li AVC_EVENT_NOW_PLAYING_CONTENT_CHANGED li AVC_EVENT_AVAILABLE_PLAYERS_CHANGED li AVC_EVENT_ADDRESSED_PLAYER_CHANGED li AVC_EVENT_UIDS_CHANGED li AVC_EVENT_VOLUME_CHANGED param playback_interval The time interval (in seconds) at which the change in playback position will be notified. Applicable for AVC_EVENT_PLAYBACK_POS_CHANGED event. return li c TRUE if the function succeeds. li c FALSE otherwise.

	bt_avrcp_register_notifications	brief Register notifications. ingroup avrcp details This function registers with the target to receive notifications asynchronously based on specific events occurring. This function is used to register multiple notifications with one call. note This function cannot be used to register for AVC_EVENT_PLAYBACK_POS_CHANGED event. param channel AVRCP channel. param event_mask Bitmask that specifies which events to register for. This value can be a combination of the following values: li AVC_EVENT_FLAG_PLAYBACK_STATUS_CHANGED li AVC_EVENT_FLAG_TRACK_CHANGED li AVC_EVENT_FLAG_TRACK_REACHED_END li AVC_EVENT_FLAG_TRACK_REACHED_START li AVC_EVENT_FLAG_BATT_STATUS_CHANGED li AVC_EVENT_FLAG_SYSTEM_STATUS_CHANGED li AVC_EVENT_FLAG_PLAYER_APPLICATION_SETTING_CHANGED li AVC_EVENT_FLAG_NOW_PLAYING_CONTENT_CHANGED li AVC_EVENT_FLAG_AVAILABLE_PLAYERS_CHANGED li AVC_EVENT_FLAG_ADDRESSED_PLAYER_CHANGED li AVC_EVENT_FLAG_UIDS_CHANGED li AVC_EVENT_FLAG_VOLUME_CHANGED li AVC_EVENT_FLAG_ALL return li c TRUE if the function... more
	bt_avrcp_send_cmd	brief Send AVRCP command ingroup avrcp details This function sends a command to the remote device. param channel AVRCP channel. param command Command to be sent. return li c TRUE if the function succeeds. li c FALSE otherwise.
	bt_avrcp_set_absolute_volume	brief Set absolute volume. ingroup avrcp details This function is used to set an absolute volume to be used by the rendering device. param channel AVRCP channel. param volume Volume return li c TRUE if the function succeeds. li c FALSE otherwise.
	bt_avrcp_set_addressed_player	brief Set addressed player. ingroup avrcp details This function is used to inform the target of which media player the controller wishes to control. param channel AVRCP channel. param player_id Player Id. return li c TRUE if the function succeeds. li c FALSE otherwise.
	bt_avrcp_set_player_application_setting_value	brief Set player setting attribute values. ingroup avrcp details This function requests to set the player application setting list of player application setting values on the target device for the corresponding defined list of setting attributes. for the requested player application setting attribute. The list of attribute value ids is stored in the setting_value_id_list member of the p response_buffer parameter. The caller has to set bt_av_player_setting_values_t::setting_value_id_list to a buffer where returned values will be stored and bt_av_player_setting_values_t::count to the number of entries in the list. param channel AVRCP channel. param attr_id_list List of setting attribute ids. param attr_value_list List of... more
	bt_avrcp_start	brief Start the AVRCP layer. ingroup avrcp details In order to be notified of various events a consumer of the AVRCP layer has to register a callback function. The stack will call the callback function whenever a new event has been generated passing the code of the event as the second parameter. bt_avrcp_start() stores pointers to the c callback and c callback_param in the bt_avrcp_mgr_t structure. param mgr AVRCP manager. param callback The callback function that will be called when the AVRCP generates an event. param callback_param A pointer to arbitrary data to be passed to the c callback callback.... more
	bt_avrcp_tg_send_element_attributes	brief Send media element attributes ingroup avrcp details This function is used to send the system status when AVRCP is running in target mode. If there are active connections with controllers, the ones that registered for AVC_EVENT_SYSTEM_STATUS_CHANGED event will be notified. param status Battery status

	bt_avrcp_tg_set_absolute_volume	brief Set absolute volume ingroup avrcp details This function is used to set the absolute volume when AVRCP is running in target mode. If there are active connections with controllers, the ones that registered for AVC_EVENT_VOLUME_CHANGED event will be notified. param track_id Track Id param song_length The length of the current track. param song_position The position of the current track.
	bt_avrcp_tg_set_battery_status	brief Set battery status ingroup avrcp details This function is used to set the battery status when AVRCP is running in target mode. If there are active connections with controllers, the ones that registered for AVC_EVENT_BATT_STATUS_CHANGED event will be notified. param status Battery status
	bt_avrcp_tg_set_channel_absolute_volume	This is function bt_avrcp_tg_set_channel_absolute_volume.
	bt_avrcp_tg_set_current_track	brief Set current track ingroup avrcp details This function is used to set the current track when AVRCP is running in target mode. If there are active connections with controllers, the ones that registered for AVC_EVENT_TRACK_CHANGED event will be notified. param track_id Track Id param song_length The length of the current track. param song_position The position of the current track.
	bt_avrcp_tg_set_play_status	local target control * • brief Set playback status • ingroup avrcp * • details This function is used to set the playback status when AVRCP is running in target mode. • If there are active connections with controllers, the ones that registered • for AVC_EVENT_PLAYBACK_STATUS_CHANGED event will be notified. * • param song_length The length of the current track. • param song_position The position of the current track. • param play_status Playback status. This value can be one of the following values: • li AVC_PLAY_STATUS_STOPPED • li AVC_PLAY_STATUS_PLAYING • li AVC_PLAY_STATUS_PAUSED • li AVC_PLAY_STATUS_FW_SEEK • li AVC_PLAY_STATUS_REV_SEEK
	bt_avrcp_tg_set_system_status	brief Set system status ingroup avrcp details This function is used to set the system status when AVRCP is running in target mode. If there are active connections with controllers, the ones that registered for AVC_EVENT_SYSTEM_STATUS_CHANGED event will be notified. param status Battery status

Macros

	Name	Description
	__AVRCP_H	This is macro __AVRCP_H.
	AVCTP_ERROR_BAD_STATE	This is macro AVCTP_ERROR_BAD_STATE.
	AVCTP_ERROR_SUCCESS	This is macro AVCTP_ERROR_SUCCESS.
	AVCTP_MESSAGE_PACKET_TYPE_CONTINUE	This is macro AVCTP_MESSAGE_PACKET_TYPE_CONTINUE.
	AVCTP_MESSAGE_PACKET_TYPE_END	This is macro AVCTP_MESSAGE_PACKET_TYPE_END.
	AVCTP_MESSAGE_PACKET_TYPE_SINGLE	This is macro AVCTP_MESSAGE_PACKET_TYPE_SINGLE.
	AVCTP_MESSAGE_PACKET_TYPE_START	This is macro AVCTP_MESSAGE_PACKET_TYPE_START.
	AVCTP_MESSAGE_TYPE_COMMAND	This is macro AVCTP_MESSAGE_TYPE_COMMAND.

	AVCTP_MESSAGE_TYPE_RESPONSE	This is macro AVCTP_MESSAGE_TYPE_RESPONSE.
	AVRCP_BTSIG_COMPANY_ID	This is macro AVRCP_BTSIG_COMPANY_ID.
	AVRCP_CHANNEL_FLAG_LISTENING	This is macro AVRCP_CHANNEL_FLAG_LISTENING.
	AVRCP_CHANNEL_FLAG_PLAY_STATUS_REQUESTED	This is macro AVRCP_CHANNEL_FLAG_PLAY_STATUS_REQUESTED.
	AVRCP_CHANNEL_FLAG_REGISTERING_NOTIFICATIONS	This is macro AVRCP_CHANNEL_FLAG_REGISTERING_NOTIFICATIONS.
	AVRCP_CHANNEL_FLAG_SENDING	This is macro AVRCP_CHANNEL_FLAG_SENDING.
	AVRCP_CHANNEL_STATE_CONNECTED	This is macro AVRCP_CHANNEL_STATE_CONNECTED.
	AVRCP_CHANNEL_STATE_CONNECTING	This is macro AVRCP_CHANNEL_STATE_CONNECTING.
	AVRCP_CHANNEL_STATE_DISCONNECTING	This is macro AVRCP_CHANNEL_STATE_DISCONNECTING.
	AVRCP_CHANNEL_STATE_FREE	This is macro AVRCP_CHANNEL_STATE_FREE.
	AVRCP_CHANNEL_STATE_IDLE	This is macro AVRCP_CHANNEL_STATE_IDLE.
	AVRCP_EVT_ADD_TO_NOW_PLAYING_COMPLETED	< This event is generated when a local device received a response to a "add to now playing" request.
	AVRCP_EVT_ADDRESSED_PLAYER_CHANGED	< This event is generated when a local device received a "addressed player changed" notification.
	AVRCP_EVT_AVAILABLE_PLAYERS_CHANGED	< This event is generated when a local device received a "available players changed" notification.
	AVRCP_EVT_BATT_STATUS_CHANGED	< This event is generated when a local device received a "battery status changed" notification.
	AVRCP_EVT_BATTERY_STATUS_OF_CT_RECEIVED	< This event is generated when a local device received a "battery status of controller" command.
	AVRCP_EVT_BROWSING_CHANNEL_CONNECTED	< This event is generated when a browsing channel between two AVRCP entities has been established.
	AVRCP_EVT_BROWSING_CHANNEL_DISCONNECTED	< This event is generated when a browsing channel between two AVRCP entities has been terminated.
	AVRCP_EVT_BROWSING_CONNECTION_FAILED	< This event is generated when a local device failed to create a browsing channel between two AVRCP entities.
	AVRCP_EVT_COMPANY_ID_LIST_RECEIVED	< This event is generated when a local device received a response to a "get company id" request.
	AVRCP_EVT_CONTROL_CHANNEL_CONNECTED	< This event is generated when a control channel between two AVRCP entities has been established.
	AVRCP_EVT_CONTROL_CHANNEL_DISCONNECTED	< This event is generated when a control channel between two AVRCP entities has been terminated.
	AVRCP_EVT_CONTROL_CONNECTION_FAILED	< This event is generated when a local device failed to create a control channel between two AVRCP entities.
	AVRCP_EVT_DISPLAYABLE_CHARACTER_SET_RECEIVED	< This event is generated when a local device received a "displayable chracter set command" request.
	AVRCP_EVT_ELEMENT_ATTRIBUTES_REQUESTED	< This event is generated when a local device received a "get element attributes" request.
	AVRCP_EVT_EVENT_ID_LIST_RECEIVED	< This event is generated when a local device received a response to a "get supported events" request.
	AVRCP_EVT_GET_ELEMENT_ATTRIBUTES_RECEIVED	< This event is generated when a local device received a response to a "get media element attributes" request.
	AVRCP_EVT_GET_PLAY_STATUS_RECEIVED	< This event is generated when a local device received a response to a "get play status" request.
	AVRCP_EVT_INFORM_BATTERY_STATUS_OF_CT_COMPLETED	< This event is generated when a local device received a response to a "inform battery status" request.

	AVRCP_EVT_INFORM_DISPLAYABLE_CHARACTER_SET_COMPLETED	< This event is generated when a local device received a response to a "inform displayable character set" request.
	AVRCP_EVT_NOTHING	addtogroup avrcp @{ @name Events details The following is a list of events AVRCP layer generates and can report to the upper layer when it completes executing an operation initiated by either local or remote device.
	AVRCP_EVT_NOW_PLAYING_CONTENT_CHANGED	< This event is generated when a local device received a "now playing content changed" notification.
	AVRCP_EVT_PANEL_COMMAND_RECEIVED	< This event is generated when a local device received a PASS THROUGH command.
	AVRCP_EVT_PANEL_RESPONSE_RECEIVED	< This event is generated when a local device received a response to a PASS THROUGH command.
	AVRCP_EVT_PLAY_ITEM_COMPLETED	< This event is generated when a local device received a response to a "play item" request.
	AVRCP_EVT_PLAYBACK_POS_CHANGED	< This event is generated when a local device received a "playback position changed" notification.
	AVRCP_EVT_PLAYBACK_STATUS_CHANGED	< This event is generated when a local device received a "play status changed" notification.
	AVRCP_EVT_PLAYER_APPLICATION_SETTING_CHANGED	< This event is generated when a local device received a "player application setting changed" notification.
	AVRCP_EVT_PLAYER_CURRENT_SETTING_VALUES_RECEIVED	< This event is generated when a local device received a response to a "get current player setting attribute values" request.
	AVRCP_EVT_PLAYER_SETTING_ATTRIBUTES_RECEIVED	< This event is generated when a local device received a response to a "get supported player setting attributes" request.
	AVRCP_EVT_PLAYER_SETTING_ATTRIBUTES_TEXT_RECEIVED	< This event is generated when a local device received a response to a "get player setting attributes displayable text" request.
	AVRCP_EVT_PLAYER_SETTING_VALUES_RECEIVED	< This event is generated when a local device received a response to a "get player setting attribute values" request.
	AVRCP_EVT_PLAYER_SETTING_VALUES_TEXT_RECEIVED	< This event is generated when a local device received a response to a "get player setting attribute values displayable text" request.
	AVRCP_EVT_REGISTER_NOTIFICATION_REQUESTED	< This event is generated when a local device received a "register notification" request.
	AVRCP_EVT_REGISTER_NOTIFICATIONS_COMPLETED	< This event is generated when a local device received a response to a "register notification" request.
	AVRCP_EVT_SEARCH_COMPLETED	< This event is generated when a local device completed searching for nearby targets.
	AVRCP_EVT_SET_ABSOLUTE_VOLUME_COMPLETED	< This event is generated when a local device received a response to a "set absolute volume" request.
	AVRCP_EVT_SET_ABSOLUTE_VOLUME_REQUESTED	< This event is generated when a local device received a "set absolute volume" request.
	AVRCP_EVT_SET_ADDRESSED_PLAYER_COMPLETED	< This event is generated when a local device received a response to a "set addressed player" request.
	AVRCP_EVT_SET_PLAYER_SETTING_VALUES_COMPLETED	< This event is generated when a local device received a response to a "set player setting attribute values" request.
	AVRCP_EVT_SYSTEM_STATUS_CHANGED	< This event is generated when a local device received a "system status changed" notification.
	AVRCP_EVT_TRACK_CHANGED	< This event is generated when a local device received a "track changed changed" notification.
	AVRCP_EVT_TRACK_REACHED_END	< This event is generated when a local device received a "track reached end" notification.

	AVRCP_EVT_TRACK_REACHED_START	< This event is generated when a local device received a "track reached start" notification.
	AVRCP_EVT_UIDS_CHANGED	< This event is generated when a local device received a "UIDs changed" notification.
	AVRCP_EVT_VOLUME_CHANGED	< This event is generated when a local device received a "volume changed" notification.
	AVRCP_MANAGER_FLAG_SEARCHING	This is macro AVRCP_MANAGER_FLAG_SEARCHING.
	AVRCP_MANAGER_STATE_IDLE	This is macro AVRCP_MANAGER_STATE_IDLE.

Structures

	Name	Description
	_bt_avrcp_channel_t	brief AVRCP channel description ingroup avrcp details This structure is used to hold information about an AVRCP channel.
	_bt_avrcp_device_s	This is type bt_avrcp_device_t.
	_bt_avrcp_evt_channel_connected_t	brief Parameter to AVRCP_EVT_CONTROL_CHANNEL_CONNECTED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::channel_connected - when a control channel between two devices has been established.
	_bt_avrcp_evt_channel_disconnected_t	brief Parameter to AVRCP_EVT_CONTROL_CHANNEL_DISCONNECTED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::channel_disconnected - when a control channel between two devices has been terminated.
	_bt_avrcp_evt_connection_failed_t	brief Parameter to AVRCP_EVT_CONTROL_CONNECTION_FAILED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::connection_failed - when a local device failed to create a control channel between two AVRCP entities.
	_bt_avrcp_evt_panel_command_received_t	brief Parameter to AVRCP_EVT_PANEL_COMMAND_RECEIVED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::panel_command_received - when a local device received a PASS THROUGH command.
	_bt_avrcp_evt_panel_response_received_t	brief Parameter to AVRCP_EVT_PANEL_RESPONSE_RECEIVED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::panel_response_received - when a local device received a response to a PASS THROUGH command.
	_bt_avrcp_evt_register_events_completed_t	This is type bt_avrcp_evt_register_events_completed_t.
	_bt_avrcp_evt_search_completed_s	brief Parameter to AVRCP_EVT_SEARCH_COMPLETED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::device_search - when searching for nearby devices has finished.
	_bt_avrcp_mgr_t	brief AVRCP manager. ingroup avrcp details A structure that glues all pieces together. There is only one instance of this structure allocated by dotstack. A pointer to the instance can be get with c bt_avrcp_get_mgr().
	bt_avrcp_device_t	This is type bt_avrcp_device_t.
	bt_avrcp_evt_channel_connected_t	brief Parameter to AVRCP_EVT_CONTROL_CHANNEL_CONNECTED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::channel_connected - when a control channel between two devices has been established.
	bt_avrcp_evt_channel_disconnected_t	brief Parameter to AVRCP_EVT_CONTROL_CHANNEL_DISCONNECTED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::channel_disconnected - when a control channel between two devices has been terminated.

	bt_avrcp_evt_connection_failed_t	brief Parameter to AVRCP_EVT_CONTROL_CONNECTION_FAILED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::connection_failed - when a local device failed to create a control channel between two AVRCP entities.
	bt_avrcp_evt_panel_command_received_t	brief Parameter to AVRCP_EVT_PANEL_COMMAND_RECEIVED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::panel_command_received - when a local device received a PASS THROUGH command.
	bt_avrcp_evt_panel_response_received_t	brief Parameter to AVRCP_EVT_PANEL_RESPONSE_RECEIVED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::panel_response_received - when a local device received a response to a PASS THROUGH command.
	bt_avrcp_evt_register_events_completed_t	This is type bt_avrcp_evt_register_events_completed_t.
	bt_avrcp_evt_search_completed_t	brief Parameter to AVRCP_EVT_SEARCH_COMPLETED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::device_search - when searching for nearby devices has finished.

Types

	Name	Description
	bt_avrcp_channel_t	This is type bt_avrcp_channel_t.
	bt_avrcp_find_callback_fp	brief Find Controller/Target callback. ingroup avrcp details This callback is called when search for Controller/Target has finished. param found This can be one of the following values: li c TRUE if Controller/Target was found. li c FALSE otherwise. param callback_param A pointer to an arbitrary data set by a call to bt_avrcp_find_target/bt_avrcp_find_controller.
	bt_avrcp_mgr_callback_fp	brief AVRCP application callback. ingroup avrcp details In order to be notified of various events a consumer of the AVRCP layer has to register a callback function (done with bt_avrcp_start()). The stack will call that function whenever a new event has been generated. param mgr AVRCP manager. param evt AVRCP event. The event can be one of the following values: @arg AVRCP_EVT_CONTROL_CHANNEL_CONNECTED This event is generated when a control channel between two AVRCP entities has been established. @arg AVRCP_EVT_CONTROL_CHANNEL_DISCONNECTED This event is generated when a control channel between two AVRCP entities has been terminated. @arg AVRCP_EVT_CONTROL_CONNECTION_FAILED This event is... more
	bt_avrcp_mgr_t	This is type bt_avrcp_mgr_t.

Unions

	Name	Description
	_bt_avrcp_event_u	brief Parameter to an application callback. ingroup avrcp details This union is used to pass event specific data to the AVRCP consumer. Which member of the union points to a valid structure depends on the event reported to the consumer. In general, each event has a corresponding member in the union.
	bt_avrcp_event_t	brief Parameter to an application callback. ingroup avrcp details This union is used to pass event specific data to the AVRCP consumer. Which member of the union points to a valid structure depends on the event reported to the consumer. In general, each event has a corresponding member in the union.

Description

Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.

avrcp_command.h

Functions

	Name	Description
	bt_avrcp_get_subuint_info	brief Get subunit info ingroup avrcp details This function is used to request subunit info from the target. param channel AVRCP channel. return li c TRUE if the function succeeds. li c FALSE otherwise.
	bt_avrcp_get_unit_info	brief Get unit info ingroup avrcp details This function is used to request unit info from the target. param channel AVRCP channel. return li c TRUE if the function succeeds. li c FALSE otherwise.
	bt_avrcp_send_button_click	brief Send AV/C Panel Subunit "click" PASS THROUGH command ingroup avrcp details This function is used to send a button click. Two PATH THROUGTH commands are sent. The first command has button state set to AVC_PANEL_BUTTON_PRESSED. The second command gas button state set to AVC_PANEL_BUTTON_RELEASED param channel AVRCP channel. param button_id Operation Id. This value can be on of the AVC_PANEL_OPID_... constants return li c TRUE if the function succeeds. li c FALSE otherwise.
	bt_avrcp_send_panel_control	brief Send AV/C Panel Subunit "control" PASS THROUGH command ingroup avrcp details This function is used to send AV/C Panel Subunit PASS THROUGH command with command type set to AVC_CTYPE_CONTROL. param channel AVRCP channel. param opid Operation Id. This value can be on of the AVC_PANEL_OPID_... constants param button_state Button state. This can be on of the following values: li AVC_PANEL_BUTTON_PRESSED li AVC_PANEL_BUTTON_RELEASED return li c TRUE if the function succeeds. li c FALSE otherwise.
	bt_avrcp_send_press_panel_control	brief Send AV/C Panel Subunit "pressed" PASS THROUGH command ingroup avrcp details This function is used to send AV/C Panel Subunit PASS THROUGH command with button state set to AVC_PANEL_BUTTON_PRESSED. param channel AVRCP channel. param opid Operation Id. This value can be on of the AVC_PANEL_OPID_... constants return li c TRUE if the function succeeds. li c FALSE otherwise.
	bt_avrcp_send_release_panel_control	brief Send AV/C Panel Subunit "released" PASS THROUGH command ingroup avrcp details This function is used to send AV/C Panel Subunit PASS THROUGH command with button state set to AVC_PANEL_BUTTON_RELEASED. param channel AVRCP channel. param opid Operation Id. This value can be on of the AVC_PANEL_OPID_... constants return li c TRUE if the function succeeds. li c FALSE otherwise.
	bt_avrcp_send_simple_panel_cmd	brief Send AV/C Panel Subunit PASS THROUGH command ingroup avrcp details This function is used to send AV/C Panel Subunit PASS THROUGH command to the target. param channel AVRCP channel. param ctype Command type. This value can be on of the following values: li AVC_CTYPE_CONTROL 0 li AVC_CTYPE_STATUS 1 li AVC_CTYPE_SPECIFIC_IQUIRY 2 li AVC_CTYPE_NOTIFY 3 li AVC_CTYPE_GENERAL_INQUORY 4 param opid Operation Id. This value can be on of the AVC_PANEL_OPID_... constants param button_state Button state. This can be on of the following values: li AVC_PANEL_BUTTON_PRESSED li AVC_PANEL_BUTTON_RELEASED return li c TRUE if the function succeeds. li c FALSE otherwise.

Macros

	Name	Description
	__AVRCP_COMMAND_H	This is macro __AVRCP_COMMAND_H.
	AVC_BATTERY_STATUS_CRITICAL	This is macro AVC_BATTERY_STATUS_CRITICAL.
	AVC_BATTERY_STATUS_EXTERNAL	This is macro AVC_BATTERY_STATUS_EXTERNAL.
	AVC_BATTERY_STATUS_FULL_CHARGE	This is macro AVC_BATTERY_STATUS_FULL_CHARGE.
	AVC_BATTERY_STATUS_NORMAL	• addtogroup avrcp • @{ • @name Battery status

	AVC_BATTERY_STATUS_WARNING	This is macro AVC_BATTERY_STATUS_WARNING.
	AVC_CAPABILITY_COMPANY_ID	This is macro AVC_CAPABILITY_COMPANY_ID.
	AVC_CAPABILITY_EVENTS_SUPPORTED	This is macro AVC_CAPABILITY_EVENTS_SUPPORTED.
	AVC_CMD_GENERAL_POWER	This is macro AVC_CMD_GENERAL_POWER.
	AVC_CMD_GENERAL_RESERVE	This is macro AVC_CMD_GENERAL_RESERVE.
	AVC_CMD_GENERAL_SUBUNIT_INFO	This is macro AVC_CMD_GENERAL_SUBUNIT_INFO.
	AVC_CMD_GENERAL_UNIT_INFO	This is macro AVC_CMD_GENERAL_UNIT_INFO.
	AVC_CMD_GENERAL_VENDOR_DEPENDENT	This is macro AVC_CMD_GENERAL_VENDOR_DEPENDENT.
	AVC_CMD_GENERAL_VERSION	This is macro AVC_CMD_GENERAL_VERSION.
	AVC_CMD_PANEL_PASS_THROUGH	This is macro AVC_CMD_PANEL_PASS_THROUGH.
	AVC_CTYPE_CONTROL	• addtogroup avrcp • @{ • @name Command types
	AVC_CTYPE_GENERAL_INQUORY	This is macro AVC_CTYPE_GENERAL_INQUORY.
	AVC_CTYPE_NOTIFY	This is macro AVC_CTYPE_NOTIFY.
	AVC_CTYPE_SPECIFIC_IQUIRY	This is macro AVC_CTYPE_SPECIFIC_IQUIRY.
	AVC_CTYPE_STATUS	This is macro AVC_CTYPE_STATUS.
	AVC_EVENT_ADDRESSED_PLAYER_CHANGED	< The Addressed Player has been changed, see 6.9.2.
	AVC_EVENT_AVAILABLE_PLAYERS_CHANGED	< The available players have changed, see 6.9
	AVC_EVENT_BATT_STATUS_CHANGED	< Change in battery status
	AVC_EVENT_FLAG_ADDRESSED_PLAYER_CHANGED	This is macro AVC_EVENT_FLAG_ADDRESSED_PLAYER_CHANGED.
	AVC_EVENT_FLAG_ALL	This is macro AVC_EVENT_FLAG_ALL.
	AVC_EVENT_FLAG_AVAILABLE_PLAYERS_CHANGED	This is macro AVC_EVENT_FLAG_AVAILABLE_PLAYERS_CHANGED.
	AVC_EVENT_FLAG_BATT_STATUS_CHANGED	This is macro AVC_EVENT_FLAG_BATT_STATUS_CHANGED.
	AVC_EVENT_FLAG_NOW_PLAYING_CONTENT_CHANGED	This is macro AVC_EVENT_FLAG_NOW_PLAYING_CONTENT_CHANGED.
	AVC_EVENT_FLAG_PLAYBACK_POS_CHANGED	This is macro AVC_EVENT_FLAG_PLAYBACK_POS_CHANGED.
	AVC_EVENT_FLAG_PLAYBACK_STATUS_CHANGED	• addtogroup avrcp • @{ • @name Notifications mask
	AVC_EVENT_FLAG_PLAYER_APPLICATION_SETTING_CHANGED	This is macro AVC_EVENT_FLAG_PLAYER_APPLICATION_SETTING_CHANGED.
	AVC_EVENT_FLAG_SYSTEM_STATUS_CHANGED	This is macro AVC_EVENT_FLAG_SYSTEM_STATUS_CHANGED.
	AVC_EVENT_FLAG_TRACK_CHANGED	This is macro AVC_EVENT_FLAG_TRACK_CHANGED.
	AVC_EVENT_FLAG_TRACK_REACHED_END	This is macro AVC_EVENT_FLAG_TRACK_REACHED_END.
	AVC_EVENT_FLAG_TRACK_REACHED_START	This is macro AVC_EVENT_FLAG_TRACK_REACHED_START.
	AVC_EVENT_FLAG_UIDS_CHANGED	This is macro AVC_EVENT_FLAG_UIDS_CHANGED.
	AVC_EVENT_FLAG_VOLUME_CHANGED	This is macro AVC_EVENT_FLAG_VOLUME_CHANGED.
	AVC_EVENT_NOW_PLAYING_CONTENT_CHANGED	< The content of the Now Playing list has changed, see 6.9.5.

	AVC_EVENT_PLAYBACK_POS_CHANGED	< Change in playback position. Returned after the specified playback notification change notification interval
	AVC_EVENT_PLAYBACK_STATUS_CHANGED	< Change in playback status of the current track.
	AVC_EVENT_PLAYER_APPLICATION_SETTING_CHANGED	< Change in player application setting
	AVC_EVENT_SYSTEM_STATUS_CHANGED	< Change in system status
	AVC_EVENT_TRACK_CHANGED	< Change of current track
	AVC_EVENT_TRACK_REACHED_END	< Reached end of a track
	AVC_EVENT_TRACK_REACHED_START	< Reached start of a track
	AVC_EVENT_UIDS_CHANGED	< The UIDs have changed, see 6.10.3.3.
	AVC_EVENT_VOLUME_CHANGED	< The volume has been changed locally on the TG, see 6.13.3
	AVC_FLAG_BROWSING_CMD	This is macro AVC_FLAG_BROWSING_CMD.
	AVC_FLAG_PANEL_CLICK	This is macro AVC_FLAG_PANEL_CLICK.
	AVC_FLAG_RESPONSE	This is macro AVC_FLAG_RESPONSE.
	AVC_MAXEVENTS	This is macro AVC_MAXEVENTS.
	AVC_MEDIA_ATTR_FLAG_ALBUM	This is macro AVC_MEDIA_ATTR_FLAG_ALBUM.
	AVC_MEDIA_ATTR_FLAG_ALL	This is macro AVC_MEDIA_ATTR_FLAG_ALL.
	AVC_MEDIA_ATTR_FLAG_ARTIST	This is macro AVC_MEDIA_ATTR_FLAG_ARTIST.
	AVC_MEDIA_ATTR_FLAG_GENRE	This is macro AVC_MEDIA_ATTR_FLAG_GENRE.
	AVC_MEDIA_ATTR_FLAG_NUMBER	This is macro AVC_MEDIA_ATTR_FLAG_NUMBER.
	AVC_MEDIA_ATTR_FLAG_PLAYING_TIME	This is macro AVC_MEDIA_ATTR_FLAG_PLAYING_TIME.
	AVC_MEDIA_ATTR_FLAG_TITLE	• addtogroup avrcp • @{ • @name Media attribute bitmask
	AVC_MEDIA_ATTR_FLAG_TOTAL_NUMBER	This is macro AVC_MEDIA_ATTR_FLAG_TOTAL_NUMBER.
	AVC_MEDIA_ATTR_ID_ALBUM	This is macro AVC_MEDIA_ATTR_ID_ALBUM.
	AVC_MEDIA_ATTR_ID_ARTIST	This is macro AVC_MEDIA_ATTR_ID_ARTIST.
	AVC_MEDIA_ATTR_ID_GENRE	This is macro AVC_MEDIA_ATTR_ID_GENRE.
	AVC_MEDIA_ATTR_ID_NUMBER	This is macro AVC_MEDIA_ATTR_ID_NUMBER.
	AVC_MEDIA_ATTR_ID_PLAYING_TIME	This is macro AVC_MEDIA_ATTR_ID_PLAYING_TIME.
	AVC_MEDIA_ATTR_ID_TITLE	• addtogroup avrcp • @{ • @name Media attribute IDs
	AVC_MEDIA_ATTR_ID_TOTAL_NUMBER	This is macro AVC_MEDIA_ATTR_ID_TOTAL_NUMBER.
	AVC_MEDIA_PLAYER_VIRTUAL_FILESYSTEM	Folder Item, Media Element Item The virtual filesystem containing the Browsed media content of the browsed player
	AVC_NOW_PLAYING	Media Element Item The Now Playing list (or queue) Addressed of the addressed player
	AVC_PACKET_TYPE_CONTINUE	This is macro AVC_PACKET_TYPE_CONTINUE.
	AVC_PACKET_TYPE_END	This is macro AVC_PACKET_TYPE_END.
	AVC_PACKET_TYPE_SINGLE	This is macro AVC_PACKET_TYPE_SINGLE.
	AVC_PACKET_TYPE_START	This is macro AVC_PACKET_TYPE_START.
	AVC_PANEL_BUTTON_PRESSED	This is macro AVC_PANEL_BUTTON_PRESSED.
	AVC_PANEL_BUTTON_RELEASED	This is macro AVC_PANEL_BUTTON_RELEASED.

	AVC_PANEL_OPID_0	define AVC_PANEL_OPID_RESERVER 0x10 define AVC_PANEL_OPID_RESERVER 0x11 define AVC_PANEL_OPID_RESERVER 0x12 define AVC_PANEL_OPID_RESERVER 0x13 define AVC_PANEL_OPID_RESERVER 0x14 define AVC_PANEL_OPID_RESERVER 0x15 define AVC_PANEL_OPID_RESERVER 0x16 define AVC_PANEL_OPID_RESERVER 0x17 define AVC_PANEL_OPID_RESERVER 0x18 define AVC_PANEL_OPID_RESERVER 0x19 define AVC_PANEL_OPID_RESERVER 0x1A define AVC_PANEL_OPID_RESERVER 0x1B define AVC_PANEL_OPID_RESERVER 0x1C define AVC_PANEL_OPID_RESERVER 0x1D define AVC_PANEL_OPID_RESERVER 0x1E define AVC_PANEL_OPID_RESERVER 0x1F
	AVC_PANEL_OPID_1	This is macro AVC_PANEL_OPID_1.
	AVC_PANEL_OPID_2	This is macro AVC_PANEL_OPID_2.
	AVC_PANEL_OPID_3	This is macro AVC_PANEL_OPID_3.
	AVC_PANEL_OPID_4	This is macro AVC_PANEL_OPID_4.
	AVC_PANEL_OPID_5	This is macro AVC_PANEL_OPID_5.
	AVC_PANEL_OPID_6	This is macro AVC_PANEL_OPID_6.
	AVC_PANEL_OPID_7	This is macro AVC_PANEL_OPID_7.
	AVC_PANEL_OPID_8	This is macro AVC_PANEL_OPID_8.
	AVC_PANEL_OPID_9	This is macro AVC_PANEL_OPID_9.
	AVC_PANEL_OPID_A	This is macro AVC_PANEL_OPID_A.
	AVC_PANEL_OPID_ANGLE	define AVC_PANEL_OPID_RESERVED 0x4E define AVC_PANEL_OPID_RESERVED 0x4F
	AVC_PANEL_OPID_APPS_MENU	This is macro AVC_PANEL_OPID_APPS_MENU.
	AVC_PANEL_OPID_B	This is macro AVC_PANEL_OPID_B.
	AVC_PANEL_OPID_BACKWARD	This is macro AVC_PANEL_OPID_BACKWARD.
	AVC_PANEL_OPID_C	This is macro AVC_PANEL_OPID_C.
	AVC_PANEL_OPID_CHANNEL_DOWN	This is macro AVC_PANEL_OPID_CHANNEL_DOWN.
	AVC_PANEL_OPID_CHANNEL_UP	define AVC_PANEL_OPID_RESRVED 0x2D define AVC_PANEL_OPID_RESRVED 0x2E define AVC_PANEL_OPID_RESRVED 0x2F
	AVC_PANEL_OPID_CLEAR	This is macro AVC_PANEL_OPID_CLEAR.
	AVC_PANEL_OPID_CONTENTS_MENU	This is macro AVC_PANEL_OPID_CONTENTS_MENU.
	AVC_PANEL_OPID_D	This is macro AVC_PANEL_OPID_D.
	AVC_PANEL_OPID_DISPLAY_INFORMATION	This is macro AVC_PANEL_OPID_DISPLAY_INFORMATION.
	AVC_PANEL_OPID_DOT	This is macro AVC_PANEL_OPID_DOT.
	AVC_PANEL_OPID_DOWN	This is macro AVC_PANEL_OPID_DOWN.
	AVC_PANEL_OPID_EJECT	This is macro AVC_PANEL_OPID_EJECT.
	AVC_PANEL_OPID_ENTER	This is macro AVC_PANEL_OPID_ENTER.
	AVC_PANEL_OPID_EXIT	This is macro AVC_PANEL_OPID_EXIT.
	AVC_PANEL_OPID_F1	define AVC_PANEL_OPID_RESERVED 0x6D define AVC_PANEL_OPID_RESERVED 0x6E define AVC_PANEL_OPID_RESERVED 0x6F define AVC_PANEL_OPID_RESERVED 0x70
	AVC_PANEL_OPID_F2	This is macro AVC_PANEL_OPID_F2.
	AVC_PANEL_OPID_F3	This is macro AVC_PANEL_OPID_F3.
	AVC_PANEL_OPID_F4	This is macro AVC_PANEL_OPID_F4.
	AVC_PANEL_OPID_F5	This is macro AVC_PANEL_OPID_F5.
	AVC_PANEL_OPID_F6	This is macro AVC_PANEL_OPID_F6.
	AVC_PANEL_OPID_F7	This is macro AVC_PANEL_OPID_F7.
	AVC_PANEL_OPID_F8	This is macro AVC_PANEL_OPID_F8.
	AVC_PANEL_OPID_F9	This is macro AVC_PANEL_OPID_F9.

	AVC_PANEL_OPID_FAST_FORWARD	This is macro AVC_PANEL_OPID_FAST_FORWARD.
	AVC_PANEL_OPID_FAVORITE_MENU	This is macro AVC_PANEL_OPID_FAVORITE_MENU.
	AVC_PANEL_OPID_FORWARD	This is macro AVC_PANEL_OPID_FORWARD.
	AVC_PANEL_OPID_HELP	This is macro AVC_PANEL_OPID_HELP.
	AVC_PANEL_OPID_INPUT_SELECT	This is macro AVC_PANEL_OPID_INPUT_SELECT.
	AVC_PANEL_OPID_KEYBORD_FUNCTION	This is macro AVC_PANEL_OPID_KEYBORD_FUNCTION.
	AVC_PANEL_OPID_LEFT	This is macro AVC_PANEL_OPID_LEFT.
	AVC_PANEL_OPID_LEFT_DOWN	This is macro AVC_PANEL_OPID_LEFT_DOWN.
	AVC_PANEL_OPID_LEFT_UP	This is macro AVC_PANEL_OPID_LEFT_UP.
	AVC_PANEL_OPID_LINKED_CONTENT	This is macro AVC_PANEL_OPID_LINKED_CONTENT.
	AVC_PANEL_OPID_LIST	This is macro AVC_PANEL_OPID_LIST.
	AVC_PANEL_OPID_LIVE_TV	This is macro AVC_PANEL_OPID_LIVE_TV.
	AVC_PANEL_OPID_LOCK	This is macro AVC_PANEL_OPID_LOCK.
	AVC_PANEL_OPID_MUTE	This is macro AVC_PANEL_OPID_MUTE.
	AVC_PANEL_OPID_MUTE_FUNCTION	This is macro AVC_PANEL_OPID_MUTE_FUNCTION.
	AVC_PANEL_OPID_NEXT_DAY	This is macro AVC_PANEL_OPID_NEXT_DAY.
	AVC_PANEL_OPID_ON_DEMAND_MENU	This is macro AVC_PANEL_OPID_ON_DEMAND_MENU.
	AVC_PANEL_OPID_PAGE_DOWN	This is macro AVC_PANEL_OPID_PAGE_DOWN.
	AVC_PANEL_OPID_PAGE_UP	This is macro AVC_PANEL_OPID_PAGE_UP.
	AVC_PANEL_OPID_PAUSE	This is macro AVC_PANEL_OPID_PAUSE.
	AVC_PANEL_OPID_PAUSE_PLAY_FUNCTION	This is macro AVC_PANEL_OPID_PAUSE_PLAY_FUNCTION.
	AVC_PANEL_OPID_PAUSE_RECORD_FUNCTION	This is macro AVC_PANEL_OPID_PAUSE_RECORD_FUNCTION .
	AVC_PANEL_OPID_PIP_DOWN	This is macro AVC_PANEL_OPID_PIP_DOWN.
	AVC_PANEL_OPID_PIP_MOVE	This is macro AVC_PANEL_OPID_PIP_MOVE.
	AVC_PANEL_OPID_PIP_UP	This is macro AVC_PANEL_OPID_PIP_UP.
	AVC_PANEL_OPID_PLAY	This is macro AVC_PANEL_OPID_PLAY.
	AVC_PANEL_OPID_PLAY_FUNCTION	define AVC_PANEL_OPID_RESERVED 0x56 define AVC_PANEL_OPID_RESERVED 0x57 define AVC_PANEL_OPID_RESERVED 0x58 define AVC_PANEL_OPID_RESERVED 0x5A define AVC_PANEL_OPID_RESERVED 0x5B define AVC_PANEL_OPID_RESERVED 0x5C define AVC_PANEL_OPID_RESERVED 0x5D define AVC_PANEL_OPID_RESERVED 0x5E define AVC_PANEL_OPID_RESERVED 0x5F
	AVC_PANEL_OPID_POWER_STATE_FUNCTION	This is macro AVC_PANEL_OPID_POWER_STATE_FUNCTION.
	AVC_PANEL_OPID_POWER_TOGGLE	This is macro AVC_PANEL_OPID_POWER_TOGGLE.
	AVC_PANEL_OPID_PREVIOUS_CHANNEL	This is macro AVC_PANEL_OPID_PREVIOUS_CHANNEL.
	AVC_PANEL_OPID_PREVIOUS_DAY	This is macro AVC_PANEL_OPID_PREVIOUS_DAY.
	AVC_PANEL_OPID_RECORD	This is macro AVC_PANEL_OPID_RECORD.
	AVC_PANEL_OPID_RECORD_FUNCTION	This is macro AVC_PANEL_OPID_RECORD_FUNCTION.
	AVC_PANEL_OPID_RESTORE_FOLUME_FUNCTION	This is macro AVC_PANEL_OPID_RESTORE_FOLUME_FUNCTI ON.
	AVC_PANEL_OPID_REWIND	This is macro AVC_PANEL_OPID_REWIND.
	AVC_PANEL_OPID_RF_BYPASS	This is macro AVC_PANEL_OPID_RF_BYPASS.
	AVC_PANEL_OPID_RIGHT	This is macro AVC_PANEL_OPID_RIGHT.

	AVC_PANEL_OPID_RIGHT_DOWN	This is macro AVC_PANEL_OPID_RIGHT_DOWN.
	AVC_PANEL_OPID_RIGHT_UP	This is macro AVC_PANEL_OPID_RIGHT_UP.
	AVC_PANEL_OPID_ROOT_MENU	This is macro AVC_PANEL_OPID_ROOT_MENU.
	AVC_PANEL_OPID_SELECT	• addtogroup avrcp • @{ • @name AV/C Panel PASS THROUGH operation IDs
	AVC_PANEL_OPID_SELECT_AUDIO_INPUT_FUNCTION	This is macro AVC_PANEL_OPID_SELECT_AUDIO_INPUT_FUNCTION.
	AVC_PANEL_OPID_SELECT_AV_INPUT_FUNCTION	This is macro AVC_PANEL_OPID_SELECT_AV_INPUT_FUNCTION.
	AVC_PANEL_OPID_SELECT_DISK_FUNCTION	This is macro AVC_PANEL_OPID_SELECT_DISK_FUNCTION.
	AVC_PANEL_OPID_SETUP_MENU	This is macro AVC_PANEL_OPID_SETUP_MENU.
	AVC_PANEL_OPID_SKIP	This is macro AVC_PANEL_OPID_SKIP.
	AVC_PANEL_OPID_SOUND_SELECT	This is macro AVC_PANEL_OPID_SOUND_SELECT.
	AVC_PANEL_OPID_STOP	This is macro AVC_PANEL_OPID_STOP.
	AVC_PANEL_OPID_STOP_FUNCTION	This is macro AVC_PANEL_OPID_STOP_FUNCTION.
	AVC_PANEL_OPID_SUBPICTURE	This is macro AVC_PANEL_OPID_SUBPICTURE.
	AVC_PANEL_OPID_TUNE_FUNCTION	This is macro AVC_PANEL_OPID_TUNE_FUNCTION.
	AVC_PANEL_OPID_UP	This is macro AVC_PANEL_OPID_UP.
	AVC_PANEL_OPID_VENDOR_UNIQUE	This is macro AVC_PANEL_OPID_VENDOR_UNIQUE.
	AVC_PANEL_OPID_VOLUME_DOWN	This is macro AVC_PANEL_OPID_VOLUME_DOWN.
	AVC_PANEL_OPID_VOLUME_UP	This is macro AVC_PANEL_OPID_VOLUME_UP.
	AVC_PANEL_OPID_ZOOM	This is macro AVC_PANEL_OPID_ZOOM.
	AVC_PDUID_ABORT_CONTINUING_RESPONSE	This is macro AVC_PDUID_ABORT_CONTINUING_RESPONSE.
	AVC_PDUID_ADD_TO_NOW_PLAYING	This is macro AVC_PDUID_ADD_TO_NOW_PLAYING.
	AVC_PDUID_CHANGE_PATH	This is macro AVC_PDUID_CHANGE_PATH.
	AVC_PDUID_GENERAL_REJECT	This is macro AVC_PDUID_GENERAL_REJECT.
	AVC_PDUID_GET_CURRENT_PLAYER_APPLICATION_SETTING_VALUE	This is macro AVC_PDUID_GET_CURRENT_PLAYER_APPLICATION_SETTING_VALUE.
	AVC_PDUID_GET_ELEMENT_ATTRIBUTES	This is macro AVC_PDUID_GET_ELEMENT_ATTRIBUTES.
	AVC_PDUID_GET_FOLDER_ITEMS	This is macro AVC_PDUID_GET_FOLDER_ITEMS.
	AVC_PDUID_GET_ITEM_ATTRIBUTES	This is macro AVC_PDUID_GET_ITEM_ATTRIBUTES.
	AVC_PDUID_GET_PLAY_STATUS	This is macro AVC_PDUID_GET_PLAY_STATUS.
	AVC_PDUID_GET_PLAYER_APPLICATION_SETTING_ATTRIBUTE_TEXT	This is macro AVC_PDUID_GET_PLAYER_APPLICATION_SETTING_ATTRIBUTE_TEXT.
	AVC_PDUID_GET_PLAYER_APPLICATION_SETTING_VALUE_TEXT	This is macro AVC_PDUID_GET_PLAYER_APPLICATION_SETTING_VALUE_TEXT.
	AVC_PDUID_GETCAPABILITIES	This is macro AVC_PDUID_GETCAPABILITIES.
	AVC_PDUID_INFORM_BATTERY_STATUS_OF_CT	This is macro AVC_PDUID_INFORM_BATTERY_STATUS_OF_CT.
	AVC_PDUID_INFORM_DISPLAYABLE_CHARACTER_SET	This is macro AVC_PDUID_INFORM_DISPLAYABLE_CHARACTER_SET.

	AVC_PDUID_LIST_PLAYER_APPLICATION_SETTING_ATTRIBUTES	This is macro AVC_PDUID_LIST_PLAYER_APPLICATION_SETTING_ATTRIBUTES.
	AVC_PDUID_LIST_PLAYER_APPLICATION_SETTING_VALUES	This is macro AVC_PDUID_LIST_PLAYER_APPLICATION_SETTING_VALUES.
	AVC_PDUID_PLAY_ITEM	This is macro AVC_PDUID_PLAY_ITEM.
	AVC_PDUID_REGISTER_NOTIFICATION	This is macro AVC_PDUID_REGISTER_NOTIFICATION.
	AVC_PDUID_REQUEST_CONTINUING_RESPONSE	This is macro AVC_PDUID_REQUEST_CONTINUING_RESPONSE.
	AVC_PDUID_SEARCH	This is macro AVC_PDUID_SEARCH.
	AVC_PDUID_SET_ABSOLUTE_VOLUME	This is macro AVC_PDUID_SET_ABSOLUTE_VOLUME.
	AVC_PDUID_SET_ADDRESSED_PLAYER	This is macro AVC_PDUID_SET_ADDRESSED_PLAYER.
	AVC_PDUID_SET_BROWSED_PLAYER	This is macro AVC_PDUID_SET_BROWSED_PLAYER.
	AVC_PDUID_SET_PLAYER_APPLICATION_SETTING_VALUE	This is macro AVC_PDUID_SET_PLAYER_APPLICATION_SETTING_VALUE.
	AVC_PLAY_STATUS_ERROR	This is macro AVC_PLAY_STATUS_ERROR.
	AVC_PLAY_STATUS_FW_SEEK	This is macro AVC_PLAY_STATUS_FW_SEEK.
	AVC_PLAY_STATUS_PAUSED	This is macro AVC_PLAY_STATUS_PAUSED.
	AVC_PLAY_STATUS_PLAYING	This is macro AVC_PLAY_STATUS_PLAYING.
	AVC_PLAY_STATUS_REV_SEEK	This is macro AVC_PLAY_STATUS_REV_SEEK.
	AVC_PLAY_STATUS_STOPPED	• addtogroup avrcp • @{ • @name Play status
	AVC_PLAYER_SETTING_EQUALIZER_OFF	This is macro AVC_PLAYER_SETTING_EQUALIZER_OFF.
	AVC_PLAYER_SETTING_EQUALIZER_ON	This is macro AVC_PLAYER_SETTING_EQUALIZER_ON.
	AVC_PLAYER_SETTING_EQUALIZER_STATUS	This is macro AVC_PLAYER_SETTING_EQUALIZER_STATUS.
	AVC_PLAYER_SETTING_REPEAT_ALL_TRACKS	This is macro AVC_PLAYER_SETTING_REPEAT_ALL_TRACKS.
	AVC_PLAYER_SETTING_REPEAT_GROUP	This is macro AVC_PLAYER_SETTING_REPEAT_GROUP.
	AVC_PLAYER_SETTING_REPEAT_MODE_OFF	This is macro AVC_PLAYER_SETTING_REPEAT_MODE_OFF.
	AVC_PLAYER_SETTING_REPEAT_MODE_STATUS	This is macro AVC_PLAYER_SETTING_REPEAT_MODE_STATUS.
	AVC_PLAYER_SETTING_REPEAT_SINGLE_TRACK	This is macro AVC_PLAYER_SETTING_REPEAT_SINGLE_TRACK.
	AVC_PLAYER_SETTING_SCAN_ALL_TRACKS	This is macro AVC_PLAYER_SETTING_SCAN_ALL_TRACKS.
	AVC_PLAYER_SETTING_SCAN_GROUP	This is macro AVC_PLAYER_SETTING_SCAN_GROUP.
	AVC_PLAYER_SETTING_SCAN_OFF	This is macro AVC_PLAYER_SETTING_SCAN_OFF.
	AVC_PLAYER_SETTING_SCAN_STATUS	This is macro AVC_PLAYER_SETTING_SCAN_STATUS.
	AVC_PLAYER_SETTING_SHUFFLE_ALL_TRACKS	This is macro AVC_PLAYER_SETTING_SHUFFLE_ALL_TRACKS.
	AVC_PLAYER_SETTING_SHUFFLE_GROUP	This is macro AVC_PLAYER_SETTING_SHUFFLE_GROUP.
	AVC_PLAYER_SETTING_SHUFFLE_OFF	This is macro AVC_PLAYER_SETTING_SHUFFLE_OFF.

	AVC_PLAYER_SETTING_SHUFFLE_STATUS	This is macro AVC_PLAYER_SETTING_SHUFFLE_STATUS.
	AVC_RESPONSE_ACCEPTED	This is macro AVC_RESPONSE_ACCEPTED.
	AVC_RESPONSE_CHANGED	This is macro AVC_RESPONSE_CHANGED.
	AVC_RESPONSE_IMPLEMENTED	This is macro AVC_RESPONSE_IMPLEMENTED.
	AVC_RESPONSE_IN_TRANSITION	This is macro AVC_RESPONSE_IN_TRANSITION.
	AVC_RESPONSE_INTERIM	This is macro AVC_RESPONSE_INTERIM.
	AVC_RESPONSE_NOT_IMPLEMENTED	• addtogroup avrcp • @{ • @name Response types
	AVC_RESPONSE_REJECTED	This is macro AVC_RESPONSE_REJECTED.
	AVC_RESPONSE_STABLE	This is macro AVC_RESPONSE_STABLE.
	AVC_RESPONSE_TIMEOUT	This is macro AVC_RESPONSE_TIMEOUT.
	AVC_SCOPE_MEDIA_PLAYER_LIST	Media Player Item Contains all available media players None
	AVC_SEARCH	Media Element Item The results of a search operation Browsed on the browsed player
	AVC_SUBUNIT_ID_EXTENDED_TO_NEXT_BYTE	This is macro AVC_SUBUNIT_ID_EXTENDED_TO_NEXT_BYTE.
	AVC_SUBUNIT_ID_IGNORE	This is macro AVC_SUBUNIT_ID_IGNORE.
	AVC_SUBUNIT_TYPE_AUDIO	This is macro AVC_SUBUNIT_TYPE_AUDIO.
	AVC_SUBUNIT_TYPE_BULLETIN_BOARD	This is macro AVC_SUBUNIT_TYPE_BULLETIN_BOARD.
	AVC_SUBUNIT_TYPE_CA	This is macro AVC_SUBUNIT_TYPE_CA.
	AVC_SUBUNIT_TYPE_CAMERA	This is macro AVC_SUBUNIT_TYPE_CAMERA.
	AVC_SUBUNIT_TYPE_CAMERA_STORAGE	This is macro AVC_SUBUNIT_TYPE_CAMERA_STORAGE.
	AVC_SUBUNIT_TYPE_DISC	This is macro AVC_SUBUNIT_TYPE_DISC.
	AVC_SUBUNIT_TYPE_EXTENDED_TO_NEXT_BYTE	This is macro AVC_SUBUNIT_TYPE_EXTENDED_TO_NEXT_BYTE.
	AVC_SUBUNIT_TYPE_MONITOR	• addtogroup avrcp • @{ • @name Subunit types
	AVC_SUBUNIT_TYPE_PANEL	This is macro AVC_SUBUNIT_TYPE_PANEL.
	AVC_SUBUNIT_TYPE_PRINTER	This is macro AVC_SUBUNIT_TYPE_PRINTER.
	AVC_SUBUNIT_TYPE_TAPE_RECORDER_PLAYER	This is macro AVC_SUBUNIT_TYPE_TAPE_RECORDER_PLAYER.
	AVC_SUBUNIT_TYPE_TUNER	This is macro AVC_SUBUNIT_TYPE_TUNER.
	AVC_SUBUNIT_TYPE_UNIT	This is macro AVC_SUBUNIT_TYPE_UNIT.
	AVC_SUBUNIT_TYPE_VENDOR_UNIQUE	This is macro AVC_SUBUNIT_TYPE_VENDOR_UNIQUE.
	AVC_VOLUME_MAX	100
	AVC_VOLUME_MIN	0
	AVRCP_COMMAND_TYPE_BROWSING	This is macro AVRCP_COMMAND_TYPE_BROWSING.
	AVRCP_COMMAND_TYPE_CONTROL	This is macro AVRCP_COMMAND_TYPE_CONTROL.
	bt_avrcp_0_click	This is macro bt_avrcp_0_click.
	bt_avrcp_0_press	This is macro bt_avrcp_0_press.
	bt_avrcp_0_release	This is macro bt_avrcp_0_release.
	bt_avrcp_1_click	This is macro bt_avrcp_1_click.
	bt_avrcp_1_press	This is macro bt_avrcp_1_press.
	bt_avrcp_1_release	This is macro bt_avrcp_1_release.
	bt_avrcp_2_click	This is macro bt_avrcp_2_click.
	bt_avrcp_2_press	This is macro bt_avrcp_2_press.
	bt_avrcp_2_release	This is macro bt_avrcp_2_release.

	bt_avrcp_3_click	This is macro bt_avrcp_3_click.
	bt_avrcp_3_press	This is macro bt_avrcp_3_press.
	bt_avrcp_3_release	This is macro bt_avrcp_3_release.
	bt_avrcp_4_click	This is macro bt_avrcp_4_click.
	bt_avrcp_4_press	This is macro bt_avrcp_4_press.
	bt_avrcp_4_release	This is macro bt_avrcp_4_release.
	bt_avrcp_5_click	This is macro bt_avrcp_5_click.
	bt_avrcp_5_press	This is macro bt_avrcp_5_press.
	bt_avrcp_5_release	This is macro bt_avrcp_5_release.
	bt_avrcp_6_click	This is macro bt_avrcp_6_click.
	bt_avrcp_6_press	This is macro bt_avrcp_6_press.
	bt_avrcp_6_release	This is macro bt_avrcp_6_release.
	bt_avrcp_7_click	This is macro bt_avrcp_7_click.
	bt_avrcp_7_press	This is macro bt_avrcp_7_press.
	bt_avrcp_7_release	This is macro bt_avrcp_7_release.
	bt_avrcp_8_click	This is macro bt_avrcp_8_click.
	bt_avrcp_8_press	This is macro bt_avrcp_8_press.
	bt_avrcp_8_release	This is macro bt_avrcp_8_release.
	bt_avrcp_9_click	This is macro bt_avrcp_9_click.
	bt_avrcp_9_press	This is macro bt_avrcp_9_press.
	bt_avrcp_9_release	This is macro bt_avrcp_9_release.
	bt_avrcp_angle_click	This is macro bt_avrcp_angle_click.
	bt_avrcp_angle_press	This is macro bt_avrcp_angle_press.
	bt_avrcp_angle_release	This is macro bt_avrcp_angle_release.
	bt_avrcp_backward_click	This is macro bt_avrcp_backward_click.
	bt_avrcp_backward_press	This is macro bt_avrcp_backward_press.
	bt_avrcp_backward_release	This is macro bt_avrcp_backward_release.
	bt_avrcp_channel_down_click	This is macro bt_avrcp_channel_down_click.
	bt_avrcp_channel_down_press	This is macro bt_avrcp_channel_down_press.
	bt_avrcp_channel_down_release	This is macro bt_avrcp_channel_down_release.
	bt_avrcp_channel_up_click	This is macro bt_avrcp_channel_up_click.
	bt_avrcp_channel_up_press	This is macro bt_avrcp_channel_up_press.
	bt_avrcp_channel_up_release	This is macro bt_avrcp_channel_up_release.
	bt_avrcp_clear_click	This is macro bt_avrcp_clear_click.
	bt_avrcp_clear_press	This is macro bt_avrcp_clear_press.
	bt_avrcp_clear_release	This is macro bt_avrcp_clear_release.
	bt_avrcp_content_menu_click	This is macro bt_avrcp_content_menu_click.
	bt_avrcp_content_menu_release	This is macro bt_avrcp_content_menu_release.
	bt_avrcp_contents_menu_press	This is macro bt_avrcp_contents_menu_press.
	bt_avrcp_display_info_click	This is macro bt_avrcp_display_info_click.
	bt_avrcp_display_info_press	This is macro bt_avrcp_display_info_press.
	bt_avrcp_display_info_release	This is macro bt_avrcp_display_info_release.
	bt_avrcp_dot_click	This is macro bt_avrcp_dot_click.
	bt_avrcp_dot_press	This is macro bt_avrcp_dot_press.
	bt_avrcp_dot_release	This is macro bt_avrcp_dot_release.
	bt_avrcp_down_click	This is macro bt_avrcp_down_click.
	bt_avrcp_down_press	This is macro bt_avrcp_down_press.
	bt_avrcp_down_release	This is macro bt_avrcp_down_release.
	bt_avrcp_eject_click	This is macro bt_avrcp_eject_click.
	bt_avrcp_eject_press	This is macro bt_avrcp_eject_press.
	bt_avrcp_eject_release	This is macro bt_avrcp_eject_release.
	bt_avrcp_enter_click	This is macro bt_avrcp_enter_click.
	bt_avrcp_enter_press	This is macro bt_avrcp_enter_press.
	bt_avrcp_enter_release	This is macro bt_avrcp_enter_release.
	bt_avrcp_exit_click	This is macro bt_avrcp_exit_click.

	bt_avrcp_exit_press	This is macro bt_avrcp_exit_press.
	bt_avrcp_exit_release	This is macro bt_avrcp_exit_release.
	bt_avrcp_f1_click	This is macro bt_avrcp_f1_click.
	bt_avrcp_f1_press	This is macro bt_avrcp_f1_press.
	bt_avrcp_f1_release	This is macro bt_avrcp_f1_release.
	bt_avrcp_f2_click	This is macro bt_avrcp_f2_click.
	bt_avrcp_f2_press	This is macro bt_avrcp_f2_press.
	bt_avrcp_f2_release	This is macro bt_avrcp_f2_release.
	bt_avrcp_f3_click	This is macro bt_avrcp_f3_click.
	bt_avrcp_f3_press	This is macro bt_avrcp_f3_press.
	bt_avrcp_f3_release	This is macro bt_avrcp_f3_release.
	bt_avrcp_f4_click	This is macro bt_avrcp_f4_click.
	bt_avrcp_f4_press	This is macro bt_avrcp_f4_press.
	bt_avrcp_f4_release	This is macro bt_avrcp_f4_release.
	bt_avrcp_f5_click	This is macro bt_avrcp_f5_click.
	bt_avrcp_f5_press	This is macro bt_avrcp_f5_press.
	bt_avrcp_f5_release	This is macro bt_avrcp_f5_release.
	bt_avrcp_f6_click	This is macro bt_avrcp_f6_click.
	bt_avrcp_f6_press	This is macro bt_avrcp_f6_press.
	bt_avrcp_f6_release	This is macro bt_avrcp_f6_release.
	bt_avrcp_f7_click	This is macro bt_avrcp_f7_click.
	bt_avrcp_f7_press	This is macro bt_avrcp_f7_press.
	bt_avrcp_f7_release	This is macro bt_avrcp_f7_release.
	bt_avrcp_f8_click	This is macro bt_avrcp_f8_click.
	bt_avrcp_f8_press	This is macro bt_avrcp_f8_press.
	bt_avrcp_f8_release	This is macro bt_avrcp_f8_release.
	bt_avrcp_f9_click	This is macro bt_avrcp_f9_click.
	bt_avrcp_f9_press	This is macro bt_avrcp_f9_press.
	bt_avrcp_f9_release	This is macro bt_avrcp_f9_release.
	bt_avrcp_fast_forward_click	This is macro bt_avrcp_fast_forward_click.
	bt_avrcp_fast_forward_press	This is macro bt_avrcp_fast_forward_press.
	bt_avrcp_fast_forward_release	This is macro bt_avrcp_fast_forward_release.
	bt_avrcp_favorite_menu_click	This is macro bt_avrcp_favorite_menu_click.
	bt_avrcp_favorite_menu_release	This is macro bt_avrcp_favorite_menu_release.
	bt_avrcp_favorive_menu_press	This is macro bt_avrcp_favorive_menu_press.
	bt_avrcp_forward_click	This is macro bt_avrcp_forward_click.
	bt_avrcp_forward_press	This is macro bt_avrcp_forward_press.
	bt_avrcp_forward_release	This is macro bt_avrcp_forward_release.
	bt_avrcp_help_click	This is macro bt_avrcp_help_click.
	bt_avrcp_help_press	This is macro bt_avrcp_help_press.
	bt_avrcp_help_release	This is macro bt_avrcp_help_release.
	bt_avrcp_input_select_click	This is macro bt_avrcp_input_select_click.
	bt_avrcp_input_select_press	This is macro bt_avrcp_input_select_press.
	bt_avrcp_input_select_release	This is macro bt_avrcp_input_select_release.
	bt_avrcp_left_click	This is macro bt_avrcp_left_click.
	bt_avrcp_left_down_click	This is macro bt_avrcp_left_down_click.
	bt_avrcp_left_down_press	This is macro bt_avrcp_left_down_press.
	bt_avrcp_left_down_release	This is macro bt_avrcp_left_down_release.
	bt_avrcp_left_press	This is macro bt_avrcp_left_press.
	bt_avrcp_left_release	This is macro bt_avrcp_left_release.
	bt_avrcp_left_up_click	This is macro bt_avrcp_left_up_click.
	bt_avrcp_left_up_press	This is macro bt_avrcp_left_up_press.
	bt_avrcp_left_up_release	This is macro bt_avrcp_left_up_release.
	bt_avrcp_mute_click	This is macro bt_avrcp_mute_click.
	bt_avrcp_mute_press	This is macro bt_avrcp_mute_press.

	bt_avrcp_mute_release	This is macro bt_avrcp_mute_release.
	bt_avrcp_page_down_click	This is macro bt_avrcp_page_down_click.
	bt_avrcp_page_down_press	This is macro bt_avrcp_page_down_press.
	bt_avrcp_page_down_release	This is macro bt_avrcp_page_down_release.
	bt_avrcp_page_up_click	This is macro bt_avrcp_page_up_click.
	bt_avrcp_page_up_press	This is macro bt_avrcp_page_up_press.
	bt_avrcp_page_up_release	This is macro bt_avrcp_page_up_release.
	bt_avrcp_pause_click	This is macro bt_avrcp_pause_click.
	bt_avrcp_pause_press	This is macro bt_avrcp_pause_press.
	bt_avrcp_pause_release	This is macro bt_avrcp_pause_release.
	bt_avrcp_play_click	This is macro bt_avrcp_play_click.
	bt_avrcp_play_press	This is macro bt_avrcp_play_press.
	bt_avrcp_play_release	This is macro bt_avrcp_play_release.
	bt_avrcp_power_click	This is macro bt_avrcp_power_click.
	bt_avrcp_power_press	This is macro bt_avrcp_power_press.
	bt_avrcp_power_release	This is macro bt_avrcp_power_release.
	bt_avrcp_previous_channel_click	This is macro bt_avrcp_previous_channel_click.
	bt_avrcp_previous_channel_press	This is macro bt_avrcp_previous_channel_press.
	bt_avrcp_previous_channel_release	This is macro bt_avrcp_previous_channel_release.
	bt_avrcp_record_click	This is macro bt_avrcp_record_click.
	bt_avrcp_record_press	This is macro bt_avrcp_record_press.
	bt_avrcp_record_release	This is macro bt_avrcp_record_release.
	bt_avrcp_rewind_click	This is macro bt_avrcp_rewind_click.
	bt_avrcp_rewind_press	This is macro bt_avrcp_rewind_press.
	bt_avrcp_rewind_release	This is macro bt_avrcp_rewind_release.
	bt_avrcp_right_click	This is macro bt_avrcp_right_click.
	bt_avrcp_right_down_click	This is macro bt_avrcp_right_down_click.
	bt_avrcp_right_down_press	This is macro bt_avrcp_right_down_press.
	bt_avrcp_right_down_release	This is macro bt_avrcp_right_down_release.
	bt_avrcp_right_press	This is macro bt_avrcp_right_press.
	bt_avrcp_right_release	This is macro bt_avrcp_right_release.
	bt_avrcp_right_up_click	This is macro bt_avrcp_right_up_click.
	bt_avrcp_right_up_press	This is macro bt_avrcp_right_up_press.
	bt_avrcp_right_up_release	This is macro bt_avrcp_right_up_release.
	bt_avrcp_root_menu_click	This is macro bt_avrcp_root_menu_click.
	bt_avrcp_root_menu_press	This is macro bt_avrcp_root_menu_press.
	bt_avrcp_root_menu_release	This is macro bt_avrcp_root_menu_release.
	bt_avrcp_select_click	This is macro bt_avrcp_select_click.
	bt_avrcp_select_press	Panel operations
	bt_avrcp_select_release	This is macro bt_avrcp_select_release.
	bt_avrcp_setup_menu_click	This is macro bt_avrcp_setup_menu_click.
	bt_avrcp_setup_menu_press	This is macro bt_avrcp_setup_menu_press.
	bt_avrcp_setup_menu_release	This is macro bt_avrcp_setup_menu_release.
	bt_avrcp_sound_select_click	This is macro bt_avrcp_sound_select_click.
	bt_avrcp_sound_select_press	This is macro bt_avrcp_sound_select_press.
	bt_avrcp_sound_select_release	This is macro bt_avrcp_sound_select_release.
	bt_avrcp_stop_click	This is macro bt_avrcp_stop_click.
	bt_avrcp_stop_press	This is macro bt_avrcp_stop_press.
	bt_avrcp_stop_release	This is macro bt_avrcp_stop_release.
	bt_avrcp_subpicture_click	This is macro bt_avrcp_subpicture_click.
	bt_avrcp_subpicture_press	This is macro bt_avrcp_subpicture_press.
	bt_avrcp_subpicture_release	This is macro bt_avrcp_subpicture_release.
	bt_avrcp_up_click	This is macro bt_avrcp_up_click.
	bt_avrcp_up_press	This is macro bt_avrcp_up_press.
	bt_avrcp_up_release	This is macro bt_avrcp_up_release.

	bt_avrcp_volume_down_click	This is macro bt_avrcp_volume_down_click.
	bt_avrcp_volume_down_press	This is macro bt_avrcp_volume_down_press.
	bt_avrcp_volume_down_release	This is macro bt_avrcp_volume_down_release.
	bt_avrcp_volume_up_click	This is macro bt_avrcp_volume_up_click.
	bt_avrcp_volume_up_press	This is macro bt_avrcp_volume_up_press.
	bt_avrcp_volume_up_release	This is macro bt_avrcp_volume_up_release.

Structures

	Name	Description
	_bt_av_add_to_now_playing_s	brief Parameter to AVRCP_EVT_ADD_TO_NOW_PLAYING_COMPLETED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::add_to_now_playing_status - when a local device received a response to a "add to now playing" request.
	_bt_av_battery_status_of_ct_s	brief Parameter to AVRCP_EVT_BATTERY_STATUS_OF_CT_RECEIVED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::battery_status_of_ct - when a local device received a "battery status of controller" command.
	_bt_av_capability_company_id_s	brief Parameter to AVRCP_EVT_COMPANY_ID_LIST_RECEIVED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::company_id - when a local device received a response to a "get company id" request.
	_bt_av_capability_event_id_s	brief Parameter to AVRCP_EVT_EVENT_ID_LIST_RECEIVED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::supported_event_id - when a local device received a response to a "get supported events" request.
	_bt_av_command_t	This is record _bt_av_command_t.
	_bt_av_displayable_character_set_s	brief Parameter to AVRCP_EVT_DISPLAYABLE_CHARACTER_SET_RECEIVED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::displayable_character_set - when a local device received a "displayable chracter set command" request.
	_bt_av_element_attribute_s	brief Media element attribute ingroup avrcp details This structure is used to store media element attribute.
	_bt_av_element_attributes_s	brief Parameter to AVRCP_EVT_GET_ELEMENT_ATTRIBUTES_RECEIVED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::element_attributes - when a local device received a response to a "get media element attributes" request.
	_bt_av_element_id_s	brief Media element UID ingroup avrcp details This structure is used to store media element UID.
	_bt_av_get_element_attributes_s	brief Parameter to AVRCP_EVT_ELEMENT_ATTRIBUTES_REQUESTED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::get_element_attributes - when a local device received a "get element attributes" request.
	_bt_av_notification_addressed_player_changed_s	brief Parameter to AVRCP_EVT_ADDRESSED_PLAYER_CHANGED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::notification::params:addressed_player - when a local device received a "addressed player changed" notification.
	_bt_av_notification_app_setting_changed_s	This is type bt_av_notification_app_setting_changed_t.

	_bt_av_notification_battery_status_s	brief Parameter to AVRCP_EVT_BATT_STATUS_CHANGED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::notification::params:battery_status - when a local device received a "battery status changed" notification.
	_bt_av_notification_playback_pos_changed_s	brief Parameter to AVRCP_EVT_PLAYBACK_POS_CHANGED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::notification::params:playback_pos - when a local device received a "playback position changed" notification.
	_bt_av_notification_playback_status_changed_s	brief Parameter to AVRCP_EVT_PLAYBACK_STATUS_CHANGED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::notification::params:play_status - when a local device received a "play status changed" notification.
	_bt_av_notification_s	brief Parameter to the following events: li AVRCP_EVT_PLAYBACK_STATUS_CHANGED li AVRCP_EVT_TRACK_CHANGED li AVRCP_EVT_PLAYBACK_POS_CHANGED li AVRCP_EVT_BATT_STATUS_CHANGED li AVRCP_EVT_SYSTEM_STATUS_CHANGED li AVRCP_EVT_NOW_PLAYING_CONTENT_CHANGED li AVRCP_EVT_AVAILABLE_PLAYERS_CHANGED li AVRCP_EVT_ADDRESSED_PLAYER_CHANGED li AVRCP_EVT_UIDS_CHANGED li AVRCP_EVT_VOLUME_CHANGED ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::notification - when a local device received one of the following notifications from the target: li Play status changed li Track changed changed li Playback position changed li Battery status changed li System status changed li Addressed player changed li UIDs changed li Volume changed li Player application setting changed... more
	_bt_av_notification_system_status_changed_s	brief Parameter to AVRCP_EVT_SYSTEM_STATUS_CHANGED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::notification::params:system_status - when a local device received a "system status changed" notification.
	_bt_av_notification_track_changed_s	brief Parameter to AVRCP_EVT_TRACK_CHANGED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::notification::params:track - when a local device received a "track changed" notification.
	_bt_av_notification_uids_changed_s	brief Parameter to AVRCP_EVT_UIDS_CHANGED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::notification::params:uids - when a local device received a "UIDs changed" notification.
	_bt_av_notification_volume_changed_s	brief Parameter to AVRCP_EVT_VOLUME_CHANGED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::notification::params:volume - when a local device received a "UIDs changed" notification.
	_bt_av_play_item_s	brief Parameter to AVRCP_EVT_PLAY_ITEM_COMPLETED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::play_item_status - when a local device received a response to a "play item" request.
	_bt_av_play_status_s	brief Parameter to AVRCP_EVT_GET_PLAY_STATUS_RECEIVED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::play_status - when a local device received a response to a "get play status" request.

	_bt_av_player_setting_current_values_s	brief Parameter to AVRCP_EVT_PLAYER_CURRENT_SETTING_VALUES_RECEIVED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::player_setting_current_values - when a local device received a response to a "get current player setting attribute values" request.
	_bt_av_player_setting_values_s	brief Parameter to AVRCP_EVT_PLAYER_SETTING_VALUES_RECEIVED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::player_setting_values - when a local device received a response to a "get player setting attribute values" request.
	_bt_av_player_setting_values_text_s	brief Parameter to AVRCP_EVT_PLAYER_SETTING_VALUES_TEXT_RECEIVED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::player_setting_values_text - when a local device received a response to a "get player setting attribute values displayable text" request.
	_bt_av_player_settings_s	brief Parameter to AVRCP_EVT_PLAYER_SETTING_ATTRIBUTES_RECEIVED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::player_settings - when a local device received a response to a "get supported player setting attributes" request.
	_bt_av_player_settings_text_s	brief Parameter to AVRCP_EVT_PLAYER_SETTING_ATTRIBUTES_TEXT_RECEIVED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::player_settings_text - when a local device received a response to a "get player setting attributes displayable text" request.
	_bt_av_player_text_s	This is type bt_av_player_text_t.
	_bt_av_register_notification_t	brief Parameter to AVRCP_EVT_REGISTER_NOTIFICATION_REQUESTED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::register_notification - when a local device received a "register notification" request.
	_bt_av_response_t	brief AV/C response header ingroup avrcp details This structure is used to store fields present in every AV/C response.
	_bt_av_set_absolute_volume_s	brief Parameter to AVRCP_EVT_SET_ABSOLUTE_VOLUME_COMPLETED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::absolute_volume - when a local device received a response to a "set absolute volume" request.
	_bt_av_set_addressed_player_s	brief Parameter to AVRCP_EVT_SET_ADDRESSED_PLAYER_COMPLETED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::addressed_player - when a local device received a response to a "set addressed player" request.
	bt_av_add_to_now_playing_t	brief Parameter to AVRCP_EVT_ADD_TO_NOW_PLAYING_COMPLETED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::add_to_now_playing_status - when a local device received a response to a "add to now playing" request.
	bt_av_battery_status_of_ct_t	brief Parameter to AVRCP_EVT_BATTERY_STATUS_OF_CT_RECEIVED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::battery_status_of_ct - when a local device received a "battery status of controller" command.

	bt_av_capability_company_id_t	brief Parameter to AVRCP_EVT_COMPANY_ID_LIST_RECEIVED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::company_id - when a local device received a response to a "get company id" request.
	bt_av_capability_event_id_t	brief Parameter to AVRCP_EVT_EVENT_ID_LIST_RECEIVED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::supported_event_id - when a local device received a response to a "get supported events" request.
	bt_av_displayable_character_set_t	brief Parameter to AVRCP_EVT_DISPLAYABLE_CHARACTER_SET_RECEIVED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::displayable_character_set - when a local device received a "displayable chracter set command" request.
	bt_av_element_attribute_t	brief Media element attribute ingroup avrcp details This structure is used to store media element attribute.
	bt_av_element_attributes_t	brief Parameter to AVRCP_EVT_GET_ELEMENT_ATTRIBUTES_RECEIVED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::element_attributes - when a local device received a response to a "get media element attributes" request.
	bt_av_element_id_t	brief Media element UID ingroup avrcp details This structure is used to store media element UID.
	bt_av_get_element_attributes_t	brief Parameter to AVRCP_EVT_ELEMENT_ATTRIBUTES_REQUESTED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::get_element_attributes - when a local device received a "get element attributes" request.
	bt_av_notification_addressed_player_changed_t	brief Parameter to AVRCP_EVT_ADDRESSED_PLAYER_CHANGED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::notification::params:addressed_player - when a local device received a "addressed player changed" notification.
	bt_av_notification_app_setting_changed_t	This is type bt_av_notification_app_setting_changed_t.
	bt_av_notification_battery_status_t	brief Parameter to AVRCP_EVT_BATT_STATUS_CHANGED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::notification::params:battery_status - when a local device received a "battery status changed" notification.
	bt_av_notification_playback_pos_changed_t	brief Parameter to AVRCP_EVT_PLAYBACK_POS_CHANGED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::notification::params:playback_pos - when a local device received a "playback position changed" notification.
	bt_av_notification_playback_status_changed_t	brief Parameter to AVRCP_EVT_PLAYBACK_STATUS_CHANGED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::notification::params:play_status - when a local device received a "play status changed" notification.
	bt_av_notification_system_status_changed_t	brief Parameter to AVRCP_EVT_SYSTEM_STATUS_CHANGED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::notification::params:system_status - when a local device received a "system status changed" notification.

	bt_av_notification_t	brief Parameter to the following events: li AVRCP_EVT_PLAYBACK_STATUS_CHANGED li AVRCP_EVT_TRACK_CHANGED li AVRCP_EVT_PLAYBACK_POS_CHANGED li AVRCP_EVT_BATT_STATUS_CHANGED li AVRCP_EVT_SYSTEM_STATUS_CHANGED li AVRCP_EVT_NOW_PLAYING_CONTENT_CHANGED li AVRCP_EVT_AVAILABLE_PLAYERS_CHANGED li AVRCP_EVT_ADDRESSED_PLAYER_CHANGED li AVRCP_EVT_UIDS_CHANGED li AVRCP_EVT_VOLUME_CHANGED ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::notification - when a local device received one of the following notifications from the target: li Play status changed li Track changed changed li Playback position changed li Battery status changed li System status changed li Addressed player changed li UIDs changed li Volume changed li Player application setting changed... more
	bt_av_notification_track_changed_t	brief Parameter to AVRCP_EVT_TRACK_CHANGED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::notification::params:track - when a local device received a "track changed" notification.
	bt_av_notification_uids_changed_t	brief Parameter to AVRCP_EVT_UIDS_CHANGED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::notification::params:uids - when a local device received a "UIDs changed" notification.
	bt_av_notification_volume_changed_t	brief Parameter to AVRCP_EVT_VOLUME_CHANGED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::notification::params:volume - when a local device received a "UIDs changed" notification.
	bt_av_play_item_t	brief Parameter to AVRCP_EVT_PLAY_ITEM_COMPLETED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::play_item_status - when a local device received a response to a "play item" request.
	bt_av_play_status_t	brief Parameter to AVRCP_EVT_GET_PLAY_STATUS_RECEIVED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::play_status - when a local device received a response to a "get play status" request.
	bt_av_player_setting_current_values_t	brief Parameter to AVRCP_EVT_PLAYER_CURRENT_SETTING_VALUES_RECEIVED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::player_setting_current_values - when a local device received a response to a "get current player setting attribute values" request.
	bt_av_player_setting_values_t	brief Parameter to AVRCP_EVT_PLAYER_SETTING_VALUES_RECEIVED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::player_setting_values - when a local device received a response to a "get player setting attribute values" request.
	bt_av_player_setting_values_text_t	brief Parameter to AVRCP_EVT_PLAYER_SETTING_VALUES_TEXT_RECEIVED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::player_setting_values_text - when a local device received a response to a "get player setting attribute values displayable text" request.

	bt_av_player_settings_t	brief Parameter to AVRCP_EVT_PLAYER_SETTING_ATTRIBUTES_RECEIVED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::player_settings - when a local device received a response to a "get supported player setting attributes" request.
	bt_av_player_settings_text_t	brief Parameter to AVRCP_EVT_PLAYER_SETTING_ATTRIBUTES_TEXT_RECEIVED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::player_settings_text - when a local device received a response to a "get player setting attributes displayable text" request.
	bt_av_player_text_t	This is type bt_av_player_text_t.
	bt_av_register_notification_t	brief Parameter to AVRCP_EVT_REGISTER_NOTIFICATION_REQUESTED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::register_notification - when a local device received a "register notification" request.
	bt_av_response_t	brief AV/C response header ingroup avrcp details This structure is used to store fields present in every AV/C response.
	bt_av_set_absolute_volume_t	brief Parameter to AVRCP_EVT_SET_ABSOLUTE_VOLUME_COMPLETED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::absolute_volume - when a local device received a response to a "set absolute volume" request.
	bt_av_set_addressed_player_t	brief Parameter to AVRCP_EVT_SET_ADDRESSED_PLAYER_COMPLETED event ingroup avrcp details A pointer to this structure is passed to the AVRCP application callback as a valid member of the bt_avrcp_event_t union - bt_avrcp_event_t::addressed_player - when a local device received a response to a "set addressed player" request.

Types

	Name	Description
	bt_av_command_t	This is type bt_av_command_t.

Description

Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.

avrcp_config.h

Macros

	Name	Description
	__AVRCP_CONFIG_H	This is macro __AVRCP_CONFIG_H.

	AVRCP_ALLOCATE_BUFFERS_RAM_SIZE_VAR	• defgroup avrcp_config Configuration
		• ingroup avrcp
		*
		• This module describes parameters used to configure AVRCP layer.
		*
		• dotstack is customized using a configuration file. The configuration file tailors the dotstack to the application being built. It has to have the structure shown below.
		* code #include "cdbt/bt/bt_std.h"
		// HCI, L2CAP and SDP must always be present
		// HCI configuration parameters #define HCI_MAX_CMD_BUFFERS ... #define HCI_MAX_DATA_BUFFERS ... #define HCI_MAX_HCI_CONNECTIONS ... #define HCI_RX_BUFFER_LEN ... #define HCI_TX_BUFFER_LEN ... #define HCI_L2CAP_BUFFER_LEN ... #define HCI_MAX_CMD_PARAM_LEN ...
		// L2CAP configuration parameters #define L2CAP_MAX_CMD_BUFFERS ... #define L2CAP_MAX_FRAME_BUFFERS ... #define L2CAP_MAX_PSMS ... #define L2CAP_MAX_CHANNELS ...
		// SDP... more

	AVRCP_ALLOCATE_BUFFERS_VARS	This is macro AVRCP_ALLOCATE_BUFFERS_VARS.
	AVRCP_CMD_TIMEOUT	brief Command timeout ingroup avrcp_config details This parameter defines the amount of time in milliseconds AVRCP waits for a response to a request. If not defined the default value of 10000 (10 secconds) is used.
	AVRCP_MAX_CHANNELS	brief Maximum number of remote devices a local device can be connected to ingroup avrcp_config details This parameter defines the number of remote devices a local device can have simultaneous connections to (i.e. channels). This value should not exceed AVCTP_MAX_CHANNELS.
	AVRCP_MAX_CMD_BUFFERS	brief Maximum number of command buffers. ingroup avrcp_config details This parameter defines the number of buffers reserved for sending commands to a remote device over its control channel. Each channel uses its own buffers so the total number of buffers is AVRCP_MAX_CHANNELS * AVRCP_MAX_CMD_BUFFERS. The minimum value is 1. The maximum value is 255. If not define one buffer for each channel is reserved.
	AVRCP_MAX_CMD_PARAM_LEN	brief Maximum length of command parameters ingroup avrcp_config details This parameter defines the maximum length of all command parameters. If not defined the default value of 512 is used.
	AVRCP_MAX_DEVICE_NAME_LEN	brief Maximum length of device name ingroup avrcp_config details This parameter defines the size of the buffer used to store device's name while searching for nearby targets with bt_avrcp_find_targets. If the name of the device is longer than AVRCP_MAX_DEVICE_NAME_LEN it is truncated to AVRCP_MAX_DEVICE_NAME_LEN. If not defined the default value of 20 is used.
	AVRCP_MAX_SEARCH_RESULTS	brief Maximum number of devices to find ingroup avrcp_config details This parameter defines the maximum number of devices bt_avrcp_find_targets can find. If not defined the default value of 7 is used.

Description

Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.

avrcp_config_event_handlers.h

Macros

	Name	Description
	__AVRCP_CONFIG_EVENT_HANDLERS_H	This is macro __AVRCP_CONFIG_EVENT_HANDLERS_H.
	AVRCP_COMMAND_HANDLER	This is macro AVRCP_COMMAND_HANDLER.
	AVRCP_COMMAND_SENT_HANDLER	This is macro AVRCP_COMMAND_SENT_HANDLER.
	AVRCP_RESPONSE_HANDLER	This is macro AVRCP_RESPONSE_HANDLER.
	AVRCP_RESPONSE_SENT_HANDLER	This is macro AVRCP_RESPONSE_SENT_HANDLER.

Description

Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.

SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements.

Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.

avrcp_private.h

Functions

	Name	Description
	_bt_avrcp_allocate_browsing_cmd	This is function _bt_avrcp_allocate_browsing_cmd.
	_bt_avrcp_allocate_browsing_response	This is function _bt_avrcp_allocate_browsing_response.
	_bt_avrcp_allocate_bt_specific_cmd	This is function _bt_avrcp_allocate_bt_specific_cmd.
	_bt_avrcp_allocate_bt_specific_response	This is function _bt_avrcp_allocate_bt_specific_response.
	_bt_avrcp_allocate_channel	void _bt_avctp_l2cap_read_data_callback(struct _channel *pch, bt_byte* pdata, bt_int len); bt_avctp_channel_t* _bt_avctp_find_channel(bt_avctp_mgr_t* mgr, bt_bdaddr_t* remote_addr, bt_uint profile_id);
	_bt_avrcp_allocate_cmd	This is function _bt_avrcp_allocate_cmd.
	_bt_avrcp_allocate_response	This is function _bt_avrcp_allocate_response.
	_bt_avrcp_allocate_simple_panel_cmd	This is function _bt_avrcp_allocate_simple_panel_cmd.
	_bt_avrcp_allocate_simple_panel_response	This is function _bt_avrcp_allocate_simple_panel_response.
	_bt_avrcp_find_channel	This is function _bt_avrcp_find_channel.
	_bt_avrcp_free_channel	This is function _bt_avrcp_free_channel.
	_bt_avrcp_free_cmd	This is function _bt_avrcp_free_cmd.
	_bt_avrcp_general_reject	This is function _bt_avrcp_general_reject.
	_bt_avrcp_get_tick_count	This is function _bt_avrcp_get_tick_count.
	_bt_avrcp_handle_add_to_now_playing	This is function _bt_avrcp_handle_add_to_now_playing.
	_bt_avrcp_handle_command	This is function _bt_avrcp_handle_command.
	_bt_avrcp_handle_command_sent	This is function _bt_avrcp_handle_command_sent.
	_bt_avrcp_handle_get_capabilities	This is function _bt_avrcp_handle_get_capabilities.
	_bt_avrcp_handle_get_current_player_application_setting_value	This is function _bt_avrcp_handle_get_current_player_application_setting_value.
	_bt_avrcp_handle_get_element_attributes	This is function _bt_avrcp_handle_get_element_attributes.
	_bt_avrcp_handle_get_play_status	This is function _bt_avrcp_handle_get_play_status.
	_bt_avrcp_handle_get_player_application_setting_attribute_text	This is function _bt_avrcp_handle_get_player_application_setting_attribute_text.
	_bt_avrcp_handle_get_player_application_setting_value_text	This is function _bt_avrcp_handle_get_player_application_setting_value_text.
	_bt_avrcp_handle_inform_battery_status_of_ct	This is function _bt_avrcp_handle_inform_battery_status_of_ct.
	_bt_avrcp_handle_inform_displayable_character_set	This is function _bt_avrcp_handle_inform_displayable_character_set.
	_bt_avrcp_handle_list_player_application_setting_attributes	This is function _bt_avrcp_handle_list_player_application_setting_attributes.
	_bt_avrcp_handle_list_player_application_setting_values	This is function _bt_avrcp_handle_list_player_application_setting_values.
	_bt_avrcp_handle_play_item	This is function _bt_avrcp_handle_play_item.
	_bt_avrcp_handle_register_notification	This is function _bt_avrcp_handle_register_notification.
	_bt_avrcp_handle_request_continuing_response	This is function _bt_avrcp_handle_request_continuing_response.
	_bt_avrcp_handle_response	This is function _bt_avrcp_handle_response.
	_bt_avrcp_handle_response_sent	This is function _bt_avrcp_handle_response_sent.
	_bt_avrcp_handle_set_absolute_volume	This is function _bt_avrcp_handle_set_absolute_volume.
	_bt_avrcp_handle_set_addressed_player	This is function _bt_avrcp_handle_set_addressed_player.
	_bt_avrcp_handle_set_player_application_setting_value	This is function _bt_avrcp_handle_set_player_application_setting_value.
	_bt_avrcp_init_signal	void _bt_avctp_init_message_buffers();

	_bt_avrcp_init_timer	This is function _bt_avrcp_init_timer.
	_bt_avrcp_register_next_notification	This is function _bt_avrcp_register_next_notification.
	_bt_avrcp_register_pending_notification	This is function _bt_avrcp_register_pending_notification.
	_bt_avrcp_send_notifications	This is function _bt_avrcp_send_notifications.
	_bt_avrcp_send_rejected_response	This is function _bt_avrcp_send_rejected_response.
	_bt_avrcp_set_signal	This is function _bt_avrcp_set_signal.
	_bt_avrcp_start_timer	This is function _bt_avrcp_start_timer.
	_bt_avrcp_tg_handle_get_capabilities	This is function _bt_avrcp_tg_handle_get_capabilities.
	_bt_avrcp_tg_handle_get_element_attributes	This is function _bt_avrcp_tg_handle_get_element_attributes.
	_bt_avrcp_tg_handle_get_play_status	This is function _bt_avrcp_tg_handle_get_play_status.
	_bt_avrcp_tg_handle_inform_battery_status_of_ct	This is function _bt_avrcp_tg_handle_inform_battery_status_of_ct.
	_bt_avrcp_tg_handle_inform_displayable_character_set	This is function _bt_avrcp_tg_handle_inform_displayable_character_set.
	_bt_avrcp_tg_handle_register_notification	This is function _bt_avrcp_tg_handle_register_notification.
	_bt_avrcp_tg_handle_set_absolute_volume	This is function _bt_avrcp_tg_handle_set_absolute_volume.
	_bt_avrcp_write_command_header	This is function _bt_avrcp_write_command_header.
	bt_avrcp_abort_continuing_response	This is function bt_avrcp_abort_continuing_response.
	bt_avrcp_avrcpt_request_continuing_response	This is function bt_avrcp_avrcpt_request_continuing_response.

Macros

	Name	Description
	__AVRCP_PRIVATE_H	This is macro __AVRCP_PRIVATE_H.
	AVRCP_MAX_ELEMENT_ATTRIBUTES	This is macro AVRCP_MAX_ELEMENT_ATTRIBUTES.

Description

Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.

at_parser.h

Functions

	Name	Description
	bt_at_parse_fragment	This is function bt_at_parse_fragment.
	bt_at_parser_reset	This is function bt_at_parser_reset.

Macros

	Name	Description
	AT_EVT_CMD_CODE	This is macro AT_EVT_CMD_CODE.
	AT_EVT_CMD_COMPLETED	This is macro AT_EVT_CMD_COMPLETED.
	AT_EVT_CMD_PARAM	This is macro AT_EVT_CMD_PARAM.
	AT_EVT_CMD_READ_CODE	This is macro AT_EVT_CMD_READ_CODE.
	AT_EVT_ERROR	This is macro AT_EVT_ERROR.
	AT_EVT_OK	This is macro AT_EVT_OK.
	AT_EVT_RING	This is macro AT_EVT_RING.
	ATCMD_BUFFER_LEN	This is macro ATCMD_BUFFER_LEN.

Structures

	Name	Description
	_bt_at_parser_t	This is type bt_at_parser_t.
	bt_at_parser_t	This is type bt_at_parser_t.

Types

	Name	Description
	bt_at_parser_callback_pf	This is type bt_at_parser_callback_pf.

Description

Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.

SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements.

Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.

bt_bdaddr.h

Functions

	Name	Description
	bt_bdaddr_is_null	This is function bt_bdaddr_is_null.
	bt_bdaddrs_are_equal	This is function bt_bdaddrs_are_equal.

Description

Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.

baseband.h

Macros

	Name	Description
	COD_MAJOR_AUDIO	This is macro COD_MAJOR_AUDIO.
	COD_MAJOR_COMPUTER	This is macro COD_MAJOR_COMPUTER.
	COD_MAJOR_HEALTH	This is macro COD_MAJOR_HEALTH.
	COD_MAJOR_IMAGING	This is macro COD_MAJOR_IMAGING.
	COD_MAJOR_MISC	This is macro COD_MAJOR_MISC.
	COD_MAJOR_NET_ACCESS_POINT	This is macro COD_MAJOR_NET_ACCESS_POINT.
	COD_MAJOR_PERIPHERAL	This is macro COD_MAJOR_PERIPHERAL.
	COD_MAJOR_PHONE	This is macro COD_MAJOR_PHONE.
	COD_MAJOR_TOY	This is macro COD_MAJOR_TOY.
	COD_MAJOR_UNCATEGORIZED	This is macro COD_MAJOR_UNCATEGORIZED.
	COD_MAJOR_WEARABLE	This is macro COD_MAJOR_WEARABLE.
	COD_MINOR_AV_CAMCORDER	This is macro COD_MINOR_AV_CAMCORDER.
	COD_MINOR_AV_CAR_AUDIO	This is macro COD_MINOR_AV_CAR_AUDIO.
	COD_MINOR_AV_GAMING	This is macro COD_MINOR_AV_GAMING.
	COD_MINOR_AV_HANDSFREE	This is macro COD_MINOR_AV_HANDSFREE.
	COD_MINOR_AV_HEADPHONES	This is macro COD_MINOR_AV_HEADPHONES.
	COD_MINOR_AV_HEADSET	This is macro COD_MINOR_AV_HEADSET.
	COD_MINOR_AV_HIFI_AUDIO	This is macro COD_MINOR_AV_HIFI_AUDIO.
	COD_MINOR_AV_LOUDSPEAKER	This is macro COD_MINOR_AV_LOUDSPEAKER.
	COD_MINOR_AV_MICROPHONE	This is macro COD_MINOR_AV_MICROPHONE.
	COD_MINOR_AV_PORTABLE_AUDIO	This is macro COD_MINOR_AV_PORTABLE_AUDIO.
	COD_MINOR_AV_RESERVED	This is macro COD_MINOR_AV_RESERVED.
	COD_MINOR_AV_RESERVERD2	This is macro COD_MINOR_AV_RESERVERD2.
	COD_MINOR_AV_SET_TOP_BOX	This is macro COD_MINOR_AV_SET_TOP_BOX.
	COD_MINOR_AV_UNCATEGORIZED	This is macro COD_MINOR_AV_UNCATEGORIZED.
	COD_MINOR_AV_VCR	This is macro COD_MINOR_AV_VCR.
	COD_MINOR_AV_VIDE_DISPLAY_AND_LOUDSPEAKER	This is macro COD_MINOR_AV_VIDE_DISPLAY_AND_LOUDSPEAKER.
	COD_MINOR_AV_VIDEO_CAMERA	This is macro COD_MINOR_AV_VIDEO_CAMERA.
	COD_MINOR_AV_VIDEO_CONFERENCING	This is macro COD_MINOR_AV_VIDEO_CONFERENCING.
	COD_MINOR_AV_VIDEO_MONITOR	This is macro COD_MINOR_AV_VIDEO_MONITOR.
	COD_MINOR_COMPUTER_DESKTOP	This is macro COD_MINOR_COMPUTER_DESKTOP.
	COD_MINOR_COMPUTER_HANDHELD	This is macro COD_MINOR_COMPUTER_HANDHELD.

	COD_MINOR_COMPUTER_LAPTOP	This is macro COD_MINOR_COMPUTER_LAPTOP.
	COD_MINOR_COMPUTER_PALMSIZED	This is macro COD_MINOR_COMPUTER_PALMSIZED.
	COD_MINOR_COMPUTER_SERVER	This is macro COD_MINOR_COMPUTER_SERVER.
	COD_MINOR_COMPUTER_UNCATEGORIZED	This is macro COD_MINOR_COMPUTER_UNCATEGORIZED.
	COD_MINOR_COMPUTER_WEARABLE	This is macro COD_MINOR_COMPUTER_WEARABLE.
	COD_MINOR_HEALTH_BPM	This is macro COD_MINOR_HEALTH_BPM.
	COD_MINOR_HEALTH_DATA_DISPLAY	This is macro COD_MINOR_HEALTH_DATA_DISPLAY.
	COD_MINOR_HEALTH_GLUCOSE_METER	This is macro COD_MINOR_HEALTH_GLUCOSE_METER.
	COD_MINOR_HEALTH_HEART_MONITOR	This is macro COD_MINOR_HEALTH_HEART_MONITOR.
	COD_MINOR_HEALTH_PULSE_OXIMETER	This is macro COD_MINOR_HEALTH_PULSE_OXIMETER.
	COD_MINOR_HEALTH_THERMOMETER	This is macro COD_MINOR_HEALTH_THERMOMETER.
	COD_MINOR_HEALTH_UNCATEGORIZED	This is macro COD_MINOR_HEALTH_UNCATEGORIZED.
	COD_MINOR_HEALTH_WEIGHING_SCALE	This is macro COD_MINOR_HEALTH_WEIGHING_SCALE.
	COD_MINOR_IMAGING_CAMERA	This is macro COD_MINOR_IMAGING_CAMERA.
	COD_MINOR_IMAGING_DISPLAY	This is macro COD_MINOR_IMAGING_DISPLAY.
	COD_MINOR_IMAGING_PRINTER	This is macro COD_MINOR_IMAGING_PRINTER.
	COD_MINOR_IMAGING_SCANNER	This is macro COD_MINOR_IMAGING_SCANNER.
	COD_MINOR_IMAGING_UNCATEGORIZED	This is macro COD_MINOR_IMAGING_UNCATEGORIZED.
	COD_MINOR_LAN_01_17	This is macro COD_MINOR_LAN_01_17.
	COD_MINOR_LAN_17_33	This is macro COD_MINOR_LAN_17_33.
	COD_MINOR_LAN_33_50	This is macro COD_MINOR_LAN_33_50.
	COD_MINOR_LAN_50_67	This is macro COD_MINOR_LAN_50_67.
	COD_MINOR_LAN_67_83	This is macro COD_MINOR_LAN_67_83.
	COD_MINOR_LAN_83_99	This is macro COD_MINOR_LAN_83_99.
	COD_MINOR_LAN_FULLY_AVAILABLE	This is macro COD_MINOR_LAN_FULLY_AVAILABLE.
	COD_MINOR_LAN_NO_SERVICE	This is macro COD_MINOR_LAN_NO_SERVICE.
	COD_MINOR_LAN_UNCATEGORIZED	This is macro COD_MINOR_LAN_UNCATEGORIZED.
	COD_MINOR_PERIPHERAL_CARD_READER	This is macro COD_MINOR_PERIPHERAL_CARD_READER.
	COD_MINOR_PERIPHERAL_COMBO	This is macro COD_MINOR_PERIPHERAL_COMBO.
	COD_MINOR_PERIPHERAL_DIGITIZER	This is macro COD_MINOR_PERIPHERAL_DIGITIZER.
	COD_MINOR_PERIPHERAL_GAMEPAD	This is macro COD_MINOR_PERIPHERAL_GAMEPAD.
	COD_MINOR_PERIPHERAL_JOYSTICK	This is macro COD_MINOR_PERIPHERAL_JOYSTICK.
	COD_MINOR_PERIPHERAL_KEYBOARD	This is macro COD_MINOR_PERIPHERAL_KEYBOARD.
	COD_MINOR_PERIPHERAL_MOUSE	This is macro COD_MINOR_PERIPHERAL_MOUSE.
	COD_MINOR_PERIPHERAL_OTHER	This is macro COD_MINOR_PERIPHERAL_OTHER.
	COD_MINOR_PERIPHERAL_REMOTE	This is macro COD_MINOR_PERIPHERAL_REMOTE.
	COD_MINOR_PERIPHERAL_SENSING	This is macro COD_MINOR_PERIPHERAL_SENSING.
	COD_MINOR_PERIPHERAL_UNCATEGORIZED	This is macro COD_MINOR_PERIPHERAL_UNCATEGORIZED.
	COD_MINOR_PHONE_CELLULAR	This is macro COD_MINOR_PHONE_CELLULAR.
	COD_MINOR_PHONE_CORDLESS	This is macro COD_MINOR_PHONE_CORDLESS.
	COD_MINOR_PHONE_ISDN	This is macro COD_MINOR_PHONE_ISDN.
	COD_MINOR_PHONE_SMART	This is macro COD_MINOR_PHONE_SMART.
	COD_MINOR_PHONE_UNCATEGORIZED	This is macro COD_MINOR_PHONE_UNCATEGORIZED.
	COD_MINOR_PHONE_WIREDMODEM	This is macro COD_MINOR_PHONE_WIREDMODEM.
	COD_MINOR_TOY_CONTROLLER	This is macro COD_MINOR_TOY_CONTROLLER.
	COD_MINOR_TOY_DOLL	This is macro COD_MINOR_TOY_DOLL.
	COD_MINOR_TOY_GAME	This is macro COD_MINOR_TOY_GAME.
	COD_MINOR_TOY_ROBOT	This is macro COD_MINOR_TOY_ROBOT.
	COD_MINOR_TOY_UNCATEGORIZED	This is macro COD_MINOR_TOY_UNCATEGORIZED.
	COD_MINOR_TOY_VEHICLE	This is macro COD_MINOR_TOY_VEHICLE.
	COD_MINOR_WEARABLE_GLASSES	This is macro COD_MINOR_WEARABLE_GLASSES.
	COD_MINOR_WEARABLE_HELMET	This is macro COD_MINOR_WEARABLE_HELMET.
	COD_MINOR_WEARABLE_JACKET	This is macro COD_MINOR_WEARABLE_JACKET.
	COD_MINOR_WEARABLE_PAGER	This is macro COD_MINOR_WEARABLE_PAGER.
	COD_MINOR_WEARABLE_UNCATEGORIZED	This is macro COD_MINOR_WEARABLE_UNCATEGORIZED.

	COD_MINOR_WEARABLE_WATCH	This is macro COD_MINOR_WEARABLE_WATCH.
	COS_AUDIO	This is macro COS_AUDIO.
	COS_CAPTURING	This is macro COS_CAPTURING.
	COS_INFORMATION	This is macro COS_INFORMATION.
	COS_LIMITED_DISCOVERABLE_MODE	This is macro COS_LIMITED_DISCOVERABLE_MODE.
	COS_NETWORKING	This is macro COS_NETWORKING.
	COS_OBJECTTRANSFER	This is macro COS_OBJECTTRANSFER.
	COS_POSITIONING	This is macro COS_POSITIONING.
	COS_RENDERING	This is macro COS_RENDERING.
	COS_TELEPHONY	This is macro COS_TELEPHONY.

Description

Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.

SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements.

Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.

bt_config.h

Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.

SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements.

Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.

bt_hcitr.h

Functions

	Name	Description
	bt_oem_recv	• Receive data. This function is called by the HCI layer when it needs more data from the HCI controller. Implementation of this function must receive the specified number of bytes from the HCI controller and call the provided callback function. @param buffer Pointer to a buffer for the received data. The buffer must be long enough to accommodate the number of bytes specified by the par len parameter. @param len Number of bytes to receive. @param callback A callback function that must be called when the requested number of bytes have been received.
	bt_oem_send	• Send data. This function is called by the HCI layer when it needs to send data to the HCI controller. Implementation of this function must send the specified number of bytes to the HCI controller and call the provided callback function. @param buffer Pointer to the data to be sent . @param len Number of bytes to send. @param callback A callback function that must be called when all data have been sent.

Types

	Name	Description
	bt_oem_recv_callback_fp	• Receive callback. This callback function is called when a receive operation initiated by bt_oem_recv() has completed. @param len Number of received bytes. The value of this parameter should always be the same as the number of bytes requested in a call to bt_oem_recv().
	bt_oem_send_callback_fp	• Send callback. This callback function is called when a send operation initiated by bt_oem_send() has completed.

Description

Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.

SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements.

Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.

bt_log.h

Functions

	Name	Description
	bt_log_bdaddr	This is function bt_log_bdaddr.
	bt_log_int	This is function bt_log_int.
	bt_log_linkkey	This is function bt_log_linkkey.
	bt_log_memory	This is function bt_log_memory.
	bt_log_msg	This is function bt_log_msg.
	bt_oem_log_write	brief Output log message. ingroup log details DotStack calls this function to output its debug information. Implementation should output or store the specified message to whatever device or medium where it can be examined and analyzed.

Macros

	Name	Description
	BT_LOG	This is macro BT_LOG.
	BT_LOG_EX	This is macro BT_LOG_EX.
	BT_LOG_LEVEL_A2DP_PAKCET	This is macro BT_LOG_LEVEL_A2DP_PAKCET.
	BT_LOG_LEVEL_ALL	This is macro BT_LOG_LEVEL_ALL.
	BT_LOG_LEVEL_DEBUG	This is macro BT_LOG_LEVEL_DEBUG.
	BT_LOG_LEVEL_ERROR	This is macro BT_LOG_LEVEL_ERROR.
	BT_LOG_LEVEL_INFO	This is macro BT_LOG_LEVEL_INFO.
	BT_LOG_LEVEL_OFF	This is macro BT_LOG_LEVEL_OFF.
	BT_LOGADDR	This is macro BT_LOGADDR.
	BT_LOGADDR_EX	This is macro BT_LOGADDR_EX.
	BT_LOGINT	This is macro BT_LOGINT.
	BT_LOGINT_EX	This is macro BT_LOGINT_EX.
	BT_LOGLINKKEY	This is macro BT_LOGLINKKEY.
	BT_LOGLINKKEY_EX	This is macro BT_LOGLINKKEY_EX.
	BT_LOGMEMORY	This is macro BT_LOGMEMORY.
	BT_LOGMEMORY_EX	This is macro BT_LOGMEMORY_EX.
	BT_LOGWRITE	This is macro BT_LOGWRITE.

Description

Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.

SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements.

Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.

bt_oem.h

Functions

	Name	Description
	bt_oem_assert	This is function bt_oem_assert.
	bt_oem_get_device_class	This is function bt_oem_get_device_class.
	bt_oem_get_device_name	This is function bt_oem_get_device_name.

	bt_oem_get_pin_code	This is function bt_oem_get_pin_code.
	bt_oem_linkkey_notification	This is function bt_oem_linkkey_notification.
	bt_oem_linkkey_request	This is function bt_oem_linkkey_request.
	bt_oem_schedule_signals	This is function bt_oem_schedule_signals.
	bt_oem_ssp_callback	This is function bt_oem_ssp_callback.

Structures

	Name	Description
	_bt_linkkey_notification_t	This is type bt_linkkey_notification_t.
	_bt_linkkey_request_t	This is type bt_linkkey_request_t.
	bt_linkkey_notification_t	This is type bt_linkkey_notification_t.
	bt_linkkey_request_t	This is type bt_linkkey_request_t.

Description

Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.

SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements.

Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.

bt_oem_config.h

Functions

	Name	Description
	ATT_ALLOCATE_BUFFERS	This is function ATT_ALLOCATE_BUFFERS.
	ATT_CLIENT_ALLOCATE_BUFFERS	This is function ATT_CLIENT_ALLOCATE_BUFFERS.
	FTP_ALLOCATE_BUFFERS	This is function FTP_ALLOCATE_BUFFERS.
	GATT_CLIENT_ALLOCATE_BUFFERS	This is function GATT_CLIENT_ALLOCATE_BUFFERS.
	HCI_ALLOCATE_BUFFERS	This is function HCI_ALLOCATE_BUFFERS.
	HCITR_BCSP_ALLOCATE_BUFFERS	This is function HCITR_BCSP_ALLOCATE_BUFFERS.
	HCRP_ALLOCATE_BUFFERS	This is function HCRP_ALLOCATE_BUFFERS.
	HFP_ALLOCATE_BUFFERS	This is function HFP_ALLOCATE_BUFFERS.
	HID_ALLOCATE_BUFFERS	This is function HID_ALLOCATE_BUFFERS.
	HSP_AG_ALLOCATE_BUFFERS	This is function HSP_AG_ALLOCATE_BUFFERS.
	HSP_ALLOCATE_BUFFERS	This is function HSP_ALLOCATE_BUFFERS.
	IAP_ALLOCATE_BUFFERS	This is function IAP_ALLOCATE_BUFFERS.
	IAP_BT_ALLOCATE_BUFFERS	This is function IAP_BT_ALLOCATE_BUFFERS.
	IAP2_ALLOCATE_BUFFERS	This is function IAP2_ALLOCATE_BUFFERS.
	IAPEA_ALLOCATE_BUFFERS	This is function IAPEA_ALLOCATE_BUFFERS.
	L2CAP_ALLOCATE_BUFFERS	This is function L2CAP_ALLOCATE_BUFFERS.
	MAP_ALLOCATE_BUFFERS	This is function MAP_ALLOCATE_BUFFERS.
	OBEX_ALLOCATE_BUFFERS	This is function OBEX_ALLOCATE_BUFFERS.
	PBAP_ALLOCATE_BUFFERS	This is function PBAP_ALLOCATE_BUFFERS.
	RFCOMM_ALLOCATE_BUFFERS	This is function RFCOMM_ALLOCATE_BUFFERS.
	SDP_ALLOCATE_BUFFERS	This is function SDP_ALLOCATE_BUFFERS.
	SMP_ALLOCATE_BUFFERS	This is function SMP_ALLOCATE_BUFFERS.
	SPP_ALLOCATE_BUFFERS	This is function SPP_ALLOCATE_BUFFERS.

Macros

	Name	Description
	__BT_APP_CONFIG_H	This is macro __BT_APP_CONFIG_H.
	BT_ENABLE_BLE	This is macro BT_ENABLE_BLE.
	BT_INCLUDE_IAP	This is macro BT_INCLUDE_IAP.
	BT_INCLUDE_IAP2	This is macro BT_INCLUDE_IAP2.
	BT_INCLUDE_RFCOMM	This is macro BT_INCLUDE_RFCOMM.

	BT_LOG_LEVEL_MAX	This is macro BT_LOG_LEVEL_MAX.
	BT_LOG_LEVEL_MIN	This is macro BT_LOG_LEVEL_MIN.
	IAP_BT_MAX_TRANSPORTS	This is macro IAP_BT_MAX_TRANSPORTS.
	IAP_MAX_SESSIONS	This is macro IAP_MAX_SESSIONS.
	IAP_RX_BUFFER_SIZE	This is macro IAP_RX_BUFFER_SIZE.
	IAP2_MAX_PACKET_SIZE	This is macro IAP2_MAX_PACKET_SIZE.
	IAP2_MAX_SESSIONS	This is macro IAP2_MAX_SESSIONS.

Description

Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.

SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements.

Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.

bt_private.h

Macros

	Name	Description
	ARG_NOT_USED	This is macro ARG_NOT_USED.
	BT_ASSERT	This is macro BT_ASSERT.
	FALSE	This is macro FALSE.
	LOG	This is macro LOG.
	LOG_EX	This is macro LOG_EX.
	LOGADDR	This is macro LOGADDR.
	LOGADDR_EX	This is macro LOGADDR_EX.
	LOGCLEAR	This is macro LOGCLEAR.
	LOGINT	This is macro LOGINT.
	LOGINT_EX	This is macro LOGINT_EX.
	LOGMEMORY	This is macro LOGMEMORY.
	LOGMEMORY_EX	This is macro LOGMEMORY_EX.
	LOGWRITE	This is macro LOGWRITE.
	NULL	This is macro NULL.
	TRUE	This is macro TRUE.

Description

Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.

SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements.

Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.

bt_signal.h

Functions

	Name	Description
	bt_signal_init	This is function bt_signal_init.
	bt_signal_process_pending	This is function bt_signal_process_pending.
	bt_signal_register	This is function bt_signal_register.
	bt_signal_set	This is function bt_signal_set.
	bt_signal_unregister	This is function bt_signal_unregister.

Structures

	Name	Description
	_bt_signal_t	This is record _bt_signal_t.

Types

	Name	Description
	bt_signal_handler_fp	This is type bt_signal_handler_fp.
	bt_signal_t	This is type bt_signal_t.

Description

Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.

SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements.

Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.

bt_std.h

Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.

SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements.

Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.

bt_storage.h

Functions

	Name	Description
	bt_oem_storage_get_capacity	brief Get non-volatile storage capacity. ingroup stg details Implementation of this function must return the capacity of its non-volatile storage.
	bt_oem_storage_read	brief Read from the non-volatile storage. ingroup stg details This function is called by the stack to read from the non-volatile storage. This function must be implemented by the application. When this function is called the application must start a read operation. When the number of bytes specified by the c len parameter is read, the application must call the callback function specified by the c callback parameter. The application does not have to read the whole number of bytes during the call to this function. It may complete reading later and then call the completion callback. The stack guarantees... more
	bt_oem_storage_start	brief Begin a sequence of non-volatile storage operations. ingroup stg details DotStack calls this function when it starts a sequence of non-volatile storage operations. When the sequence is finished, DotStack will call bt_oem_storage_stop().
	bt_oem_storage_stop	brief End a sequence of non-volatile storage operations. ingroup stg details DotStack calls this function when it finishes executing a sequence of non-volatile storage operations.
	bt_oem_storage_write	brief Write to non-volatile storage. ingroup stg details This function is called by the stack to write data to the non-volatile storage. This function must be implemented by the application. When this function is called, the application must start writing specified data to the non-volatile storage. When all data has been written, the application must call the callback function passed in the c callback parameter. The application does not have to complete the write operation during the call to this function. It may complete the operation later and then call the callback function. In this case, the application does not... more

Macros

	Name	Description
	CDS_LAST_DEVICE_ADDR	This is macro CDS_LAST_DEVICE_ADDR.
	CDS_SIGNATURE	This is macro CDS_SIGNATURE.
	CDS_SIGNATURE_ADDR	This is macro CDS_SIGNATURE_ADDR.
	HCI_BDADDR_LEN	This is macro HCI_BDADDR_LEN.
	HIDS_LAST_DEVICE_ADDR	This is macro HIDS_LAST_DEVICE_ADDR.
	HIDS_SIGNATURE	This is macro HIDS_SIGNATURE.
	HIDS_SIGNATURE_ADDR	This is macro HIDS_SIGNATURE_ADDR.
	LKS_FIRST_KEY_ADDR	This is macro LKS_FIRST_KEY_ADDR.

	LKS_MAX_LINK_KEYS	This is macro LKS_MAX_LINK_KEYS.
	LKS_SIGNATURE	This is macro LKS_SIGNATURE.
	LKS_SIGNATURE_ADDR	This is macro LKS_SIGNATURE_ADDR.

Types

	Name	Description
	bt_storage_callback_fp	brief Storage callback. details This callback is called when a non-volatile storage operation completes.

Description

Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.

SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements.

Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.

bt_system.h

Functions

	Name	Description
	bt_sys_get_connectable	This is function bt_sys_get_connectable.
	bt_sys_get_discoverable	This is function bt_sys_get_discoverable.
	bt_sys_get_l2cap_manager	• Get the L2CAP manager. This function returns the L2CAP manager. The L2CAP manager is created as part of the start up sequence. @return The L2CAP manager.
	bt_sys_init	• Initialize the Bluetooth system. This function initializes all internal variables of HCI, L2CAP and SDP modules. It must be called by the application before it can access any functionality provided by the library. In addition to this initialization function the application must call initialization functions of all other profile modules the application is intended to use. E.g., if the application is using the SPP module the bt_spp_init() must be called right after calling bt_sys_init(). This function essentially calls bt_sys_init_ex(HCI_LINK_POLICY_ENABLE_ALL) so all link policy setting are enabled.
	bt_sys_init_ex	• Initialize the Bluetooth system. This function initializes all internal variables of HCI, L2CAP and SDP modules. It must be called by the application before it can access any functionality provided by the library. In addition to this initialization function the application must call initialization functions of all other profile modules the application is intended to use. E.g., if the application is using the SPP module the bt_spp_init() must be called right after calling bt_sys_init(). Also, the caller must provide an SDP database. @param default_link_policy default link policy settings. This is a bitmask that defines the initial value of... more
	bt_sys_set_modes	This is function bt_sys_set_modes.
	bt_sys_start	• Start the Bluetooth system. After all modules used by the application have been initialized this function should be called to start the Bluetooth system operation. During the start up sequence it will reset and initialize the HCI controller and then create the L2CAP manager. The application will be notified when the start up sequence completes by calling the provided callback function. Also, the caller must provide an SDP database. @param discoverable defines whether the device is discoverable after reset. @param connectable defines whether the device is connectable after reset. @param sdp_db SDP database data. @param sdp_db_len Length of SDP database... more

Types

	Name	Description
	bt_sys_callback_fp	• System start callback. This callback function is called when system start initiated by bt_sys_start() has completed. @param success Success of the operation: c BT_TRUE if successfull, c BT_FALSE otherwise. @param param Callback parameter that was specified when bt_sys_start() was called.

Description

Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.

SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements.

Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.

bt_timer.h

Enumerations

	Name	Description
	_bt_timer_id_enum	This is type bt_timer_id.
	bt_timer_id	This is type bt_timer_id.

Functions

	Name	Description
	bt_oem_timer_clear	brief Clear timer. details This function must be implemented by the application. When this function is called the application must clear the specified timer. If it is already expired and a callback is currently pending, the application should also take measures to cancel the callback. param timerId ID of the timer to clear.
	bt_oem_timer_set	brief Set timer. details This function must be implemented by the application. When it is called, the application must set the specified timer. When the timer expires, the application must call the passed callback function. The function must not wait until the timer expires. It must set the timer and exit immediately. param timerId ID of the timer to set. param milliseconds Timer interval in milliseconds param callback Timer expiration callback function.

Types

	Name	Description
	bt_timer_callback_fp	brief Timer callback. details This callback is called when a timer expires.

Description

Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.

SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements.

Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.

bt_types.h

Macros

	Name	Description
	BT_FALSE	This is macro BT_FALSE.
	BT_LINKKEY_LENGTH	This is macro BT_LINKKEY_LENGTH.
	BT_MAKE_BDADDR	This is macro BT_MAKE_BDADDR.
	BT_MAKE_BDADDR_LE	This is macro BT_MAKE_BDADDR_LE.

	BT_NO	This is macro BT_NO.
	BT_TRUE	This is macro BT_TRUE.
	BT_YES	This is macro BT_YES.

Structures

	Name	Description
	_bt_bdaddr_s	This is type bt_bdaddr_t.
	_bt_linkkey_t	This is type bt_linkkey_t.
	_bt_uuid_s	This is type bt_uuid_t.
	bt_bdaddr_t	This is type bt_bdaddr_t.
	bt_linkkey_t	This is type bt_linkkey_t.
	bt_uuid_t	This is type bt_uuid_t.

Types

	Name	Description
	bt_bdaddr_cp	This is type bt_bdaddr_cp.
	bt_bdaddr_p	This is type bt_bdaddr_p.
	bt_bool	This is type bt_bool.
	bt_byte_cp	This is type bt_byte_cp.
	bt_byte_p	This is type bt_byte_p.
	bt_char	This is type bt_char.
	bt_char_cp	This is type bt_char_cp.
	bt_char_p	This is type bt_char_p.
	bt_id	This is type bt_id.
	bt_int_cp	This is type bt_int_cp.
	bt_int_p	This is type bt_int_p.
	bt_linkkey_cp	This is type bt_linkkey_cp.
	bt_linkkey_p	This is type bt_linkkey_p.
	bt_long_cp	This is type bt_long_cp.
	bt_long_p	This is type bt_long_p.
	bt_uint_cp	This is type bt_uint_cp.
	bt_uint_p	This is type bt_uint_p.
	bt_ulong_cp	This is type bt_ulong_cp.
	bt_ulong_p	This is type bt_ulong_p.
	bt_uuid_cp	This is type bt_uuid_cp.
	bt_uuid_p	This is type bt_uuid_p.
	bt_uuid16	This is type bt_uuid16.
	bt_uuid32	This is type bt_uuid32.

Description

Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.

SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements.

Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.

buffer.h

Functions

	Name	Description
	bt_alloc_buffer	This is function bt_alloc_buffer.
	bt_free_buffer	This is function bt_free_buffer.
	bt_get_buffer	This is function bt_get_buffer.
	bt_get_buffer_header	This is function bt_get_buffer_header.
	bt_get_buffer_index	This is function bt_get_buffer_index.

	bt_init_buffer_mgr	This is function bt_init_buffer_mgr.

Macros

	Name	Description
	BUFFER_HDR_LEN	This is macro BUFFER_HDR_LEN.
	BUFFER_STATE_FREE	for offsetof
	BUFFER_STATE_USED	This is macro BUFFER_STATE_USED.

Structures

	Name	Description
	_bt_buffer_header_t	This is type bt_buffer_header_t.
	_bt_buffer_mgr_t	This is type bt_buffer_mgr_t.
	bt_bt_buffer_header_p	This is type bt_bt_buffer_header_p.
	bt_buffer_header_t	This is type bt_buffer_header_t.
	bt_buffer_mgr_p	This is type bt_buffer_mgr_p.
	bt_buffer_mgr_t	This is type bt_buffer_mgr_t.

Description

Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.

SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements.

Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.

bufferutils.h

Functions

	Name	Description
	_bt_memcpy	This is function _bt_memcpy.
	_compact_buffer	This is function _compact_buffer.
	_expand_buffer	This is function _expand_buffer.
	_is_empty_str	This is function _is_empty_str.
	_read_bdaddr	This is function _read_bdaddr.
	_readb	This is function _readb.
	_readi	This is function _readi.
	_readin	This is function _readin.
	_readl	This is function _readl.
	_readln	This is function _readln.
	_readui	This is function _readui.
	_readuin	This is function _readuin.
	_readul	This is function _readul.
	_readuln	This is function _readuln.
	_str2ulong	This is function _str2ulong.
	_str2ulong_dec	This is function _str2ulong_dec.
	_to_lower_case	This is function _to_lower_case.
	_ulong2str	This is function _ulong2str.
	_ulong2str_dec	This is function _ulong2str_dec.
	_write_bdaddr	This is function _write_bdaddr.
	_writeb	This is function _writeb.
	_writei	This is function _writei.
	_writein	This is function _writein.
	_writel	This is function _writel.
	_writeln	This is function _writeln.
	_writes	This is function _writes.
	_writesx	This is function _writesx.

	_zero_memory	This is function _zero_memory.

Macros

	Name	Description
	_readbn	This is macro _readbn.
	_readui	This is macro _readui.
	_readuin	This is macro _readuin.
	_readul	This is macro _readul.
	_readuln	This is macro _readuln.
	_writebn	This is macro _writebn.
	bt_max	This is macro bt_max.
	bt_min	This is macro bt_min.
	max	This is macro max.
	min	This is macro min.

Description

Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.

SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements.

Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.

channel.h

Functions

	Name	Description
	bt_l2cap_read_data	This is function bt_l2cap_read_data.
	bt_l2cap_send_cmd	This is function bt_l2cap_send_cmd.
	bt_l2cap_send_data	brief Send data over an L2CAP channel ingroup l2cap details This function sends data over the specified L2CAP channel. param channel The L2CAP channel to send data over. param data The pointer to the data. param len The length of the data. param callback The callback function that is called when sending the data has been completed. return li c TRUE if the function succeeds. li c FALSE otherwise. The callback function is not called in this case.
	bt_l2cap_send_smart_data	This is function bt_l2cap_send_smart_data.

Macros

	Name	Description
	CID_ATT	This is macro CID_ATT.
	CID_LE_SIG	This is macro CID_LE_SIG.
	CID_MAX	This is macro CID_MAX.
	CID_MAX_FIXED	This is macro CID_MAX_FIXED.
	CID_NULL	This is macro CID_NULL.
	CID_RECV	This is macro CID_RECV.
	CID_SIG	This is macro CID_SIG.
	CID_SM	This is macro CID_SM.
	CMODE_BASIC	This is macro CMODE_BASIC.
	CMODE_ERETR	This is macro CMODE_ERETR.
	CMODE_FLOW	This is macro CMODE_FLOW.
	CMODE_RETR	This is macro CMODE_RETR.
	CMODE_STRM	This is macro CMODE_STRM.
	CSTATE_CLOSED	This is macro CSTATE_CLOSED.
	CSTATE_FREE	This is macro CSTATE_FREE.
	CSTATE_OPEN	This is macro CSTATE_OPEN.
	CSTATE_WAIT_CONFIG	This is macro CSTATE_WAIT_CONFIG.

	CSTATE_WAIT_CONFIG_REQ	This is macro CSTATE_WAIT_CONFIG_REQ.
	CSTATE_WAIT_CONFIG_RSP	This is macro CSTATE_WAIT_CONFIG_RSP.
	CSTATE_WAIT_CONNECT	This is macro CSTATE_WAIT_CONNECT.
	CSTATE_WAIT_CONNECT_RSP	This is macro CSTATE_WAIT_CONNECT_RSP.
	CSTATE_WAIT_DISCONNECT	This is macro CSTATE_WAIT_DISCONNECT.
	CTYPE_CL	connectionless bt_l2cap_channel
	CTYPE_CO	connection-oriented bt_l2cap_channel
	L2CAP_CHANNEL_FLAG_FORCE_HCI_DISCONNECT	This is macro L2CAP_CHANNEL_FLAG_FORCE_HCI_DISCONNECT.
	L2CAP_CHANNEL_FLAG_INCOMING	This is macro L2CAP_CHANNEL_FLAG_INCOMING.
	L2CAP_CHANNEL_FLAG_SENDING	This is macro L2CAP_CHANNEL_FLAG_SENDING.
	L2CAP_ERETR_RECV_STATE_RECV	This is macro L2CAP_ERETR_RECV_STATE_RECV.
	L2CAP_ERETR_RECV_STATE_REJ_SENT	This is macro L2CAP_ERETR_RECV_STATE_REJ_SENT.
	L2CAP_ERETR_RECV_STATE_SREJ_SENT	This is macro L2CAP_ERETR_RECV_STATE_SREJ_SENT.
	L2CAP_ERETR_XMIT_STATE_WAIT_ACK	This is macro L2CAP_ERETR_XMIT_STATE_WAIT_ACK.
	L2CAP_ERETR_XMIT_STATE_WAIT_F	This is macro L2CAP_ERETR_XMIT_STATE_WAIT_F.
	L2CAP_ERETR_XMIT_STATE_XMIT	This is macro L2CAP_ERETR_XMIT_STATE_XMIT.

Structures

	Name	Description
	_bt_l2cap_channel_ext_t	This is record _bt_l2cap_channel_ext_t.
	_bt_l2cap_channel_t	This is record _bt_l2cap_channel_t.
	_bt_l2cap_frame_desc_s	typedef union _bt_l2cap_frame_control_s { struct { bt_byte frame_type:1; bt_byte txSeq:6; bt_byte f:1; bt_byte reqSeq:6; bt_byte sar:2; } iframe; struct { bt_byte frame_type:1; bt_byte reserved:1; bt_byte s:2; bt_byte p:1; bt_byte reserved2:2; bt_byte f:1; bt_byte reqSeq:6; bt_byte reserved3:2; } sframe; } bt_l2cap_frame_control_t;
	bt_l2cap_frame_desc_t	typedef union _bt_l2cap_frame_control_s { struct { bt_byte frame_type:1; bt_byte txSeq:6; bt_byte f:1; bt_byte reqSeq:6; bt_byte sar:2; } iframe; struct { bt_byte frame_type:1; bt_byte reserved:1; bt_byte s:2; bt_byte p:1; bt_byte reserved2:2; bt_byte f:1; bt_byte reqSeq:6; bt_byte reserved3:2; } sframe; } bt_l2cap_frame_control_t;

Types

	Name	Description
	bt_l2cap_channel_ext_t	This is type bt_l2cap_channel_ext_t.
	bt_l2cap_channel_t	This is type bt_l2cap_channel_t.
	bt_l2cap_read_data_callback_fp	This is type bt_l2cap_read_data_callback_fp.
	bt_l2cap_send_data_callback_fp	This is type bt_l2cap_send_data_callback_fp.
	bt_l2cap_state_changed_callback_fp	This is type bt_l2cap_state_changed_callback_fp.

Description

Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.

SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements.

Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.

chmanager.h

Functions

	Name	Description
	bt_l2cap_allocate_channel	This is function bt_l2cap_allocate_channel.
	bt_l2cap_free_channel	This is function bt_l2cap_free_channel.
	bt_l2cap_get_channel	This is function bt_l2cap_get_channel.
	bt_l2cap_get_channel_by_bdaddr_cid	This is function bt_l2cap_get_channel_by_bdaddr_cid.
	bt_l2cap_get_channel_by_hconn_cid	This is function bt_l2cap_get_channel_by_hconn_cid.
	bt_l2cap_get_channel_by_hconn_dest_cid	This is function bt_l2cap_get_channel_by_hconn_dest_cid.
	bt_l2cap_get_channel_by_psm	This is function bt_l2cap_get_channel_by_psm.
	bt_l2cap_init_channels	This is function bt_l2cap_init_channels.

Description

Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.

SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements.

Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.

cmdbuffer.h

Functions

	Name	Description
	bt_l2cap_alloc_cmd_buffer	This is function bt_l2cap_alloc_cmd_buffer.
	bt_l2cap_alloc_cmd_config_req	This is function bt_l2cap_alloc_cmd_config_req.
	bt_l2cap_alloc_cmd_config_res	This is function bt_l2cap_alloc_cmd_config_res.
	bt_l2cap_alloc_cmd_conn_param_update_req	This is function bt_l2cap_alloc_cmd_conn_param_update_req.
	bt_l2cap_alloc_cmd_conn_param_update_res	This is function bt_l2cap_alloc_cmd_conn_param_update_res.
	bt_l2cap_alloc_cmd_connection_req	This is function bt_l2cap_alloc_cmd_connection_req.
	bt_l2cap_alloc_cmd_connection_res	This is function bt_l2cap_alloc_cmd_connection_res.
	bt_l2cap_alloc_cmd_disconnection_req	This is function bt_l2cap_alloc_cmd_disconnection_req.
	bt_l2cap_alloc_cmd_disconnection_res	This is function bt_l2cap_alloc_cmd_disconnection_res.
	bt_l2cap_alloc_cmd_echo_req	This is function bt_l2cap_alloc_cmd_echo_req.
	bt_l2cap_alloc_cmd_echo_res	This is function bt_l2cap_alloc_cmd_echo_res.
	bt_l2cap_alloc_cmd_info_req	This is function bt_l2cap_alloc_cmd_info_req.
	bt_l2cap_alloc_cmd_info_res	This is function bt_l2cap_alloc_cmd_info_res.
	bt_l2cap_alloc_cmd_reject	This is function bt_l2cap_alloc_cmd_reject.
	bt_l2cap_free_cmd_buffer	This is function bt_l2cap_free_cmd_buffer.
	bt_l2cap_init_cmd_buffers	This is function bt_l2cap_init_cmd_buffers.

Description

Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.

SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements.

Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.

command.h

Macros

	Name	Description
	L2CAP_CMD_CONFIG_REQUEST	This is macro L2CAP_CMD_CONFIG_REQUEST.
	L2CAP_CMD_CONFIG_RESPONSE	This is macro L2CAP_CMD_CONFIG_RESPONSE.
	L2CAP_CMD_CONN_PARAM_UPDATE_REQUEST	This is macro L2CAP_CMD_CONN_PARAM_UPDATE_REQUEST.
	L2CAP_CMD_CONN_PARAM_UPDATE_RESPONSE	This is macro L2CAP_CMD_CONN_PARAM_UPDATE_RESPONSE.
	L2CAP_CMD_CONN_REQUEST	This is macro L2CAP_CMD_CONN_REQUEST.
	L2CAP_CMD_CONN_RESPONSE	This is macro L2CAP_CMD_CONN_RESPONSE.
	L2CAP_CMD_DATA_LEN_CMD_REJECT	This is macro L2CAP_CMD_DATA_LEN_CMD_REJECT.
	L2CAP_CMD_DATA_LEN_CONFIG_REQUEST	This is macro L2CAP_CMD_DATA_LEN_CONFIG_REQUEST.
	L2CAP_CMD_DATA_LEN_CONFIG_RESPONSE	This is macro L2CAP_CMD_DATA_LEN_CONFIG_RESPONSE.
	L2CAP_CMD_DATA_LEN_CONN_REQUEST	This is macro L2CAP_CMD_DATA_LEN_CONN_REQUEST.
	L2CAP_CMD_DATA_LEN_CONN_RESPONSE	This is macro L2CAP_CMD_DATA_LEN_CONN_RESPONSE.

	L2CAP_CMD_DATA_LEN_DCONN_REQUEST	This is macro L2CAP_CMD_DATA_LEN_DCONN_REQUEST.
	L2CAP_CMD_DATA_LEN_DCONN_RESPONSE	This is macro L2CAP_CMD_DATA_LEN_DCONN_RESPONSE.
	L2CAP_CMD_DATA_LEN_ECHO_REQUEST	This is macro L2CAP_CMD_DATA_LEN_ECHO_REQUEST.
	L2CAP_CMD_DATA_LEN_ECHO_RESPONSE	This is macro L2CAP_CMD_DATA_LEN_ECHO_RESPONSE.
	L2CAP_CMD_DATA_LEN_INFO_REQUEST	This is macro L2CAP_CMD_DATA_LEN_INFO_REQUEST.
	L2CAP_CMD_DATA_LEN_INFO_RESPONSE	This is macro L2CAP_CMD_DATA_LEN_INFO_RESPONSE.
	L2CAP_CMD_DCONN_REQUEST	This is macro L2CAP_CMD_DCONN_REQUEST.
	L2CAP_CMD_DCONN_RESPONSE	This is macro L2CAP_CMD_DCONN_RESPONSE.
	L2CAP_CMD_ECHO_REQUEST	This is macro L2CAP_CMD_ECHO_REQUEST.
	L2CAP_CMD_ECHO_RESPONSE	This is macro L2CAP_CMD_ECHO_RESPONSE.
	L2CAP_CMD_HEADER_LEN	This is macro L2CAP_CMD_HEADER_LEN.
	L2CAP_CMD_INFO_REQUEST	This is macro L2CAP_CMD_INFO_REQUEST.
	L2CAP_CMD_INFO_RESPONSE	This is macro L2CAP_CMD_INFO_RESPONSE.
	L2CAP_CMD_LAST	This is macro L2CAP_CMD_LAST.
	L2CAP_CMD_REJECT	This is macro L2CAP_CMD_REJECT.
	L2CAP_CMD_RESERVED	This is macro L2CAP_CMD_RESERVED.
	L2CAP_CMD_STATUS_BEING_SENT	This is macro L2CAP_CMD_STATUS_BEING_SENT.
	L2CAP_CMD_STATUS_PENDING	This is macro L2CAP_CMD_STATUS_PENDING.
	L2CAP_CMD_STATUS_WAITING_RESPONSE	This is macro L2CAP_CMD_STATUS_WAITING_RESPONSE.
	L2CAP_CONFIG_RESULT_REJECTED	This is macro L2CAP_CONFIG_RESULT_REJECTED.
	L2CAP_CONFIG_RESULT_SUCCESS	This is macro L2CAP_CONFIG_RESULT_SUCCESS.
	L2CAP_CONFIG_RESULT_UNACCEPTABLE_PARAMETER	This is macro L2CAP_CONFIG_RESULT_UNACCEPTABLE_PARAMETER.
	L2CAP_CONFIG_RESULT_UNKNOWN_OPTION	This is macro L2CAP_CONFIG_RESULT_UNKNOWN_OPTION.
	L2CAP_CONN_REQ_RESULT_INVALID_PSM	This is macro L2CAP_CONN_REQ_RESULT_INVALID_PSM.
	L2CAP_CONN_REQ_RESULT_INVALID_SOURCE_CID	This is macro L2CAP_CONN_REQ_RESULT_INVALID_SOURCE_CID.
	L2CAP_CONN_REQ_RESULT_NO_RESOURCES	This is macro L2CAP_CONN_REQ_RESULT_NO_RESOURCES.
	L2CAP_CONN_REQ_RESULT_PENDING	This is macro L2CAP_CONN_REQ_RESULT_PENDING.
	L2CAP_CONN_REQ_RESULT_SECURITY_BLOCK	This is macro L2CAP_CONN_REQ_RESULT_SECURITY_BLOCK.
	L2CAP_CONN_REQ_RESULT_SRC_CID_ALREADY_ALLOCATED	This is macro L2CAP_CONN_REQ_RESULT_SRC_CID_ALREADY_ALLOCATED.
	L2CAP_CONN_REQ_RESULT_SUCCESS	This is macro L2CAP_CONN_REQ_RESULT_SUCCESS.
	L2CAP_CONN_REQ_STATUS_AUTHENTICATION_PENDING	This is macro L2CAP_CONN_REQ_STATUS_AUTHENTICATION_PENDING.
	L2CAP_CONN_REQ_STATUS_AUTHORIZATION_PENDING	This is macro L2CAP_CONN_REQ_STATUS_AUTHORIZATION_PENDING.
	L2CAP_CONN_REQ_STATUS_NO_INFO	This is macro L2CAP_CONN_REQ_STATUS_NO_INFO.
	L2CAP_ECHO_MAX_DATA_LEN	echo request and response
	L2CAP_EXT_BI_QOS	This is macro L2CAP_EXT_BI_QOS.
	L2CAP_EXT_ENHANCED_RETRANSMISSION	This is macro L2CAP_EXT_ENHANCED_RETRANSMISSION.
	L2CAP_EXT_EXT_FLOW_SPEC_BR_EDR	This is macro L2CAP_EXT_EXT_FLOW_SPEC_BR_EDR.
	L2CAP_EXT_EXT_WINDOW_SIZE	This is macro L2CAP_EXT_EXT_WINDOW_SIZE.
	L2CAP_EXT_FCS_OPTION	This is macro L2CAP_EXT_FCS_OPTION.
	L2CAP_EXT_FIXED_CHANNELS	This is macro L2CAP_EXT_FIXED_CHANNELS.
	L2CAP_EXT_FLOW_CONTROL	This is macro L2CAP_EXT_FLOW_CONTROL.

	L2CAP_EXT_RETRANSMISSION	This is macro L2CAP_EXT_RETRANSMISSION.
	L2CAP_EXT_STREAMING	This is macro L2CAP_EXT_STREAMING.
	L2CAP_EXT_UNICAST_CONNLESS_DATA_RCPT	This is macro L2CAP_EXT_UNICAST_CONNLESS_DATA_RCPT.
	L2CAP_INFO_NOT_SUPPORTED	REDFLAG: should it be non zero?
	L2CAP_INFO_RESULT_SUCCESS	This is macro L2CAP_INFO_RESULT_SUCCESS.
	L2CAP_INFO_TYPE_CONNECTIONLESS_MTU	information request
	L2CAP_INFO_TYPE_EXTENDED_SUPPORT	This is macro L2CAP_INFO_TYPE_EXTENDED_SUPPORT.
	L2CAP_INFO_TYPE_FIXED_CHANNELS	This is macro L2CAP_INFO_TYPE_FIXED_CHANNELS.
	L2CAP_MAX_OPTIONS	configuration request/response
	L2CAP_OPTION_LEN_FLASH_QOS	This is macro L2CAP_OPTION_LEN_FLASH_QOS.
	L2CAP_OPTION_LEN_FLASH_RFC	This is macro L2CAP_OPTION_LEN_FLASH_RFC.
	L2CAP_OPTION_LEN_FLASH_TIMEOUT	This is macro L2CAP_OPTION_LEN_FLASH_TIMEOUT.
	L2CAP_OPTION_LEN_MAX_MTU	This is macro L2CAP_OPTION_LEN_MAX_MTU.
	L2CAP_OPTION_LEN_UNKNOWN	This is macro L2CAP_OPTION_LEN_UNKNOWN.
	L2CAP_OPTION_LEN_UNKNOWN_DATA	21
	L2CAP_OPTION_TYPE_FLASH_QOS	This is macro L2CAP_OPTION_TYPE_FLASH_QOS.
	L2CAP_OPTION_TYPE_FLASH_QOS_FLAG	This is macro L2CAP_OPTION_TYPE_FLASH_QOS_FLAG.
	L2CAP_OPTION_TYPE_FLASH_RFC	This is macro L2CAP_OPTION_TYPE_FLASH_RFC.
	L2CAP_OPTION_TYPE_FLASH_RFC_FLAG	This is macro L2CAP_OPTION_TYPE_FLASH_RFC_FLAG.
	L2CAP_OPTION_TYPE_FLASH_TIMEOUT	This is macro L2CAP_OPTION_TYPE_FLASH_TIMEOUT.
	L2CAP_OPTION_TYPE_FLASH_TIMEOUT_FLAG	This is macro L2CAP_OPTION_TYPE_FLASH_TIMEOUT_FLAG.
	L2CAP_OPTION_TYPE_MAX_MTU	This is macro L2CAP_OPTION_TYPE_MAX_MTU.
	L2CAP_OPTION_TYPE_MAX_MTU_FLAG	This is macro L2CAP_OPTION_TYPE_MAX_MTU_FLAG.
	L2CAP_OPTION_TYPE_UNKNOWN_FLAG	This is macro L2CAP_OPTION_TYPE_UNKNOWN_FLAG.
	L2CAP_REJECT_REASON_INVALID_CHANNEL	This is macro L2CAP_REJECT_REASON_INVALID_CHANNEL.
	L2CAP_REJECT_REASON_MTU_EXCEEDED	This is macro L2CAP_REJECT_REASON_MTU_EXCEEDED.
	L2CAP_REJECT_REASON_NOT_UNDERSTOOD	This is macro L2CAP_REJECT_REASON_NOT_UNDERSTOOD.
	L2CAP_RFC_BASIC	This is macro L2CAP_RFC_BASIC.
	L2CAP_RFC_ENHANCED_RETRANSMISSION	This is macro L2CAP_RFC_ENHANCED_RETRANSMISSION.
	L2CAP_RFC_ERETR_TIMEOUT	3 secs
	L2CAP_RFC_ERETR_TX_WINDOW	This is macro L2CAP_RFC_ERETR_TX_WINDOW.
	L2CAP_RFC_FLOW_CONTROL	This is macro L2CAP_RFC_FLOW_CONTROL.
	L2CAP_RFC_MONITOR_TIMEOUT	12 secs
	L2CAP_RFC_RETRANSMISSION	This is macro L2CAP_RFC_RETRANSMISSION.
	L2CAP_RFC_STREAMING	This is macro L2CAP_RFC_STREAMING.
	L2CAP_SERVICE_TYPE_BEST_EFFORT	This is macro L2CAP_SERVICE_TYPE_BEST_EFFORT.
	L2CAP_SERVICE_TYPE_GUARANTEED	This is macro L2CAP_SERVICE_TYPE_GUARANTEED.
	L2CAP_SERVICE_TYPE_NO_TRAFFIC	This is macro L2CAP_SERVICE_TYPE_NO_TRAFFIC.

Structures

	Name	Description
	_bt_l2cap_cfg_option_t	This is type bt_l2cap_cfg_option_t.
	_bt_l2cap_cmd_config_req_t	This is type bt_l2cap_cmd_config_req_t.
	_bt_l2cap_cmd_config_res_t	This is type bt_l2cap_cmd_config_res_t.
	_bt_l2cap_cmd_conn_param_update_req_t	connection parameter update
	_bt_l2cap_cmd_conn_param_update_res_t	This is type bt_l2cap_cmd_conn_param_update_res_t.
	_bt_l2cap_cmd_connection_req_t	This is type bt_l2cap_cmd_connection_req_t.
	_bt_l2cap_cmd_connection_res_t	This is type bt_l2cap_cmd_connection_res_t.
	_bt_l2cap_cmd_disconnection_req_t	This is type bt_l2cap_cmd_disconnection_req_t.
	_bt_l2cap_cmd_echo_req_t	This is type bt_l2cap_cmd_echo_req_t.

	_bt_l2cap_cmd_echo_res_t	This is type bt_l2cap_cmd_echo_res_t.
	_bt_l2cap_cmd_header_t	This is type bt_l2cap_cmd_header_t.
	_bt_l2cap_cmd_info_req_t	This is type bt_l2cap_cmd_info_req_t.
	_bt_l2cap_cmd_info_res_t	This is type bt_l2cap_cmd_info_res_t.
	_bt_l2cap_cmd_reject_t	This is type bt_l2cap_cmd_reject_t.
	_bt_l2cap_option_flash_timeout_t	This is type bt_l2cap_option_flash_timeout_t.
	_bt_l2cap_option_max_mtu_t	This is type bt_l2cap_option_max_mtu_t.
	_bt_l2cap_option_qos_t	This is type bt_l2cap_option_qos_t.
	_bt_l2cap_option_rfc_t	This is type bt_l2cap_option_rfc_t.
	_bt_l2cap_option_unknown_t	This is type bt_l2cap_option_unknown_t.
	_cmd_disconnection_res	This is type pcmd_disconnection_res.
	bt_l2cap_cfg_option_p	This is type bt_l2cap_cfg_option_p.
	bt_l2cap_cfg_option_t	This is type bt_l2cap_cfg_option_t.
	bt_l2cap_cmd_config_req_p	This is type bt_l2cap_cmd_config_req_p.
	bt_l2cap_cmd_config_req_t	This is type bt_l2cap_cmd_config_req_t.
	bt_l2cap_cmd_config_res_p	This is type bt_l2cap_cmd_config_res_p.
	bt_l2cap_cmd_config_res_t	This is type bt_l2cap_cmd_config_res_t.
	bt_l2cap_cmd_conn_param_update_req_t	connection parameter update
	bt_l2cap_cmd_conn_param_update_res_t	This is type bt_l2cap_cmd_conn_param_update_res_t.
	bt_l2cap_cmd_connection_req_p	This is type bt_l2cap_cmd_connection_req_p.
	bt_l2cap_cmd_connection_req_t	This is type bt_l2cap_cmd_connection_req_t.
	bt_l2cap_cmd_connection_res_p	This is type bt_l2cap_cmd_connection_res_p.
	bt_l2cap_cmd_connection_res_t	This is type bt_l2cap_cmd_connection_res_t.
	bt_l2cap_cmd_disconnection_req_p	This is type bt_l2cap_cmd_disconnection_req_p.
	bt_l2cap_cmd_disconnection_req_t	This is type bt_l2cap_cmd_disconnection_req_t.
	bt_l2cap_cmd_echo_req_p	This is type bt_l2cap_cmd_echo_req_p.
	bt_l2cap_cmd_echo_req_t	This is type bt_l2cap_cmd_echo_req_t.
	bt_l2cap_cmd_echo_res_p	This is type bt_l2cap_cmd_echo_res_p.
	bt_l2cap_cmd_echo_res_t	This is type bt_l2cap_cmd_echo_res_t.
	bt_l2cap_cmd_header_p	This is type bt_l2cap_cmd_header_p.
	bt_l2cap_cmd_header_t	This is type bt_l2cap_cmd_header_t.
	bt_l2cap_cmd_info_req_p	This is type bt_l2cap_cmd_info_req_p.
	bt_l2cap_cmd_info_req_t	This is type bt_l2cap_cmd_info_req_t.
	bt_l2cap_cmd_info_res_p	This is type bt_l2cap_cmd_info_res_p.
	bt_l2cap_cmd_info_res_t	This is type bt_l2cap_cmd_info_res_t.
	bt_l2cap_cmd_reject_p	This is type bt_l2cap_cmd_reject_p.
	bt_l2cap_cmd_reject_t	This is type bt_l2cap_cmd_reject_t.
	bt_l2cap_option_flash_timeout_t	This is type bt_l2cap_option_flash_timeout_t.
	bt_l2cap_option_max_mtu_t	This is type bt_l2cap_option_max_mtu_t.
	bt_l2cap_option_qos_t	This is type bt_l2cap_option_qos_t.
	bt_l2cap_option_rfc_t	This is type bt_l2cap_option_rfc_t.
	bt_l2cap_option_unknown_t	This is type bt_l2cap_option_unknown_t.
	cmd_disconnection_res	This is type cmd_disconnection_res.
	pcmd_disconnection_res	This is type pcmd_disconnection_res.

Types

	Name	Description
	pf_l2cap_cmd_callback	This is type pf_l2cap_cmd_callback.

Unions

	Name	Description
	_bt_l2cap_cmd_reject_param_t	This is type bt_l2cap_cmd_reject_param_t.
	_bt_l2cap_command_t	This is type bt_l2cap_command_t.
	bt_l2cap_cmd_reject_param_t	This is type bt_l2cap_cmd_reject_param_t.
	bt_l2cap_command_t	This is type bt_l2cap_command_t.

Description

Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.

SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements.

Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.

config.h

Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC.

Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.

csr.h

Functions

	Name	Description
	btx_csr_alloc_bccmd_getreq	This is function btx_csr_alloc_bccmd_getreq.
	btx_csr_alloc_bccmd_setreq	This is function btx_csr_alloc_bccmd_setreq.
	btx_csr_autobaud	brief Configure controller's UART speed. ingroup btx_csr details This function makes the controller auto-configure its UART speed. The host transport must be set to H4. This function works only with BC6 controllers.
	btx_csr_bc7_sel_host_interface_h4	brief Configure controller's UART speed and host interface. ingroup btx_csr details This function makes the controller auto-configure its UART speed and select H4 as host interface. PS_KEY_HOST_INTERFACE must not be set. PS_KEY_UART_BITRATE must be set to 0. This function works only with BC7 controllers.
	btx_csr_enable_tx	brief Enable/disable transmitter ingroup btx_csr param enable Specifies whether the transmitter should be enable or disabled. param callback The callback function that will be called after the command has completed. param callback_param A pointer to arbitrary data to be passed to the c callback callback.
	btx_csr_exec_hq_script	This is function btx_csr_exec_hq_script.
	btx_csr_exec_script	brief Patch controller's firmware ingroup btx_csr details This function executes a script that patches the controller's firmware. The c script must point to a structure that contain a complete patch script for a particular controller model and revision. If the revision specified in the script and revision read from the controller are the same btx_csr_patch_controller() loads the script to the controller and calls the c callback with the first parameter TRUE. Otherwise the c callback is called with the first parameter FALSE. If support for multiple firmware revisions is neede use btx_csr_patch_controller(). param script Array of... more
	btx_csr_get_cached_temperature	brief Get chip's cached temperature ingroup btx_csr param callback The callback function that will be called after the command has completed. param callback_param A pointer to arbitrary data to be passed to the c callback callback.
	btx_csr_get_ps_var	This is function btx_csr_get_ps_var.
	btx_csr_get_ps_var_ex	This is function btx_csr_get_ps_var_ex.
	btx_csr_get_rssi_acl	brief Get RSSI ingroup btx_csr details This function retrieves the RSSI for a given HCI ACL handle. param hconn ACL connection handle. param callback The callback function that will be called after the command has completed. param callback_param A pointer to arbitrary data to be passed to the c callback callback.
	btx_csr_get_script__dsp_script__PB_109_DSP_rev8	brief Return script for patching DSP in CSR8x11 A08 (BlueCore 7) ingroup btx_csr
	btx_csr_get_script__PB_101_CSR8811_CSP28_UART	brief Return script for patching CSR8x11 A06 (BlueCore 7) ingroup btx_csr

	btx_csr_get_script__PB_109_CSR8811_REV16	brief Return script for patching CSR8x11 A08 (BlueCore 7) ingroup btx_csr
	btx_csr_get_script__PB_173_CSR8X11_REV1	brief Return script for patching CSR8x11 A12 (BlueCore 7) ingroup btx_csr
	btx_csr_get_script__PB_27_R20_BC6ROM_A04	brief Return script for patching BlueCore 6 ingroup btx_csr
	btx_csr_get_script__PB_90_REV6	brief Return script for patching CSR8810 (BlueCore 7) ingroup btx_csr
	btx_csr_get_var	This is function btx_csr_get_var.
	btx_csr_init	brief Initialize CSR support layer. ingroup btx_csr details This function initializes all internal variables of the CSR support layer. CSR controllers use vendor specific event (0xFF) to carry the BCCMD protocol. They also do not report number of completed packets for BCCMD commands. This function installs a vendor specific event handler that makes sure that callback are called on corresponding vendor specific commands and the number of free command buffers in the controller is kept correct.
	btx_csr_init_hq_script	This is function btx_csr_init_hq_script.
	btx_csr_patch_controller	brief Patch controller's firmware ingroup btx_csr details This function executes a script that patches the controller's firmware. Each entry of the c scripts array must be a complete patch script for a particular controller model and revision. btx_csr_patch_controller() reads the revision number from the controller then tries to find the corresponding script in the c scripts. If there is a matching script it is loaded to the controller and c callback is called with the first parameter TRUE. If no suitable script found c callback is called with the first parameter FALSE. param scripts Array of patch scripts.... more
	btx_csr_register_bccmd_listener	This is function btx_csr_register_bccmd_listener.
	btx_csr_send_dsp_config_data	This is function btx_csr_send_dsp_config_data.
	btx_csr_send_next_hq_script_packet	This is function btx_csr_send_next_hq_script_packet.
	btx_csr_set_ps_var	This is function btx_csr_set_ps_var.
	btx_csr_set_ps_var_ex	This is function btx_csr_set_ps_var_ex.
	btx_csr_set_ps_vars	brief Write PS variables ingroup btx_csr details param ps_vars PS values param buffer A buffer for storing temporary data during function execution. param callback The callback function that will be called when all PS values have been sent to the controller or error occurred. param callback_param A pointer to arbitrary data to be passed to the c callback callback..
	btx_csr_set_ps_vars_ex	brief Write PS variables ingroup btx_csr param ps_vars PS values param buffer A buffer for storing temporary data during function execution. param store param callback The callback function that will be called when all PS values have been sent to the controller or error occurred. param callback_param A pointer to arbitrary data to be passed to the c callback callback..
	btx_csr_set_var	This is function btx_csr_set_var.
	btx_csr_unregister_bccmd_listener	This is function btx_csr_unregister_bccmd_listener.
	btx_csr_warm_reset	brief Warm reset ingroup btx_csr details This function performs warm reset of the controller. All patches and configuration parameters sent to the controller before warm reset are kept intact.
	btx_csr_warm_reset_ex	brief Warm reset ingroup btx_csr details This function performs warm reset of the controller. All patches and configuration parameters sent to the controller before warm reset are kept intact. Since the controller does not respond to the warm reset command as it starts resetting immediately upon receiving the command, the c callback is called right after the command packet has been transmitted to the controller. param callback The callback function that will be called after the warm reset command has been sent to the controller. param callback_param A pointer to arbitrary data to be passed to the c callback callback.... more

Macros

	Name	Description
	CSR_I2S_STREAM_CFG_KEY_AUDIO_ATTEN	This is macro CSR_I2S_STREAM_CFG_KEY_AUDIO_ATTEN.

	CSR_I2S_STREAM_CFG_KEY_AUDIO_ATTEN_ENABLE	This is macro CSR_I2S_STREAM_CFG_KEY_AUDIO_ATTEN_ENABLE.
	CSR_I2S_STREAM_CFG_KEY_BITS_PER_SAMPLE	This is macro CSR_I2S_STREAM_CFG_KEY_BITS_PER_SAMPLE.
	CSR_I2S_STREAM_CFG_KEY_CROP_ENABLE	This is macro CSR_I2S_STREAM_CFG_KEY_CROP_ENABLE.
	CSR_I2S_STREAM_CFG_KEY_JUSTIFY_DELAY	This is macro CSR_I2S_STREAM_CFG_KEY_JUSTIFY_DELAY.
	CSR_I2S_STREAM_CFG_KEY_JUSTIFY_FORMAT	This is macro CSR_I2S_STREAM_CFG_KEY_JUSTIFY_FORMAT.
	CSR_I2S_STREAM_CFG_KEY_JUSTIFY_RESOLUTION	This is macro CSR_I2S_STREAM_CFG_KEY_JUSTIFY_RESOLUTION.
	CSR_I2S_STREAM_CFG_KEY_MASTER	This is macro CSR_I2S_STREAM_CFG_KEY_MASTER.
	CSR_I2S_STREAM_CFG_KEY_MCLK	This is macro CSR_I2S_STREAM_CFG_KEY_MCLK.
	CSR_I2S_STREAM_CFG_KEY_POLARITY	This is macro CSR_I2S_STREAM_CFG_KEY_POLARITY.
	CSR_I2S_STREAM_CFG_KEY_RX_START_SAMPLE	This is macro CSR_I2S_STREAM_CFG_KEY_RX_START_SAMPLE.
	CSR_I2S_STREAM_CFG_KEY_SYNC_RATE	This is macro CSR_I2S_STREAM_CFG_KEY_SYNC_RATE.
	CSR_I2S_STREAM_CFG_KEY_TX_START_SAMPLE	This is macro CSR_I2S_STREAM_CFG_KEY_TX_START_SAMPLE.
	CSR_MAX_HQ_PACKET_LEN	This is macro CSR_MAX_HQ_PACKET_LEN.
	CSR_MSG_TYPE_GETREQ	This is macro CSR_MSG_TYPE_GETREQ.
	CSR_MSG_TYPE_GETRESP	This is macro CSR_MSG_TYPE_GETRESP.
	CSR_MSG_TYPE_SETREQ	This is macro CSR_MSG_TYPE_SETREQ.
	CSR_PCM_STREAM_CFG_KEY_LSB_FIRST	This is macro CSR_PCM_STREAM_CFG_KEY_LSB_FIRST.
	CSR_PCM_STREAM_CFG_KEY_MANCHESTER	This is macro CSR_PCM_STREAM_CFG_KEY_MANCHESTER.
	CSR_PCM_STREAM_CFG_KEY_MANCHESTER_SLAVE	This is macro CSR_PCM_STREAM_CFG_KEY_MANCHESTER_SLAVE.
	CSR_PCM_STREAM_CFG_KEY_MASTER	This is macro CSR_PCM_STREAM_CFG_KEY_MASTER.
	CSR_PCM_STREAM_CFG_KEY_MCLK	This is macro CSR_PCM_STREAM_CFG_KEY_MCLK.
	CSR_PCM_STREAM_CFG_KEY_SHORT_SYNC	This is macro CSR_PCM_STREAM_CFG_KEY_SHORT_SYNC.
	CSR_PCM_STREAM_CFG_KEY_SIGN_EXTEND	This is macro CSR_PCM_STREAM_CFG_KEY_SIGN_EXTEND.
	CSR_PCM_STREAM_CFG_KEY_SLOT_COUNT	This is macro CSR_PCM_STREAM_CFG_KEY_SLOT_COUNT.
	CSR_PCM_STREAM_CFG_KEY_SYNC_RATE	This is macro CSR_PCM_STREAM_CFG_KEY_SYNC_RATE.
	CSR_PCM_STREAM_CFG_KEY_TX_TRISTATE	This is macro CSR_PCM_STREAM_CFG_KEY_TX_TRISTATE.
	CSR_SNK_ADC	This is macro CSR_SNK_ADC.
	CSR_SNK_FASTPIPE	This is macro CSR_SNK_FASTPIPE.
	CSR_SNK_FM	This is macro CSR_SNK_FM.
	CSR_SNK_I2S	This is macro CSR_SNK_I2S.
	CSR_SNK_L2CAP	This is macro CSR_SNK_L2CAP.
	CSR_SNK_PCM	This is macro CSR_SNK_PCM.
	CSR_SNK_SCO	This is macro CSR_SNK_SCO.
	CSR_SNK_SPDIF	This is macro CSR_SNK_SPDIF.
	CSR_SRC_ADC	This is macro CSR_SRC_ADC.
	CSR_SRC_FASTPIPE	This is macro CSR_SRC_FASTPIPE.
	CSR_SRC_FM	This is macro CSR_SRC_FM.
	CSR_SRC_I2S	This is macro CSR_SRC_I2S.
	CSR_SRC_L2CAP	This is macro CSR_SRC_L2CAP.
	CSR_SRC_MIC	This is macro CSR_SRC_MIC.
	CSR_SRC_PCM	This is macro CSR_SRC_PCM.
	CSR_SRC_SCO	This is macro CSR_SRC_SCO.
	CSR_SRC_SPDIF	This is macro CSR_SRC_SPDIF.
	CSR_VARID_CACHED_TEMPERATURE	This is macro CSR_VARID_CACHED_TEMPERATURE.
	CSR_VARID_CAPABILITY_DOWNLOAD_INDICATION	This is macro CSR_VARID_CAPABILITY_DOWNLOAD_INDICATION.
	CSR_VARID_CREATE_OPERATOR_C	This is macro CSR_VARID_CREATE_OPERATOR_C.
	CSR_VARID_CREATE_OPERATOR_P	This is macro CSR_VARID_CREATE_OPERATOR_P.
	CSR_VARID_DESTROY_OPERATORS	This is macro CSR_VARID_DESTROY_OPERATORS.
	CSR_VARID_DSPMANAGER_CONFIG_REQUEST	This is macro CSR_VARID_DSPMANAGER_CONFIG_REQUEST.
	CSR_VARID_ENABLE_SCO_STREAMS	This is macro CSR_VARID_ENABLE_SCO_STREAMS.

	CSR_VARID_MAP_SCO_AUDIO	This is macro CSR_VARID_MAP_SCO_AUDIO.
	CSR_VARID_MESSAGE_FROM_OPERATOR	Select Variable definitions
	CSR_VARID_PIO	This is macro CSR_VARID_PIO.
	CSR_VARID_PIO_DIRECTION_MASK	This is macro CSR_VARID_PIO_DIRECTION_MASK.
	CSR_VARID_PIO_PROTECT_MASK	This is macro CSR_VARID_PIO_PROTECT_MASK.
	CSR_VARID_RSSI_ACL	This is macro CSR_VARID_RSSI_ACL.
	CSR_VARID_START_OPERATORS	This is macro CSR_VARID_START_OPERATORS.
	CSR_VARID_STOP_OPERATORS	This is macro CSR_VARID_STOP_OPERATORS.
	CSR_VARID_STREAM_CLOSE_SINK	This is macro CSR_VARID_STREAM_CLOSE_SINK.
	CSR_VARID_STREAM_CLOSE_SOURCE	This is macro CSR_VARID_STREAM_CLOSE_SOURCE.
	CSR_VARID_STREAM_CONFIGURE	This is macro CSR_VARID_STREAM_CONFIGURE.
	CSR_VARID_STREAM_CONNECT	This is macro CSR_VARID_STREAM_CONNECT.
	CSR_VARID_STREAM_DRAINED_NOTIFICATION	This is macro CSR_VARID_STREAM_DRAINED_NOTIFICATION.
	CSR_VARID_STREAM_GET_SINK	This is macro CSR_VARID_STREAM_GET_SINK.
	CSR_VARID_STREAM_GET_SOURCE	This is macro CSR_VARID_STREAM_GET_SOURCE.
	CSR_VARID_STREAM_TRANSFORM_DISCONNECT	This is macro CSR_VARID_STREAM_TRANSFORM_DISCONNECT.
	GETRESP_BAD_REQ	This is macro GETRESP_BAD_REQ.
	GETRESP_ERROR	This is macro GETRESP_ERROR.
	GETRESP_NO_ACCESS	This is macro GETRESP_NO_ACCESS.
	GETRESP_NO_SUCH_VARID	This is macro GETRESP_NO_SUCH_VARID.
	GETRESP_NO_VALUE	This is macro GETRESP_NO_VALUE.
	GETRESP_OK	BCCMD GETRESP status codes
	GETRESP_PERMISSION_DENIED	This is macro GETRESP_PERMISSION_DENIED.
	GETRESP_READ_ONLY	This is macro GETRESP_READ_ONLY.
	GETRESP_TOO_BIG	This is macro GETRESP_TOO_BIG.
	GETRESP_WRITE_ONLY	This is macro GETRESP_WRITE_ONLY.
	PS_DEFAULT	This is macro PS_DEFAULT.
	PS_F	This is macro PS_F.
	PS_I	This is macro PS_I.
	PS_RAM	This is macro PS_RAM.
	PS_ROM	This is macro PS_ROM.
	PSKEY_ANA_FREQ	This is macro PSKEY_ANA_FREQ.
	PSKEY_BDADDR	Select PS key definitions
	PSKEY_BLE_DEFAULT_TX_POWER	This is macro PSKEY_BLE_DEFAULT_TX_POWER.
	PSKEY_CLOCK_REQUEST_ENABLE	This is macro PSKEY_CLOCK_REQUEST_ENABLE.
	PSKEY_DEEP_SLEEP_CLEAR_RTS	This is macro PSKEY_DEEP_SLEEP_CLEAR_RTS.
	PSKEY_DEEP_SLEEP_EXTERNAL_CLOCK_SOURCE_PIO	This is macro PSKEY_DEEP_SLEEP_EXTERNAL_CLOCK_SOURCE_PIO.
	PSKEY_DEEP_SLEEP_STATE	This is macro PSKEY_DEEP_SLEEP_STATE.
	PSKEY_DEEP_SLEEP_USE_EXTERNAL_CLOCK	This is macro PSKEY_DEEP_SLEEP_USE_EXTERNAL_CLOCK.
	PSKEY_DEEP_SLEEP_WAKE_CTS	This is macro PSKEY_DEEP_SLEEP_WAKE_CTS.
	PSKEY_DIGITAL_AUDIO_BITS_PER_SAMPLE	This is macro PSKEY_DIGITAL_AUDIO_BITS_PER_SAMPLE.
	PSKEY_DIGITAL_AUDIO_CONFIG	This is macro PSKEY_DIGITAL_AUDIO_CONFIG.
	PSKEY_H_HC_FC_MAX_SCO_PKT_LEN	This is macro PSKEY_H_HC_FC_MAX_SCO_PKT_LEN.
	PSKEY_H_HC_FC_MAX_SCO_PKTS	This is macro PSKEY_H_HC_FC_MAX_SCO_PKTS.
	PSKEY_HCI_NOP_DISABLE	This is macro PSKEY_HCI_NOP_DISABLE.
	PSKEY_HOST_INTERFACE	This is macro PSKEY_HOST_INTERFACE.
	PSKEY_HOSTIO_MAP_SCO_PCM	This is macro PSKEY_HOSTIO_MAP_SCO_PCM.
	PSKEY_HOSTIO_UART_RESET_TIMEOUT	This is macro PSKEY_HOSTIO_UART_RESET_TIMEOUT.
	PSKEY_LC_DEFAULT_TX_POWER	This is macro PSKEY_LC_DEFAULT_TX_POWER.
	PSKEY_LC_MAX_TX_POWER	This is macro PSKEY_LC_MAX_TX_POWER.
	PSKEY_LC_MAX_TX_POWER_NO_RSSI	This is macro PSKEY_LC_MAX_TX_POWER_NO_RSSI.
	PSKEY_PCM_CLOCK_RATE	This is macro PSKEY_PCM_CLOCK_RATE.
	PSKEY_PCM_CONFIG32	This is macro PSKEY_PCM_CONFIG32.

	PSKEY_PCM_FORMAT	This is macro PSKEY_PCM_FORMAT.
	PSKEY_PCM_PULL_CONTROL	This is macro PSKEY_PCM_PULL_CONTROL.
	PSKEY_PCM_SYNC_RATE	This is macro PSKEY_PCM_SYNC_RATE.
	PSKEY_PCM_USE_LOW_JITTER_MODE	This is macro PSKEY_PCM_USE_LOW_JITTER_MODE.
	PSKEY_PIO_DEEP_SLEEP_EITHER_LEVEL	This is macro PSKEY_PIO_DEEP_SLEEP_EITHER_LEVEL.
	PSKEY_UART_BAUDRATE	This is macro PSKEY_UART_BAUDRATE.
	PSKEY_UART_BITRATE	This is macro PSKEY_UART_BITRATE.
	PSKEY_UART_CONFIG_BCSP	This is macro PSKEY_UART_CONFIG_BCSP.
	PSKEY_UART_CONFIG_H4	This is macro PSKEY_UART_CONFIG_H4.
	PSKEY_UART_CONFIG_H4DS	This is macro PSKEY_UART_CONFIG_H4DS.
	PSKEY_UART_CONFIG_H5	This is macro PSKEY_UART_CONFIG_H5.
	PSKEY_UART_TX_WINDOW_SIZE	This is macro PSKEY_UART_TX_WINDOW_SIZE.
	PSKEY_VM_DISABLE	This is macro PSKEY_VM_DISABLE.
	SET_PS_VALUE_BDADDR	This is macro SET_PS_VALUE_BDADDR.
	SET_PS_VALUE_UINT16	Macros for defining lists of PS values for use with btx_csr_set_ps_vars().
	SET_PS_VALUE_UINT32	This is macro SET_PS_VALUE_UINT32.

Structures

	Name	Description
	_btx_csr_autobaud_buffer_t	This is record _btx_csr_autobaud_buffer_t.
	_btx_csr_bccmd_header_s	This is type btx_csr_bccmd_header_t.
	_btx_csr_bccmd_listener_t	This is record _btx_csr_bccmd_listener_t.
	_btx_csr_cached_temperature_s	This is type btx_csr_cached_temperature_t.
	_btx_csr_create_operator_c_s	This is type btx_csr_create_operator_c_t.
	_btx_csr_exec_hq_script_buffer_t	This is record _btx_csr_exec_hq_script_buffer_t.
	_btx_csr_exec_script_buffer_t	This is record _btx_csr_exec_script_buffer_t.
	_btx_csr_pio_direction_mask_s	This is type btx_csr_pio_direction_mask_t.
	_btx_csr_pio_protection_mask_s	This is type btx_csr_pio_protection_mask_t.
	_btx_csr_pio_s	This is type btx_csr_pio_t.
	_btx_csr_rssi_acl_s	This is type btx_csr_rssi_acl_t.
	_btx_csr_script_t	This is type btx_csr_script_t.
	_btx_csr_set_ps_vars_buffer_t	This is record _btx_csr_set_ps_vars_buffer_t.
	_btx_csr_strm_connect_s	This is type btx_csr_strm_connect_t.
	_btx_csr_strm_get_sink_s	This is type btx_csr_strm_get_sink_t.
	_btx_csr_strm_get_source_s	This is type btx_csr_strm_get_source_t.
	btx_csr_bccmd_header_t	This is type btx_csr_bccmd_header_t.
	btx_csr_cached_temperature_t	This is type btx_csr_cached_temperature_t.
	btx_csr_create_operator_c_t	This is type btx_csr_create_operator_c_t.
	btx_csr_pio_direction_mask_t	This is type btx_csr_pio_direction_mask_t.
	btx_csr_pio_protection_mask_t	This is type btx_csr_pio_protection_mask_t.
	btx_csr_pio_t	This is type btx_csr_pio_t.
	btx_csr_rssi_acl_t	This is type btx_csr_rssi_acl_t.
	btx_csr_script_t	This is type btx_csr_script_t.
	btx_csr_strm_connect_t	This is type btx_csr_strm_connect_t.
	btx_csr_strm_get_sink_t	This is type btx_csr_strm_get_sink_t.
	btx_csr_strm_get_source_t	This is type btx_csr_strm_get_source_t.

Types

	Name	Description
	btx_csr_autobaud_buffer_t	This is type btx_csr_autobaud_buffer_t.
	btx_csr_autobaud_callback_fp	This is type btx_csr_autobaud_callback_fp.
	btx_csr_bccmd_callback_fp	This is type btx_csr_bccmd_callback_fp.
	btx_csr_bccmd_listener_t	This is type btx_csr_bccmd_listener_t.
	btx_csr_exec_hq_script_buffer_t	This is type btx_csr_exec_hq_script_buffer_t.

	btx_csr_exec_hq_script_callback_fp	This is type btx_csr_exec_hq_script_callback_fp.
	btx_csr_exec_script_buffer_t	This is type btx_csr_exec_script_buffer_t.
	btx_csr_exec_script_callback_fp	This is type btx_csr_exec_script_callback_fp.
	btx_csr_get_ps_var_callback_fp	This is type btx_csr_get_ps_var_callback_fp.
	btx_csr_get_script_fp	This is type btx_csr_get_script_fp.
	btx_csr_get_var_callback_fp	This is type btx_csr_get_var_callback_fp.
	btx_csr_set_ps_vars_buffer_t	This is type btx_csr_set_ps_vars_buffer_t.
	btx_csr_set_ps_vars_callback_fp	This is type btx_csr_set_ps_vars_callback_fp.
	btx_csr_set_var_callback_fp	This is type btx_csr_set_var_callback_fp.

Unions

	Name	Description
	_btx_csr_var_u	This is type btx_csr_var_t.
	btx_csr_var_t	This is type btx_csr_var_t.

Description

Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.

frame_buffer.h

Functions

	Name	Description
	bt_l2cap_alloc_frame_buffer	This is function bt_l2cap_alloc_frame_buffer.
	bt_l2cap_free_frame_buffer	This is function bt_l2cap_free_frame_buffer.
	bt_l2cap_init_frame_buffers	This is function bt_l2cap_init_frame_buffers.

Macros

	Name	Description
	L2CAP_MAX_FRAME_BUFFERS	This is macro L2CAP_MAX_FRAME_BUFFERS.

Description

Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.

SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements.

Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.

gap.h

Functions

	Name	Description
	bt_gap_find_devices	This is function bt_gap_find_devices.

Structures

	Name	Description
	_bt_device_t	This is type bt_device_t.
	bt_device_t	This is type bt_device_t.

Types

	Name	Description
	bt_find_devices_callback_fp	This is type bt_find_devices_callback_fp.

Description

Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.

hci.h

Functions

	Name	Description
	bt_hci_add_param_bdaddr	brief Add BD address parameter to an HCI command ingroup hci
	bt_hci_add_param_byte	brief Add byte parameter to an HCI command ingroup hci
	bt_hci_add_param_cod	brief Add class of device parameter to an HCI command ingroup hci
	bt_hci_add_param_int	brief Add int parameter to an HCI command ingroup hci
	bt_hci_add_param_linkkey	brief Add link key parameter to an HCI command ingroup hci
	bt_hci_add_param_long	brief Add long parameter to an HCI command ingroup hci
	bt_hci_add_param_string	brief Add string parameter to an HCI command ingroup hci
	bt_hci_alloc_canned_command	brief Allocate and initialize an HCI command structure for a canned (pre-formatted) command. ingroup hci
	bt_hci_alloc_command	brief Allocate and initialize an HCI command structure. ingroup hci
	bt_hci_authenticate_ex	This is function bt_hci_authenticate_ex.
	bt_hci_cancel_find_devices	brief Stop inquiry
	bt_hci_cancel_find_devices_ex	brief Stop inquiry
	bt_hci_cancel_request_remote_name	brief Cancel remote device name request
	bt_hci_cancel_send_acl_data	brief Cancel sending data over ACL connection
	bt_hci_connect	brief Connect to a remote device. ingroup hci details This function tries to establish an HCI connection with a remote device specified by the Bluetooth address c dest. Upon completion, the callback function specified by the c callback parameter is called. param dest Bluetooth address of the remote device. param packet_type param role_switch param acl_config param callback Pointer to a callback function that is called when the connect operation completes. param param Pointer to arbitrary data that is to be passed to the callback function. return li c TRUE when the function succeeds. li c FALSE otherwise. The callback function... more
	bt_hci_connect_sco	This is function bt_hci_connect_sco.
	bt_hci_disconnect	brief Abort connection ingroup hci
	bt_hci_exit_park_state	brief Exit park state
	bt_hci_exit_sniff_mode_ex	This is function bt_hci_exit_sniff_mode_ex.
	bt_hci_find_devices	brief Start inquiry
	bt_hci_find_devices_ex	This is function bt_hci_find_devices_ex.
	bt_hci_free_command	brief Free HCI command. ingroup hci
	bt_hci_get_evt_param_bdaddr	brief Get bd address parameter from HCI event ingroup hci
	bt_hci_get_evt_param_byte	brief Get byte parameter from HCI event ingroup hci
	bt_hci_get_evt_param_devclass	brief Get class of device parameter from HCI event ingroup hci
	bt_hci_get_evt_param_int	brief Get int parameter from HCI event ingroup hci
	bt_hci_get_evt_param_linkkey	brief Get link key parameter from HCI event ingroup hci
	bt_hci_get_evt_param_long	brief Get long parameter from HCI event ingroup hci
	bt_hci_get_evt_param_uint	brief Get unsigned int parameter from HCI event ingroup hci
	bt_hci_get_evt_param_ulong	brief Get unsigned long parameter from HCI event ingroup hci
	bt_hci_get_inquiry_response_tx_power_level	This is function bt_hci_get_inquiry_response_tx_power_level.
	bt_hci_get_last_cmd_status	This is function bt_hci_get_last_cmd_status.
	bt_hci_get_param_bdaddr	brief Get BD address parameter from HCI command ingroup hci
	bt_hci_get_param_byte	brief Get byte parameter from HCI command ingroup hci
	bt_hci_get_param_int	brief Get int parameter from HCI command ingroup hci
	bt_hci_get_param_linkkey	brief Get link key parameter from HCI command ingroup hci
	bt_hci_get_param_long	brief Get long parameter from HCI command ingroup hci
	bt_hci_get_recv_buffer	This is function bt_hci_get_recv_buffer.
	bt_hci_get_recv_buffer_len	This is function bt_hci_get_recv_buffer_len.
	bt_hci_get_send_buffer	This is function bt_hci_get_send_buffer.
	bt_hci_get_send_buffer_len	This is function bt_hci_get_send_buffer_len.

	bt_hci_init	brief Initialize the HCI layer. ingroup hci This function initializes all internal variables of the HCI layer. The application, unless it's going to use only HCI layer, does not need to call this function as it is implicitly called by bt_sys_init. This function essentially calls bt_hci_init_ex(HCI_LINK_POLICY_ENABLE_ALL) so all link policy setting are enabled.
	bt_hci_init_ex	brief Initialize the HCI layer. ingroup hci This function initializes all internal variables of the HCI layer. The application, unless it's going to use only HCI layer, does not need to call this function as it is implicitly called by bt_sys_init_ex. @param default_link_policy default link policy settings. This is a bitmask that defines the initial value of the link policy settings for all new BR/EDR connections. This value can be a combination of the following values: li HCI_LINK_POLICY_ENABLE_ROLE_SWITCH li HCI_LINK_POLICY_ENABLE_HOLD_MODE li HCI_LINK_POLICY_ENABLE_SNIFF_MODE li HCI_LINK_POLICY_ENABLE_PARK_STATE To enable all settings pass HCI_LINK_POLICY_ENABLE_ALL.
	bt_hci_listen	brief Listen for incoming connections. ingroup hci details param callback Pointer to a callback function that is called when a new incoming connection has been established. param param Pointer to arbitrary data that is to be passed to the callback function. return li c TRUE when the function succeeds. li c FALSE otherwise. The callback function is not called in this case.
	bt_hci_listen_sco	This is function bt_hci_listen_sco.
	bt_hci_park_state	brief Put local device to park state
	bt_hci_read_inquiry_mode	brief Get current inquiry mode
	bt_hci_read_inquiry_scan_activity	brief Get current inquiry scan activity configuration
	bt_hci_read_inquiry_scan_type	brief Get current inquiry scan type
	bt_hci_read_page_scan_activity	brief Get current page scan activity configuration
	bt_hci_read_page_scan_period_mode	brief Get current page scan period mode
	bt_hci_read_page_scan_type	brief Get current page scan type
	bt_hci_read_page_timeout	brief Get current page timeout
	bt_hci_reject_pin_code	This is function bt_hci_reject_pin_code.
	bt_hci_request_remote_name	brief Request remote device's name
	bt_hci_reset	brief Reset controller. ingroup hci details This function resets the BT controller. param callback Completion callback. Called when the controller has been reset. param callback_param A pointer to arbitrary data to be passed to the c callback callback.
	bt_hci_role_change_ex	brief Change local device's role
	bt_hci_send_acl_data	brief Send data over ACL connection
	bt_hci_send_cmd	brief Send a command to local device
	bt_hci_send_linkkey	This is function bt_hci_send_linkkey.
	bt_hci_send_pin_code_ex	This is function bt_hci_send_pin_code_ex.
	bt_hci_send_sco_data	brief Send data over SCO connection
	bt_hci_set_encryption_ex	brief Set connection encryption
	bt_hci_set_incoming_connection_role	This is function bt_hci_set_incoming_connection_role.
	bt_hci_set_scan	This is function bt_hci_set_scan.
	bt_hci_set_scan_ex	This is function bt_hci_set_scan_ex.
	bt_hci_sniff_mode_ex	This is function bt_hci_sniff_mode_ex.
	bt_hci_sniff_subrating_ex	This is function bt_hci_sniff_subrating_ex.

	bt_hci_start	brief Start HCI layer. ingroup hci details This function starts the HCI layer of the stack. Starting the HCI layer consists essentially of two steps: 1. Make the HCI transport receive packets from the controller. This results in a call to bt_oem_recv. 2. Reset and configure the controller. Upon completion of controller initialization the callback function passed in the c callback parameter is called. param callback Completion callback. Called when controller initialization is complete. param callback_param A pointer to arbitrary data to be passed to the c callback callback. param enable_scan This is a bitmask that defines which scans are enabled... more
	bt_hci_start_no_init	brief Start HCI layer without controller configuration. ingroup hci details This function is similar to bt_hci_start but unlike the former it does not perform the controller configuration. I.e., bt_hci_start_no_init simply calls the HCI transport and makes it receive packets from the controller. The main purpose of this function is make the HCI transport ready to exchange packets if controller needs some vendor specific configuration before it can be used with the stack. E.g., controllers based on CRS8811 chip need loading various values that configure its operating mode using CSR's proprietary protocol. So the application after configuring the HCI transport would... more
	bt_hci_stop	brief Stop HCI layer. ingroup hci details This function makes the HCI layer inoperable. After this call the application must perform the full reset of the HCI transport and stack. param callback Completion callback. Called when the HCI layer has been stopped. param callback_param A pointer to arbitrary data to be passed to the c callback callback.
	bt_hci_write_default_link_policy_settings	This is function bt_hci_write_default_link_policy_settings.
	bt_hci_write_inquiry_mode	brief Configure inquiry mode
	bt_hci_write_inquiry_scan_activity	brief Configure inquiry scan activity
	bt_hci_write_inquiry_scan_type	brief Configure inquiry scan type
	bt_hci_write_link_policy_settings	This is function bt_hci_write_link_policy_settings.
	bt_hci_write_local_name	brief Write local device name ingroup hci details The Write_Local_Name command provides the ability to modify the userfriendly name for the BR/EDR Controller.
	bt_hci_write_local_name_ex	This is function bt_hci_write_local_name_ex.
	bt_hci_write_page_scan_activity	brief Configure page scan activity
	bt_hci_write_page_scan_period_mode	brief Configure page scan period mode
	bt_hci_write_page_scan_type	brief Configure page scan type
	bt_hci_write_page_timeout	brief Configure page timeout
	hci_sleep	brief Put local device to sleep
	hci_wakeup	brief Awaken local device

Macros

	Name	Description
	bt_hci_add_param_hconn	brief Add connection handle parameter to an HCI command ingroup hci
	bt_hci_add_param_lap	This is macro bt_hci_add_param_lap.
	bt_hci_add_param_uint	brief Add unsigned int parameter to an HCI command ingroup hci
	bt_hci_add_param_ulong	brief Add unsigned long parameter to an HCI command ingroup hci
	bt_hci_authenticate	This is macro bt_hci_authenticate.
	bt_hci_exit_sniff_mode	brief Cancel sniff mode
	bt_hci_get_evt_param_hconn	brief Get connection handle parameter from HCI event ingroup hci
	bt_hci_get_evt_param_uint	This is macro bt_hci_get_evt_param_uint.
	bt_hci_get_evt_param_ulong	This is macro bt_hci_get_evt_param_ulong.
	bt_hci_get_param_hconn	brief Get connection handle parameter from HCI command ingroup hci
	bt_hci_role_change	This is macro bt_hci_role_change.
	bt_hci_send_pin_code	This is macro bt_hci_send_pin_code.

	bt_hci_set_encryption	This is macro bt_hci_set_encryption.
	bt_hci_sniff_mode	brief Put local device to sniff mode
	bt_hci_sniff_subrating	This is macro bt_hci_sniff_subrating.
	HCI_ACCEPT_CONNECTION_REQUEST	This is macro HCI_ACCEPT_CONNECTION_REQUEST.
	HCI_ACCEPT_SYNCH_CONNECTION_REQUEST	This is macro HCI_ACCEPT_SYNCH_CONNECTION_REQUEST.
	HCI_ACL_DATA_HEADER_LEN	This is macro HCI_ACL_DATA_HEADER_LEN.
	HCI_AUTHENTICATION_REQUESTED	This is macro HCI_AUTHENTICATION_REQUESTED.
	HCI_BB_PACKET_TYPE_DH1	This is macro HCI_BB_PACKET_TYPE_DH1.
	HCI_BB_PACKET_TYPE_DH3	This is macro HCI_BB_PACKET_TYPE_DH3.
	HCI_BB_PACKET_TYPE_DH5	This is macro HCI_BB_PACKET_TYPE_DH5.
	HCI_BB_PACKET_TYPE_DM1	This is macro HCI_BB_PACKET_TYPE_DM1.
	HCI_BB_PACKET_TYPE_DM3	This is macro HCI_BB_PACKET_TYPE_DM3.
	HCI_BB_PACKET_TYPE_DM5	This is macro HCI_BB_PACKET_TYPE_DM5.
	HCI_BB_PACKET_TYPE_NO_2_DH1	This is macro HCI_BB_PACKET_TYPE_NO_2_DH1.
	HCI_BB_PACKET_TYPE_NO_2_DH3	This is macro HCI_BB_PACKET_TYPE_NO_2_DH3.
	HCI_BB_PACKET_TYPE_NO_2_DH5	This is macro HCI_BB_PACKET_TYPE_NO_2_DH5.
	HCI_BB_PACKET_TYPE_NO_3_DH1	This is macro HCI_BB_PACKET_TYPE_NO_3_DH1.
	HCI_BB_PACKET_TYPE_NO_3_DH3	This is macro HCI_BB_PACKET_TYPE_NO_3_DH3.
	HCI_BB_PACKET_TYPE_NO_3_DH5	This is macro HCI_BB_PACKET_TYPE_NO_3_DH5.
	HCI_C2H_BROADCAST_NOT_PARCKED_SLAVE	This is macro HCI_C2H_BROADCAST_NOT_PARCKED_SLAVE.
	HCI_C2H_BROADCAST_P2P	This is macro HCI_C2H_BROADCAST_P2P.
	HCI_C2H_BROADCAST_PARCKED_SLAVE	This is macro HCI_C2H_BROADCAST_PARCKED_SLAVE.
	HCI_C2H_BROADCAST_RESERVED	This is macro HCI_C2H_BROADCAST_RESERVED.
	HCI_CHANGE_CONNECTION_LINK_KEY	This is macro HCI_CHANGE_CONNECTION_LINK_KEY.
	HCI_CHANGE_CONNECTION_PACKET_TYPE	This is macro HCI_CHANGE_CONNECTION_PACKET_TYPE.
	HCI_CMD_HEADER_LEN	This is macro HCI_CMD_HEADER_LEN.
	HCI_CMD_STATUS_BEING_SENT	This is macro HCI_CMD_STATUS_BEING_SENT.
	HCI_CMD_STATUS_PENDING	This is macro HCI_CMD_STATUS_PENDING.
	HCI_CMD_STATUS_WAITING_RESPONSE	This is macro HCI_CMD_STATUS_WAITING_RESPONSE.
	HCI_CONFIG_BECOME_MASTER	This is macro HCI_CONFIG_BECOME_MASTER.
	HCI_CONFIG_ENABLE_AUTHENTICATION	This is macro HCI_CONFIG_ENABLE_AUTHENTICATION.
	HCI_CONFIG_ENABLE_ENCRYPTION	This is macro HCI_CONFIG_ENABLE_ENCRYPTION.
	HCI_CONNECTABLE	This is macro HCI_CONNECTABLE.
	HCI_CONTROLLER	This is macro HCI_CONTROLLER.
	HCI_CREATE_CONNECTION	This is macro HCI_CREATE_CONNECTION.
	HCI_CREATE_CONNECTION_CANCEL	This is macro HCI_CREATE_CONNECTION_CANCEL.
	HCI_CREATE_NEW_UNIT_KEY	This is macro HCI_CREATE_NEW_UNIT_KEY.
	HCI_DATA_STATUS_BEING_SENT	This is macro HCI_DATA_STATUS_BEING_SENT.
	HCI_DATA_STATUS_PENDING	This is macro HCI_DATA_STATUS_PENDING.
	HCI_DEFAULT_ACL_CONFIG	Default value for acl_config parameter of bt_hci_connect()
	HCI_DELETE_STORED_LINK_KEY	This is macro HCI_DELETE_STORED_LINK_KEY.
	HCI_DISCONNECT	This is macro HCI_DISCONNECT.
	HCI_DISCOVERABLE	This is macro HCI_DISCOVERABLE.
	HCI_DISCOVERABLE_MODE_GENERAL	This is macro HCI_DISCOVERABLE_MODE_GENERAL.
	HCI_DISCOVERABLE_MODE_LIMITED	This is macro HCI_DISCOVERABLE_MODE_LIMITED.
	HCI_ENABLE_DEVICE_UNDER_TEST_MODE	This is macro HCI_ENABLE_DEVICE_UNDER_TEST_MODE.
	HCI_ENCRYPTION_OFF	This is macro HCI_ENCRYPTION_OFF.

	HCI_ENCRYPTION_ON	This is macro HCI_ENCRYPTION_ON.
	HCI_ENHANCED_FLUSH	This is macro HCI_ENHANCED_FLUSH.
	HCI_EVT_ALL_HCI_EVENTS	Used in bt_hci_ctrl_register_event_listener to indicate that the listener is called for all HCI events (excluding dotstack internal ones)
	HCI_EVT_AUTHENTICATION_COMPLETE	This is macro HCI_EVT_AUTHENTICATION_COMPLETE.
	HCI_EVT_CHANGE_CONN_LINK_COMPLETE	This is macro HCI_EVT_CHANGE_CONN_LINK_COMPLETE.
	HCI_EVT_CMD_SEND_FINISHED	This is macro HCI_EVT_CMD_SEND_FINISHED.
	HCI_EVT_CMD_SEND_STARTED	dotstack internal events
	HCI_EVT_COMMAND_COMPLETE	This is macro HCI_EVT_COMMAND_COMPLETE.
	HCI_EVT_COMMAND_COMPLETE_PARAM_LEN	This is macro HCI_EVT_COMMAND_COMPLETE_PARAM_LEN.
	HCI_EVT_COMMAND_STATUS	This is macro HCI_EVT_COMMAND_STATUS.
	HCI_EVT_CONN_PACKET_TYPE_CHANGED	This is macro HCI_EVT_CONN_PACKET_TYPE_CHANGED.
	HCI_EVT_CONNECTION_COMPLETE	This is macro HCI_EVT_CONNECTION_COMPLETE.
	HCI_EVT_CONNECTION_REQUEST	This is macro HCI_EVT_CONNECTION_REQUEST.
	HCI_EVT_DATA_BUFFER_OVERFLOW	This is macro HCI_EVT_DATA_BUFFER_OVERFLOW.
	HCI_EVT_DISCONNECTION_COMPLETE	This is macro HCI_EVT_DISCONNECTION_COMPLETE.
	HCI_EVT_ENCRYPTION_CHANGE	This is macro HCI_EVT_ENCRYPTION_CHANGE.
	HCI_EVT_ENCRYPTION_KEY_REFRESH_COMPLETE	This is macro HCI_EVT_ENCRYPTION_KEY_REFRESH_COMPLETE.
	HCI_EVT_ENHANCED_FLUSH_COMPLETE	This is macro HCI_EVT_ENHANCED_FLUSH_COMPLETE.
	HCI_EVT_EXTENDED_INQUIRY_RESULT	This is macro HCI_EVT_EXTENDED_INQUIRY_RESULT.
	HCI_EVT_FIRST	This is macro HCI_EVT_FIRST.
	HCI_EVT_FLOW_SPECIFICATION_COMPLETE	This is macro HCI_EVT_FLOW_SPECIFICATION_COMPLETE.
	HCI_EVT_FLUSH_OCCURED	This is macro HCI_EVT_FLUSH_OCCURED.
	HCI_EVT_HARDWARE_ERROR	This is macro HCI_EVT_HARDWARE_ERROR.
	HCI_EVT_INQUIRY_COMPLETE	• addtogroup hci • @{ • @name Events
	HCI_EVT_INQUIRY_RESULT	This is macro HCI_EVT_INQUIRY_RESULT.
	HCI_EVT_INQUIRY_RESULT_WITH_RSSI	This is macro HCI_EVT_INQUIRY_RESULT_WITH_RSSI.
	HCI_EVT_IO_CAPABILITY_REQUEST	This is macro HCI_EVT_IO_CAPABILITY_REQUEST.
	HCI_EVT_IO_CAPABILITY_RESPONSE	This is macro HCI_EVT_IO_CAPABILITY_RESPONSE.
	HCI_EVT_KEYPRESS_NOTIFICATION	This is macro HCI_EVT_KEYPRESS_NOTIFICATION.
	HCI_EVT_LAST	This is macro HCI_EVT_LAST.
	HCI_EVT_LE_META_EVENT	This is macro HCI_EVT_LE_META_EVENT.
	HCI_EVT_LINK_IS_BUSY	This is macro HCI_EVT_LINK_IS_BUSY.
	HCI_EVT_LINK_IS_IDLE	This is macro HCI_EVT_LINK_IS_IDLE.
	HCI_EVT_LINK_KEY_NOTIFICATION	This is macro HCI_EVT_LINK_KEY_NOTIFICATION.
	HCI_EVT_LINK_KEY_REQUEST	This is macro HCI_EVT_LINK_KEY_REQUEST.
	HCI_EVT_LINK_SUPERVISION_TO_CHANGED	This is macro HCI_EVT_LINK_SUPERVISION_TO_CHANGED.
	HCI_EVT_LOOPBACK_COMMAND	This is macro HCI_EVT_LOOPBACK_COMMAND.
	HCI_EVT_MASTER_LINK_KEY_COMPLETE	This is macro HCI_EVT_MASTER_LINK_KEY_COMPLETE.
	HCI_EVT_MAX_SLOTS_CHANGE	This is macro HCI_EVT_MAX_SLOTS_CHANGE.
	HCI_EVT_MODE_CHANGE	This is macro HCI_EVT_MODE_CHANGE.
	HCI_EVT_NUM_OF_COMPLETED_PACKETS	This is macro HCI_EVT_NUM_OF_COMPLETED_PACKETS.

	HCI_EVT_PAGE_SCAN_REPET_MODE_CHANGE	This is macro HCI_EVT_PAGE_SCAN_REPET_MODE_CHANGE.
	HCI_EVT_PIN_CODE_REQUEST	This is macro HCI_EVT_PIN_CODE_REQUEST.
	HCI_EVT_QOS_SETUP_COMPLETE	This is macro HCI_EVT_QOS_SETUP_COMPLETE.
	HCI_EVT_QOS_VIOLATION	This is macro HCI_EVT_QOS_VIOLATION.
	HCI_EVT_READ_CLOCK_OFFSET_COMPLETE	This is macro HCI_EVT_READ_CLOCK_OFFSET_COMPLETE.
	HCI_EVT_READ_RMT_EXT_FEATURES_COMP	This is macro HCI_EVT_READ_RMT_EXT_FEATURES_COMP.
	HCI_EVT_READ_RMT_SUP_FEATURES_COMP	This is macro HCI_EVT_READ_RMT_SUP_FEATURES_COMP.
	HCI_EVT_READ_RMT_VERSION_INFO_COMP	This is macro HCI_EVT_READ_RMT_VERSION_INFO_COMP.
	HCI_EVT_REMOTE_NAME_REQUEST_COMPLETE	This is macro HCI_EVT_REMOTE_NAME_REQUEST_COMPLETE.
	HCI_EVT_REMOTE_OOB_DATA_REQUEST	This is macro HCI_EVT_REMOTE_OOB_DATA_REQUEST.
	HCI_EVT_RETURN_LINK_KEYS	This is macro HCI_EVT_RETURN_LINK_KEYS.
	HCI_EVT_RMT_HOST_SUPP_FEATURES_NTF	This is macro HCI_EVT_RMT_HOST_SUPP_FEATURES_NTF.
	HCI_EVT_ROLE_CHANGE	This is macro HCI_EVT_ROLE_CHANGE.
	HCI_EVT_SIMPLE_PAIRING_COMPLETE	This is macro HCI_EVT_SIMPLE_PAIRING_COMPLETE.
	HCI_EVT_SNIFF_SUBRATING	This is macro HCI_EVT_SNIFF_SUBRATING.
	HCI_EVT_SYNCH_CONNECTION_CHANGED	This is macro HCI_EVT_SYNCH_CONNECTION_CHANGED.
	HCI_EVT_SYNCH_CONNECTION_COMPLETE	This is macro HCI_EVT_SYNCH_CONNECTION_COMPLETE.
	HCI_EVT_USER_CONFIRMATION_REQUEST	This is macro HCI_EVT_USER_CONFIRMATION_REQUEST.
	HCI_EVT_USER_PASSKEY_NOTIFICATION	This is macro HCI_EVT_USER_PASSKEY_NOTIFICATION.
	HCI_EVT_USER_PASSKEY_REQUEST	This is macro HCI_EVT_USER_PASSKEY_REQUEST.
	HCI_EXIT_PARK_STATE	This is macro HCI_EXIT_PARK_STATE.
	HCI_EXIT_PERIODIC_INQUIRY_MODE	This is macro HCI_EXIT_PERIODIC_INQUIRY_MODE.
	HCI_EXIT_SNIFF_MODE	This is macro HCI_EXIT_SNIFF_MODE.
	HCI_FLOW_SPECIFICATION	This is macro HCI_FLOW_SPECIFICATION.
	HCI_FLUSH	This is macro HCI_FLUSH.
	HCI_H2C_BROADCAST_ACTIVE_SLAVE	This is macro HCI_H2C_BROADCAST_ACTIVE_SLAVE.
	HCI_H2C_BROADCAST_NO_BROADCATS	This is macro HCI_H2C_BROADCAST_NO_BROADCATS.
	HCI_H2C_BROADCAST_PARCKED_SLAVE	This is macro HCI_H2C_BROADCAST_PARCKED_SLAVE.
	HCI_H2C_BROADCAST_RESERVED	This is macro HCI_H2C_BROADCAST_RESERVED.
	HCI_HOLD_MODE	addtogroup hci @{ @name Link policy commands details The Link Policy Commands provide methods for the Host to affect how the Link Manager manages the piconet.
	HCI_HOST_BUFFER_SIZE	This is macro HCI_HOST_BUFFER_SIZE.
	HCI_HOST_NUM_OF_COMPLETED_PACKETS	This is macro HCI_HOST_NUM_OF_COMPLETED_PACKETS.
	HCI_INQUIRY	addtogroup hci @{ @name Link control commands details The Link Control commands allow a Controller to control connections to other BR/EDR Controllers.
	HCI_INQUIRY_CANCEL	This is macro HCI_INQUIRY_CANCEL.
	HCI_INQUIRY_MODE_EXTENDED	This is macro HCI_INQUIRY_MODE_EXTENDED.
	HCI_INQUIRY_MODE_STANDARD	This is macro HCI_INQUIRY_MODE_STANDARD.
	HCI_INQUIRY_MODE_WITH_RSSI	This is macro HCI_INQUIRY_MODE_WITH_RSSI.

	HCI_IO_CAPABILITY_REQUEST_NEGATIVE_REPLY	This is macro HCI_IO_CAPABILITY_REQUEST_NEGATIVE_REPLY.
	HCI_IO_CAPABILITY_REQUEST_REPLY	This is macro HCI_IO_CAPABILITY_REQUEST_REPLY.
	HCI_LE_ADD_DEVICE_TO_WHITE_LIST	This is macro HCI_LE_ADD_DEVICE_TO_WHITE_LIST.
	HCI_LE_CLEAR_WHITE_LIST	This is macro HCI_LE_CLEAR_WHITE_LIST.
	HCI_LE_CONNECTION_UPDATE	This is macro HCI_LE_CONNECTION_UPDATE.
	HCI_LE_CREATE_CONNECTION	This is macro HCI_LE_CREATE_CONNECTION.
	HCI_LE_CREATE_CONNECTION_CANCEL	This is macro HCI_LE_CREATE_CONNECTION_CANCEL.
	HCI_LE_ENCRYPT	This is macro HCI_LE_ENCRYPT.
	HCI_LE_LONG_TERM_KEY_REQUEST_NEGATIVE_REPLY	This is macro HCI_LE_LONG_TERM_KEY_REQUEST_NEGATIVE_REPLY.
	HCI_LE_LONG_TERM_KEY_REQUEST_REPLY	This is macro HCI_LE_LONG_TERM_KEY_REQUEST_REPLY.
	HCI_LE_MAX_AD_LEN	This is macro HCI_LE_MAX_AD_LEN.
	HCI_LE_RAND	This is macro HCI_LE_RAND.
	HCI_LE_READ_ADVERTISING_CHANNEL_TX_POWER	This is macro HCI_LE_READ_ADVERTISING_CHANNEL_TX_POWER.
	HCI_LE_READ_BUFFER_SIZE	This is macro HCI_LE_READ_BUFFER_SIZE.
	HCI_LE_READ_CHANNEL_MAP	This is macro HCI_LE_READ_CHANNEL_MAP.
	HCI_LE_READ_LOCAL_SUPPORTED_FEATURES	This is macro HCI_LE_READ_LOCAL_SUPPORTED_FEATURES.
	HCI_LE_READ_REMOTE_USED_FEATURES	This is macro HCI_LE_READ_REMOTE_USED_FEATURES.
	HCI_LE_READ_SUPPORTED_STATES	This is macro HCI_LE_READ_SUPPORTED_STATES.
	HCI_LE_READ_WHITE_LIST_SIZE	This is macro HCI_LE_READ_WHITE_LIST_SIZE.
	HCI_LE_RECEIVE_TEST	This is macro HCI_LE_RECEIVE_TEST.
	HCI_LE_REMOVE_DEVICE_FROM_WHITE_LIST	This is macro HCI_LE_REMOVE_DEVICE_FROM_WHITE_LIST.
	HCI_LE_SET_ADVERTISE_ENABLE	This is macro HCI_LE_SET_ADVERTISE_ENABLE.
	HCI_LE_SET_ADVERTISING_DATA	This is macro HCI_LE_SET_ADVERTISING_DATA.
	HCI_LE_SET_ADVERTISING_PARAMETERS	This is macro HCI_LE_SET_ADVERTISING_PARAMETERS.
	HCI_LE_SET_EVENT_MASK	addtogroup hci @{ @name LE controller commands details The LE Controller Commands provide access and control to various capabilities of the Bluetooth hardware, as well as methods for the Host to affect how the Link Layer manages the piconet, and controls connections.
	HCI_LE_SET_HOST_CHANNEL_CLASSIFICATION	This is macro HCI_LE_SET_HOST_CHANNEL_CLASSIFICATION.
	HCI_LE_SET_RANDOM_ADDRESS	This is macro HCI_LE_SET_RANDOM_ADDRESS.
	HCI_LE_SET_SCAN_ENABLE	This is macro HCI_LE_SET_SCAN_ENABLE.
	HCI_LE_SET_SCAN_PARAMETERS	This is macro HCI_LE_SET_SCAN_PARAMETERS.
	HCI_LE_SET_SCAN_RESPONSE_DATA	This is macro HCI_LE_SET_SCAN_RESPONSE_DATA.
	HCI_LE_START_ENCRYPTION	This is macro HCI_LE_START_ENCRYPTION.
	HCI_LE_TEST_END	This is macro HCI_LE_TEST_END.
	HCI_LE_TRANSMITTER_TEST	This is macro HCI_LE_TRANSMITTER_TEST.
	HCI_LINK_KEY_LEN	This is macro HCI_LINK_KEY_LEN.
	HCI_LINK_KEY_REQUEST_NEGATIVE_REPLY	This is macro HCI_LINK_KEY_REQUEST_NEGATIVE_REPLY.
	HCI_LINK_KEY_REQUEST_REPLY	This is macro HCI_LINK_KEY_REQUEST_REPLY.
	HCI_LINK_KEY_TYPE_COMBINATION	This is macro HCI_LINK_KEY_TYPE_COMBINATION.
	HCI_LINK_KEY_TYPE_LOCAL_UNIT	This is macro HCI_LINK_KEY_TYPE_LOCAL_UNIT.
	HCI_LINK_KEY_TYPE_REMOTE_UNIT	This is macro HCI_LINK_KEY_TYPE_REMOTE_UNIT.

	HCI_LINK_POLICY_ENABLE_ALL	This is macro HCI_LINK_POLICY_ENABLE_ALL.
	HCI_LINK_POLICY_ENABLE_HOLD_MODE	This is macro HCI_LINK_POLICY_ENABLE_HOLD_MODE.
	HCI_LINK_POLICY_ENABLE_PARK_STATE	This is macro HCI_LINK_POLICY_ENABLE_PARK_STATE.
	HCI_LINK_POLICY_ENABLE_ROLE_SWITCH	This is macro HCI_LINK_POLICY_ENABLE_ROLE_SWITCH.
	HCI_LINK_POLICY_ENABLE_SNIFF_MODE	This is macro HCI_LINK_POLICY_ENABLE_SNIFF_MODE.
	HCI_MASTER_LINK_KEY	This is macro HCI_MASTER_LINK_KEY.
	HCI_MAX_EVENT_PARAM_LEN	This is macro HCI_MAX_EVENT_PARAM_LEN.
	HCI_MAX_PARAM_LEN	This is macro HCI_MAX_PARAM_LEN.
	HCI_MAX_PIN_LENGTH	This is macro HCI_MAX_PIN_LENGTH.
	HCI_OPCODE	This is macro HCI_OPCODE.
	HCI_PACKET_BOUNDARY_COMPLETE	This is macro HCI_PACKET_BOUNDARY_COMPLETE.
	HCI_PACKET_BOUNDARY_CONTINUE	This is macro HCI_PACKET_BOUNDARY_CONTINUE.
	HCI_PACKET_BOUNDARY_FIRST	This is macro HCI_PACKET_BOUNDARY_FIRST.
	HCI_PACKET_BOUNDARY_FIRST_AUTO_FLUSH	This is macro HCI_PACKET_BOUNDARY_FIRST_AUTO_FLUSH.
	HCI_PACKET_BOUNDARY_FIRST_NO_AUTO_FLUSH	This is macro HCI_PACKET_BOUNDARY_FIRST_NO_AUTO_FLUSH .
	HCI_PACKET_TYPE_ACL_DATA	This is macro HCI_PACKET_TYPE_ACL_DATA.
	HCI_PACKET_TYPE_COMMAND	This is macro HCI_PACKET_TYPE_COMMAND.
	HCI_PACKET_TYPE_EVENT	This is macro HCI_PACKET_TYPE_EVENT.
	HCI_PACKET_TYPE_NONE	This is macro HCI_PACKET_TYPE_NONE.
	HCI_PACKET_TYPE_SCO_DATA	This is macro HCI_PACKET_TYPE_SCO_DATA.
	HCI_PAGE_SCAN_REPETITION_MODE_R0	This is macro HCI_PAGE_SCAN_REPETITION_MODE_R0.
	HCI_PAGE_SCAN_REPETITION_MODE_R1	This is macro HCI_PAGE_SCAN_REPETITION_MODE_R1.
	HCI_PAGE_SCAN_REPETITION_MODE_R2	This is macro HCI_PAGE_SCAN_REPETITION_MODE_R2.
	HCI_PARAM_LEN_BD_ADDR	This is macro HCI_PARAM_LEN_BD_ADDR.
	HCI_PARAM_LEN_BYTE	This is macro HCI_PARAM_LEN_BYTE.
	HCI_PARAM_LEN_DEV_CLASS	This is macro HCI_PARAM_LEN_DEV_CLASS.
	HCI_PARAM_LEN_HANDLE	This is macro HCI_PARAM_LEN_HANDLE.
	HCI_PARAM_LEN_INT	This is macro HCI_PARAM_LEN_INT.
	HCI_PARAM_LEN_LONG	This is macro HCI_PARAM_LEN_LONG.
	HCI_PARAM_TYPE_BD_ADDR	This is macro HCI_PARAM_TYPE_BD_ADDR.
	HCI_PARAM_TYPE_BYTE	This is macro HCI_PARAM_TYPE_BYTE.
	HCI_PARAM_TYPE_DEV_CLASS	This is macro HCI_PARAM_TYPE_DEV_CLASS.
	HCI_PARAM_TYPE_HANDLE	This is macro HCI_PARAM_TYPE_HANDLE.
	HCI_PARAM_TYPE_INT	This is macro HCI_PARAM_TYPE_INT.
	HCI_PARAM_TYPE_LONG	This is macro HCI_PARAM_TYPE_LONG.
	HCI_PARAM_TYPE_STRING	This is macro HCI_PARAM_TYPE_STRING.
	HCI_PARK_STATE	This is macro HCI_PARK_STATE.
	HCI_PERIODIC_INQUIRY_MODE	This is macro HCI_PERIODIC_INQUIRY_MODE.
	HCI_PIN_CODE_REQUEST_NEGATIVE_REPLY	This is macro HCI_PIN_CODE_REQUEST_NEGATIVE_REPLY.
	HCI_PIN_CODE_REQUEST_REPLY	This is macro HCI_PIN_CODE_REQUEST_REPLY.
	HCI_POWER_MODE_ACTIVE	This is macro HCI_POWER_MODE_ACTIVE.
	HCI_POWER_MODE_HOLD	This is macro HCI_POWER_MODE_HOLD.
	HCI_POWER_MODE_PARK	This is macro HCI_POWER_MODE_PARK.
	HCI_POWER_MODE_SNIFF	This is macro HCI_POWER_MODE_SNIFF.
	HCI_QOS_SETUP	This is macro HCI_QOS_SETUP.
	HCI_READ_AFH_CHANNEL_ASSESSMENT_MODE	This is macro HCI_READ_AFH_CHANNEL_ASSESSMENT_MODE.

	HCI_READ_AFH_CHANNEL_MAP	This is macro HCI_READ_AFH_CHANNEL_MAP.
	HCI_READ_AUTHENTICATION_ENABLE	This is macro HCI_READ_AUTHENTICATION_ENABLE.
	HCI_READ_AUTOMATIC_FLASH_TIMEOUT	This is macro HCI_READ_AUTOMATIC_FLASH_TIMEOUT.
	HCI_READ_BD_ADDR	This is macro HCI_READ_BD_ADDR.
	HCI_READ_BUFFER_SIZE	This is macro HCI_READ_BUFFER_SIZE.
	HCI_READ_CLASS_OF_DEVICE	This is macro HCI_READ_CLASS_OF_DEVICE.
	HCI_READ_CLOCK_COMMAND	This is macro HCI_READ_CLOCK_COMMAND.
	HCI_READ_CLOCK_OFFSET	This is macro HCI_READ_CLOCK_OFFSET.
	HCI_READ_CONNECTION_ACCEPT_TIMEOUT	This is macro HCI_READ_CONNECTION_ACCEPT_TIMEOUT.
	HCI_READ_CURRENT_IAC_LAP	This is macro HCI_READ_CURRENT_IAC_LAP.
	HCI_READ_DEFAULT_ERRONEOUS_DATA_REPORTING	This is macro HCI_READ_DEFAULT_ERRONEOUS_DATA_REPORTING.
	HCI_READ_DEFAULT_POLICY_SETTINGS	This is macro HCI_READ_DEFAULT_POLICY_SETTINGS.
	HCI_READ_ENCRYPTION_MODE	This is macro HCI_READ_ENCRYPTION_MODE.
	HCI_READ_EXTENDED_INQUIRY_RESPONSE	This is macro HCI_READ_EXTENDED_INQUIRY_RESPONSE.
	HCI_READ_FAILED_CONTACT_COUNTER	addtogroup hci @{ @name Status Parameters details The Controller modifies all status parameters. These parameters provide information about the current state of the Link Manager and Baseband in the BR/EDR Controller and the PAL in an AMP Controller. The host device cannot modify any of these parameters other than to reset certain specific parameters.
	HCI_READ_HOLD_MODE_ACTIVITY	This is macro HCI_READ_HOLD_MODE_ACTIVITY.
	HCI_READ_INQUIRY_MODE	This is macro HCI_READ_INQUIRY_MODE.
	HCI_READ_INQUIRY_RESPONSE_TX_POWER_LEVEL	This is macro HCI_READ_INQUIRY_RESPONSE_TX_POWER_LEVEL.
	HCI_READ_INQUIRY_SCAN_ACTIVITY	This is macro HCI_READ_INQUIRY_SCAN_ACTIVITY.
	HCI_READ_INQUIRY_SCAN_TYPE	This is macro HCI_READ_INQUIRY_SCAN_TYPE.
	HCI_READ_LE_HOST_SUPPORT	This is macro HCI_READ_LE_HOST_SUPPORT.
	HCI_READ_LINK_POLICY_SETTINGS	This is macro HCI_READ_LINK_POLICY_SETTINGS.
	HCI_READ_LINK_QUALITY	This is macro HCI_READ_LINK_QUALITY.
	HCI_READ_LINK_SUPERVISION_TIMEOUT	This is macro HCI_READ_LINK_SUPERVISION_TIMEOUT.
	HCI_READ_LMP_HANDLE	This is macro HCI_READ_LMP_HANDLE.
	HCI_READ_LOCAL_EXTENDED_FEATURES	This is macro HCI_READ_LOCAL_EXTENDED_FEATURES.
	HCI_READ_LOCAL_NAME	This is macro HCI_READ_LOCAL_NAME.
	HCI_READ_LOCAL_OOB_DATA	This is macro HCI_READ_LOCAL_OOB_DATA.
	HCI_READ_LOCAL_SUPPORTED_COMMANDS	This is macro HCI_READ_LOCAL_SUPPORTED_COMMANDS.
	HCI_READ_LOCAL_SUPPORTED_FEATURES	This is macro HCI_READ_LOCAL_SUPPORTED_FEATURES.
	HCI_READ_LOCAL_VERSION_INFORMATION	addtogroup hci @{ @name Informational Parameters details The Informational Parameters are fixed by the manufacturer of the Bluetooth hardware. These parameters provide information about the BR/EDR Controller and the capabilities of the Link Manager and Baseband in the BR/EDR Controller and PAL in the AMP Controller. The host device cannot modify any of these parameters.

	HCI_READ_LOOPBACK_MODE	addtogroup hci @{ @name Testing Commands details The Testing commands are used to provide the ability to test various functional capabilities of the Bluetooth hardware.
	HCI_READ_NUM_BROADCST_RETR	This is macro HCI_READ_NUM_BROADCST_RETR.
	HCI_READ_NUM_OF_SUPPORTED_IAC	This is macro HCI_READ_NUM_OF_SUPPORTED_IAC.
	HCI_READ_PAGE_SCAN_ACTIVITY	This is macro HCI_READ_PAGE_SCAN_ACTIVITY.
	HCI_READ_PAGE_SCAN_PERIOD_MODE	This is macro HCI_READ_PAGE_SCAN_PERIOD_MODE.
	HCI_READ_PAGE_SCAN_TYPE	This is macro HCI_READ_PAGE_SCAN_TYPE.
	HCI_READ_PAGE_TIMEOUT	This is macro HCI_READ_PAGE_TIMEOUT.
	HCI_READ_PIN_TYPE	This is macro HCI_READ_PIN_TYPE.
	HCI_READ_REFRESH_ENCRYPTION_KEY	This is macro HCI_READ_REFRESH_ENCRYPTION_KEY.
	HCI_READ_REMOTE_EXTENDED_FEATURES	This is macro HCI_READ_REMOTE_EXTENDED_FEATURES.
	HCI_READ_REMOTE_SUPPORTED_FEATURES	This is macro HCI_READ_REMOTE_SUPPORTED_FEATURES.
	HCI_READ_REMOTE_VERSION_INFORMATION	This is macro HCI_READ_REMOTE_VERSION_INFORMATION.
	HCI_READ_RSSI	This is macro HCI_READ_RSSI.
	HCI_READ_SCAN_ENABLE	This is macro HCI_READ_SCAN_ENABLE.
	HCI_READ_SIMPLE_PAIRING_MODE	This is macro HCI_READ_SIMPLE_PAIRING_MODE.
	HCI_READ_STORED_LINK_KEY	This is macro HCI_READ_STORED_LINK_KEY.
	HCI_READ_SYNC_FLOW_CONTROL_ENABLE	This is macro HCI_READ_SYNC_FLOW_CONTROL_ENABLE.
	HCI_READ_TRANSMIT_POWER_LEVEL	This is macro HCI_READ_TRANSMIT_POWER_LEVEL.
	HCI_READ_VOICE_SETTING	This is macro HCI_READ_VOICE_SETTING.
	HCI_REJECT_CONNECTION_REQUEST	This is macro HCI_REJECT_CONNECTION_REQUEST.
	HCI_REJECT_SYNCH_CONNECTION_REQUEST	This is macro HCI_REJECT_SYNCH_CONNECTION_REQUEST.
	HCI_REMOTE_NAME_REQUEST	This is macro HCI_REMOTE_NAME_REQUEST.
	HCI_REMOTE_NAME_REQUEST_CANCEL	This is macro HCI_REMOTE_NAME_REQUEST_CANCEL.
	HCI_REMOTE_OOB_DATA_REQUEST_NEGATIVE_REPLY	This is macro HCI_REMOTE_OOB_DATA_REQUEST_NEGATIVE_REPLY.
	HCI_REMOTE_OOB_DATA_REQUEST_REPLY	This is macro HCI_REMOTE_OOB_DATA_REQUEST_REPLY.
	HCI_RESET	This is macro HCI_RESET.
	HCI_RESET_FAILED_CONTACT_COUNTER	This is macro HCI_RESET_FAILED_CONTACT_COUNTER.
	HCI_ROLE_DISCOVERY	This is macro HCI_ROLE_DISCOVERY.
	HCI_ROLE_SWITCH_ALLOW	This is macro HCI_ROLE_SWITCH_ALLOW.
	HCI_ROLE_SWITCH_DISALLOW	This is macro HCI_ROLE_SWITCH_DISALLOW.
	HCI_SCAN_INQUIRY	Flags for HCI_READ_SCAN_ENABLE/HCI_WRITE_SCAN_ENABLE
	HCI_SCAN_PAGE	This is macro HCI_SCAN_PAGE.
	HCI_SCO_CONTENT_FORMAT_AIR_CODING_A_LAW	This is macro HCI_SCO_CONTENT_FORMAT_AIR_CODING_A_LAW.
	HCI_SCO_CONTENT_FORMAT_AIR_CODING_CSVD	This is macro HCI_SCO_CONTENT_FORMAT_AIR_CODING_CSVD.
	HCI_SCO_CONTENT_FORMAT_AIR_CODING_N_LAW	This is macro HCI_SCO_CONTENT_FORMAT_AIR_CODING_N_LAW.

	HCI_SCO_CONTENT_FORMAT_AIR_CODING_TRANSPARENT	This is macro HCI_SCO_CONTENT_FORMAT_AIR_CODING_TRANSPARENT.
	HCI_SCO_CONTENT_FORMAT_INPUT_CODING_A_LAW	This is macro HCI_SCO_CONTENT_FORMAT_INPUT_CODING_A_LAW.
	HCI_SCO_CONTENT_FORMAT_INPUT_CODING_LINEAR	This is macro HCI_SCO_CONTENT_FORMAT_INPUT_CODING_LINEAR.
	HCI_SCO_CONTENT_FORMAT_INPUT_CODING_N_LAW	This is macro HCI_SCO_CONTENT_FORMAT_INPUT_CODING_N_LAW.
	HCI_SCO_CONTENT_FORMAT_INPUT_FORMAT_1_COMPLEMENT	This is macro HCI_SCO_CONTENT_FORMAT_INPUT_FORMAT_1_COMPLEMENT.
	HCI_SCO_CONTENT_FORMAT_INPUT_FORMAT_2_COMPLEMENT	This is macro HCI_SCO_CONTENT_FORMAT_INPUT_FORMAT_2_COMPLEMENT.
	HCI_SCO_CONTENT_FORMAT_INPUT_FORMAT_SIGN_MAGNITUDE	This is macro HCI_SCO_CONTENT_FORMAT_INPUT_FORMAT_SIGN_MAGNITUDE.
	HCI_SCO_CONTENT_FORMAT_INPUT_FORMAT_UNSIGNED	This is macro HCI_SCO_CONTENT_FORMAT_INPUT_FORMAT_UNSIGNED.
	HCI_SCO_CONTENT_FORMAT_INPUT_SAMPLE_SIZE_16	This is macro HCI_SCO_CONTENT_FORMAT_INPUT_SAMPLE_SIZE_16.
	HCI_SCO_CONTENT_FORMAT_INPUT_SAMPLE_SIZE_8	This is macro HCI_SCO_CONTENT_FORMAT_INPUT_SAMPLE_SIZE_8.
	HCI_SCO_DATA_HEADER_LEN	This is macro HCI_SCO_DATA_HEADER_LEN.
	HCI_SCO_MAX_DATA_LEN	This is macro HCI_SCO_MAX_DATA_LEN.
	HCI_SCO_MAX_LATENCY_DONTCARE	This is macro HCI_SCO_MAX_LATENCY_DONTCARE.
	HCI_SCO_PACKET_TYPE_ALL	This is macro HCI_SCO_PACKET_TYPE_ALL.
	HCI_SCO_PACKET_TYPE_EV3	This is macro HCI_SCO_PACKET_TYPE_EV3.
	HCI_SCO_PACKET_TYPE_EV4	This is macro HCI_SCO_PACKET_TYPE_EV4.
	HCI_SCO_PACKET_TYPE_EV5	This is macro HCI_SCO_PACKET_TYPE_EV5.
	HCI_SCO_PACKET_TYPE_HV1	This is macro HCI_SCO_PACKET_TYPE_HV1.
	HCI_SCO_PACKET_TYPE_HV2	This is macro HCI_SCO_PACKET_TYPE_HV2.
	HCI_SCO_PACKET_TYPE_HV3	This is macro HCI_SCO_PACKET_TYPE_HV3.
	HCI_SCO_PACKET_TYPE_NO_2_EV3	This is macro HCI_SCO_PACKET_TYPE_NO_2_EV3.
	HCI_SCO_PACKET_TYPE_NO_2_EV5	This is macro HCI_SCO_PACKET_TYPE_NO_2_EV5.
	HCI_SCO_PACKET_TYPE_NO_3_EV3	This is macro HCI_SCO_PACKET_TYPE_NO_3_EV3.
	HCI_SCO_PACKET_TYPE_NO_3_EV5	This is macro HCI_SCO_PACKET_TYPE_NO_3_EV5.
	HCI_SCO_RTX_EFFORT_DONTCARE	This is macro HCI_SCO_RTX_EFFORT_DONTCARE.
	HCI_SCO_RTX_EFFORT_NO_RETRANSMISSION	This is macro HCI_SCO_RTX_EFFORT_NO_RETRANSMISSION.
	HCI_SCO_RTX_EFFORT_OPTIMIZE_LINK_QUALITY	This is macro HCI_SCO_RTX_EFFORT_OPTIMIZE_LINK_QUALITY.
	HCI_SCO_RTX_EFFORT_OPTIMIZE_POWER_CONSUMPTION	This is macro HCI_SCO_RTX_EFFORT_OPTIMIZE_POWER_CONSUMPTION.
	HCI_SCO_RX_BANDWIDTH_DONTCARE	This is macro HCI_SCO_RX_BANDWIDTH_DONTCARE.
	HCI_SCO_TX_BANDWIDTH_DONTCARE	This is macro HCI_SCO_TX_BANDWIDTH_DONTCARE.
	HCI_SEND_DATA_STATUS_INTERRUPTED	This is macro HCI_SEND_DATA_STATUS_INTERRUPTED.
	HCI_SEND_DATA_STATUS_SUCCESS	This is macro HCI_SEND_DATA_STATUS_SUCCESS.
	HCI_SEND_KEY_PRESS_NOTIFICATION	This is macro HCI_SEND_KEY_PRESS_NOTIFICATION.
	HCI_SET_AFH_HOST_CHANNEL_CLASSIFICATION	This is macro HCI_SET_AFH_HOST_CHANNEL_CLASSIFICATION.

	HCI_SET_CONNECTION_ENCRYPTION	This is macro HCI_SET_CONNECTION_ENCRYPTION.
	HCI_SET_CTRL_TO_HOST_FLOW_CONTROL	This is macro HCI_SET_CTRL_TO_HOST_FLOW_CONTROL.
	HCI_SET_EVENT_FILTER	This is macro HCI_SET_EVENT_FILTER.
	HCI_SET_EVENT_MASK	addtogroup hci @{ @name Controller & Baseband commands details The Controller & Baseband Commands provide access and control to various capabilities of the Bluetooth hardware.
	HCI_SETUP_SYNCHRONOUS_CONNECTION	This is macro HCI_SETUP_SYNCHRONOUS_CONNECTION.
	HCI_SNIFF_MODE	This is macro HCI_SNIFF_MODE.
	HCI_SNIFF_SUBRATING	This is macro HCI_SNIFF_SUBRATING.
	HCI_SWITCH_ROLE	This is macro HCI_SWITCH_ROLE.
	HCI_USER_CONFIRMATION_REQ_NEGATIVE_REPLY	This is macro HCI_USER_CONFIRMATION_REQ_NEGATIVE_REPLY.
	HCI_USER_CONFIRMATION_REQUEST_REPLY	This is macro HCI_USER_CONFIRMATION_REQUEST_REPLY.
	HCI_USER_PASSKEY_REQUEST_NEGATIVE_REPLY	This is macro HCI_USER_PASSKEY_REQUEST_NEGATIVE_REPLY.
	HCI_USER_PASSKEY_REQUEST_REPLY	This is macro HCI_USER_PASSKEY_REQUEST_REPLY.
	HCI_WRITE_AFH_CHANNEL_ASSESSMENT_MODE	This is macro HCI_WRITE_AFH_CHANNEL_ASSESSMENT_MODE.
	HCI_WRITE_AUTHENTICATION_ENABLE	This is macro HCI_WRITE_AUTHENTICATION_ENABLE.
	HCI_WRITE_AUTOMATIC_FLASH_TIMEOUT	This is macro HCI_WRITE_AUTOMATIC_FLASH_TIMEOUT.
	HCI_WRITE_CLASS_OF_DEVICE	This is macro HCI_WRITE_CLASS_OF_DEVICE.
	HCI_WRITE_CONNECTION_ACCEPT_TIMEOUT	This is macro HCI_WRITE_CONNECTION_ACCEPT_TIMEOUT.
	HCI_WRITE_CURRENT_IAC_LAP	This is macro HCI_WRITE_CURRENT_IAC_LAP.
	HCI_WRITE_DEFAULT_ERRONEOUS_DATA_REPORTING	This is macro HCI_WRITE_DEFAULT_ERRONEOUS_DATA_REPORTING.
	HCI_WRITE_DEFAULT_POLICY_SETTINGS	This is macro HCI_WRITE_DEFAULT_POLICY_SETTINGS.
	HCI_WRITE_ENCRYPTION_MODE	This is macro HCI_WRITE_ENCRYPTION_MODE.
	HCI_WRITE_EXTENDED_INQUIRY_RESPONSE	This is macro HCI_WRITE_EXTENDED_INQUIRY_RESPONSE.
	HCI_WRITE_EXTENDED_INQUIRY_RESPONSE_PARAM_LEN	This is macro HCI_WRITE_EXTENDED_INQUIRY_RESPONSE_PARAM_LEN.
	HCI_WRITE_HOLD_MODE_ACTIVITY	This is macro HCI_WRITE_HOLD_MODE_ACTIVITY.
	HCI_WRITE_INQUIRY_MODE	This is macro HCI_WRITE_INQUIRY_MODE.
	HCI_WRITE_INQUIRY_SCAN_ACTIVITY	This is macro HCI_WRITE_INQUIRY_SCAN_ACTIVITY.
	HCI_WRITE_INQUIRY_SCAN_TYPE	This is macro HCI_WRITE_INQUIRY_SCAN_TYPE.
	HCI_WRITE_INQUIRY_TX_POWER_LEVEL	This is macro HCI_WRITE_INQUIRY_TX_POWER_LEVEL.
	HCI_WRITE_LE_HOST_SUPPORT	This is macro HCI_WRITE_LE_HOST_SUPPORT.
	HCI_WRITE_LINK_POLICY_SETTINGS	This is macro HCI_WRITE_LINK_POLICY_SETTINGS.
	HCI_WRITE_LINK_SUPERVISION_TIMEOUT	This is macro HCI_WRITE_LINK_SUPERVISION_TIMEOUT.
	HCI_WRITE_LOCAL_NAME	This is macro HCI_WRITE_LOCAL_NAME.
	HCI_WRITE_LOCAL_NAME_PARAM_LEN	This is macro HCI_WRITE_LOCAL_NAME_PARAM_LEN.
	HCI_WRITE_LOOPBACK_MODE	This is macro HCI_WRITE_LOOPBACK_MODE.
	HCI_WRITE_NUM_BROADCST_RETR	This is macro HCI_WRITE_NUM_BROADCST_RETR.
	HCI_WRITE_PAGE_SCAN_ACTIVITY	This is macro HCI_WRITE_PAGE_SCAN_ACTIVITY.

	HCI_WRITE_PAGE_SCAN_PERIOD_MODE	This is macro HCI_WRITE_PAGE_SCAN_PERIOD_MODE.
	HCI_WRITE_PAGE_SCAN_TYPE	This is macro HCI_WRITE_PAGE_SCAN_TYPE.
	HCI_WRITE_PAGE_TIMEOUT	This is macro HCI_WRITE_PAGE_TIMEOUT.
	HCI_WRITE_PIN_TYPE	This is macro HCI_WRITE_PIN_TYPE.
	HCI_WRITE_SCAN_ENABLE	This is macro HCI_WRITE_SCAN_ENABLE.
	HCI_WRITE_SIMPLE_PAIRING_DEBUG_MODE	This is macro HCI_WRITE_SIMPLE_PAIRING_DEBUG_MODE.
	HCI_WRITE_SIMPLE_PAIRING_MODE	This is macro HCI_WRITE_SIMPLE_PAIRING_MODE.
	HCI_WRITE_STORED_LINK_KEY	This is macro HCI_WRITE_STORED_LINK_KEY.
	HCI_WRITE_SYNC_FLOW_CONTROL_ENABLE	This is macro HCI_WRITE_SYNC_FLOW_CONTROL_ENABLE.
	HCI_WRITE_VOICE_SETTING	This is macro HCI_WRITE_VOICE_SETTING.
	OGF_CTRL_BASEBAND	This is macro OGF_CTRL_BASEBAND.
	OGF_INFORMATION	This is macro OGF_INFORMATION.
	OGF_LE	This is macro OGF_LE.
	OGF_LINK_CONTROL	This is macro OGF_LINK_CONTROL.
	OGF_LINK_POLICY	This is macro OGF_LINK_POLICY.
	OGF_STATUS	This is macro OGF_STATUS.
	OGF_TESTING	This is macro OGF_TESTING.
	OGF_VENDOR	This is macro OGF_VENDOR.

Structures

	Name	Description
	_bt_hci_command_s	This is type bt_hci_command_t.
	_bt_hci_data_s	This is type bt_hci_data_t.
	_bt_hci_event_s	This is type bt_hci_event_t.
	_bt_hci_inquiry_response_t	This is type bt_hci_inquiry_response_t.
	bt_hci_command_p	This is type bt_hci_command_p.
	bt_hci_command_t	This is type bt_hci_command_t.
	bt_hci_data_p	This is type bt_hci_data_p.
	bt_hci_data_t	This is type bt_hci_data_t.
	bt_hci_event_p	This is type bt_hci_event_p.
	bt_hci_event_t	This is type bt_hci_event_t.
	bt_hci_inquiry_response_t	This is type bt_hci_inquiry_response_t.

Types

	Name	Description
	bt_hci_cmd_callback_fp	This is type bt_hci_cmd_callback_fp.
	bt_hci_conn_state_p	This is type bt_hci_conn_state_p.
	bt_hci_conn_state_t	This is type bt_hci_conn_state_t.
	bt_hci_connect_callback_fp	brief HCI connect callback. ingroup hci details This typedef defines a type for the callback function that is called when HCI connect operation initiated by a call to bt_hci_connect() is complete. param status Operation status. c It is 0 if connection was successfully established. param pconn pointer to a structure representing the established connection. param param pointer to arbitrary data passed to the bt_hci_connect() function through its c param parameter.
	bt_hci_data_callback_fp	This is type bt_hci_data_callback_fp.
	bt_hci_disconnect_callback_fp	brief HCI disconnect callback. ingroup hci details This typedef defines a type for the callback function that is called when an HCI connection has been terminated. param status Operation status. c It is 0 if connection has been successfully terminated. param reason Reason for disconnection. param pconn pointer to a structure representing the connection. param param pointer to arbitrary data associated with an event listener.
	bt_hci_event_listener_fp	This is type bt_hci_event_listener_fp.
	bt_hci_hconn_p	This is type bt_hci_hconn_p.
	bt_hci_hconn_t	This is type bt_hci_hconn_t.

	bt_hci_inquiry_callback_fp	This is type bt_hci_inquiry_callback_fp.
	bt_hci_read_inquiry_mode_callback_fp	This is type bt_hci_read_inquiry_mode_callback_fp.
	bt_hci_read_inquiry_scan_activity_callback_fp	This is type bt_hci_read_inquiry_scan_activity_callback_fp.
	bt_hci_read_inquiry_scan_type_callback_fp	This is type bt_hci_read_inquiry_scan_type_callback_fp.
	bt_hci_read_page_scan_activity_callback_fp	This is type bt_hci_read_page_scan_activity_callback_fp.
	bt_hci_read_page_scan_period_mode_callback_fp	This is type bt_hci_read_page_scan_period_mode_callback_fp.
	bt_hci_read_page_scan_type_callback_fp	This is type bt_hci_read_page_scan_type_callback_fp.
	bt_hci_read_page_timeout_callback_fp	This is type bt_hci_read_page_timeout_callback_fp.
	bt_hci_request_remote_name_callback_fp	This is type bt_hci_request_remote_name_callback_fp.
	bt_hci_start_callback_fp	brief HCI initialization callback. ingroup hci details This typedef defines a function pointer type for the HCI initialization callback functions. Such a function must be passed to the bt_hci_start() function. param success Specifies whether HCI initialization succeeded or not.
	bt_hci_stop_callback_fp	brief HCI stop callback. ingroup hci details This typedef defines a function pointer type for the HCI stop callback functions. Such a function must be passed to the bt_hci_stop() function.
	pf_hci_sleep_callback	This is type pf_hci_sleep_callback.
	pf_hci_wakeup_callback	This is type pf_hci_wakeup_callback.

Description

Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.

SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements.

Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.

hci_cmd_buffer.h

Functions

	Name	Description
	_hci_allocate_cmd	This is function _hci_allocate_cmd.
	_hci_free_cmd	This is function _hci_free_cmd.
	_hci_init_cmd_buffers	This is function _hci_init_cmd_buffers.

Description

Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.

SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements.

Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.

hci_cmd_queue.h

Functions

	Name	Description
	_hci_init_cmd_queues	This is function _hci_init_cmd_queues.
	hci_cq_find_by_bdaddr_and_opcode	This is function hci_cq_find_by_bdaddr_and_opcode.
	hci_cq_find_by_hconn	This is function hci_cq_find_by_hconn.
	hci_cq_find_by_hconn_and_opcode	This is function hci_cq_find_by_hconn_and_opcode.
	hci_cq_find_by_opcode	This is function hci_cq_find_by_opcode.

Description

Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.

SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements.

Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with

a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.

hci_config.h

Macros

	Name	Description
	__HCI_CONFIG_H	This is macro __HCI_CONFIG_H.
	BT_ENABLE_SCO	This is macro BT_ENABLE_SCO.
	HCI_ALLOCATE_BUFFERS_FUNCTION	This is macro HCI_ALLOCATE_BUFFERS_FUNCTION.
	HCI_ALLOCATE_BUFFERS_RAM_SIZE_VAR	This is macro HCI_ALLOCATE_BUFFERS_RAM_SIZE_VAR.
	HCI_ALLOCATE_BUFFERS_VARS	This is macro HCI_ALLOCATE_BUFFERS_VARS.
	HCI_DECLARE_LE_CONN_STATES	This is macro HCI_DECLARE_LE_CONN_STATES.
	HCI_DECLARE_LE_CTRL_STATE	This is macro HCI_DECLARE_LE_CTRL_STATE.
	HCI_ENABLE_CTRL_TO_HOST_FLOW_CONTROL	This is macro HCI_ENABLE_CTRL_TO_HOST_FLOW_CONTROL.
	HCI_INIT_LE_CONN_STATES	This is macro HCI_INIT_LE_CONN_STATES.
	HCI_INIT_LE_CTRL_STATE	This is macro HCI_INIT_LE_CTRL_STATE.
	HCI_INIT_SCO_HANDLERS	This is macro HCI_INIT_SCO_HANDLERS.
	HCI_INIT_SSP_HANDLERS	This is macro HCI_INIT_SSP_HANDLERS.
	HCI_L2CAP_BUFFER_LEN	This is macro HCI_L2CAP_BUFFER_LEN.
	HCI_MAX_CONNECT_ATTEMPTS	This is macro HCI_MAX_CONNECT_ATTEMPTS.
	HCI_SIZEOF_LE_CONN_STATES	This is macro HCI_SIZEOF_LE_CONN_STATES.
	HCI_SIZEOF_LE_CTRL_STATE	This is macro HCI_SIZEOF_LE_CTRL_STATE.
	HCI_TX_BUFFER_LEN	This is macro HCI_TX_BUFFER_LEN.

Description

Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.

SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements.

Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.

hci_config_event_handlers.h

Macros

	Name	Description
	__HCI_CONFIG_EVENT_HANDLERS_H	This is macro __HCI_CONFIG_EVENT_HANDLERS_H.
	BT_HCI_EVT_AUTHENTICATION_COMPLETE_HANDLER	This is macro BT_HCI_EVT_AUTHENTICATION_COMPLETE_HANDLER.
	BT_HCI_EVT_CONNECTION_COMPLETE_HANDLER	This is macro BT_HCI_EVT_CONNECTION_COMPLETE_HANDLER.
	BT_HCI_EVT_CONNECTION_REQUEST_HANDLER	This is macro BT_HCI_EVT_CONNECTION_REQUEST_HANDLER.
	BT_HCI_EVT_ENCRYPTION_CHANGE_HANDLER	This is macro BT_HCI_EVT_ENCRYPTION_CHANGE_HANDLER.
	BT_HCI_EVT_EXTENDED_INQUIRY_RESULT_HANDLER	This is macro BT_HCI_EVT_EXTENDED_INQUIRY_RESULT_HANDLER.
	BT_HCI_EVT_INQUIRY_COMPLETE_HANDLER	This is macro BT_HCI_EVT_INQUIRY_COMPLETE_HANDLER.
	BT_HCI_EVT_INQUIRY_RESULT_HANDLER	This is macro BT_HCI_EVT_INQUIRY_RESULT_HANDLER.
	BT_HCI_EVT_INQUIRY_RESULT_WITH_RSSI_HANDLER	This is macro BT_HCI_EVT_INQUIRY_RESULT_WITH_RSSI_HANDLER.
	BT_HCI_EVT_LINK_KEY_NOTIFICATION_HANDLER	This is macro BT_HCI_EVT_LINK_KEY_NOTIFICATION_HANDLER.

	BT_HCI_EVT_LINK_KEY_REQUEST_HANDLER	This is macro BT_HCI_EVT_LINK_KEY_REQUEST_HANDLER.
	BT_HCI_EVT_MODE_CHANGE_HANDLER	This is macro BT_HCI_EVT_MODE_CHANGE_HANDLER.
	BT_HCI_EVT_PIN_CODE_REQUEST_HANDLER	This is macro BT_HCI_EVT_PIN_CODE_REQUEST_HANDLER.
	BT_HCI_EVT_REMOTE_NAME_REQUEST_COMPLETE_HANDLER	This is macro BT_HCI_EVT_REMOTE_NAME_REQUEST_COMPLETE_HANDLER.
	BT_HCI_EVT_ROLE_CHANGE_HANDLER	This is macro BT_HCI_EVT_ROLE_CHANGE_HANDLER.
	BT_HCI_EVT_SYNCH_CONNECTION_COMPLETE_HANDLER	This is macro BT_HCI_EVT_SYNCH_CONNECTION_COMPLETE_HANDLER.
	BT_LE_EVT_HANDLER	This is macro BT_LE_EVT_HANDLER.
	BT_SSP_EVT_HANDLER	This is macro BT_SSP_EVT_HANDLER.

Description

Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.

SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements.

Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.

hci_conn_state.h

Functions

	Name	Description
	bt_hci_register_listener	This is function bt_hci_register_listener.
	bt_hci_unregister_listener	This is function bt_hci_unregister_listener.
	hci_allocate_conn_state	This is function hci_allocate_conn_state.
	hci_get_conn_state	This is function hci_get_conn_state.
	hci_get_conn_state_by_bdaddr	This is function hci_get_conn_state_by_bdaddr.

Macros

	Name	Description
	HCI_CONN_ROLE_MASTER	This is macro HCI_CONN_ROLE_MASTER.
	HCI_CONN_ROLE_SLAVE	This is macro HCI_CONN_ROLE_SLAVE.
	HCI_CONN_STATE_AUTHENTICATING	This is macro HCI_CONN_STATE_AUTHENTICATING.
	HCI_CONN_STATE_CLOSED	This is macro HCI_CONN_STATE_CLOSED.
	HCI_CONN_STATE_OPEN	This is macro HCI_CONN_STATE_OPEN.
	HCI_CONN_TYPE_ACL	This is macro HCI_CONN_TYPE_ACL.
	HCI_CONN_TYPE_ESCO	This is macro HCI_CONN_TYPE_ESCO.
	HCI_CONN_TYPE_SCO	This is macro HCI_CONN_TYPE_SCO.
	HCI_LINK_TYPE_BD_EDR	This is macro HCI_LINK_TYPE_BD_EDR.
	HCI_LINK_TYPE_LE	This is macro HCI_LINK_TYPE_LE.

Structures

	Name	Description
	_bt_hci_conn_state_s	This is record _bt_hci_conn_state_s.
	_bt_hci_listener_t	This is record _bt_hci_listener_t.

Types

	Name	Description
	bt_hci_listener_t	This is type bt_hci_listener_t.
	bt_hci_sco_read_data_callback_fp	This is type bt_hci_sco_read_data_callback_fp.

Unions

	Name	Description
	_bt_hci_event_e	This is type bt_hci_event_e.
	bt_hci_event_e	This is type bt_hci_event_e.

Description

Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.

SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements.

Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.

hci_ctrl_state.h

Functions

	Name	Description
	bt_hci_ctrl_register_data_listener	This is function bt_hci_ctrl_register_data_listener.
	bt_hci_ctrl_register_listener	This is function bt_hci_ctrl_register_listener.
	bt_hci_ctrl_unregister_listener	This is function bt_hci_ctrl_unregister_listener.
	bt_hci_set_event_listener	listener is triggered by the following events: HCI_EVT_AUTHENTICATION_COMPLETE HCI_EVT_CONNECTION_COMPLETE HCI_EVT_DISCONNECTION_COMPLETE HCI_EVT_ROLE_CHANGE HCI_EVT_MODE_CHANGE

Macros

	Name	Description
	HCI_CTRL_LISTENER_ACL_DATA	This is macro HCI_CTRL_LISTENER_ACL_DATA.
	HCI_CTRL_LISTENER_EVENT	This is macro HCI_CTRL_LISTENER_EVENT.
	HCI_CTRL_LISTENER_SCO_DATA	This is macro HCI_CTRL_LISTENER_SCO_DATA.
	HCI_CTRL_STATE_CLOSED	This is macro HCI_CTRL_STATE_CLOSED.
	HCI_CTRL_STATE_CONNECTABLE	This is macro HCI_CTRL_STATE_CONNECTABLE.
	HCI_CTRL_STATE_DISCOVERABLE	This is macro HCI_CTRL_STATE_DISCOVERABLE.
	HCI_CTRL_STATE_INIT	This is macro HCI_CTRL_STATE_INIT.
	HCI_CTRL_STATE_READY	This is macro HCI_CTRL_STATE_READY.
	HCI_CTRL_STATE_SLEEP	This is macro HCI_CTRL_STATE_SLEEP.
	HCI_CTRL_STATE_WAKING_UP	This is macro HCI_CTRL_STATE_WAKING_UP.

Structures

	Name	Description
	_bt_hci_ctrl_listener_t	This is record _bt_hci_ctrl_listener_t.
	_bt_hci_ctrl_state_s	This is type bt_hci_ctrl_state_t.
	bt_hci_ctrl_state_t	This is type bt_hci_ctrl_state_t.

Types

	Name	Description
	bt_hci_cmd_listener_fp	This is type bt_hci_cmd_listener_fp.
	bt_hci_ctrl_listener_t	This is type bt_hci_ctrl_listener_t.
	bt_hci_data_listener_fp	This is type bt_hci_data_listener_fp.
	pf_l2cap_receive_callback	This is type pf_l2cap_receive_callback.

Description

Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.

SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements.

Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.

hci_data_buffer.h

Functions

	Name	Description
	bt_hci_alloc_data_buffer	This is function bt_hci_alloc_data_buffer.
	bt_hci_free_data_buffer	This is function bt_hci_free_data_buffer.
	bt_hci_init_data_buffers	This is function bt_hci_init_data_buffers.

Macros

	Name	Description
	HCI_DATA_BUFFER_STATE_FREE	This is macro HCI_DATA_BUFFER_STATE_FREE.
	HCI_DATA_BUFFER_STATE_USED	This is macro HCI_DATA_BUFFER_STATE_USED.
	HCI_MAX_DATA_BUFFERS	This is macro HCI_MAX_DATA_BUFFERS.

Structures

	Name	Description
	_bt_hci_data_buffer_s	This is type bt_hci_data_buffer_t.
	bt_hci_data_buffer_p	This is type bt_hci_data_buffer_p.
	bt_hci_data_buffer_t	This is type bt_hci_data_buffer_t.

Description

Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.

SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements.

Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.

hci_data_queue.h

Functions

	Name	Description
	bt_hci_init_data_queues	This is function bt_hci_init_data_queues.

Description

Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.

SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements.

Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.

hci_eir.h

Functions

	Name	Description
	bt_hci_allocate_write_eir_command	This is function bt_hci_allocate_write_eir_command.
	bt_hci_param_eir_add	This is function bt_hci_param_eir_add.
	bt_hci_param_eir_device_id_add	This is function bt_hci_param_eir_device_id_add.
	bt_hci_param_eir_local_name_add	This is function bt_hci_param_eir_local_name_add.
	bt_hci_param_eir_uuid128_add	This is function bt_hci_param_eir_uuid128_add.
	bt_hci_param_eir_uuid16_add	This is function bt_hci_param_eir_uuid16_add.
	bt_hci_param_eir_uuid32_add	This is function bt_hci_param_eir_uuid32_add.
	bt_hci_param_eir_vendor_add	This is function bt_hci_param_eir_vendor_add.
	bt_hci_param_tx_power_level_add	This is function bt_hci_param_tx_power_level_add.
	bt_hci_write_eir	This is function bt_hci_write_eir.

Macros

	Name	Description
	HCI_EIR_FEC_NOT_REQUIRED	This is macro HCI_EIR_FEC_NOT_REQUIRED.
	HCI_EIR_FEC_REQUIRED	This is macro HCI_EIR_FEC_REQUIRED.
	HCI_EIR_TYPE_3D_Information_Data	This is macro HCI_EIR_TYPE_3D_Information_Data.
	HCI_EIR_TYPE_ADVERTISING_INTERVAL	This is macro HCI_EIR_TYPE_ADVERTISING_INTERVAL.
	HCI_EIR_TYPE_APPEARANCE	This is macro HCI_EIR_TYPE_APPEARANCE.
	HCI_EIR_TYPE_DEVICE_ID	This is macro HCI_EIR_TYPE_DEVICE_ID.
	HCI_EIR_TYPE_FLAGS	This is macro HCI_EIR_TYPE_FLAGS.
	HCI_EIR_TYPE_LE_BLUETOOTH_DEVICE_ADDRESS	This is macro HCI_EIR_TYPE_LE_BLUETOOTH_DEVICE_ADDRESS.
	HCI_EIR_TYPE_LE_ROLE	This is macro HCI_EIR_TYPE_LE_ROLE.
	HCI_EIR_TYPE_LE_SECURE_CONNECTIONS_CONFIRMATION_VALUE	This is macro HCI_EIR_TYPE_LE_SECURE_CONNECTIONS_CONFIRMATION_VALUE.
	HCI_EIR_TYPE_LE_SECURE_CONNECTIONS_RANDOM_VALUE	This is macro HCI_EIR_TYPE_LE_SECURE_CONNECTIONS_RANDOM_VALUE.
	HCI_EIR_TYPE_LOCAL_NAME_COMPLETE	This is macro HCI_EIR_TYPE_LOCAL_NAME_COMPLETE.
	HCI_EIR_TYPE_LOCAL_NAME_SHORTENED	This is macro HCI_EIR_TYPE_LOCAL_NAME_SHORTENED.
	HCI_EIR_TYPE_MANUFACTURER_SPECIFIC	This is macro HCI_EIR_TYPE_MANUFACTURER_SPECIFIC.
	HCI_EIR_TYPE_OOB_COD	This is macro HCI_EIR_TYPE_OOB_COD.
	HCI_EIR_TYPE_OOB_HASH	This is macro HCI_EIR_TYPE_OOB_HASH.
	HCI_EIR_TYPE_OOB_RANDOMIZER	This is macro HCI_EIR_TYPE_OOB_RANDOMIZER.
	HCI_EIR_TYPE_PUBLIC_TARGET_ADDRESS	This is macro HCI_EIR_TYPE_PUBLIC_TARGET_ADDRESS.
	HCI_EIR_TYPE_RANDOM_TARGET_ADDRESS	This is macro HCI_EIR_TYPE_RANDOM_TARGET_ADDRESS.
	HCI_EIR_TYPE_SERVICE_DATA	This is macro HCI_EIR_TYPE_SERVICE_DATA.
	HCI_EIR_TYPE_SERVICE_DATA_UUID128	This is macro HCI_EIR_TYPE_SERVICE_DATA_UUID128.
	HCI_EIR_TYPE_SERVICE_DATA_UUID16	This is macro HCI_EIR_TYPE_SERVICE_DATA_UUID16.
	HCI_EIR_TYPE_SERVICE_DATA_UUID32	This is macro HCI_EIR_TYPE_SERVICE_DATA_UUID32.
	HCI_EIR_TYPE_SIMPLE_PAIRING_HASH_C_256	This is macro HCI_EIR_TYPE_SIMPLE_PAIRING_HASH_C_256.
	HCI_EIR_TYPE_SIMPLE_PAIRING_RANDOMIZER_R_256	This is macro HCI_EIR_TYPE_SIMPLE_PAIRING_RANDOMIZER_R_256.
	HCI_EIR_TYPE_SLAVE_CONN_INTERVAL_RANGE	This is macro HCI_EIR_TYPE_SLAVE_CONN_INTERVAL_RANGE.
	HCI_EIR_TYPE_SM_OOB_FLAGS	This is macro HCI_EIR_TYPE_SM_OOB_FLAGS.
	HCI_EIR_TYPE_SM_TK_VALUE	This is macro HCI_EIR_TYPE_SM_TK_VALUE.
	HCI_EIR_TYPE_SOLICITATION_UUID128_LIST	This is macro HCI_EIR_TYPE_SOLICITATION_UUID128_LIST.
	HCI_EIR_TYPE_SOLICITATION_UUID16_LIST	This is macro HCI_EIR_TYPE_SOLICITATION_UUID16_LIST.
	HCI_EIR_TYPE_SOLICITATION_UUID32_LIST	This is macro HCI_EIR_TYPE_SOLICITATION_UUID32_LIST.
	HCI_EIR_TYPE_TX_POWER_LEVEL	This is macro HCI_EIR_TYPE_TX_POWER_LEVEL.
	HCI_EIR_TYPE_UUID128_LIST_COMPLETE	This is macro HCI_EIR_TYPE_UUID128_LIST_COMPLETE.

	HCI_EIR_TYPE_UUID128_LIST_MORE_AVAILABLE	This is macro HCI_EIR_TYPE_UUID128_LIST_MORE_AVAILABLE .
	HCI_EIR_TYPE_UUID16_LIST_COMPLETE	This is macro HCI_EIR_TYPE_UUID16_LIST_COMPLETE.
	HCI_EIR_TYPE_UUID16_LIST_MORE_AVAILABLE	This is macro HCI_EIR_TYPE_UUID16_LIST_MORE_AVAILABLE.
	HCI_EIR_TYPE_UUID32_LIST_COMPLETE	This is macro HCI_EIR_TYPE_UUID32_LIST_COMPLETE.
	HCI_EIR_TYPE_UUID32_LIST_MORE_AVAILABLE	This is macro HCI_EIR_TYPE_UUID32_LIST_MORE_AVAILABLE.

Description

Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.

SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements.

Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.

hci_errors.h

Macros

	Name	Description
	HCI_ERR_ACL_CONN_ALREADY_EXISTS	This is macro HCI_ERR_ACL_CONN_ALREADY_EXISTS.
	HCI_ERR_AUTHENTICATION_FAILURE	This is macro HCI_ERR_AUTHENTICATION_FAILURE.
	HCI_ERR_CONN_REJECT_LIMITED_RESOURCES	This is macro HCI_ERR_CONN_REJECT_LIMITED_RESOURCES.
	HCI_ERR_CONNECTION_TIMEOUT	This is macro HCI_ERR_CONNECTION_TIMEOUT.
	HCI_ERR_INVALID_PARAMETERS	This is macro HCI_ERR_INVALID_PARAMETERS.
	HCI_ERR_MEMORY_CAPACITY_EXCEEDED	This is macro HCI_ERR_MEMORY_CAPACITY_EXCEEDED.
	HCI_ERR_PAIRING_NOT_ALLOWED	This is macro HCI_ERR_PAIRING_NOT_ALLOWED.
	HCI_ERR_PIN_OR_KEY_MISSING	This is macro HCI_ERR_PIN_OR_KEY_MISSING.
	HCI_ERR_SCO_CONN_LIMIT_EXCEEDED	This is macro HCI_ERR_SCO_CONN_LIMIT_EXCEEDED.
	HCI_ERR_SIMPLE_PAIRING_NOT_SUPPORTED	This is macro HCI_ERR_SIMPLE_PAIRING_NOT_SUPPORTED.
	HCI_ERR_SUCCESS	• addtogroup hci • @{ • @name Errors
	HCI_ERR_UNSPECIFIED	This is macro HCI_ERR_UNSPECIFIED.
	HCI_SUCCESS	This is macro HCI_SUCCESS.

Description

Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.

SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements.

Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.

hci_evt_handlers.h

Functions

	Name	Description
	bt_hci_evt_authentication_complete_handler	This is function bt_hci_evt_authentication_complete_handler.
	bt_hci_evt_change_conn_link_complete_handler	This is function bt_hci_evt_change_conn_link_complete_handler.
	bt_hci_evt_command_complete_handler	This is function bt_hci_evt_command_complete_handler.
	bt_hci_evt_command_status_handler	This is function bt_hci_evt_command_status_handler.
	bt_hci_evt_conn_packet_type_changed_handler	This is function bt_hci_evt_conn_packet_type_changed_handler.
	bt_hci_evt_connection_complete_handler	This is function bt_hci_evt_connection_complete_handler.
	bt_hci_evt_connection_request_handler	This is function bt_hci_evt_connection_request_handler.

	bt_hci_evt_data_buffer_overflow_handler	This is function bt_hci_evt_data_buffer_overflow_handler.
	bt_hci_evt_default_handler	This is function bt_hci_evt_default_handler.
	bt_hci_evt_disconnection_complete_handler	This is function bt_hci_evt_disconnection_complete_handler.
	bt_hci_evt_encryption_change_handler	This is function bt_hci_evt_encryption_change_handler.
	bt_hci_evt_extended_inquiry_result_handler	This is function bt_hci_evt_extended_inquiry_result_handler.
	bt_hci_evt_flow_specification_complete_handler	This is function bt_hci_evt_flow_specification_complete_handler.
	bt_hci_evt_flush_occured_handler	This is function bt_hci_evt_flush_occured_handler.
	bt_hci_evt_hardware_error_handler	This is function bt_hci_evt_hardware_error_handler.
	bt_hci_evt_inquiry_complete_handler	This is function bt_hci_evt_inquiry_complete_handler.
	bt_hci_evt_inquiry_result_handler	This is function bt_hci_evt_inquiry_result_handler.
	bt_hci_evt_inquiry_result_with_rssi_handler	This is function bt_hci_evt_inquiry_result_with_rssi_handler.
	bt_hci_evt_link_key_notification_handler	This is function bt_hci_evt_link_key_notification_handler.
	bt_hci_evt_link_key_request_handler	This is function bt_hci_evt_link_key_request_handler.
	bt_hci_evt_loopback_command_handler	This is function bt_hci_evt_loopback_command_handler.
	bt_hci_evt_master_link_key_complete_handler	This is function bt_hci_evt_master_link_key_complete_handler.
	bt_hci_evt_max_slots_change_handler	This is function bt_hci_evt_max_slots_change_handler.
	bt_hci_evt_mode_change_handler	This is function bt_hci_evt_mode_change_handler.
	bt_hci_evt_num_of_completed_packets_handler	This is function bt_hci_evt_num_of_completed_packets_handler.
	bt_hci_evt_page_scan_repet_mode_change_handler	This is function bt_hci_evt_page_scan_repet_mode_change_handler.
	bt_hci_evt_pin_code_request_handler	This is function bt_hci_evt_pin_code_request_handler.
	bt_hci_evt_qos_setup_complete_handler	This is function bt_hci_evt_qos_setup_complete_handler.
	bt_hci_evt_qos_violation_handler	This is function bt_hci_evt_qos_violation_handler.
	bt_hci_evt_read_clock_offset_complete_handler	This is function bt_hci_evt_read_clock_offset_complete_handler.
	bt_hci_evt_read_rmt_ext_features_comp_handler	This is function bt_hci_evt_read_rmt_ext_features_comp_handler.
	bt_hci_evt_read_rmt_sup_features_comp_handler	This is function bt_hci_evt_read_rmt_sup_features_comp_handler.
	bt_hci_evt_read_rmt_version_info_comp_handler	This is function bt_hci_evt_read_rmt_version_info_comp_handler.
	bt_hci_evt_remote_name_request_complete_handler	This is function bt_hci_evt_remote_name_request_complete_handler.
	bt_hci_evt_return_link_keys_handler	This is function bt_hci_evt_return_link_keys_handler.
	bt_hci_evt_role_change_handler	This is function bt_hci_evt_role_change_handler.
	bt_hci_evt_synch_connection_changed_handler	This is function bt_hci_evt_synch_connection_changed_handler.
	bt_hci_evt_synch_connection_complete_handler	This is function bt_hci_evt_synch_connection_complete_handler.
	bt_hci_set_vendor_specific_event_handler	This is function bt_hci_set_vendor_specific_event_handler.

Structures

	Name	Description
	_bt_hci_evt_authentication_complete_t	This is type bt_hci_evt_authentication_complete_t.
	_bt_hci_evt_command_complete_t	This is type bt_hci_evt_command_complete_t.
	_bt_hci_evt_command_status_t	This is type bt_hci_evt_command_status_t.
	_bt_hci_evt_connection_complete_t	This is type bt_hci_evt_connection_complete_t.
	_bt_hci_evt_connection_request_t	This is type bt_hci_evt_connection_request_t.
	_bt_hci_evt_disconnection_complete_t	This is type bt_hci_evt_disconnection_complete_t.
	_bt_hci_evt_encryption_change_s	This is type bt_hci_evt_encryption_change_t.
	_bt_hci_evt_mode_change_t	This is type bt_hci_evt_mode_change_t.
	_bt_hci_evt_role_change_t	This is type bt_hci_evt_role_change_t.
	bt_hci_evt_authentication_complete_t	This is type bt_hci_evt_authentication_complete_t.
	bt_hci_evt_command_complete_t	This is type bt_hci_evt_command_complete_t.
	bt_hci_evt_command_status_t	This is type bt_hci_evt_command_status_t.
	bt_hci_evt_connection_complete_t	This is type bt_hci_evt_connection_complete_t.
	bt_hci_evt_connection_request_t	This is type bt_hci_evt_connection_request_t.
	bt_hci_evt_disconnection_complete_t	This is type bt_hci_evt_disconnection_complete_t.
	bt_hci_evt_encryption_change_t	This is type bt_hci_evt_encryption_change_t.
	bt_hci_evt_mode_change_t	This is type bt_hci_evt_mode_change_t.
	bt_hci_evt_role_change_t	This is type bt_hci_evt_role_change_t.

Types

	Name	Description
	bt_hci_event_handler_ex_fp	This is type bt_hci_event_handler_ex_fp.
	bt_hci_event_handler_fp	This is type bt_hci_event_handler_fp.

Description

Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.

SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements.

Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.

hci_le.h

Functions

	Name	Description
	bt_hci_le_add_device_to_white_list	This is function bt_hci_le_add_device_to_white_list.
	bt_hci_le_advertising_add	This is function bt_hci_le_advertising_add.
	bt_hci_le_advertising_add_local_name	This is function bt_hci_le_advertising_add_local_name.
	bt_hci_le_advertising_device_id_add	This is function bt_hci_le_advertising_device_id_add.
	bt_hci_le_advertising_flags_add	This is function bt_hci_le_advertising_flags_add.
	bt_hci_le_advertising_get_local_name	This is function bt_hci_le_advertising_get_local_name.
	bt_hci_le_advertising_tx_power_level_add	This is function bt_hci_le_advertising_tx_power_level_add.
	bt_hci_le_advertising_uuid128_add	This is function bt_hci_le_advertising_uuid128_add.
	bt_hci_le_advertising_uuid16_add	This is function bt_hci_le_advertising_uuid16_add.
	bt_hci_le_advertising_uuid32_add	This is function bt_hci_le_advertising_uuid32_add.
	bt_hci_le_advertising_vendor_add	This is function bt_hci_le_advertising_vendor_add.
	bt_hci_le_allocate_set_advertising_data_command	This is function bt_hci_le_allocate_set_advertising_data_command.
	bt_hci_le_allocate_set_scan_response_data_command	This is function bt_hci_le_allocate_set_scan_response_data_command.
	bt_hci_le_cancel_connect_ex	This is function bt_hci_le_cancel_connect_ex.
	bt_hci_le_cancel_find_devices	This is function bt_hci_le_cancel_find_devices.
	bt_hci_le_clear_white_list	This is function bt_hci_le_clear_white_list.
	bt_hci_le_connect_ex	This is function bt_hci_le_connect_ex.
	bt_hci_le_enable	This is function bt_hci_le_enable.
	bt_hci_le_encrypt	This is function bt_hci_le_encrypt.
	bt_hci_le_find_devices	This is function bt_hci_le_find_devices.
	bt_hci_le_get_connect_parameters	This is function bt_hci_le_get_connect_parameters.
	bt_hci_le_ibeacon_add	This is function bt_hci_le_ibeacon_add.
	bt_hci_le_ltk_negative_reply	This is function bt_hci_le_ltk_negative_reply.
	bt_hci_le_ltk_reply	This is function bt_hci_le_ltk_reply.
	bt_hci_le_rand	This is function bt_hci_le_rand.
	bt_hci_le_read_channel_map	This is function bt_hci_le_read_channel_map.
	bt_hci_le_read_remote_used_features	This is function bt_hci_le_read_remote_used_features.
	bt_hci_le_read_support	This is function bt_hci_le_read_support.
	bt_hci_le_read_white_list_size	This is function bt_hci_le_read_white_list_size.
	bt_hci_le_receiver_test	This is function bt_hci_le_receiver_test.
	bt_hci_le_remove_device_from_white_list	This is function bt_hci_le_remove_device_from_white_list.
	bt_hci_le_set_adevrtising_enable_ex	This is function bt_hci_le_set_adevrtising_enable_ex.
	bt_hci_le_set_advertising_parameters	This is function bt_hci_le_set_advertising_parameters.
	bt_hci_le_set_connect_parameters	This is function bt_hci_le_set_connect_parameters.
	bt_hci_le_set_host_channel_classification	This is function bt_hci_le_set_host_channel_classification.
	bt_hci_le_set_random_address	This is function bt_hci_le_set_random_address.
	bt_hci_le_set_scan_enable	This is function bt_hci_le_set_scan_enable.
	bt_hci_le_set_scan_parameters	This is function bt_hci_le_set_scan_parameters.

	bt_hci_le_start_encryption	This is function bt_hci_le_start_encryption.
	bt_hci_le_supported	This is function bt_hci_le_supported.
	bt_hci_le_test_end	This is function bt_hci_le_test_end.
	bt_hci_le_transmitter_test	This is function bt_hci_le_transmitter_test.
	bt_hci_le_update_connection	This is function bt_hci_le_update_connection.
	bt_hci_le_write_support	This is function bt_hci_le_write_support.

Macros

	Name	Description
	bt_hci_le_cancel_connect	This is macro bt_hci_le_cancel_connect.
	bt_hci_le_set_adevrtising_enable	This is macro bt_hci_le_set_adevrtising_enable.
	HCI_LE_ADDRESS_TYPE_PUBLIC	This is macro HCI_LE_ADDRESS_TYPE_PUBLIC.
	HCI_LE_ADDRESS_TYPE_RANDOM	This is macro HCI_LE_ADDRESS_TYPE_RANDOM.
	HCI_LE_ADV_CHANNEL_MAP_ENABLE_37	This is macro HCI_LE_ADV_CHANNEL_MAP_ENABLE_37.
	HCI_LE_ADV_CHANNEL_MAP_ENABLE_38	This is macro HCI_LE_ADV_CHANNEL_MAP_ENABLE_38.
	HCI_LE_ADV_CHANNEL_MAP_ENABLE_39	This is macro HCI_LE_ADV_CHANNEL_MAP_ENABLE_39.
	HCI_LE_ADV_CHANNEL_MAP_ENABLE_ALL	This is macro HCI_LE_ADV_CHANNEL_MAP_ENABLE_ALL.
	HCI_LE_ADV_CHANNEL_MAP_RESERVED	This is macro HCI_LE_ADV_CHANNEL_MAP_RESERVED.
	HCI_LE_ADV_TYPE_CONN_UNDIRECT	Connectable undirected advertising (ADV_IND) (default)
	HCI_LE_ADV_TYPE_DIRECT_HIGH	Connectable high duty cycle directed advertising (ADV_DIRECT_IND, high duty cycle)
	HCI_LE_ADV_TYPE_DIRECT_LOW	Connectable low duty cycle directed advertising (ADV_DIRECT_IND, low duty cycle)
	HCI_LE_ADV_TYPE_NONCONN	Non connectable undirected advertising (ADV_NONCONN_IND)
	HCI_LE_ADV_TYPE_SCAN	Scannable undirected advertising (ADV_SCAN_IND)
	HCI_LE_ADVERTISING_FLAG_BREDR_NOT_SUPPORTED	This is macro HCI_LE_ADVERTISING_FLAG_BREDR_NOT_SUPPORTED.
	HCI_LE_ADVERTISING_FLAG_GENERAL_DISCOVERABLE_MODE	This is macro HCI_LE_ADVERTISING_FLAG_GENERAL_DISCOVERABLE_MODE.
	HCI_LE_ADVERTISING_FLAG_LIMITED_DISCOVERABLE_MODE	This is macro HCI_LE_ADVERTISING_FLAG_LIMITED_DISCOVERABLE_MODE.
	HCI_LE_ADVERTISING_FLAG_SIMULTANEOUS_LE_BREDR_CONTROLLER	This is macro HCI_LE_ADVERTISING_FLAG_SIMULTANEOUS_LE_BREDR_CONTROLLER.
	HCI_LE_ADVERTISING_FLAG_SIMULTANEOUS_LE_BREDR_HOST	This is macro HCI_LE_ADVERTISING_FLAG_SIMULTANEOUS_LE_BREDR_HOST.
	HCI_LE_AVERTISING_DISABLED	This is macro HCI_LE_AVERTISING_DISABLED.
	HCI_LE_AVERTISING_ENABLED	This is macro HCI_LE_AVERTISING_ENABLED.
	HCI_LE_DISABLED	This is macro HCI_LE_DISABLED.
	HCI_LE_DUPLICATE_FILTERING_DISABLED	This is macro HCI_LE_DUPLICATE_FILTERING_DISABLED.
	HCI_LE_DUPLICATE_FILTERING_ENABLED	This is macro HCI_LE_DUPLICATE_FILTERING_ENABLED.
	HCI_LE_ENABLED	This is macro HCI_LE_ENABLED.
	HCI_LE_EVT_ADVERTISING_REPORT	This is macro HCI_LE_EVT_ADVERTISING_REPORT.

	HCI_LE_EVT_CONNECTION_COMPLETE	This is macro HCI_LE_EVT_CONNECTION_COMPLETE.
	HCI_LE_EVT_CONNECTION_UPDATE_COMPLETE	This is macro HCI_LE_EVT_CONNECTION_UPDATE_COMPLETE.
	HCI_LE_EVT_LONG_TERM_KEY_REQUEST	This is macro HCI_LE_EVT_LONG_TERM_KEY_REQUEST.
	HCI_LE_EVT_READ_REMOTE_USED_FEATURES_COMPLETE	This is macro HCI_LE_EVT_READ_REMOTE_USED_FEATURES_COMPLETE.
	HCI_LE_FILTER_POLICY_NOT_USED	White list is not used to determine which advertiser to connect to. Peer_Address_Type and Peer_Address shall be used.
	HCI_LE_FILTER_POLICY_WHITE_LIST	White list is used to determine which advertiser to connect to. Peer_Address_Type and Peer_Address shall be ignored
	HCI_LE_FLAG_SHARED_ACL_BUFFERS	This is macro HCI_LE_FLAG_SHARED_ACL_BUFFERS.
	HCI_LE_FLAG_SIMULTANEOUS_LE_BREDR	This is macro HCI_LE_FLAG_SIMULTANEOUS_LE_BREDR.
	HCI_LE_RANDOM_ADDRESS_TYPE_NON_RESOLVABLE	This is macro HCI_LE_RANDOM_ADDRESS_TYPE_NON_RESOLVABLE.
	HCI_LE_RANDOM_ADDRESS_TYPE_RESOLVABLE	This is macro HCI_LE_RANDOM_ADDRESS_TYPE_RESOLVABLE.
	HCI_LE_RANDOM_ADDRESS_TYPE_STATIC	This is macro HCI_LE_RANDOM_ADDRESS_TYPE_STATIC.
	HCI_LE_SCAN_DISABLED	This is macro HCI_LE_SCAN_DISABLED.
	HCI_LE_SCAN_ENABLED	This is macro HCI_LE_SCAN_ENABLED.
	HCI_LE_SCAN_EVT_CANCEL_FAILED	This is macro HCI_LE_SCAN_EVT_CANCEL_FAILED.
	HCI_LE_SCAN_EVT_CANCELLED	This is macro HCI_LE_SCAN_EVT_CANCELLED.
	HCI_LE_SCAN_EVT_DEVICE_FOUND	This is macro HCI_LE_SCAN_EVT_DEVICE_FOUND.
	HCI_LE_SCAN_EVT_START_FAILED	This is macro HCI_LE_SCAN_EVT_START_FAILED.
	HCI_LE_SCAN_EVT_STARTED	This is macro HCI_LE_SCAN_EVT_STARTED.
	HCI_LE_SCAN_FILTER_POLICY_ALL	This is macro HCI_LE_SCAN_FILTER_POLICY_ALL.
	HCI_LE_SCAN_FILTER_POLICY_WHITE_LIST	This is macro HCI_LE_SCAN_FILTER_POLICY_WHITE_LIST.
	HCI_LE_SCAN_TYPE_ACTIVE	This is macro HCI_LE_SCAN_TYPE_ACTIVE.
	HCI_LE_SCAN_TYPE_PASSIVE	This is macro HCI_LE_SCAN_TYPE_PASSIVE.
	HCI_LE_SIMULTANEOUS_DISABLED	This is macro HCI_LE_SIMULTANEOUS_DISABLED.
	HCI_LE_SIMULTANEOUS_ENABLED	This is macro HCI_LE_SIMULTANEOUS_ENABLED.

Structures

	Name	Description
	_bt_hci_le_advertising_report_t	This is record _bt_hci_le_advertising_report_t.
	_bt_hci_le_conn_state_t	This is type bt_hci_le_conn_state_t.
	_bt_hci_le_connect_parameters_t	This is type bt_hci_le_connect_parameters_t.
	_bt_hci_le_ctrl_state_t	This is type bt_hci_le_ctrl_state_t.
	_bt_hci_le_evt_connection_updated_t	This is type bt_hci_le_evt_connection_updated_t.
	_bt_hci_le_evt_read_remote_used_features_completed_t	This is type bt_hci_le_evt_read_remote_used_features_completed_t.
	_bt_hci_le_evt_read_support_params_t	This is type bt_hci_le_evt_read_support_params_t.
	bt_hci_le_conn_state_t	This is type bt_hci_le_conn_state_t.
	bt_hci_le_connect_parameters_t	This is type bt_hci_le_connect_parameters_t.
	bt_hci_le_ctrl_state_t	This is type bt_hci_le_ctrl_state_t.

	bt_hci_le_evt_connection_updated_t	This is type bt_hci_le_evt_connection_updated_t.
	bt_hci_le_evt_read_remote_used_features_completed_t	This is type bt_hci_le_evt_read_remote_used_features_completed_t.
	bt_hci_le_evt_read_support_params_t	This is type bt_hci_le_evt_read_support_params_t.

Types

	Name	Description
	bt_hci_le_advertising_report_t	This is type bt_hci_le_advertising_report_t.
	bt_hci_le_scan_callback_fp	This is type bt_hci_le_scan_callback_fp.
	bt_le_evt_handler	This is type bt_le_evt_handler.

Description

Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.

hci_linkkey_buffer.h

Functions

	Name	Description
	bt_hci_init_linkkey_buffers	This is function bt_hci_init_linkkey_buffers.

Structures

	Name	Description
	_bt_hci_link_key_s	This is type bt_hci_link_key_t.
	bt_hci_link_key_t	This is type bt_hci_link_key_t.

Description

Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.

SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements.

Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.

hci_private.h

Functions

	Name	Description
	_bt_hci_ctrl_notify_data_listeners	This is function _bt_hci_ctrl_notify_data_listeners.
	_bt_hci_ctrl_notify_listeners	This is function _bt_hci_ctrl_notify_listeners.
	_bt_hci_get_tick_count	This is function _bt_hci_get_tick_count.
	_bt_hci_init_timer	From hci_timer.c
	_bt_hci_init_transport	This is function _bt_hci_init_transport.
	_bt_hci_le_command_complete_handler	This is function _bt_hci_le_command_complete_handler.
	_bt_hci_notify_listeners	This is function _bt_hci_notify_listeners.
	_bt_hci_set_init_flags	This is function _bt_hci_set_init_flags.
	_bt_le_evt_handler	This is function _bt_le_evt_handler.
	_hci_allocate_buffers	Defined by OEM through library configuration
	_hci_receive_start	From hci_receive.c
	_hci_recv_sco_data_packet	From hci_sco.c
	_hci_send_commands_from_queue	From hci_send.c
	_hci_send_data	This is function _hci_send_data.
	_hci_send_data_fragment	This is function _hci_send_data_fragment.
	_hci_send_data_from_queue	This is function _hci_send_data_from_queue.
	bt_hci_le_init	This is function bt_hci_le_init.
	hci_check_aux_info	From hci_aux_info.c

Macros

	Name	Description
	HCI_INIT_FLAG_IGNORE_TOTAL_NUM_ACL_DATA_PACKETS	The total number of ACL data packets read from the controller will be ignored. The stack will assume that he controller can accept only 1 ACL packet and sends HCI_EVT_NUM_OF_COMPLETED_PACKETS for each packet it has processed. This seems to be for controller working over SDIO (at least for Marvell's 88W8777).
	HCI_INIT_FLAG_SEND_HCI_RESET	HCI reset will be sent when bt_hci_init is called

Description

Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.

SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements.

Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.

hci_signal.h

Functions

	Name	Description
	_bt_hci_init_signal	This is function _bt_hci_init_signal.
	_bt_hci_set_signal	This is function _bt_hci_set_signal.

Description

Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.

SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements.

Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.

hci_transport.h

Functions

	Name	Description
	bt_hci_transport_recv_packet	This is function bt_hci_transport_recv_packet.
	bt_hci_transport_send_cmd	deprecated
	bt_hci_transport_send_data	This is function bt_hci_transport_send_data.
	bt_hci_transport_send_packet	This is function bt_hci_transport_send_packet.
	bt_hci_transport_set_transport	This is function bt_hci_transport_set_transport.

Macros

	Name	Description
	HCI_TRANSPORT_HEADER_LEN	This is macro HCI_TRANSPORT_HEADER_LEN.

Structures

	Name	Description
	_hci_transport_t	This is type hci_transport_t.
	hci_transport_t	This is type hci_transport_t.

Types

	Name	Description
	bt_hci_transport_recv_packet_callback_fp	This is type bt_hci_transport_recv_packet_callback_fp.
	bt_hci_transport_send_packet_callback_fp	This is type bt_hci_transport_send_packet_callback_fp.

Description

Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a

license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.

SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements.

Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.

hci_utils.h

Functions

	Name	Description
	is_bdaddr_command	This is function is_bdaddr_command.
	is_hconn_command	This is function is_hconn_command.

Description

Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.

SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements.

Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.

hcitr_3wire.h

Functions

	Name	Description
	bt_hcitr_3wire_cancel_recv_packet	This is function bt_hcitr_3wire_cancel_recv_packet.
	bt_hcitr_3wire_init_ex	This is function bt_hcitr_3wire_init_ex.
	bt_hcitr_3wire_reset_ex	This is function bt_hcitr_3wire_reset_ex.
	bt_hcitr_3wire_start	This is function bt_hcitr_3wire_start.

Macros

	Name	Description
	bt_hcitr_3wire_init	This is macro bt_hcitr_3wire_init.
	bt_hcitr_3wire_reset	This is macro bt_hcitr_3wire_reset.
	HCITR_3WIRE_DEFAULT_ACK_TIMEOUT	This is macro HCITR_3WIRE_DEFAULT_ACK_TIMEOUT.

Description

Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.

SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements.

Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.

hcitr_bcsp.h

Functions

	Name	Description
	bt_hcitr_bcsp_init_ex	This is function bt_hcitr_bcsp_init_ex.
	bt_hcitr_bcsp_reset_ex	This is function bt_hcitr_bcsp_reset_ex.
	bt_hcitr_bcsp_start	This is function bt_hcitr_bcsp_start.

Macros

	Name	Description
	bt_hcitr_bcsp_init	This is macro bt_hcitr_bcsp_init.
	bt_hcitr_bcsp_reset	This is macro bt_hcitr_bcsp_reset.
	HCITR_BCSP_DEFAULT_ACK_TIMEOUT	This is macro HCITR_BCSP_DEFAULT_ACK_TIMEOUT.

Description

Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.

SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements.

Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.

hcitr_packet.h

Functions

	Name	Description
	bt_hcitr_packet_init	This is function bt_hcitr_packet_init.
	bt_hcitr_packet_reset	This is function bt_hcitr_packet_reset.
	bt_hcitr_packet_start	This is function bt_hcitr_packet_start.

Description

Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.

SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements.

Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.

hcitr_tih4.h

Enumerations

	Name	Description
	_hcitr_tih4_power_event_e	This is type bt_hcitr_tih4_power_event_e.
	bt_hcitr_tih4_power_event_e	This is type bt_hcitr_tih4_power_event_e.

Functions

	Name	Description
	bt_hcitr_tih4_init	This is function bt_hcitr_tih4_init.
	bt_hcitr_tih4_reset	This is function bt_hcitr_tih4_reset.
	bt_hcitr_tih4_start	This is function bt_hcitr_tih4_start.
	bt_hcitr_tih4_wake_up	This is function bt_hcitr_tih4_wake_up.

Types

	Name	Description
	bt_hcitr_tih4_power_callback_fp	This is type bt_hcitr_tih4_power_callback_fp.

Description

Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.

SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements.

Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.

hcitr_uart.h

Functions

	Name	Description
	bt_hcitr_uart_init	brief Initialize HCI UART (H4) transport protocol ingroup hcitr_uart details This function initializes internal structures of the transport. The application must call it as early as possible before bt_hcitr_uart_start and before the stack is initialized and started with bt_sys_init and bt_sys_start.

	bt_hcitr_uart_reset	brief Re-initialize HCI UART (H4) transport protocol ingroup hcitr_uart details This function re-initializes the transport. Currently it simply calls bt_hcitr_uart_init. After calling this function the application must perform the full initialization of the stack by calling bt_sys_init, bt_sys_start and initialization functions of all other profile modules the application is intending to use.
	bt_hcitr_uart_start	brief Start HCI UART (H4) transport protocol ingroup hcitr_uart details This function starts the transport, i.e., makes it able to receive and send packets.

Macros

	Name	Description
	HCI_UART_PACKET_TYPE_ACL_DATA	This is macro HCI_UART_PACKET_TYPE_ACL_DATA.
	HCI_UART_PACKET_TYPE_COMMAND	defgroup hcitr_uart HCI UART (H4) transport protocol ingroup hcitr details This module describes functions used to initialize and start HCI UART transport protocol. The transport uses common interface for exchanging data between the host CPU and HCI controller defined in bt_hcitr.h. This interface consist of two functions that must be implemented by the application: li bt_oem_send() li bt_oem_recv()
	HCI_UART_PACKET_TYPE_EVENT	This is macro HCI_UART_PACKET_TYPE_EVENT.
	HCI_UART_PACKET_TYPE_SCO_DATA	This is macro HCI_UART_PACKET_TYPE_SCO_DATA.

Description

Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.

SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements.

Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.

lm.h

Macros

	Name	Description
	LM_PACKET_TYPE_DH1	This is macro LM_PACKET_TYPE_DH1.
	LM_PACKET_TYPE_DH3	This is macro LM_PACKET_TYPE_DH3.
	LM_PACKET_TYPE_DH5	This is macro LM_PACKET_TYPE_DH5.
	LM_PACKET_TYPE_DM1	This is macro LM_PACKET_TYPE_DM1.
	LM_PACKET_TYPE_DM3	This is macro LM_PACKET_TYPE_DM3.
	LM_PACKET_TYPE_DM5	This is macro LM_PACKET_TYPE_DM5.

Description

Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.

SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements.

Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.

l2cap.h

Functions

	Name	Description
	bt_l2cap_allocate_mgr	brief Allocate L2CAP manager. ingroup l2cap details This function allocates and initializes an L2CAP manager structure. One L2CAP manager manages all L2CAP connections for a particular local device. The local device is specified by the c hci_ctrl parameter. param hci_ctrl Pointer to the hci_ctrl_state structure that represents the local device (HCI controller) for which a L2CAP manager is to be allocated. return A pointer to the allocated L2CAP manager structure. The returned L2CAP manager should be freed by a call to bt_l2cap_free_mgr when it is no longer needed.

	bt_l2cap_connect_ext	brief Connect to a remote device. ingroup l2cap details This function establishes an L2CAP connection with a remote device on a specific PSM. When connect operation completes a callback function is called. param mgr The L2CAP manager. param remote_addr The address of the remote device. param psm The PSM for the connection. param connect_cb The Callback function that is called when the connect operation completes. param param A pointer to arbitrary data to be passed to the c connect_cb callback. param state_cb The callback function that is called when the state of the established connection changes. return li c TRUE... more
	bt_l2cap_connect_fixed_channel	This is function bt_l2cap_connect_fixed_channel.
	bt_l2cap_disconnect	brief Close L2CAP channel. ingroup l2cap details This function closes an L2CAP channel. The channel can be established either by a call to bt_l2cap_connect() or by an incoming connection request. param ch The L2CAP channel to be closed. return li c TRUE when the function succeeds. li c FALSE otherwise. The callback function is not called in this case.
	bt_l2cap_disconnect_ex	This is function bt_l2cap_disconnect_ex.
	bt_l2cap_echo	brief Send echo command
	bt_l2cap_free_mgr	brief Release L2CAP manger. ingroup l2cap details This function releases the L2CAP manager structure. param mgr The L2CAP manager structure to be released.
	bt_l2cap_get_mgr	This is function bt_l2cap_get_mgr.
	bt_l2cap_hci_has_open_channels	This is function bt_l2cap_hci_has_open_channels.
	bt_l2cap_init	brief Initialize the L2CAP layer. ingroup l2cap details This function initializes the L2CAP layer of the stack. It must be called prior to any other L2CAP function can be called.
	bt_l2cap_is_channel_open	This is function bt_l2cap_is_channel_open.
	bt_l2cap_listen_ext	brief Listen for incoming connections. ingroup l2cap details This function tells the L2CAP manager to listen for incoming connections on a specific PSM. When a connection is established a callback function is called. param mgr The L2CAP manager. param psm The PSM on which the manager will listen and accept incoming connections. param callback The callback function that will be called when an incoming connection is established. param param An arbitrary data pointer that will be passed to the callback function specified by the c callback parameter. return li c TRUE when the function succeeds. li c FALSE otherwise. The... more
	bt_l2cap_listen_fixed_channel	This is function bt_l2cap_listen_fixed_channel.
	bt_l2cap_reject	brief Send reject command (used to reject unknown or invalid commands)
	bt_l2cap_send_config	This is function bt_l2cap_send_config.
	bt_l2cap_update_conn_parameters	This is function bt_l2cap_update_conn_parameters.

Macros

	Name	Description
	bt_l2cap_connect	This is macro bt_l2cap_connect.
	bt_l2cap_listen	This is macro bt_l2cap_listen.
	L2CAP_DEFAULT_ERTX	seconds
	L2CAP_DEFAULT_RTX	seconds
	L2CAP_FRAME_TYPE_I	This is macro L2CAP_FRAME_TYPE_I.
	L2CAP_FRAME_TYPE_S	This is macro L2CAP_FRAME_TYPE_S.
	L2CAP_HCI_CONNECT_PACKET_TYPE	This is macro L2CAP_HCI_CONNECT_PACKET_TYPE.
	L2CAP_HCI_CONNECT_PAGE_SCAN_REPETITION_MODE	This is macro L2CAP_HCI_CONNECT_PAGE_SCAN_REPETITION_MODE.
	L2CAP_HCI_CONNECT_ROLE_SWITCH	This is macro L2CAP_HCI_CONNECT_ROLE_SWITCH.
	L2CAP_HEADER_LEN	This is macro L2CAP_HEADER_LEN.
	L2CAP_IDLE_HCI_CONNECTION_TIMEOUT	This is macro L2CAP_IDLE_HCI_CONNECTION_TIMEOUT.
	L2CAP_MAX_MANAGERS	This is macro L2CAP_MAX_MANAGERS.
	L2CAP_MAX_MTU	This is macro L2CAP_MAX_MTU.
	L2CAP_MAX_RTX	This is macro L2CAP_MAX_RTX.
	L2CAP_MGR_STATE_FREE	This is macro L2CAP_MGR_STATE_FREE.
	L2CAP_MGR_STATE_LISTENING_L2CAP	This is macro L2CAP_MGR_STATE_LISTENING_L2CAP.
	L2CAP_MGR_STATE_USED	This is macro L2CAP_MGR_STATE_USED.

	L2CAP_MIN_MTU	This is macro L2CAP_MIN_MTU.
	L2CAP_MONITOR_TIMEOUT	This is macro L2CAP_MONITOR_TIMEOUT.
	L2CAP_RETR_TIMEOUT	This is macro L2CAP_RETR_TIMEOUT.
	L2CAP_SAR_SDU_CONTINUE	This is macro L2CAP_SAR_SDU_CONTINUE.
	L2CAP_SAR_SDU_START	This is macro L2CAP_SAR_SDU_START.
	L2CAP_SAR_SDU_STOP	This is macro L2CAP_SAR_SDU_STOP.
	L2CAP_SAR_UNSEGMENTED	This is macro L2CAP_SAR_UNSEGMENTED.
	L2CAP_SFUNCTION_REJ	This is macro L2CAP_SFUNCTION_REJ.
	L2CAP_SFUNCTION_RNR	This is macro L2CAP_SFUNCTION_RNR.
	L2CAP_SFUNCTION_RR	This is macro L2CAP_SFUNCTION_RR.
	L2CAP_SFUNCTION_SREJ	This is macro L2CAP_SFUNCTION_SREJ.
	PSM_ATT	This is macro PSM_ATT.
	PSM_AVCTP	This is macro PSM_AVCTP.
	PSM_AVCTP_Browsing	This is macro PSM_AVCTP_Browsing.
	PSM_AVDTP	This is macro PSM_AVDTP.
	PSM_BNEP	This is macro PSM_BNEP.
	PSM_HID_Control	This is macro PSM_HID_Control.
	PSM_HID_Interrupt	This is macro PSM_HID_Interrupt.
	PSM_RFCOMM	This is macro PSM_RFCOMM.
	PSM_SDP	This is macro PSM_SDP.

Structures

	Name	Description
	_bt_l2cap_connect_params_s	This is record _bt_l2cap_connect_params_s.
	_bt_l2cap_fixed_channel_s	This is type bt_l2cap_fixed_channel_t.
	_bt_l2cap_mgr_s	This is type bt_l2cap_mgr_t.
	_bt_l2cap_psm_s	This is type bt_l2cap_psm_t.
	bt_l2cap_fixed_channel_t	This is type bt_l2cap_fixed_channel_t.
	bt_l2cap_mgr_p	This is type bt_l2cap_mgr_p.
	bt_l2cap_mgr_t	This is type bt_l2cap_mgr_t.
	bt_l2cap_psm_t	This is type bt_l2cap_psm_t.

Types

	Name	Description
	bt_l2cap_connect_callback_fp	This is type bt_l2cap_connect_callback_fp.
	bt_l2cap_connect_params_t	This is type bt_l2cap_connect_params_t.
	bt_l2cap_listen_callback_fp	This is type bt_l2cap_listen_callback_fp.

Description

Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.

SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements.

Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.

l2cap_command_queue.h

Functions

	Name	Description
	_bt_l2cap_notify_and_remove	This is function _bt_l2cap_notify_and_remove.
	_bt_l2cap_send_commands_from_queue	This is function _bt_l2cap_send_commands_from_queue.

Macros

	Name	Description
	L2CAP_MAX_CMD_QUEUE_LEN	This is macro L2CAP_MAX_CMD_QUEUE_LEN.

Description

Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.

SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements.

Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.

l2cap_config.h

Macros

	Name	Description
	__L2CAP_CONFIG_H	This is macro __L2CAP_CONFIG_H.
	L2CAP_ALLOCATE_BUFFERS_FUNCTION	This is macro L2CAP_ALLOCATE_BUFFERS_FUNCTION.
	L2CAP_ALLOCATE_BUFFERS_RAM_SIZE_VAR	This is macro L2CAP_ALLOCATE_BUFFERS_RAM_SIZE_VAR.
	L2CAP_ALLOCATE_BUFFERS_VARS	This is macro L2CAP_ALLOCATE_BUFFERS_VARS.
	L2CAP_DECL_ERETR_FUNCTIONS	This is macro L2CAP_DECL_ERETR_FUNCTIONS.
	L2CAP_FIXED_CHANNELS_DECL	This is macro L2CAP_FIXED_CHANNELS_DECL.
	L2CAP_HCI_PACKET_TYPE	• brief L2CAP_HCI_PACKET_TYPE. • ingroup btconfig * • details Defines a set of packets that link manager is allowed to use when calling bt_l2cap_connect. • The default value is to enable all packet types. enable all packet types
	L2CAP_HCI_PAGE_SCAN_REPETITION_MODE	brief L2CAP_HCI_PAGE_SCAN_REPETITION_MODE. ingroup btconfig details Defines a default value of the page scan repetition mode when calling bt_l2cap_connect. Must be set to one of the following values: HCI_PAGE_SCAN_REPETITION_MODE_R0 HCI_PAGE_SCAN_REPETITION_MODE_R1 HCI_PAGE_SCAN_REPETITION_MODE_R2 The default value is HCI_PAGE_SCAN_REPETITION_MODE_R0.
	L2CAP_HCI_ROLE_SWITCH	brief L2CAP_HCI_ROLE_SWITCH. ingroup btconfig details Defines a default value of the role switch parameter when calling bt_l2cap_connect. Must be set to one of the following values: HCI_ROLE_SWITCH_ALLOW HCI_ROLE_SWITCH_DISALLOW The default value is to allow the role switch.
	L2CAP_IDLE_CONNECTION_TIMEOUT	seconds
	L2CAP_MAX_FIXED_CHANNELS	This is macro L2CAP_MAX_FIXED_CHANNELS.

Description

Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.

SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements.

Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.

l2cap_config_handlers.h

Macros

	Name	Description
	__L2CAP_CONFIG_HANDLERS_H	This is macro __L2CAP_CONFIG_HANDLERS_H.
	PACK_CONFIG_REQUEST	This is macro PACK_CONFIG_REQUEST.
	PACK_CONFIG_RESPONSE	This is macro PACK_CONFIG_RESPONSE.
	PACK_CONN_REQUEST	This is macro PACK_CONN_REQUEST.
	PACK_CONN_RESPONSE	This is macro PACK_CONN_RESPONSE.
	PACK_DCONN_REQUEST	This is macro PACK_DCONN_REQUEST.
	PACK_DCONN_RESPONSE	This is macro PACK_DCONN_RESPONSE.

	PACK_INFO_REQUEST	This is macro PACK_INFO_REQUEST.
	PACK_INFO_RESPONSE	This is macro PACK_INFO_RESPONSE.
	PROCESS_CONFIG_REQ	This is macro PROCESS_CONFIG_REQ.
	PROCESS_CONFIG_RES	This is macro PROCESS_CONFIG_RES.
	PROCESS_CONN_REQ	This is macro PROCESS_CONN_REQ.
	PROCESS_CONN_RES	This is macro PROCESS_CONN_RES.
	PROCESS_DCONN_REQ	This is macro PROCESS_DCONN_REQ.
	PROCESS_DCONN_RES	This is macro PROCESS_DCONN_RES.
	PROCESS_INFO_REQ	This is macro PROCESS_INFO_REQ.
	PROCESS_INFO_RES	This is macro PROCESS_INFO_RES.
	READ_CONFIG_REQUEST	This is macro READ_CONFIG_REQUEST.
	READ_CONFIG_RESPONSE	This is macro READ_CONFIG_RESPONSE.
	READ_CONN_REQUEST	This is macro READ_CONN_REQUEST.
	READ_CONN_RESPONSE	This is macro READ_CONN_RESPONSE.
	READ_DCONN_REQUEST	This is macro READ_DCONN_REQUEST.
	READ_DCONN_RESPONSE	This is macro READ_DCONN_RESPONSE.
	READ_INFO_REQUEST	This is macro READ_INFO_REQUEST.
	READ_INFO_RESPONSE	This is macro READ_INFO_RESPONSE.

Description

Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.

SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements.

Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.

l2cap_eretr.h

Enumerations

	Name	Description
	_bt_l2cap_eretr_xmit_event_e	This is type bt_l2cap_eretr_xmit_event_e.
	bt_l2cap_eretr_xmit_event_e	This is type bt_l2cap_eretr_xmit_event_e.

Functions

	Name	Description
	_bt_l2cap_eretr_handle_xmit_event	This is function _bt_l2cap_eretr_handle_xmit_event.
	_bt_l2cap_eretr_recv	This is function _bt_l2cap_eretr_recv.
	_bt_l2cap_eretr_retr_frames	This is function _bt_l2cap_eretr_retr_frames.
	_bt_l2cap_eretr_send_data	This is function _bt_l2cap_eretr_send_data.
	_bt_l2cap_eretr_send_pending_frames	This is function _bt_l2cap_eretr_send_pending_frames.
	_bt_l2cap_eretr_send_smart_data	This is function _bt_l2cap_eretr_send_smart_data.
	_bt_l2cap_fcs	This is function _bt_l2cap_fcs.
	_bt_l2cap_recv_req_seq_and_fbit	This is function _bt_l2cap_recv_req_seq_and_fbit.
	_bt_l2cap_send_ack	This is function _bt_l2cap_send_ack.
	_bt_l2cap_send_i_or_rr_or_rnr	This is function _bt_l2cap_send_i_or_rr_or_rnr.
	_bt_l2cap_send_rej	This is function _bt_l2cap_send_rej.
	_bt_l2cap_send_rnr	This is function _bt_l2cap_send_rnr.
	_bt_l2cap_send_rr	This is function _bt_l2cap_send_rr.
	_bt_l2cap_send_rr_or_rnr	This is function _bt_l2cap_send_rr_or_rnr.

Macros

	Name	Description
	_bt_l2cap_start_monitor_timer	This is macro _bt_l2cap_start_monitor_timer.
	_bt_l2cap_start_monitor_timer_if_not_running	This is macro _bt_l2cap_start_monitor_timer_if_not_running.
	_bt_l2cap_start_retr_timer	This is macro _bt_l2cap_start_retr_timer.

	_bt_l2cap_start_retr_timer_if_not_running	This is macro _bt_l2cap_start_retr_timer_if_not_running.
	_bt_l2cap_stop_monitor_timer	This is macro _bt_l2cap_stop_monitor_timer.
	_bt_l2cap_stop_retr_timer	This is macro _bt_l2cap_stop_retr_timer.
	GET_F_BIT	This is macro GET_F_BIT.
	GET_FRAME_TYPE	This is macro GET_FRAME_TYPE.
	GET_P_BIT	This is macro GET_P_BIT.
	GET_REQ_SEQ	This is macro GET_REQ_SEQ.
	GET_S_FUNCTION	This is macro GET_S_FUNCTION.
	GET_SAR	This is macro GET_SAR.
	GET_TX_SEQ	This is macro GET_TX_SEQ.
	SET_F_BIT	This is macro SET_F_BIT.
	SET_FRAME_TYPE	This is macro SET_FRAME_TYPE.
	SET_P_BIT	This is macro SET_P_BIT.
	SET_REQ_SEQ	This is macro SET_REQ_SEQ.
	SET_S_FUNCTION	This is macro SET_S_FUNCTION.
	SET_SAR	This is macro SET_SAR.
	SET_TX_SEQ	This is macro SET_TX_SEQ.

Structures

	Name	Description
	_bt_l2cap_xmit_event_param_t	This is type bt_l2cap_xmit_event_param_t.
	bt_l2cap_xmit_event_param_t	This is type bt_l2cap_xmit_event_param_t.

Description

Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.

SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements.

Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.

l2cap_fixed_channel.h

Functions

	Name	Description
	bt_l2cap_allocate_fixed_channel	This is function bt_l2cap_allocate_fixed_channel.
	bt_l2cap_find_fixed_channel	This is function bt_l2cap_find_fixed_channel.
	bt_l2cap_free_fixed_channel	This is function bt_l2cap_free_fixed_channel.

Macros

	Name	Description
	L2CAP_LINK_TYPE_BD_EDR	This is macro L2CAP_LINK_TYPE_BD_EDR.
	L2CAP_LINK_TYPE_LE	This is macro L2CAP_LINK_TYPE_LE.

Description

Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.

l2cap_packet.h

Functions

	Name	Description
	bt_l2cap_packet_cmd_assembler	This is function bt_l2cap_packet_cmd_assembler.
	bt_l2cap_packet_data_assembler	This is function bt_l2cap_packet_data_assembler.

Macros

	Name	Description
	L2CAP_PACKET_DATA_TYPE_RAW	This is macro L2CAP_PACKET_DATA_TYPE_RAW.
	L2CAP_PACKET_DATA_TYPE_SMART	This is macro L2CAP_PACKET_DATA_TYPE_SMART.

Structures

	Name	Description
	_bt_l2cap_packet_t	This is type bt_l2cap_packet_t.
	bt_l2cap_packet_t	This is type bt_l2cap_packet_t.

Description

Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.

l2cap_private.h

Functions

	Name	Description
	_bt_l2cap_clear_channel_cmd_queue	This is function _bt_l2cap_clear_channel_cmd_queue.
	_bt_l2cap_eretr_pack_config_request	This is function _bt_l2cap_eretr_pack_config_request.
	_bt_l2cap_process_connect_signal	This is function _bt_l2cap_process_connect_signal.
	_l2cap_allocate_buffers	Defined by OEM through library configuration
	_l2cap_data_receive_callback	From l2cap_recv.c
	_pack_cmd_reject	This is function _pack_cmd_reject.
	_pack_config_request	This is function _pack_config_request.
	_pack_config_response	This is function _pack_config_response.
	_pack_conn_param_update_request	This is function _pack_conn_param_update_request.
	_pack_conn_param_update_response	This is function _pack_conn_param_update_response.
	_pack_conn_request	This is function _pack_conn_request.
	_pack_conn_response	This is function _pack_conn_response.
	_pack_dconn_request	This is function _pack_dconn_request.
	_pack_dconn_response	This is function _pack_dconn_response.
	_pack_echo_request	This is function _pack_echo_request.
	_pack_echo_response	This is function _pack_echo_response.
	_pack_info_request	This is function _pack_info_request.
	_pack_info_response	This is function _pack_info_response.
	_process_config_req	From cmd_config.c
	_process_config_res	This is function _process_config_res.
	_process_conn_param_update_req	This is function _process_conn_param_update_req.
	_process_conn_param_update_res	This is function _process_conn_param_update_res.
	_process_conn_req	From cmd_connect.c
	_process_conn_res	This is function _process_conn_res.
	_process_dconn_req	From cmd_disconnect.c
	_process_dconn_res	This is function _process_dconn_res.
	_process_echo_req	From cmd_echo.c
	_process_echo_res	This is function _process_echo_res.
	_process_info_req	From cmd_info.c
	_process_info_res	This is function _process_info_res.
	_process_reject	From cmd_reject.c
	_process_unknown_req	From cmd_unknown.c
	_process_unknown_res	This is function _process_unknown_res.
	_read_cmd_reject	This is function _read_cmd_reject.
	_read_config_request	This is function _read_config_request.
	_read_config_response	This is function _read_config_response.

	_read_conn_param_update_request	This is function _read_conn_param_update_request.
	_read_conn_param_update_response	This is function _read_conn_param_update_response.
	_read_conn_request	This is function _read_conn_request.
	_read_conn_response	This is function _read_conn_response.
	_read_dconn_request	This is function _read_dconn_request.
	_read_dconn_response	This is function _read_dconn_response.
	_read_echo_request	This is function _read_echo_request.
	_read_echo_response	This is function _read_echo_response.
	_read_info_request	This is function _read_info_request.
	_read_info_response	This is function _read_info_response.
	read_command	From channel_cmd_recv.c

Macros

	Name	Description
	CHANNEL_SIGNAL_CMD_DISCONNECT_FIXED	This is macro CHANNEL_SIGNAL_CMD_DISCONNECT_FIXED.
	L2CAP_EXT_FEATURES_ENABLED	This is macro L2CAP_EXT_FEATURES_ENABLED.

Types

	Name	Description
	bt_l2cap_cmd_assembler_fp	This is type bt_l2cap_cmd_assembler_fp.
	bt_l2cap_cmd_parser_fp	This is type bt_l2cap_cmd_parser_fp.
	bt_l2cap_request_handler_fp	This is type bt_l2cap_request_handler_fp.
	bt_l2cap_response_handler_fp	This is type bt_l2cap_response_handler_fp.

Description

Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.

SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements.

Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.

l2cap_psm.h

Functions

	Name	Description
	bt_l2cap_allocate_psm	This is function bt_l2cap_allocate_psm.
	bt_l2cap_find_psm	This is function bt_l2cap_find_psm.
	bt_l2cap_free_psm	This is function bt_l2cap_free_psm.
	bt_l2cap_init_psms	This is function bt_l2cap_init_psms.

Description

Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.

SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements.

Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.

l2cap_signal.h

Functions

	Name	Description
	_bt_l2cap_init_signal	This is function _bt_l2cap_init_signal.
	_bt_l2cap_set_signal	This is function _bt_l2cap_set_signal.

Description

Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a

license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.

SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements.

Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.

l2cap_test.h

Macros

	Name	Description
	bt_l2cap_test_enable_local_config	This is macro bt_l2cap_test_enable_local_config.
	bt_l2cap_test_enable_remote_config	This is macro bt_l2cap_test_enable_remote_config.

Description

Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.

SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements.

Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.

l2cap_timer.h

Functions

	Name	Description
	_bt_l2cap_get_tick_count	This is function _bt_l2cap_get_tick_count.
	_bt_l2cap_init_timer	This is function _bt_l2cap_init_timer.

Description

Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.

SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements.

Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.

packet.h

Structures

	Name	Description
	_bt_packet_t	This is type bt_packet_t.
	bt_packet_t	This is type bt_packet_t.

Types

	Name	Description
	bt_packet_assembler_fp	This is type bt_packet_assembler_fp.

Description

Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.

patch.h

Macros

	Name	Description
	_W	This is macro _W.
	DCRH	Number of configuration blocks
	H	PS key store (0 = use default)
	W	This is macro W.

Description

Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.

queue.h

Functions

	Name	Description
	bt_q_add	This is function bt_q_add.
	bt_q_contains	This is function bt_q_contains.
	bt_q_get	This is function bt_q_get.
	bt_q_get_head	This is function bt_q_get_head.
	bt_q_get_length	This is function bt_q_get_length.
	bt_q_get_next	This is function bt_q_get_next.
	bt_q_push	This is function bt_q_push.
	bt_q_remove	This is function bt_q_remove.
	bt_q_remove_by_idx	This is function bt_q_remove_by_idx.

Structures

	Name	Description
	_bt_queue_element_t	This is type bt_queue_element_t.
	bt_queue_element_t	This is type bt_queue_element_t.

Description

Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.

SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements.

Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.

rfcomm.h

Functions

	Name	Description
	bt_rfcomm_allocate_dlc	brief Allocate DLC. ingroup rfcomm details This function allocates a new DLC on the specified RFCOMM session. param session The RFCOMM session. param dlci DLCI of the new DLC. return li A pointer to the new DLC if the function succeeds. li c NULL otherwise.
	bt_rfcomm_allocate_session	brief Allocate RFCOMM session. ingroup rfcomm details This function allocates a new RFCOMM session. param l2cap_mgr The L2CAP manager on which the RFCOMM session is to be created. param callback The callback function that is called when session state changes. param param An arbitrary data pointer that will be passed to the callback function specified by the c callback parameter. return li A pointer to the new RFCOMM session structure if the function succeeds. li c NULL otherwise.
	bt_rfcomm_cancel_listen	This is function bt_rfcomm_cancel_listen.
	bt_rfcomm_close_dlc	brief Close DLC. ingroup rfcomm details This function closes a DLC. If DLCI = 0, the parent RFCOMM session is also closed.
	bt_rfcomm_connect	brief Connect to a remote device. ingroup rfcomm details This function establishes an RFCOMM connection with a remote device and opens a data DLC. Changes in the session state are reported through a callback function specified when the session has been allocated via call to bt_rfcomm_allocate_session. Changes in the data DLC are reported through a callback function specified in this call. param remote_addr Address of the remote device. param server_channel A server channel of the remote RFCOMM server. param callback The callback function for reporting changes in DLC state opened by this call. param param An arbitrary data pointer... more
	bt_rfcomm_find_dlc	brief Find DLC

	bt_rfcomm_free_dlc	brief Release DLC. ingroup rfcomm details This function releases the specified DLC. param dlc The DLC to be released.
	bt_rfcomm_free_session	brief Release RFCOMM session. ingroup rfcomm details This function deallocates the specified RFCOMM session. This function does not disconnect the session. It just frees the memory used by the bt_rfcomm_session structure. The session has to be disconnected by calling bt_rfcomm_close_dlc with DLCI = 0 first. param session The RFCOMM session to be deallocated.
	bt_rfcomm_get_frame_length	This is function bt_rfcomm_get_frame_length.
	bt_rfcomm_init	brief Initialize the RFCOMM layer. ingroup rfcomm details This function initializes the RFCOMM layer of the stack. It must be called prior to any other RFCOMM function can be called.
	bt_rfcomm_listen	brief Listen for incoming connections. ingroup rfcomm details This function enables incoming connections on the specified RFCOMM session. Changes in the session state are reported through a callback function. param server_channel A server channel on which the RFCOMM session will listen and accept incoming connections. param callback The callback function that is called when session state changes. param param An arbitrary data pointer that will be passed to the callback function specified by the c callback parameter. return li c TRUE if the function succeeds. li c FALSE otherwise. The callback function is not called in this case.
	bt_rfcomm_open_dlc	brief Open DLC. ingroup rfcomm details This function opens the specified DLC. Before calling this function the RFCOMM session must be already open. This function is not to be used with DLCI = 0. Changes in DLC state are reported through a callback function. param dlc The DLC to open. param callback The callback function for reporting changes in DLC state. param param An arbitrary data pointer that will be passed to the callback function specified by the c callback parameter. return li c TRUE if the function succeeds. li c FALSE otherwise. The callback function is not called in... more
	bt_rfcomm_send_credit	This is function bt_rfcomm_send_credit.
	bt_rfcomm_send_data	brief Send data over a DLC. ingroup rfcomm details This function sends data over the specified DLC. Operation completion is reported through callback function. param dlc The DLC. param data A pointer to the data to be sent. param len Data length. param callback The callback function that is called when operation completes. return li c TRUE if the function succeeds. li c FALSE otherwise. The callback function is not called in this case.

Macros

	Name	Description
	MK_CMD_ADDRESS	define RFCOMM_MS_RTC
	MK_DLCI	This is macro MK_DLCI.
	RFCOMM_CFC_ENABLED	This is macro RFCOMM_CFC_ENABLED.
	RFCOMM_CFC_LOCAL_CREDIT	This is macro RFCOMM_CFC_LOCAL_CREDIT.
	RFCOMM_CFC_MAX_INITIAL_CREDIT	This is macro RFCOMM_CFC_MAX_INITIAL_CREDIT.
	RFCOMM_CMD_STATUS_FC_PENDING	This is macro RFCOMM_CMD_STATUS_FC_PENDING.
	RFCOMM_CMD_STATUS_PENDING	This is macro RFCOMM_CMD_STATUS_PENDING.
	RFCOMM_CMD_STATUS_WAITING_RESPONSE	This is macro RFCOMM_CMD_STATUS_WAITING_RESPONSE.
	RFCOMM_COMMAND	This is macro RFCOMM_COMMAND.
	RFCOMM_CTL_MSG_CLD	This is macro RFCOMM_CTL_MSG_CLD.
	RFCOMM_CTL_MSG_FCOFF	This is macro RFCOMM_CTL_MSG_FCOFF.
	RFCOMM_CTL_MSG_FCON	This is macro RFCOMM_CTL_MSG_FCON.
	RFCOMM_CTL_MSG_MSC	This is macro RFCOMM_CTL_MSG_MSC.
	RFCOMM_CTL_MSG_NSC	This is macro RFCOMM_CTL_MSG_NSC.
	RFCOMM_CTL_MSG_PN	This is macro RFCOMM_CTL_MSG_PN.
	RFCOMM_CTL_MSG_PSC	This is macro RFCOMM_CTL_MSG_PSC.
	RFCOMM_CTL_MSG_RLS	This is macro RFCOMM_CTL_MSG_RLS.
	RFCOMM_CTL_MSG_RPN	This is macro RFCOMM_CTL_MSG_RPN.
	RFCOMM_CTL_MSG_SNC	This is macro RFCOMM_CTL_MSG_SNC.
	RFCOMM_CTL_MSG_TEST	This is macro RFCOMM_CTL_MSG_TEST.

	RFCOMM_DLC_CHANGED_CONN_STATE	This is macro RFCOMM_DLC_CHANGED_CONN_STATE.
	RFCOMM_DLC_CHANGED_REMOTE_MSC	This is macro RFCOMM_DLC_CHANGED_REMOTE_MSC.
	RFCOMM_DLC_CONNECTION_FAILED	This is macro RFCOMM_DLC_CONNECTION_FAILED.
	RFCOMM_DLC_STATE_CLOSED	This is macro RFCOMM_DLC_STATE_CLOSED.
	RFCOMM_DLC_STATE_OPEN	This is macro RFCOMM_DLC_STATE_OPEN.
	RFCOMM_DLCI_CONTROL	This is macro RFCOMM_DLCI_CONTROL.
	RFCOMM_DLCI_FREE	This is macro RFCOMM_DLCI_FREE.
	RFCOMM_ERR_DM	This is macro RFCOMM_ERR_DM.
	RFCOMM_ERR_INTERRUPTED	This is macro RFCOMM_ERR_INTERRUPTED.
	RFCOMM_ERR_SUCCESS	This is macro RFCOMM_ERR_SUCCESS.
	RFCOMM_ERR_TIMEOUT	This is macro RFCOMM_ERR_TIMEOUT.
	RFCOMM_FC_TYPE_AGREGATE	This is macro RFCOMM_FC_TYPE_AGREGATE.
	RFCOMM_FC_TYPE_CREDIT	This is macro RFCOMM_FC_TYPE_CREDIT.
	RFCOMM_FLAG_CR	This is macro RFCOMM_FLAG_CR.
	RFCOMM_FLAG_EA	This is macro RFCOMM_FLAG_EA.
	RFCOMM_FLAG_PF	This is macro RFCOMM_FLAG_PF.
	RFCOMM_FRAME_HEADER_LEN	This is macro RFCOMM_FRAME_HEADER_LEN.
	RFCOMM_FRAME_TYPE_DISC	This is macro RFCOMM_FRAME_TYPE_DISC.
	RFCOMM_FRAME_TYPE_DM	This is macro RFCOMM_FRAME_TYPE_DM.
	RFCOMM_FRAME_TYPE_SABM	This is macro RFCOMM_FRAME_TYPE_SABM.
	RFCOMM_FRAME_TYPE_UA	This is macro RFCOMM_FRAME_TYPE_UA.
	RFCOMM_FRAME_TYPE_UI	This is macro RFCOMM_FRAME_TYPE_UI.
	RFCOMM_FRAME_TYPE_UIH	This is macro RFCOMM_FRAME_TYPE_UIH.
	RFCOMM_MAX_INFO_LEN	This is macro RFCOMM_MAX_INFO_LEN.
	RFCOMM_MIX_INFO_LEN	This is macro RFCOMM_MIX_INFO_LEN.
	RFCOMM_MX_MSG_MAX_DATA_LEN	This is macro RFCOMM_MX_MSG_MAX_DATA_LEN.
	RFCOMM_RESPONSE	This is macro RFCOMM_RESPONSE.
	RFCOMM_ROLE_INITIATOR	This is macro RFCOMM_ROLE_INITIATOR.
	RFCOMM_ROLE_RESPONDER	This is macro RFCOMM_ROLE_RESPONDER.
	RFCOMM_RPN_BAUD_RATE_1152	This is macro RFCOMM_RPN_BAUD_RATE_1152.
	RFCOMM_RPN_BAUD_RATE_192	This is macro RFCOMM_RPN_BAUD_RATE_192.
	RFCOMM_RPN_BAUD_RATE_2304	This is macro RFCOMM_RPN_BAUD_RATE_2304.
	RFCOMM_RPN_BAUD_RATE_24	This is macro RFCOMM_RPN_BAUD_RATE_24.
	RFCOMM_RPN_BAUD_RATE_384	This is macro RFCOMM_RPN_BAUD_RATE_384.
	RFCOMM_RPN_BAUD_RATE_48	This is macro RFCOMM_RPN_BAUD_RATE_48.
	RFCOMM_RPN_BAUD_RATE_576	This is macro RFCOMM_RPN_BAUD_RATE_576.
	RFCOMM_RPN_BAUD_RATE_72	This is macro RFCOMM_RPN_BAUD_RATE_72.
	RFCOMM_RPN_BAUD_RATE_96	default
	RFCOMM_RPN_DATA_BIT_5	This is macro RFCOMM_RPN_DATA_BIT_5.
	RFCOMM_RPN_DATA_BIT_6	This is macro RFCOMM_RPN_DATA_BIT_6.
	RFCOMM_RPN_DATA_BIT_7	This is macro RFCOMM_RPN_DATA_BIT_7.
	RFCOMM_RPN_DATA_BIT_8	default
	RFCOMM_RPN_FLC_N	This is macro RFCOMM_RPN_FLC_N.
	RFCOMM_RPN_FLC_RTC_INPUT	This is macro RFCOMM_RPN_FLC_RTC_INPUT.
	RFCOMM_RPN_FLC_RTC_OUTPUT	This is macro RFCOMM_RPN_FLC_RTC_OUTPUT.
	RFCOMM_RPN_FLC_RTR_INPUT	This is macro RFCOMM_RPN_FLC_RTR_INPUT.
	RFCOMM_RPN_FLC_RTR_OUTPUT	This is macro RFCOMM_RPN_FLC_RTR_OUTPUT.
	RFCOMM_RPN_FLC_XONOFF_INPUT	This is macro RFCOMM_RPN_FLC_XONOFF_INPUT.
	RFCOMM_RPN_FLC_XONOFF_OUTPUT	This is macro RFCOMM_RPN_FLC_XONOFF_OUTPUT.
	RFCOMM_RPN_PARITY_EVEN	This is macro RFCOMM_RPN_PARITY_EVEN.
	RFCOMM_RPN_PARITY_MARK	This is macro RFCOMM_RPN_PARITY_MARK.
	RFCOMM_RPN_PARITY_N	default
	RFCOMM_RPN_PARITY_ODD	This is macro RFCOMM_RPN_PARITY_ODD.
	RFCOMM_RPN_PARITY_SPACE	This is macro RFCOMM_RPN_PARITY_SPACE.
	RFCOMM_RPN_PARITY_Y	This is macro RFCOMM_RPN_PARITY_Y.

	RFCOMM_RPN_STOP_BIT_1	default
	RFCOMM_RPN_STOP_BIT_1_5	This is macro RFCOMM_RPN_STOP_BIT_1_5.
	RFCOMM_RPN_XOFF_DEFAULT	This is macro RFCOMM_RPN_XOFF_DEFAULT.
	RFCOMM_RPN_XON_DEFAULT	This is macro RFCOMM_RPN_XON_DEFAULT.
	RFCOMM_SERIAL_PORT_CH_1	This is macro RFCOMM_SERIAL_PORT_CH_1.
	RFCOMM_SERIAL_PORT_CH_10	This is macro RFCOMM_SERIAL_PORT_CH_10.
	RFCOMM_SERIAL_PORT_CH_11	This is macro RFCOMM_SERIAL_PORT_CH_11.
	RFCOMM_SERIAL_PORT_CH_12	This is macro RFCOMM_SERIAL_PORT_CH_12.
	RFCOMM_SERIAL_PORT_CH_13	This is macro RFCOMM_SERIAL_PORT_CH_13.
	RFCOMM_SERIAL_PORT_CH_14	This is macro RFCOMM_SERIAL_PORT_CH_14.
	RFCOMM_SERIAL_PORT_CH_15	This is macro RFCOMM_SERIAL_PORT_CH_15.
	RFCOMM_SERIAL_PORT_CH_16	This is macro RFCOMM_SERIAL_PORT_CH_16.
	RFCOMM_SERIAL_PORT_CH_17	This is macro RFCOMM_SERIAL_PORT_CH_17.
	RFCOMM_SERIAL_PORT_CH_18	This is macro RFCOMM_SERIAL_PORT_CH_18.
	RFCOMM_SERIAL_PORT_CH_19	This is macro RFCOMM_SERIAL_PORT_CH_19.
	RFCOMM_SERIAL_PORT_CH_2	This is macro RFCOMM_SERIAL_PORT_CH_2.
	RFCOMM_SERIAL_PORT_CH_20	This is macro RFCOMM_SERIAL_PORT_CH_20.
	RFCOMM_SERIAL_PORT_CH_21	This is macro RFCOMM_SERIAL_PORT_CH_21.
	RFCOMM_SERIAL_PORT_CH_22	This is macro RFCOMM_SERIAL_PORT_CH_22.
	RFCOMM_SERIAL_PORT_CH_23	This is macro RFCOMM_SERIAL_PORT_CH_23.
	RFCOMM_SERIAL_PORT_CH_24	This is macro RFCOMM_SERIAL_PORT_CH_24.
	RFCOMM_SERIAL_PORT_CH_25	This is macro RFCOMM_SERIAL_PORT_CH_25.
	RFCOMM_SERIAL_PORT_CH_26	This is macro RFCOMM_SERIAL_PORT_CH_26.
	RFCOMM_SERIAL_PORT_CH_27	This is macro RFCOMM_SERIAL_PORT_CH_27.
	RFCOMM_SERIAL_PORT_CH_28	This is macro RFCOMM_SERIAL_PORT_CH_28.
	RFCOMM_SERIAL_PORT_CH_29	This is macro RFCOMM_SERIAL_PORT_CH_29.
	RFCOMM_SERIAL_PORT_CH_3	This is macro RFCOMM_SERIAL_PORT_CH_3.
	RFCOMM_SERIAL_PORT_CH_30	This is macro RFCOMM_SERIAL_PORT_CH_30.
	RFCOMM_SERIAL_PORT_CH_4	This is macro RFCOMM_SERIAL_PORT_CH_4.
	RFCOMM_SERIAL_PORT_CH_5	This is macro RFCOMM_SERIAL_PORT_CH_5.
	RFCOMM_SERIAL_PORT_CH_6	This is macro RFCOMM_SERIAL_PORT_CH_6.
	RFCOMM_SERIAL_PORT_CH_7	This is macro RFCOMM_SERIAL_PORT_CH_7.
	RFCOMM_SERIAL_PORT_CH_8	This is macro RFCOMM_SERIAL_PORT_CH_8.
	RFCOMM_SERIAL_PORT_CH_9	This is macro RFCOMM_SERIAL_PORT_CH_9.
	RFCOMM_SESSION_CHANGED_AFC	This is macro RFCOMM_SESSION_CHANGED_AFC.
	RFCOMM_SESSION_CHANGED_CONN_STATE	This is macro RFCOMM_SESSION_CHANGED_CONN_STATE.
	RFCOMM_SESSION_STATE_CONNECTED	This is macro RFCOMM_SESSION_STATE_CONNECTED.
	RFCOMM_SESSION_STATE_DISCONNECTED	This is macro RFCOMM_SESSION_STATE_DISCONNECTED.
	RFCOMM_SESSION_STATE_FREE	This is macro RFCOMM_SESSION_STATE_FREE.
	RFCOMM_TIMEOUT	This is macro RFCOMM_TIMEOUT.

Structures

	Name	Description
	_bt_rfcomm_command_t	This is type bt_rfcomm_command_t.
	_bt_rfcomm_ctl_msg_t	This is type bt_rfcomm_ctl_msg_t.
	_bt_rfcomm_dlc_t	This is type bt_rfcomm_dlc_t.
	_bt_rfcomm_session_t	This is type bt_rfcomm_session_t.
	bt_rfcomm_command_p	This is type bt_rfcomm_command_p.
	bt_rfcomm_command_t	This is type bt_rfcomm_command_t.
	bt_rfcomm_ctl_msg_p	This is type bt_rfcomm_ctl_msg_p.
	bt_rfcomm_ctl_msg_t	This is type bt_rfcomm_ctl_msg_t.
	bt_rfcomm_dlc_p	This is type bt_rfcomm_dlc_p.
	bt_rfcomm_dlc_t	This is type bt_rfcomm_dlc_t.
	bt_rfcomm_session_p	This is type bt_rfcomm_session_p.
	bt_rfcomm_session_t	This is type bt_rfcomm_session_t.

Types

	Name	Description
	bt_rfcomm_cmd_callback_fp	This is type bt_rfcomm_cmd_callback_fp.
	bt_rfcomm_dlc_state_callback_fp	This is type bt_rfcomm_dlc_state_callback_fp.
	bt_rfcomm_read_data_callback_fp	This is type bt_rfcomm_read_data_callback_fp.
	bt_rfcomm_send_data_callback_fp	This is type bt_rfcomm_send_data_callback_fp.
	bt_rfcomm_state_callback_fp	This is type bt_rfcomm_state_callback_fp.

Description

Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.

SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements.

Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.

rfcomm_cmd_queue.h

Functions

	Name	Description
	rfcomm_cq_ack_cmd	This is function rfcomm_cq_ack_cmd.
	rfcomm_cq_ack_mx_cmd	This is function rfcomm_cq_ack_mx_cmd.
	rfcomm_cq_find_failed_pn	This is function rfcomm_cq_find_failed_pn.
	rfcomm_send_commands_from_queue	This is function rfcomm_send_commands_from_queue.

Description

Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.

SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements.

Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.

rfcomm_config.h

Macros

	Name	Description
	__RFCOMM_CONFIG_H	This is macro __RFCOMM_CONFIG_H.
	RFCOMM_ALLOCATE_BUFFERS_FUNCTION	This is macro RFCOMM_ALLOCATE_BUFFERS_FUNCTION.
	RFCOMM_ALLOCATE_BUFFERS_RAM_SIZE_VAR	This is macro RFCOMM_ALLOCATE_BUFFERS_RAM_SIZE_VAR.
	RFCOMM_ALLOCATE_BUFFERS_VARS	This is macro RFCOMM_ALLOCATE_BUFFERS_VARS.
	RFCOMM_BUFFER_SIZE	This is macro RFCOMM_BUFFER_SIZE.
	RFCOMM_ENABLE_MULTIDEVICE_CHANNELS	brief Enable multi-device server channels. ingroup rfcomm_config details Normally each server channel can be used only once. I.e. if device A connected to channel 1, device B cannot connect to channel 1 until device A disconnects. With this option it is possible to make channels accept connections from several devices at the same time. I.e., if RFCOMM_ENABLE_MULTIDEVICE_CHANNELS is TRUE both device A and device B can connect to channel 1 at the same time.
	RFCOMM_INFO_LEN	brief Maximum size of the data portion of a UIH frame. ingroup rfcomm_config details This parameter defines the maximum size of the data portion of a UIH frame. If CFC is used the actual length of the data portion will be 1 byte less. This value must be less than or equal to HCI_L2CAP_BUFFER_LEN - RFCOMM_FRAME_HEADER_LEN - L2CAP_HEADER_LEN.

	RFCOMM_LOCAL_CREDIT	brief The number of receive buffers. ingroup rfcomm_config details This parameter defines the number of received UIH frames that can be stored on the local device. The flow control mechanism used in RFCOMM ensures that the remote side of the link always knows how many free buffers left on the local device. When the number of free buffers reaches 0, the transmitter stops sending data frames until the receiver frees some buffers. The RFCOMM layer does not actually allocate space for buffers. It uses RFCOMM_LOCAL_CREDIT to keep track of free buffers and report them to the remote side. Actual memory... more
	RFCOMM_LOCAL_CREDIT_SEND_THRESHOLD_DECL	This is macro RFCOMM_LOCAL_CREDIT_SEND_THRESHOLD_DECL.
	RFCOMM_MAX_CMD_BUFFERS	brief Maximum number of command buffers. ingroup rfcomm_config details This parameter defines the maximum number of commands that can be sent at the same time. It is usually enough to reserve 2 buffers for each DLC excluding control DLC. Therefore, this value can be defined as rn #define RFCOMM_MAX_CMD_BUFFERS (RFCOMM_MAX_DLCS - 1) * 2
	RFCOMM_MAX_DLCS	brief Maximum number of DLCs ingroup rfcomm_config details This parameter defines the maximum number of DLCs on each session. This value should be at least 2 because each session uses one DLC to convey multiplexer control messages. All other DLCs are used to emulate serial ports.
	RFCOMM_MAX_SERVER_CHANNELS	brief Maximum number of Server channels ingroup rfcomm_config details This parameter defines the maximum number of server channels exposed by the local device. This value should not exceed RFCOMM_MAX_DLCS - 1.
	RFCOMM_MAX_SESSIONS	brief Maximum number of remote devices a local device can be connected to ingroup rfcomm_config details This parameter defines the maximum number of remote devices a local device can have simultaneous connections to. This value should not exceed HCI_MAX_HCI_CONNECTIONS.

Description

Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.

SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements.

Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.

rfcomm_mgr.h

Functions

	Name	Description
	_bt_rfcomm_allocate_channel	This is function _bt_rfcomm_allocate_channel.
	_bt_rfcomm_find_channel	This is function _bt_rfcomm_find_channel.
	_bt_rfcomm_get_mgr	This is function _bt_rfcomm_get_mgr.
	_bt_rfcomm_mgr_l2cap_listen_callback	This is function _bt_rfcomm_mgr_l2cap_listen_callback.
	_bt_rfcomm_mgr_notify_listeners	This is function _bt_rfcomm_mgr_notify_listeners.
	_rfcomm_init_mgr	This is function _rfcomm_init_mgr.
	bt_rfcomm_register_listener	This is function bt_rfcomm_register_listener.
	bt_rfcomm_unregister_listener	This is function bt_rfcomm_unregister_listener.

Structures

	Name	Description
	_bt_rfcomm_mgr_t	This is type bt_rfcomm_mgr_t.
	_bt_rfcomm_server_channel_t	This is type bt_rfcomm_server_channel_t.
	_bt_rfcomm_session_listener_t	This is type bt_rfcomm_session_listener_t.
	bt_rfcomm_mgr_t	This is type bt_rfcomm_mgr_t.
	bt_rfcomm_server_channel_t	This is type bt_rfcomm_server_channel_t.
	bt_rfcomm_session_listener_t	This is type bt_rfcomm_session_listener_t.

Description

Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.

SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements.

Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.

rfcomm_mx.h

Functions

	Name	Description
	_rfcomm_allocate_mx_cmd	This is function _rfcomm_allocate_mx_cmd.
	_rfcomm_mx_process_fc	This is function _rfcomm_mx_process_fc.
	_rfcomm_mx_process_pn	This is function _rfcomm_mx_process_pn.
	_rfcomm_mx_process_rpn	This is function _rfcomm_mx_process_rpn.
	_rfcomm_process_mx_fc_response	This is function _rfcomm_process_mx_fc_response.
	_rfcomm_process_mx_msc_response	This is function _rfcomm_process_mx_msc_response.
	_rfcomm_process_mx_pn_response	This is function _rfcomm_process_mx_pn_response.
	_rfcomm_process_mx_rls_response	This is function _rfcomm_process_mx_rls_response.
	_rfcomm_send_mx_fc_cmd	This is function _rfcomm_send_mx_fc_cmd.
	_rfcomm_send_mx_msc_response	This is function _rfcomm_send_mx_msc_response.
	_rfcomm_send_mx_nsc_response	This is function _rfcomm_send_mx_nsc_response.
	_rfcomm_send_mx_rls_cmd	This is function _rfcomm_send_mx_rls_cmd.
	_rfcomm_send_mx_rls_response	This is function _rfcomm_send_mx_rls_response.
	_rfcomm_send_mx_test_response	This is function _rfcomm_send_mx_test_response.
	rfcomm_send_mx_msc_cmd	This is function rfcomm_send_mx_msc_cmd.
	rfcomm_send_mx_pn_cmd	This is function rfcomm_send_mx_pn_cmd.

Macros

	Name	Description
	RFCOMM_LINE_STATUS_FRAMING	This is macro RFCOMM_LINE_STATUS_FRAMING.
	RFCOMM_LINE_STATUS_OVERRUN	This is macro RFCOMM_LINE_STATUS_OVERRUN.
	RFCOMM_LINE_STATUS_PARITY	This is macro RFCOMM_LINE_STATUS_PARITY.
	RFCOMM_MODEM_STATUS_DV	This is macro RFCOMM_MODEM_STATUS_DV.
	RFCOMM_MODEM_STATUS_FC	This is macro RFCOMM_MODEM_STATUS_FC.
	RFCOMM_MODEM_STATUS_IC	This is macro RFCOMM_MODEM_STATUS_IC.
	RFCOMM_MODEM_STATUS_RTC	This is macro RFCOMM_MODEM_STATUS_RTC.
	RFCOMM_MODEM_STATUS_RTR	This is macro RFCOMM_MODEM_STATUS_RTR.
	RFCOMM_MX_MSG_FCOFF	This is macro RFCOMM_MX_MSG_FCOFF.
	RFCOMM_MX_MSG_FCON	This is macro RFCOMM_MX_MSG_FCON.
	RFCOMM_MX_MSG_MSC	This is macro RFCOMM_MX_MSG_MSC.
	RFCOMM_MX_MSG_NSC	This is macro RFCOMM_MX_MSG_NSC.
	RFCOMM_MX_MSG_PN	This is macro RFCOMM_MX_MSG_PN.
	RFCOMM_MX_MSG_RLS	This is macro RFCOMM_MX_MSG_RLS.
	RFCOMM_MX_MSG_RPN	This is macro RFCOMM_MX_MSG_RPN.
	RFCOMM_MX_MSG_TEST	This is macro RFCOMM_MX_MSG_TEST.

Description

Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.

SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements.

Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.

rfcomm_private.h

Functions

	Name	Description
	_bt_rfcomm_clear_queue	From command_queue.c
	_bt_rfcomm_mgr_allocate_session	This is function _bt_rfcomm_mgr_allocate_session.
	_calc_fcs	This is function _calc_fcs.
	_rfcomm_alloc_cmd_buffer	This is function _rfcomm_alloc_cmd_buffer.
	_rfcomm_allocate_buffers	Defined by OEM through library configuration
	_rfcomm_free_cmd_buffer	This is function _rfcomm_free_cmd_buffer.
	_rfcomm_init_cmd_buffers	From rfcomm_cmdbuffer.c
	_rfcomm_init_sessions	From rfcomm_session.c
	_rfcomm_l2cap_read_data_callback	From rfcomm_cmd_recv.c
	_rfcomm_l2cap_state_changed_callback	From rfcomm.c
	_rfcomm_process_cmd_frame_disc	From frame_disc.c
	_rfcomm_process_cmd_frame_dm	This is function _rfcomm_process_cmd_frame_dm.
	_rfcomm_process_cmd_frame_sabm	This is function _rfcomm_process_cmd_frame_sabm.
	_rfcomm_process_cmd_frame_ua	This is function _rfcomm_process_cmd_frame_ua.
	_rfcomm_process_cmd_frame_uih	From frame_uih.c
	_rfcomm_process_res_frame_disc	This is function _rfcomm_process_res_frame_disc.
	_rfcomm_process_res_frame_dm	This is function _rfcomm_process_res_frame_dm.
	_rfcomm_process_res_frame_sabm	This is function _rfcomm_process_res_frame_sabm.
	_rfcomm_process_res_frame_ua	This is function _rfcomm_process_res_frame_ua.
	_rfcomm_process_res_frame_uih	This is function _rfcomm_process_res_frame_uih.
	_rfcomm_send_command	From rfcomm_cmd_send.c
	_rfcomm_send_dm_response	From frame_dm.c
	_rfcomm_send_sabm_cmd	From frame_sabm.c
	_rfcomm_send_ua_response	From frame_ua.c
	check_fcs	From rfcomm_fcs.c
	rfcomm_find_session	This is function rfcomm_find_session.
	rfcomm_send_cmd	This is function rfcomm_send_cmd.

Description

Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.

SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements.

Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.

rfcomm_signal.h

Functions

	Name	Description
	_bt_rfcomm_init_signal	This is function _bt_rfcomm_init_signal.
	_bt_rfcomm_set_signal	This is function _bt_rfcomm_set_signal.

Description

Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.

SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements.

Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.

rfcomm_timer.h

Functions

	Name	Description
	_rfcomm_get_tick_count	This is function _rfcomm_get_tick_count.
	_rfcomm_init_timer	This is function _rfcomm_init_timer.

Description

Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.

SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements.

Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.

sbc.h

Provides SBC Decoder Library interface services.

Functions

	Name	Description
	sbc_decode	Decodes the encoded SBC frame.
	sbc_get_codesize	Gets the input block size.
	sbc_get_frame_duration	Gets the time one input/output block takes to play in microseconds.
	sbc_get_frame_length	Gets the output block size.
	sbc_get_state_size	Gets the SBC Decoder's state data structure.
	sbc_init	Initializes the SBC Decoder.

Macros

	Name	Description
	__SBC_H	This is macro __SBC_H.
	INP_BUF_SIZE	Bytes
	MAX_SBC_DEC_STATE_SIZE	Bytes
	OUT_BUF_SIZE	Bytes
	SBC_AM_LOUDNESS	Allocation method
	SBC_AM_SNR	This is macro SBC_AM_SNR.
	SBC_BE	This is macro SBC_BE.
	SBC_BLK_12	This is macro SBC_BLK_12.
	SBC_BLK_16	This is macro SBC_BLK_16.
	SBC_BLK_4	Blocks
	SBC_BLK_8	This is macro SBC_BLK_8.
	SBC_FREQ_16000	Sampling frequency
	SBC_FREQ_32000	This is macro SBC_FREQ_32000.
	SBC_FREQ_44100	This is macro SBC_FREQ_44100.
	SBC_FREQ_48000	This is macro SBC_FREQ_48000.
	SBC_LE	Data endianness
	SBC_MODE_DUAL_CHANNEL	This is macro SBC_MODE_DUAL_CHANNEL.
	SBC_MODE_JOINT_STEREO	This is macro SBC_MODE_JOINT_STEREO.
	SBC_MODE_MONO	Channel mode
	SBC_MODE_STEREO	This is macro SBC_MODE_STEREO.
	SBC_SB_4	Sub-bands
	SBC_SB_8	This is macro SBC_SB_8.
	ssize_t	

Structures

	Name	Description
	sbc_struct	

Types

	Name	Description
	sbc_t	This is type sbc_t.

Description

SBC Decoder Library Interface File

SBC Decoder interface services to decode the encoded data received, from the Bluetooth device, which will be executed in audioTask.

File Name

sbc.h

Company

Microchip Technology Inc.

sdp.h

Functions

	Name	Description
	bt_sdp_start	brief Start SDP server ingroup sdp details This function starts the SDP server. param l2cap_mgr The L2CAP manager on which the SDP server is to be started. param sdp_db A pointer to the SDP database define with BEGIN_SDP_DB and END_SDP_DB macros. return li c TRUE if the function succeeds. li c FALSE otherwise.

Macros

	Name	Description
	BEGIN_DE_SEQUENCE	• brief Begin a data element sequence • ingroup sdp * • details BEGIN_DE_SEQUENCE and END_DE_SEQUENCE are used to define a data element sequence which is an array of sdp_data_element structures. • The array is used a search pattern in bt_sdp_request_service_search() and bt_sdp_request_service_attribute(). • For example, to find a AVRCP Target the following code can be used: code const bt_uuid_t AVRCP_AV_REMOTE_CONTROL_CLSID = { 0x5F9B34FB, 0x80000080, 0x00001000, SDP_CLSID_AV_REMOTE_CONTROL }; const bt_uuid_t AVRCP_AV_REMOTE_CONTROL_TARGET_CLSID = { 0x5F9B34FB, 0x80000080, 0x00001000, SDP_CLSID_AV_REMOTE_CONTROL_TARGET }; BEGIN_DE_SEQUENCE(avrcp_target_service_search, 2) DE_UUID128(AVRCP_AV_REMOTE_CONTROL_CLSID) DE_UUID128(AVRCP_AV_REMOTE_CONTROL_TARGET_CLSID) END_DE_SEQUENCE(avrcp_target_service_search) . . . void findAvrcpTarget(void) { INIT_DE_SEQUENCE(avrcp_target_service_search); bt_sdp_request_service_search(channel, &seq_avrcp_target_service_search, &callback, NULL); } endcode * • param... more

	DE_BOOL	brief Declare a boolean data element ingroup sdp details This macro adds a boolean data element to a data element sequence. This macro is to be used between BEGIN_DE_SEQUENCE and END_DE_SEQUENCE. param value The data element value.
	DE_INT	brief Declare a 1-byte signed integer data element ingroup sdp details This macro adds a 1-byte signed integer data element to a data element sequence. This macro is to be used between BEGIN_DE_SEQUENCE and END_DE_SEQUENCE. param value The data element value.
	DE_STRING	brief Declare a text string data element ingroup sdp details This macro adds a text string data element to a data element sequence. The length of the generated data element will be the actual length of the string. This macro is to be used between BEGIN_DE_SEQUENCE and END_DE_SEQUENCE. param value The data element value.
	DE_STRING2	brief Declare a text string data element ingroup sdp details This macro adds a text string data element to a data element sequence. The length of the generated data element will be the value specified by the "len" parameter even if the actual length of the string is not equal to the "len" value. This macro is to be used between BEGIN_DE_SEQUENCE and END_DE_SEQUENCE. param value The data element value. param len The length of the data element value.
	DE_UINT	brief Declare a 1-byte unsigned integer data element ingroup sdp details This macro adds a 1-byte unsigned integer data element to a data element sequence. This macro is to be used between BEGIN_DE_SEQUENCE and END_DE_SEQUENCE. param value The data element value.
	DE_UINT16	brief Declare a 2-byte unsigned integer data element ingroup sdp details This macro adds a 2-byte unsigned integer data element to a data element sequence. This macro is to be used between BEGIN_DE_SEQUENCE and END_DE_SEQUENCE. param value The data element value.
	DE_URL	brief Declare a URL data element ingroup sdp details This macro adds a URL data element to a data element sequence. This macro is to be used between BEGIN_DE_SEQUENCE and END_DE_SEQUENCE. param value The data element value which is a pointer to a string.
	DE_UUID128	brief Declare a 128-bit UUID data element ingroup sdp details This macro adds a 128-bit UUID data element to a data element sequence. This macro is to be used between BEGIN_DE_SEQUENCE and END_DE_SEQUENCE. param value The data element value. The value must be a name of a variable of type bt_uuid.
	DE_UUID16	brief Declare a 16-bit UUID data element ingroup sdp details This macro adds a 16-bit UUID data element to a data element sequence. This macro is to be used between BEGIN_DE_SEQUENCE and END_DE_SEQUENCE. param value The data element value.
	DE_UUID32	brief Declare a 32-bit UUID data element ingroup sdp details This macro adds a 32-bit UUID data element to a data element sequence. This macro is to be used between BEGIN_DE_SEQUENCE and END_DE_SEQUENCE. param value The data element value.
	END_DE_SEQUENCE	brief End a data element sequence ingroup sdp details BEGIN_DE_SEQUENCE and END_DE_SEQUENCE are used to define a data element sequence which is an array of bt_sdp_data_element structures. param id The data element sequence identifier.
	INIT_DE_SEQUENCE	brief Initialize a data element sequence ingroup sdp details This macro calls a function defined in BEGIN_DE_SEQUENCE which initializes the data element sequence. param id The data element sequence identifier.
	SDP_ATTRID_AdditionalProtocolDescriptorLists	This is macro SDP_ATTRID_AdditionalProtocolDescriptorLists.
	SDP_ATTRID_BluetoothProfileDescriptorList	This is macro SDP_ATTRID_BluetoothProfileDescriptorList.

	SDP_ATTRID_BrowseGroupList	This is macro SDP_ATTRID_BrowseGroupList.
	SDP_ATTRID_ClientExecutableURL	This is macro SDP_ATTRID_ClientExecutableURL.
	SDP_ATTRID_DIPrimaryRecord	This is macro SDP_ATTRID_DIPrimaryRecord.
	SDP_ATTRID_DIProductId	This is macro SDP_ATTRID_DIProductId.
	SDP_ATTRID_DISpecificationId	This is macro SDP_ATTRID_DISpecificationId.
	SDP_ATTRID_DIVendorId	This is macro SDP_ATTRID_DIVendorId.
	SDP_ATTRID_DIVendorIdSource	This is macro SDP_ATTRID_DIVendorIdSource.
	SDP_ATTRID_DIVersion	This is macro SDP_ATTRID_DIVersion.
	SDP_ATTRID_DocumentationURL	This is macro SDP_ATTRID_DocumentationURL.
	SDP_ATTRID_GAPRemoteAudioVolumeControl	This is macro SDP_ATTRID_GAPRemoteAudioVolumeControl.
	SDP_ATTRID_GroupID	This is macro SDP_ATTRID_GroupID.
	SDP_ATTRID_HCRP_1284ID	This is macro SDP_ATTRID_HCRP_1284ID.
	SDP_ATTRID_HCRP_DeviceLocation	This is macro SDP_ATTRID_HCRP_DeviceLocation.
	SDP_ATTRID_HCRP_DeviceName	This is macro SDP_ATTRID_HCRP_DeviceName.
	SDP_ATTRID_HCRP_FriendlyName	This is macro SDP_ATTRID_HCRP_FriendlyName.
	SDP_ATTRID_HDPDataExchangeSpecification	This is macro SDP_ATTRID_HDPDataExchangeSpecification.
	SDP_ATTRID_HDPMCAPSupportedProcedures	This is macro SDP_ATTRID_HDPMCAPSupportedProcedures.
	SDP_ATTRID_HDPSuportedFeatures	This is macro SDP_ATTRID_HDPSuportedFeatures.
	SDP_ATTRID_HFPAGNetwork	This is macro SDP_ATTRID_HFPAGNetwork.
	SDP_ATTRID_HFPSupportedFeatures	This is macro SDP_ATTRID_HFPSupportedFeatures.
	SDP_ATTRID_HIDBatteryPower	This is macro SDP_ATTRID_HIDBatteryPower.
	SDP_ATTRID_HIDBootDevice	This is macro SDP_ATTRID_HIDBootDevice.
	SDP_ATTRID_HIDCountryCode	This is macro SDP_ATTRID_HIDCountryCode.
	SDP_ATTRID_HIDDescriptorList	This is macro SDP_ATTRID_HIDDescriptorList.
	SDP_ATTRID_HIDDeviceReleaseNumber	This is macro SDP_ATTRID_HIDDeviceReleaseNumber.
	SDP_ATTRID_HIDDeviceSubclass	This is macro SDP_ATTRID_HIDDeviceSubclass.
	SDP_ATTRID_HIDLANGIDBaseList	This is macro SDP_ATTRID_HIDLANGIDBaseList.
	SDP_ATTRID_HIDNormallyConnectable	This is macro SDP_ATTRID_HIDNormallyConnectable.
	SDP_ATTRID_HIDParserVersion	This is macro SDP_ATTRID_HIDParserVersion.
	SDP_ATTRID_HIDProfileVersion	This is macro SDP_ATTRID_HIDProfileVersion.
	SDP_ATTRID_HIDReconnectInitiate	This is macro SDP_ATTRID_HIDReconnectInitiate.
	SDP_ATTRID_HIDRemoteWake	This is macro SDP_ATTRID_HIDRemoteWake.
	SDP_ATTRID_HIDSDPDisable	This is macro SDP_ATTRID_HIDSDPDisable.
	SDP_ATTRID_HIDSupervisionTimeout	This is macro SDP_ATTRID_HIDSupervisionTimeout.
	SDP_ATTRID_HIDVirtualCable	This is macro SDP_ATTRID_HIDVirtualCable.
	SDP_ATTRID_IconURL	This is macro SDP_ATTRID_IconURL.
	SDP_ATTRID_INVALID	This is macro SDP_ATTRID_INVALID.
	SDP_ATTRID_LanguageBaseAttributeIDList	This is macro SDP_ATTRID_LanguageBaseAttributeIDList.
	SDP_ATTRID_OFFSET_ProviderName	This is macro SDP_ATTRID_OFFSET_ProviderName.
	SDP_ATTRID_OFFSET_ServiceDescription	This is macro SDP_ATTRID_OFFSET_ServiceDescription.
	SDP_ATTRID_OFFSET_ServiceName	This is macro SDP_ATTRID_OFFSET_ServiceName.
	SDP_ATTRID_PrimaryLanguageBaseId	This is macro SDP_ATTRID_PrimaryLanguageBaseId.
	SDP_ATTRID_ProtocolDescriptorList	This is macro SDP_ATTRID_ProtocolDescriptorList.
	SDP_ATTRID_ServiceAvailability	This is macro SDP_ATTRID_ServiceAvailability.
	SDP_ATTRID_ServiceClassIDList	This is macro SDP_ATTRID_ServiceClassIDList.
	SDP_ATTRID_ServiceDatabaseState	This is macro SDP_ATTRID_ServiceDatabaseState.
	SDP_ATTRID_ServiceID	This is macro SDP_ATTRID_ServiceID.
	SDP_ATTRID_ServiceInfoTimeToLive	This is macro SDP_ATTRID_ServiceInfoTimeToLive.
	SDP_ATTRID_ServiceRecordHandle	This is macro SDP_ATTRID_ServiceRecordHandle.
	SDP_ATTRID_ServiceRecordState	This is macro SDP_ATTRID_ServiceRecordState.
	SDP_ATTRID_SupportedFeatures	This is macro SDP_ATTRID_SupportedFeatures.
	SDP_ATTRID_VersionNumberList	This is macro SDP_ATTRID_VersionNumberList.
	SDP_CLSID_ADVANCED_AUDIO_DISTRIBUTION	This is macro SDP_CLSID_ADVANCED_AUDIO_DISTRIBUTION.
	SDP_CLSID_AUDIO_SINK	This is macro SDP_CLSID_AUDIO_SINK.
	SDP_CLSID_AUDIO_SOURCE	This is macro SDP_CLSID_AUDIO_SOURCE.

	SDP_CLSID_AV_REMOTE_CONTROL	This is macro SDP_CLSID_AV_REMOTE_CONTROL.
	SDP_CLSID_AV_REMOTE_CONTROL_CONTROLLER	This is macro SDP_CLSID_AV_REMOTE_CONTROL_CONTROLLER.
	SDP_CLSID_AV_REMOTE_CONTROL_PROFILE_ID	This is macro SDP_CLSID_AV_REMOTE_CONTROL_PROFILE_ID.
	SDP_CLSID_AV_REMOTE_CONTROL_TARGET	This is macro SDP_CLSID_AV_REMOTE_CONTROL_TARGET.
	SDP_CLSID_AVCTP	This is macro SDP_CLSID_AVCTP.
	SDP_CLSID_AVDTP	This is macro SDP_CLSID_AVDTP.
	SDP_CLSID_BrowseGroupDescriptorServiceClassID	This is macro SDP_CLSID_BrowseGroupDescriptorServiceClassID.
	SDP_CLSID_DialupNetworking	This is macro SDP_CLSID_DialupNetworking.
	SDP_CLSID_GENERIC_AUDIO	This is macro SDP_CLSID_GENERIC_AUDIO.
	SDP_CLSID_HARD_COPY_CABLE_REPLACEMENT	This is macro SDP_CLSID_HARD_COPY_CABLE_REPLACEMENT.
	SDP_CLSID_HARD_COPY_CONTROL_CHANNEL	This is macro SDP_CLSID_HARD_COPY_CONTROL_CHANNEL.
	SDP_CLSID_HARD_COPY_DATA_CHANNEL	This is macro SDP_CLSID_HARD_COPY_DATA_CHANNEL.
	SDP_CLSID_HARD_COPY_NOTIFICATION	This is macro SDP_CLSID_HARD_COPY_NOTIFICATION.
	SDP_CLSID_HCR_PRINT	This is macro SDP_CLSID_HCR_PRINT.
	SDP_CLSID_HCR_SCAN	This is macro SDP_CLSID_HCR_SCAN.
	SDP_CLSID_HDP	This is macro SDP_CLSID_HDP.
	SDP_CLSID_HDP_SINK	This is macro SDP_CLSID_HDP_SINK.
	SDP_CLSID_HDP_SOURCE	This is macro SDP_CLSID_HDP_SOURCE.
	SDP_CLSID_HFP	This is macro SDP_CLSID_HFP.
	SDP_CLSID_HFP_AG	This is macro SDP_CLSID_HFP_AG.
	SDP_CLSID_HID	This is macro SDP_CLSID_HID.
	SDP_CLSID_HIDProtocol	This is macro SDP_CLSID_HIDProtocol.
	SDP_CLSID_HSP	This is macro SDP_CLSID_HSP.
	SDP_CLSID_HSP_AG	This is macro SDP_CLSID_HSP_AG.
	SDP_CLSID_HSP_HS	This is macro SDP_CLSID_HSP_HS.
	SDP_CLSID_L2CAP	This is macro SDP_CLSID_L2CAP.
	SDP_CLSID_MCAP_CONTROL	This is macro SDP_CLSID_MCAP_CONTROL.
	SDP_CLSID_MCAP_DATA	This is macro SDP_CLSID_MCAP_DATA.
	SDP_CLSID_OBEXFileTransfer	This is macro SDP_CLSID_OBEXFileTransfer.
	SDP_CLSID_OBEXObjectPush	This is macro SDP_CLSID_OBEXObjectPush.
	SDP_CLSID_PBAP_PCE	This is macro SDP_CLSID_PBAP_PCE.
	SDP_CLSID_PBAP_PSE	This is macro SDP_CLSID_PBAP_PSE.
	SDP_CLSID_PNPInformation	This is macro SDP_CLSID_PNPInformation.
	SDP_CLSID_PublicBrowseGroup	This is macro SDP_CLSID_PublicBrowseGroup.
	SDP_CLSID_RFCOMM	This is macro SDP_CLSID_RFCOMM.
	SDP_CLSID_SerialPort	This is macro SDP_CLSID_SerialPort.
	SDP_CLSID_ServiceDiscoveryServerServiceClassID	defgroup sdp SDP This module describe functions and data structures used to start the SDP server and perform SDP queries.
	SDP_DATA_TYPE_ALTERNATIVE	This is macro SDP_DATA_TYPE_ALTERNATIVE.
	SDP_DATA_TYPE_BOOL	This is macro SDP_DATA_TYPE_BOOL.
	SDP_DATA_TYPE_INT	This is macro SDP_DATA_TYPE_INT.
	SDP_DATA_TYPE_INT128	This is macro SDP_DATA_TYPE_INT128.
	SDP_DATA_TYPE_INT16	This is macro SDP_DATA_TYPE_INT16.
	SDP_DATA_TYPE_INT32	This is macro SDP_DATA_TYPE_INT32.
	SDP_DATA_TYPE_INT64	This is macro SDP_DATA_TYPE_INT64.
	SDP_DATA_TYPE_INT8	This is macro SDP_DATA_TYPE_INT8.
	SDP_DATA_TYPE_NIL	This is macro SDP_DATA_TYPE_NIL.
	SDP_DATA_TYPE_SEQUENCE	This is macro SDP_DATA_TYPE_SEQUENCE.
	SDP_DATA_TYPE_STRING	This is macro SDP_DATA_TYPE_STRING.
	SDP_DATA_TYPE_UINT	This is macro SDP_DATA_TYPE_UINT.
	SDP_DATA_TYPE_UINT128	This is macro SDP_DATA_TYPE_UINT128.
	SDP_DATA_TYPE_UINT16	This is macro SDP_DATA_TYPE_UINT16.
	SDP_DATA_TYPE_UINT32	This is macro SDP_DATA_TYPE_UINT32.
	SDP_DATA_TYPE_UINT64	This is macro SDP_DATA_TYPE_UINT64.

	SDP_DATA_TYPE_UINT8	This is macro SDP_DATA_TYPE_UINT8.
	SDP_DATA_TYPE_URL	This is macro SDP_DATA_TYPE_URL.
	SDP_DATA_TYPE_UUID	This is macro SDP_DATA_TYPE_UUID.
	SDP_DATA_TYPE_UUID128	This is macro SDP_DATA_TYPE_UUID128.
	SDP_DATA_TYPE_UUID16	This is macro SDP_DATA_TYPE_UUID16.
	SDP_DATA_TYPE_UUID32	This is macro SDP_DATA_TYPE_UUID32.
	SDP_ERROR_INSUFFICIENT_RESOURCE	This is macro SDP_ERROR_INSUFFICIENT_RESOURCE.
	SDP_ERROR_INVALID_CONTINUATION_STATE	This is macro SDP_ERROR_INVALID_CONTINUATION_STATE.
	SDP_ERROR_INVALID_PDU_SIZE	This is macro SDP_ERROR_INVALID_PDU_SIZE.
	SDP_ERROR_INVALID_REQUEST_SYNTAX	This is macro SDP_ERROR_INVALID_REQUEST_SYNTAX.
	SDP_ERROR_INVALID_SDP_VERSION	This is macro SDP_ERROR_INVALID_SDP_VERSION.
	SDP_ERROR_INVALID_SR_HANDLE	This is macro SDP_ERROR_INVALID_SR_HANDLE.
	SDP_ERROR_RESERVED	This is macro SDP_ERROR_RESERVED.
	SDP_ErrorResponse	This is macro SDP_ErrorResponse.
	SDP_FTP_SERVICE_ID	This is macro SDP_FTP_SERVICE_ID.
	SDP_HID_SERVICE_ID	This is macro SDP_HID_SERVICE_ID.
	SDP_HSP_AG_SERVICE_ID	This is macro SDP_HSP_AG_SERVICE_ID.
	SDP_HSP_HS_SERVICE_ID	This is macro SDP_HSP_HS_SERVICE_ID.
	SDP_MAX_ATTRIBUTE_PATTERN_LEN	This is macro SDP_MAX_ATTRIBUTE_PATTERN_LEN.
	SDP_MAX_DATA_ELEMENT_LEN	This is macro SDP_MAX_DATA_ELEMENT_LEN.
	SDP_MAX_DATA_ELEMENTS	This is macro SDP_MAX_DATA_ELEMENTS.
	SDP_MAX_SEARCH_PATTERN_LEN	This is macro SDP_MAX_SEARCH_PATTERN_LEN.
	SDP_MAX_TRANSACTIONS	typedef struct _sdp_db { bt_int count; bt_sdp_service_record_p *records; } bt_sdp_db, *bt_sdp_db_p;
	SDP_PDU_HEADER_LEN	This is macro SDP_PDU_HEADER_LEN.
	SDP_RFCOMM_SERVICE_ID	This is macro SDP_RFCOMM_SERVICE_ID.
	SDP_ServiceAttributeRequest	This is macro SDP_ServiceAttributeRequest.
	SDP_ServiceAttributeResponse	This is macro SDP_ServiceAttributeResponse.
	SDP_ServiceSearchAttributeRequest	This is macro SDP_ServiceSearchAttributeRequest.
	SDP_ServiceSearchAttributeResponse	This is macro SDP_ServiceSearchAttributeResponse.
	SDP_ServiceSearchRequest	This is macro SDP_ServiceSearchRequest.
	SDP_ServiceSearchResponse	This is macro SDP_ServiceSearchResponse.
	SDP_SR_HANDLE_HDP_SINK	This is macro SDP_SR_HANDLE_HDP_SINK.
	SDP_SR_HANDLE_HDP_SOURCE	This is macro SDP_SR_HANDLE_HDP_SOURCE.
	SDP_SR_HANDLE_HFP_HF	This is macro SDP_SR_HANDLE_HFP_HF.
	SDP_SR_HANDLE_HID	This is macro SDP_SR_HANDLE_HID.
	SDP_SR_HANDLE_HID_KEYBOARD	This is macro SDP_SR_HANDLE_HID_KEYBOARD.
	SDP_SR_HANDLE_HSP_HS	This is macro SDP_SR_HANDLE_HSP_HS.
	SDP_SR_HANDLE_OBEXFileTransfer	This is macro SDP_SR_HANDLE_OBEXFileTransfer.
	SDP_SR_HANDLE_OBEXObjectPush	This is macro SDP_SR_HANDLE_OBEXObjectPush.
	SDP_SR_HANDLE_PNPINFORMATION	This is macro SDP_SR_HANDLE_PNPINFORMATION.
	SDP_SR_HANDLE_RFCOMM	This is macro SDP_SR_HANDLE_RFCOMM.
	SDP_SR_HANDLE_SERVER	This is macro SDP_SR_HANDLE_SERVER.
	SDP_SR_HANDLE_TEST	for simulating service search using continuation state

Structures

	Name	Description
	_bt_sdp_data_element_t	This is type bt_sdp_data_element_t.
	_bt_sdp_found_attr_list_t	This is type bt_sdp_found_attr_list_t.
	_bt_sdp_sequence_t	This is type bt_sdp_sequence_t.
	_bt_sdp_serialization_state_t	This is type bt_sdp_serialization_state_t.
	_bt_sdp_service_transaction_t	This is type bt_sdp_service_transaction_t.
	_bt_sdp_transaction_t	This is type bt_sdp_transaction_t.
	bt_sdp_data_element_t	This is type bt_sdp_data_element_t.
	bt_sdp_found_attr_list_t	This is type bt_sdp_found_attr_list_t.

	bt_sdp_sequence_t	This is type bt_sdp_sequence_t.
	bt_sdp_serialization_state_p	This is type bt_sdp_serialization_state_p.
	bt_sdp_serialization_state_t	This is type bt_sdp_serialization_state_t.
	bt_sdp_service_transaction_p	This is type bt_sdp_service_transaction_p.
	bt_sdp_service_transaction_t	This is type bt_sdp_service_transaction_t.
	bt_sdp_transaction_t	This is type bt_sdp_transaction_t.

Types

	Name	Description
	bt_sdp_data_element_cp	This is type bt_sdp_data_element_cp.
	bt_sdp_data_element_p	This is type bt_sdp_data_element_p.
	bt_sdp_sequence_cp	This is type bt_sdp_sequence_cp.
	bt_sdp_sequence_p	This is type bt_sdp_sequence_p.
	bt_sr_handle_p	This is type bt_sr_handle_p.
	bt_sr_handle_t	This is type bt_sr_handle_t.

Description

Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.

SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements.

Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.

sdp_client.h

Functions

	Name	Description
	bt_sdp_request_service_attribute	brief Search attributes ingroup sdp details This function retrieves attribute values from a service record. param channel The L2CAP channel used to communicate to the remote SDP server. param sr The service record handle specifies the service record from which attribute values are to be retrieved. param pattern The attribute search pattern is a data element sequence where each element in the list is either an attribute ID or a range of attribute IDs. The pattern buffer must be valid for the duration of the search operation, i.e. until c callback is called for the first time. To define a... more
	bt_sdp_request_service_search	brief Search service records ingroup sdp details This function locates service records on a remote SDP server that match the given service search pattern. param channel The L2CAP channel used to communicate to the remote SDP server. param pattern The service search pattern is a data element sequence where each element in the sequence is a UUID. The sequence must contain at least one UUID. The maximum number of UUIDs in the sequence is 12. The pattern buffer must be valid for the duration of the search operation, i.e. until c callback is called. To define a data element sequence... more

Macros

	Name	Description
	SDP_CLIENT_EVT_CONNECTED	This is macro SDP_CLIENT_EVT_CONNECTED.
	SDP_CLIENT_EVT_CONNECTION_FAILED	This is macro SDP_CLIENT_EVT_CONNECTION_FAILED.
	SDP_CLIENT_EVT_DISCONNECTED	This is macro SDP_CLIENT_EVT_DISCONNECTED.
	SDP_CLIENT_EVT_NULL	This is macro SDP_CLIENT_EVT_NULL.

Structures

	Name	Description
	bt_sdp_client_evt_connected_t	This is record bt_sdp_client_evt_connected_t.
	bt_sdp_client_evt_disconnected_t	This is record bt_sdp_client_evt_disconnected_t.

Types

	Name	Description
	bt_sdp_client_callback_fp	This is type bt_sdp_client_callback_fp.

	bt_sdp_read_de_callback_ftp	This is type bt_sdp_read_de_callback_ftp.
	bt_sdp_service_attribute_callback_fp	This is type bt_sdp_service_attribute_callback_fp.
	bt_sdp_service_search_callback_fp	This is type bt_sdp_service_search_callback_fp.

Description

Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.

SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements.

Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.

sdp_config.h

Macros

	Name	Description
	__SDP_CONFIG_H	This is macro __SDP_CONFIG_H.

	SDP_ALLOCATE_BUFFERS_RAM_SIZE_VAR	• defgroup sdp_config Configuration
		• ingroup sdp
		*
		• This module describes parameters used to configure SDP.
		*
		• dotstack is customized using a configuration file. The configuration file tailors the dotstack to the application being built. It has to have the structure shown below.
		* code #include "cdbt/bt/bt_std.h"
		// HCI and L2CAP must always be present // SDP is required only if stack is running in dual mode. This is the default mode. // To run the stack in single mode (i.e. only BLE is supported) a BT_BLE_SINGLE_MODE symbol // must be defined: // #define BT_BLE_SINGLE_MODE
		// HCI configuration parameters #define... more

	SDP_ALLOCATE_BUFFERS_VARS	This is macro SDP_ALLOCATE_BUFFERS_VARS.
	SDP_MAX_ATTRIBUTE_RESULT_LEN	brief Maximum number of attributes to find ingroup sdp_config details This parameter defines the maximum number of attributes withing a service record the SDP server will return to the client.
	SDP_MAX_PDU_BUFFERS	brief Maximum number of SDP server PDU buffers. ingroup sdp_config details This parameter defines the maximum number of responses the SDP server can send at the same time.
	SDP_MAX_SEARCH_RESULT_LEN	brief Maximum number of service records to find. ingroup sdp_config details This parameter defines the maximum number of service records the SDP server will return to the client.

Description

Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.

SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements.

Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.

sdp_packet.h

Functions

	Name	Description
	bt_sdp_packet_assembler	This is function bt_sdp_packet_assembler.

Structures

	Name	Description
	_bt_sdp_packet_t	This is type bt_sdp_packet_t.
	bt_sdp_packet_t	This is type bt_sdp_packet_t.

Description

Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.

sdp_private.h

Functions

	Name	Description
	_bt_sdp_client_init	This is function _bt_sdp_client_init.
	_sdp_alloc_svc_tran_buffer	This is function _sdp_alloc_svc_tran_buffer.
	_sdp_alloc_tran_buffer	This is function _sdp_alloc_tran_buffer.
	_sdp_find_svc_transaction	This is function _sdp_find_svc_transaction.
	_sdp_find_transaction	This is function _sdp_find_transaction.
	_sdp_free_svc_tran_buffer	This is function _sdp_free_svc_tran_buffer.
	_sdp_free_tran_buffer	This is function _sdp_free_tran_buffer.
	_sdp_init_tran_buffers	From sdp_tran_buffer.c
	bt_sdp_read_attribute	From sdp_db_utils.c
	sdp_compare_uuid_de	This is function sdp_compare_uuid_de.
	sdp_find_attributes	This is function sdp_find_attributes.
	sdp_find_service_records2	From sdp_utils.c

Structures

	Name	Description
	_bt_sdp_server_attribute_t	This is type bt_sdp_server_attribute_t.
	_bt_sdp_server_data_element_t	Private types
	_bt_sdp_server_record_t	This is type bt_sdp_server_record_t.
	bt_sdp_server_attribute_t	This is type bt_sdp_server_attribute_t.
	bt_sdp_server_data_element_t	Private types
	bt_sdp_server_record_t	This is type bt_sdp_server_record_t.

Description

Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.

SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements.

Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.

sdp_utils.h

Functions

	Name	Description
	_sdp_get_de_data_len	This is function _sdp_get_de_data_len.
	_sdp_get_de_hdr_len	This is function _sdp_get_de_hdr_len.
	_sdp_read_de_header	This is function _sdp_read_de_header.
	_sdp_write_data_element	This is function _sdp_write_data_element.
	bt_sdp_de_to_uuid	This is function bt_sdp_de_to_uuid.

Description

Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.

SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements.

Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.

spp.h

Enumerations

	Name	Description
	_bt_spp_port_event_e	• Serial port event. Values of this enumeration represent serial port events. Events are reported to the application through a callback function. The callback function is specified when the port is allocated using bt_spp_allocate().
	_bt_spp_port_state_e	• Serial port state. Values of this enumeration represent states of the serial port.
	_bt_spp_send_status_e	Send operation status.
	bt_spp_port_event_e	• Serial port event. Values of this enumeration represent serial port events. Events are reported to the application through a callback function. The callback function is specified when the port is allocated using bt_spp_allocate().
	bt_spp_port_state_e	• Serial port state. Values of this enumeration represent states of the serial port.
	bt_spp_send_status_e	Send operation status.

Functions

	Name	Description
	bt_spp_allocate	• Allocate a serial port. The returned serial port is initially in the ::SPP_PORT_STATE_DISCONNECTED state. To establish a connection with a remote device, call bt_spp_connect(). To listen for incoming connections from other devices, call bt_spp_listen(). The p callback parameter must specify a callback function that will be used to notify about serial port events and state changes. When the port is not needed any more it must be deallocated by bt_spp_deallocate(). The maximum number of serial ports that can be allocated simultaneously is specified by the ::SPP_MAX_PORTS configuration parameter. @param l2cap_mgr L2CAP manager. @param callback Pointer to... more
	bt_spp_cancel_listen	• Stop listening for incoming connections. This function stops the port to accept incoming connections on a given server channel. @param port Serial port. @param channel The RFCOMM server channel to accept connections on. @return c TRUE if successful, c FALSE otherwise.
	bt_spp_cancel_receive	• Cancel receive data. If a receive operation is currently in progress this function will cancel it. After calling this function the receive callback specified in bt_spp_receive() will not be called. If there is no receive operation in progress calling this function has no effect. @param port Serial port.
	bt_spp_cancel_send	• Cancel send data. If a send operation is currently in progress this function will try to cancel it. When the operation is canceled the send callback function will be called with the ::SPP_SEND_STATUS_CANCELED status. If this function is called but the active send operation completes successfuly before the stack can actually cancel it the call back function will still be called with the ::SPP_SEND_STATUS_CANCELED status. If there is no send operation in progess calling this function has no effect. @param port Serial port.
	bt_spp_clr_port_options	This is function bt_spp_clr_port_options.

	bt_spp_connect	• Connect to a remote device. This function initiates a connection to a remote device. When the connection is successfully established the port's callback is called with the ::SPP_PORT_EVENT_CONNECTED event. If connection fails the callback is called with the ::SPP_PORT_EVENT_CONNECTION_FAILED event. The port must be in ::SPP_PORT_STATE_DISCONNECTED state. Otherwise, the function will fail. @param port Serial port. @param remote_addr Bluetooth address of the remote device. @param channel RFCOMM server channel on which the connection is to be established. @return c TRUE if successful, c FALSE otherwise.
	bt_spp_deallocate	• Deallocate serial port. This function deallocates the specified port structure and other resources associated with it. The port must be in ::SPP_PORT_STATE_DISCONNECTED state. Otherwise, the function will fail. If the function completes successfully the application must not try to access any fields in the structure and must not use it with any other SPP functions. Also, it becomes available for subsequent allocation by bt_spp_port_allocate(). @param port Serial port structure to deallocate. @return c TRUE if successful, c FALSE otherwise.
	bt_spp_disconnect	• Disconnect from the remote device. This function initiates the disconnection process. When it is complete the the port's callback is called with the SPP_PORT_EVENT_DISCONNECTED event. If the port is already in the disconnected state the function does nothing and the callback is not called. @param port Serial port.
	bt_spp_find_server	This is function bt_spp_find_server.
	bt_spp_find_server_ex	This is function bt_spp_find_server_ex.
	bt_spp_get_frame_length	• Get frame length. This function returns the RFCOMM frame length used by the RFCOMM protocol. The frame length depends on configuration of DotStack and configuration of the Bluetooth stack running on the remote device. In order to achieve maximum throughput over the serial port connection the application should send and receive data in chunks that are multiple of this frame length. @return RFCOMM frame length in bytes.
	bt_spp_get_hci_connection	• Get SPP port's ACL connection. This function returns a pinter to the structure that describes the ACL connection this port is on. @return Pointer to ACL connection description if the port is connected, NULL otherwise.
	bt_spp_get_local_modem_status	• Get local device's TS 07.10 control signals. This function returns current state of the local device's TS 07.10 controls signals. The signals are defined as a mask of the following constants: SPP_RS232_DSR SPP_RS232_RTS SPP_RS232_RI SPP_RS232_DCD @param port Serial port. @return local device's TS 07.10 control signals.
	bt_spp_get_remote_address	brief Get the address of the remote device this device is connected to. ingroup spp param port Serial port. return li c A pointer to bt_bdaddr structure that contains the address of the remote device.
	bt_spp_get_remote_modem_status	• Get remote device's TS 07.10 control signals. This function returns current state of the remote device's V.24 controls signals. The signals are defined as a mask of the following constants: SPP_RS232_DTR SPP_RS232_CTS SPP_RS232_RI SPP_RS232_DCD @param port Serial port. @return remote device's TS 07.10 control signals.
	bt_spp_init	• Initialize the SPP module. This function initializes all internal variables of the SPP module. It must be called prior to using any other functions in this module.

	bt_spp_listen	• Listen for incoming connections. This function registers the port to accept incoming connections from remote devices on a particular RFCOMM server channel. The specified server channel should be listed in the SDP database. Otherwise, remote devices will not be able to find out which server channel to use. When a remote device successfully establishes a connection on the specified port the port's callback is called with the ::SPP_PORT_EVENT_CONNECTED event. The port must be in ::SPP_PORT_STATE_DISCONNECTED state. Otherwise, the function will fail. @param port Serial port. @param channel The RFCOMM server channel on which to listen for connections. @return c TRUE... more
	bt_spp_receive	• Receive data. This function receives data from the serial port connection. The caller must provide a buffer and a callback function. Whenever the port receives data they are copied to the provided buffer and the callback function is called. The callback function is passed the length of received data and the same callback parameter that was specified when the port was allocated with bt_spp_allocate(). This function does not wait until the buffer is filled out completely. Any amount of received data will complete the operation. The port must be in ::SPP_PORT_STATE_CONNECTED state. Otherwise, the function will fail. Also, the... more
	bt_spp_send	• Send data. This function starts sending data over the serial port connection. Along with the data the caller must provide a callback function that is called when all data has been sent. Also, during execution of this operation the port's state callback function is called periodically with the ::SPP_PORT_EVENT_SEND_PROGRESS event. The port must be in ::SPP_PORT_STATE_CONNECTED state. Otherwise, the function will fail. Also, the function will fail if a previously started send operation is still in progress. The callback function is passed the same callback parameter that was specified when the port was allocated with bt_spp_allocate(). @param port Serial... more
	bt_spp_set_dtr	• Set local device's RS-232 DTR signal. Changes the state of local device's RS-232 DTR signal and notifies the remote device of the change. This signal corresponds to the TS 07.10's RFCOMM_MODEM_STATUS_RTC signal. If there are resources to send a command to the remote device this command will return TRUE. If the remote party has been successfully notified SPP_PORT_EVENT_LOCAL_MODEM_STATUS_CHANGED event will be reported. If the command could not be send to the remote party for any reason other than lack of resources SPP_PORT_EVENT_LOCAL_MODEM_STATUS_CHANGE_FAILED will be reported. If the status cannot be changed because there is no resources to send a... more
	bt_spp_set_local_modem_status	• Set local device's TS 07.10 control signals. Changes the state of local device's TS 07.10 control signals and notifies the remote device of the change. If there are resources to send a command to the remote device this command will return TRUE. If the remote party has been successfully notified SPP_PORT_EVENT_LOCAL_MODEM_STATUS_CHANGED event will be reported. If the command could not be send to the remote party for any reason other than lack of resources SPP_PORT_EVENT_LOCAL_MODEM_STATUS_CHANGE_FAILED will be reported. If the status cannot be changed because there is no resources to send a command to the remote device this function... more
	bt_spp_set_port_options	This is function bt_spp_set_port_options.

	bt_spp_set_rts	• Set local device's RS-232 RTS signal. Changes the state of local device's RS-232 DTR signal and notifies the remote device of the change. This signal corresponds to the TS 07.10's RFCOMM_MODEM_STATUS_RTR signal. If there are resources to send a command to the remote device this command will return TRUE. If the remote party has been successfully notified SPP_PORT_EVENT_LOCAL_MODEM_STATUS_CHANGED event will be reported. If the command could not be send to the remote party for any reason other than lack of resources SPP_PORT_EVENT_LOCAL_MODEM_STATUS_CHANGE_FAILED will be reported. If the status cannot be changed because there is no resources to send a... more

Macros

	Name	Description
	SPP_PORT_OPTION_ENCRYPTED	This is macro SPP_PORT_OPTION_ENCRYPTED.
	SPP_PORT_OPTION_MASTER	This is macro SPP_PORT_OPTION_MASTER.
	SPP_PORT_OPTION_SECURE	This is macro SPP_PORT_OPTION_SECURE.
	SPP_PORT_TYPE_INCOMING	This is macro SPP_PORT_TYPE_INCOMING.
	SPP_PORT_TYPE_OUTGOING	This is macro SPP_PORT_TYPE_OUTGOING.
	SPP_RS232_CTS	This is macro SPP_RS232_CTS.
	SPP_RS232_DCD	This is macro SPP_RS232_DCD.
	SPP_RS232_DSR	• @defgroup spp Serial Port Profile (SPP) The DotStack SPP API is a simple API for communicating over a Bluetooth link using the Bluetooth Serial Port Profile. Here are the steps for using this API: • Call the bt_spp_init() function. • Allocate a serial port structure with bt_spp_allocate(). One of the parameters to this function is a pointer to a callback function. That callback function will be called by the stack whenever the state of the serial port changes. • To connect to a remote device call bt_spp_connect(). The stack will notify when the connection is established by calling the state... more
	SPP_RS232_DTR	This is macro SPP_RS232_DTR.
	SPP_RS232_RI	This is macro SPP_RS232_RI.
	SPP_RS232_RTS	This is macro SPP_RS232_RTS.

Structures

	Name	Description
	_bt_spp_port_t	• Serial port structure. This structure represents a Bluetooth serial port. Application code may only use those fields that are documented. The rest of the fields are private to the SPP implementation.

Types

	Name	Description
	bt_spp_find_server_callback_fp	This is type bt_spp_find_server_callback_fp.
	bt_spp_port_t	This is type bt_spp_port_t.
	bt_spp_receive_callback_fp	• Serial port receive callback. This callback function is called when a receive operation initiated by bt_spp_receive() completes. @param port Serial port on which the receive operation completed. @param bytes_received Number of received bytes. @param param Callback parameter that was specified when bt_spp_allocate() was called.

	bt_spp_send_callback_fp	• Serial port send callback. This callback function is called when a send operation initiated by bt_spp_send() completes. @param port Serial port on which the send operation completed. @param bytes_sent Number of bytes sent. This parameter is just a convenience as it always specifies the same number of bytes that was passed to bt_spp_send(); @param status Completion status. It is one of the values defined in the ::bt_spp_send_status_e enumeration. @param param Callback parameter that was specified when bt_spp_allocate() was called.
	bt_spp_state_callback_fp	• Serial port state callback. This callback function is called whenever the state of a serial port is changed. @param port Serial port which state has changed. @param evt Event describing the nature of state change. It is one of the values defined in the ::bt_spp_port_event_e enumeration. @param param Callback parameter that was specified when bt_spp_allocate() was called.

Description

Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.

SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements.

Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.

spp_config.h

Macros

	Name	Description
	__SPP_CONFIG_H	This is macro __SPP_CONFIG_H.
	SPP_ALLOCATE_BUFFERS_RAM_SIZE_VAR	This is macro SPP_ALLOCATE_BUFFERS_RAM_SIZE_VAR.
	SPP_ALLOCATE_BUFFERS_VARS	This is macro SPP_ALLOCATE_BUFFERS_VARS.
	SPP_DECLARE_FRAME_BUFFERS	This is macro SPP_DECLARE_FRAME_BUFFERS.
	SPP_DISABLE_BUFFERING	This is macro SPP_DISABLE_BUFFERING.
	SPP_FRAME_BUFFERS_RAM_SIZE	This is macro SPP_FRAME_BUFFERS_RAM_SIZE.
	SPP_FRAME_BUFFERS_SIZE	This is macro SPP_FRAME_BUFFERS_SIZE.
	SPP_MAX_PORTS	brief Maximum number of SPP ports. ingroup spp_config details This parameter defines the maximum number of SPP port that can be open between the local and remote devices. If RFCOMM_ENABLE_MULTIDEVICE_CHANNELS is FALSE (default) this value should be equal to RFCOMM_MAX_SERVER_CHANNELS. If RFCOMM_ENABLE_MULTIDEVICE_CHANNELS is TRUE this value should be between RFCOMM_MAX_SERVER_CHANNELS and RFCOMM_MAX_SERVER_CHANNELS * RFCOMM_MAX_SESSIONS.

Description

Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.

SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements.

Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.

spp_private.h

Functions

	Name	Description
	_bt_spp_client_init	This is function _bt_spp_client_init.
	_bt_spp_find_port	This is function _bt_spp_find_port.
	_bt_spp_handle_rx	This is function _bt_spp_handle_rx.
	_bt_spp_handle_tx	This is function _bt_spp_handle_tx.
	_bt_spp_rfcomm_read_data_callback	This is function _bt_spp_rfcomm_read_data_callback.

	_bt_spp_rfcomm_send_data_callback	This is function _bt_spp_rfcomm_send_data_callback.

Description

Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.

SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements.

Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.

ssp.h

Enumerations

	Name	Description
	_SSP_AUTHENTICATION_REQUIREMENTS	This is type SSP_AUTHENTICATION_REQUIREMENTS.
	_SSP_IO_CAPABILITY	This is type SSP_IO_CAPABILITY.
	_SSP_KEYPRESS_NOTIFICATION_TYPE	This is type SSP_KEYPRESS_NOTIFICATION_TYPE.
	_SSP_MODE	This is type SSP_MODE.
	_SSP_OOB_DATA_PRESENT	This is type SSP_OOB_DATA_PRESENT.
	SSP_AUTHENTICATION_REQUIREMENTS	This is type SSP_AUTHENTICATION_REQUIREMENTS.
	SSP_IO_CAPABILITY	This is type SSP_IO_CAPABILITY.
	SSP_KEYPRESS_NOTIFICATION_TYPE	This is type SSP_KEYPRESS_NOTIFICATION_TYPE.
	SSP_MODE	This is type SSP_MODE.
	SSP_OOB_DATA_PRESENT	This is type SSP_OOB_DATA_PRESENT.

Functions

	Name	Description
	bt_ssp_evt_handler	This is function bt_ssp_evt_handler.
	bt_ssp_init	This is function bt_ssp_init.
	bt_ssp_read_local_oob_data	This is function bt_ssp_read_local_oob_data.
	bt_ssp_send_keypress_notification	This is function bt_ssp_send_keypress_notification.
	bt_ssp_send_user_confirmation	This is function bt_ssp_send_user_confirmation.
	bt_ssp_send_user_passkey	This is function bt_ssp_send_user_passkey.
	bt_ssp_set_io_capabilities	This is function bt_ssp_set_io_capabilities.
	bt_ssp_set_mode	This is function bt_ssp_set_mode.
	bt_ssp_set_oob_data	This is function bt_ssp_set_oob_data.

Macros

	Name	Description
	OOB_DATA_HASH_LENGTH	Remote OOB Data Request Event
	OOB_DATA_RANDOMIZER_LENGTH	This is macro OOB_DATA_RANDOMIZER_LENGTH.
	SSP_MAX_MANAGERS	This is macro SSP_MAX_MANAGERS.

Structures

	Name	Description
	_bt_ssp_io_capability	This is type bt_ssp_io_capability.
	_bt_ssp_keypress_notification	Keypress Notification Event
	_bt_ssp_oob_data	This is type bt_ssp_oob_data.
	_bt_ssp_simple_pairing_complete	Simple Pairing Complete Event
	_bt_ssp_user_confirmation_request	User Confirmation Request Event
	_bt_ssp_user_passkey_notification	User Passkey Notification Event
	_bt_ssp_user_passkey_request	User Passkey Request Event
	bt_ssp_io_capability	This is type bt_ssp_io_capability.
	bt_ssp_keypress_notification	Keypress Notification Event
	bt_ssp_oob_data	This is type bt_ssp_oob_data.
	bt_ssp_simple_pairing_complete	Simple Pairing Complete Event

	bt_ssp_user_confirmation_request	User Confirmation Request Event
	bt_ssp_user_passkey_notification	User Passkey Notification Event
	bt_ssp_user_passkey_request	User Passkey Request Event

Types

	Name	Description
	bt_spp_read_local_oob_data_callback_fp	This is type bt_spp_read_local_oob_data_callback_fp.

Description

Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.

SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements.

Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.

ssp_event.h

Enumerations

	Name	Description
	_SSP_EVENT	This is type SSP_EVENT.
	SSP_EVENT	This is type SSP_EVENT.

Description

Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.

SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements.

Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.

ssp_event_handler.h

Functions

	Name	Description
	ssp_evt_io_capability_request	This is function ssp_evt_io_capability_request.
	ssp_evt_io_capability_response	This is function ssp_evt_io_capability_response.
	ssp_evt_keypress_notification	This is function ssp_evt_keypress_notification.
	ssp_evt_oob_data_request	This is function ssp_evt_oob_data_request.
	ssp_evt_ssp_complete	This is function ssp_evt_ssp_complete.
	ssp_evt_user_confirmation_request	This is function ssp_evt_user_confirmation_request.
	ssp_evt_user_passkey_notification	This is function ssp_evt_user_passkey_notification.
	ssp_evt_user_passkey_request	This is function ssp_evt_user_passkey_request.

Description

Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.

SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements.

Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.

ti.h

Functions

	Name	Description
	btx_ti_a3dp_sink_close_stream	This is function btx_ti_a3dp_sink_close_stream.
	btx_ti_a3dp_sink_codec_config	This is function btx_ti_a3dp_sink_codec_config.
	btx_ti_a3dp_sink_open_stream	This is function btx_ti_a3dp_sink_open_stream.

	btx_ti_a3dp_sink_start_stream	This is function btx_ti_a3dp_sink_start_stream.
	btx_ti_a3dp_sink_stop_stream	This is function btx_ti_a3dp_sink_stop_stream.
	btx_ti_avpr_debug	This is function btx_ti_avpr_debug.
	btx_ti_avpr_enable	This is function btx_ti_avpr_enable.
	btx_ti_drpb_enable_rf_calibration	This is function btx_ti_drpb_enable_rf_calibration.
	btx_ti_drpb_set_power_vector	This is function btx_ti_drpb_set_power_vector.
	btx_ti_drpb_tester_con_tx	This is function btx_ti_drpb_tester_con_tx.
	btx_ti_enable_deep_sleep	This is function btx_ti_enable_deep_sleep.
	btx_ti_enable_fast_clock_crystal	This is function btx_ti_enable_fast_clock_crystal.
	btx_ti_enable_low_power_scan	This is function btx_ti_enable_low_power_scan.
	btx_ti_enable_low_power_scan_default	This is function btx_ti_enable_low_power_scan_default.
	btx_ti_exec_script	This is function btx_ti_exec_script.
	btx_ti_exec_script_oem	This is function btx_ti_exec_script_oem.
	btx_ti_get_script__BL6450_2_0_BT_Service_Pack_2_36	CC2560 Scripts (Service pack 2.36)
	btx_ti_get_script__BL6450_2_0_BT_Service_Pack_2_44	CC2560 Scripts (Service pack 2.44)
	btx_ti_get_script__BL6450L_BT_Service_Pack_2_10	CC2564 Scripts (Service pack 2.10)
	btx_ti_get_script__BL6450L_BT_Service_Pack_2_10_BLE_AddOn	This is function btx_ti_get_script__BL6450L_BT_Service_Pack_2_10_BLE_AddOn.
	btx_ti_get_script__BL6450L_BT_Service_Pack_2_12	CC2564 Scripts (Service pack 2.12)
	btx_ti_get_script__BL6450L_BT_Service_Pack_2_12_BLE_AddOn	This is function btx_ti_get_script__BL6450L_BT_Service_Pack_2_12_BLE_AddOn.
	btx_ti_get_script__BL6450L_BT_Service_Pack_2_14	CC2564 Scripts (Service pack 2.14)
	btx_ti_get_script__BL6450L_BT_Service_Pack_2_14_BLE_AddOn	This is function btx_ti_get_script__BL6450L_BT_Service_Pack_2_14_BLE_AddOn.
	btx_ti_get_script__BL6450L_BT_Service_Pack_2_7	CC2564 Scripts (Service pack 2.7)
	btx_ti_get_script__BL6450L_BT_Service_Pack_2_8_AVPR_AddOn	This is function btx_ti_get_script__BL6450L_BT_Service_Pack_2_8_AVPR_AddOn.
	btx_ti_get_script__BL6450L_BT_Service_Pack_2_8_BLE_AddOn	This is function btx_ti_get_script__BL6450L_BT_Service_Pack_2_8_BLE_AddOn.
	btx_ti_get_script__BL6450L_BT_Service_Pack_2_8_Short	CC2564 Scripts (Service pack 2.8)
	btx_ti_get_script__BL6450x_BT_Service_Pack_2_7_AVPR_AddOn	This is function btx_ti_get_script__BL6450x_BT_Service_Pack_2_7_AVPR_AddOn.
	btx_ti_get_script__BL6450x_BT_Service_Pack_2_7_BLE_AddOn	This is function btx_ti_get_script__BL6450x_BT_Service_Pack_2_7_BLE_AddOn.
	btx_ti_get_script__CC2564B_BT_Service_Pack_0_1	CC2564B Scripts (Service pack 0.1)
	btx_ti_get_script__CC2564B_BT_Service_Pack_0_1_BLE_AddOn	This is function btx_ti_get_script__CC2564B_BT_Service_Pack_0_1_BLE_AddOn.
	btx_ti_get_script__CC2564B_BT_Service_Pack_0_2	CC2564B Scripts (Service pack 0.2)
	btx_ti_get_script__CC2564B_BT_Service_Pack_0_2_BLE_AddOn	This is function btx_ti_get_script__CC2564B_BT_Service_Pack_0_2_BLE_AddOn.
	btx_ti_get_script__CC2564B_BT_Service_Pack_1_0	CC2564B Scripts (Service pack 1.0)
	btx_ti_get_script__CC2564B_BT_Service_Pack_1_0_BLE_AddOn	This is function btx_ti_get_script__CC2564B_BT_Service_Pack_1_0_BLE_AddOn.
	btx_ti_get_script__CC2564B_BT_Service_Pack_1_1	CC2564B Scripts (Service pack 1.1)
	btx_ti_get_script__CC2564B_BT_Service_Pack_1_1_BLE_AddOn	This is function btx_ti_get_script__CC2564B_BT_Service_Pack_1_1_BLE_AddOn.
	btx_ti_get_script__CC2564B_BT_Service_Pack_1_2	CC2564B Scripts (Service pack 1.2)
	btx_ti_get_script__CC2564B_BT_Service_Pack_1_2_AVPR_AddOn	This is function btx_ti_get_script__CC2564B_BT_Service_Pack_1_2_AVPR_AddOn.

	btx_ti_get_script__CC2564B_BT_Service_Pack_1_2_BLE_AddOn	This is function btx_ti_get_script__CC2564B_BT_Service_Pack_1_2_BLE_AddOn.
	btx_ti_get_script__WL127xL_BT_Service_Pack_2_3	CC2564 Scripts (Service pack 2.3)
	btx_ti_get_script__WL127xL_BT_Service_Pack_2_3_AVPR_AddOn	This is function btx_ti_get_script__WL127xL_BT_Service_Pack_2_3_AVPR_AddOn.
	btx_ti_get_script__WL127xL_BT_Service_Pack_2_3_BLE_AddOn	This is function btx_ti_get_script__WL127xL_BT_Service_Pack_2_3_BLE_AddOn.
	btx_ti_get_script__WL127xL_BT_Service_Pack_2_3_DC2DC_AddOn	This is function btx_ti_get_script__WL127xL_BT_Service_Pack_2_3_DC2DC_AddOn.
	btx_ti_get_script__WL127xL_BT_Service_Pack_2_4	CC2564 Scripts (Service pack 2.4)
	btx_ti_get_script__WL127xL_BT_Service_Pack_2_4_AVPR_AddOn	This is function btx_ti_get_script__WL127xL_BT_Service_Pack_2_4_AVPR_AddOn.
	btx_ti_get_script__WL127xL_BT_Service_Pack_2_4_BLE_AddOn	This is function btx_ti_get_script__WL127xL_BT_Service_Pack_2_4_BLE_AddOn.
	btx_ti_get_script__WL127xL_BT_Service_Pack_2_4_DC2DC_AddOn	This is function btx_ti_get_script__WL127xL_BT_Service_Pack_2_4_DC2DC_AddOn.
	btx_ti_get_script__XWL1271L1_BT_ServicePack_1_3	CC2564 Scripts (Service pack 1.3)
	btx_ti_get_script__XWL1271L1_BT_ServicePack_1_3_BLE_Init	This is function btx_ti_get_script__XWL1271L1_BT_ServicePack_1_3_BLE_Init.
	btx_ti_init_ble_controller	This is function btx_ti_init_ble_controller.
	btx_ti_init_controller	This is function btx_ti_init_controller.
	btx_ti_le_enable	This is function btx_ti_le_enable.
	btx_ti_set_afh_mode	This is function btx_ti_set_afh_mode.
	btx_ti_set_uart_baud_rate	This is function btx_ti_set_uart_baud_rate.
	btx_ti_write_bdaddr	This is function btx_ti_write_bdaddr.
	btx_ti_write_codec_config	This is function btx_ti_write_codec_config.
	btx_ti_write_hardware_register	This is function btx_ti_write_hardware_register.

Macros

	Name	Description
	BTX_TI_A3DP_ROLE_SINK	This is macro BTX_TI_A3DP_ROLE_SINK.
	BTX_TI_A3DP_ROLE_SOURCE	This is macro BTX_TI_A3DP_ROLE_SOURCE.
	BTX_TI_AVPR_DISABLE	This is macro BTX_TI_AVPR_DISABLE.
	BTX_TI_AVPR_DO_NOT_LOAD_A3DP_CODE	This is macro BTX_TI_AVPR_DO_NOT_LOAD_A3DP_CODE.
	BTX_TI_AVPR_ENABLE	This is macro BTX_TI_AVPR_ENABLE.
	BTX_TI_AVPR_LOAD_A3DP_CODE	This is macro BTX_TI_AVPR_LOAD_A3DP_CODE.
	BTX_TI_EXEC_SCRIPT_OEM_RX_BUFFER_SIZE	This is macro BTX_TI_EXEC_SCRIPT_OEM_RX_BUFFER_SIZE.
	btx_ti_get_script__CC2560_ServicePack	Alias for the latest script
	btx_ti_get_script__CC2564_BLE_Init	This is macro btx_ti_get_script__CC2564_BLE_Init.
	btx_ti_get_script__CC2564_ServicePack	Aliases for latest scripts
	btx_ti_get_script__CC2564B_AVPR_Init	This is macro btx_ti_get_script__CC2564B_AVPR_Init.
	btx_ti_get_script__CC2564B_BLE_Init	This is macro btx_ti_get_script__CC2564B_BLE_Init.
	btx_ti_get_script__CC2564B_ServicePack	Aliases for latest scripts
	BTX_TI_MODULATION_SCHEME_8_DPSK	This is macro BTX_TI_MODULATION_SCHEME_8_DPSK.
	BTX_TI_MODULATION_SCHEME_CW	This is macro BTX_TI_MODULATION_SCHEME_CW.
	BTX_TI_MODULATION_SCHEME_GFSK	This is macro BTX_TI_MODULATION_SCHEME_GFSK.
	BTX_TI_MODULATION_SCHEME_P4_DPSK	This is macro BTX_TI_MODULATION_SCHEME_P4_DPSK.
	BTX_TI_MODULATION_TYPE_EDR2	This is macro BTX_TI_MODULATION_TYPE_EDR2.
	BTX_TI_MODULATION_TYPE_EDR3	This is macro BTX_TI_MODULATION_TYPE_EDR3.
	BTX_TI_MODULATION_TYPE_GFSK	This is macro BTX_TI_MODULATION_TYPE_GFSK.
	BTX_TI_SBC_ALLOCATION_METHOD_LOUDNESS	This is macro BTX_TI_SBC_ALLOCATION_METHOD_LOUDNESS.

	BTX_TI_SBC_ALLOCATION_METHOD_SNR	This is macro BTX_TI_SBC_ALLOCATION_METHOD_SNR.
	BTX_TI_SBC_BLOCKS_12	This is macro BTX_TI_SBC_BLOCKS_12.
	BTX_TI_SBC_BLOCKS_16	This is macro BTX_TI_SBC_BLOCKS_16.
	BTX_TI_SBC_BLOCKS_4	This is macro BTX_TI_SBC_BLOCKS_4.
	BTX_TI_SBC_BLOCKS_8	This is macro BTX_TI_SBC_BLOCKS_8.
	BTX_TI_SBC_CHANNEL_MODE_DUAL_CHANNEL	This is macro BTX_TI_SBC_CHANNEL_MODE_DUAL_CHANNEL.
	BTX_TI_SBC_CHANNEL_MODE_JOINT_STEREO	This is macro BTX_TI_SBC_CHANNEL_MODE_JOINT_STEREO.
	BTX_TI_SBC_CHANNEL_MODE_MONO	This is macro BTX_TI_SBC_CHANNEL_MODE_MONO.
	BTX_TI_SBC_CHANNEL_MODE_STEREO	This is macro BTX_TI_SBC_CHANNEL_MODE_STEREO.
	BTX_TI_SBC_SAMPLE_FREQUENCY_16000	This is macro BTX_TI_SBC_SAMPLE_FREQUENCY_16000.
	BTX_TI_SBC_SAMPLE_FREQUENCY_32000	This is macro BTX_TI_SBC_SAMPLE_FREQUENCY_32000.
	BTX_TI_SBC_SAMPLE_FREQUENCY_44100	This is macro BTX_TI_SBC_SAMPLE_FREQUENCY_44100.
	BTX_TI_SBC_SAMPLE_FREQUENCY_48000	This is macro BTX_TI_SBC_SAMPLE_FREQUENCY_48000.
	BTX_TI_TEST_PATTERN_ALL_0	This is macro BTX_TI_TEST_PATTERN_ALL_0.
	BTX_TI_TEST_PATTERN_ALL_1	This is macro BTX_TI_TEST_PATTERN_ALL_1.
	BTX_TI_TEST_PATTERN_F0F0	This is macro BTX_TI_TEST_PATTERN_F0F0.
	BTX_TI_TEST_PATTERN_FF00	This is macro BTX_TI_TEST_PATTERN_FF00.
	BTX_TI_TEST_PATTERN_PN15	This is macro BTX_TI_TEST_PATTERN_PN15.
	BTX_TI_TEST_PATTERN_PN9	This is macro BTX_TI_TEST_PATTERN_PN9.
	BTX_TI_TEST_PATTERN_USER	This is macro BTX_TI_TEST_PATTERN_USER.
	BTX_TI_TEST_PATTERN_Z0Z0	This is macro BTX_TI_TEST_PATTERN_Z0Z0.

Structures

	Name	Description
	_btx_ti_codec_config_s	This is type btx_ti_codec_config_t.
	_btx_ti_exec_script_buffer_t	This is record _btx_ti_exec_script_buffer_t.
	_btx_ti_exec_script_oem_buffer_t	This is record _btx_ti_exec_script_oem_buffer_t.
	_btx_ti_script_t	This is type btx_ti_script_t.
	btx_ti_codec_config_t	This is type btx_ti_codec_config_t.
	btx_ti_script_t	This is type btx_ti_script_t.

Types

	Name	Description
	btx_ti_completion_callback_fp	This is type btx_ti_completion_callback_fp.
	btx_ti_exec_script_buffer_t	This is type btx_ti_exec_script_buffer_t.
	btx_ti_exec_script_oem_buffer_t	This is type btx_ti_exec_script_oem_buffer_t.
	btx_ti_exec_script_oem_callback_fp	This is type btx_ti_exec_script_oem_callback_fp.

Description

Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.

ti_private.h

Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.

types.h

Macros

	Name	Description
	BYTE_SIZE	This is macro BYTE_SIZE.

Types

	Name	Description
	bt_byte	This is type bt_byte.

	bt_int	This is type bt_int.
	bt_long	This is type bt_long.
	bt_uint	This is type bt_uint.
	bt_ulong	This is type bt_ulong.

Description

Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC.

Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.

vcard_parser.h

Functions

	Name	Description
	bt_vcard_parse_fragment	This is function bt_vcard_parse_fragment.
	bt_vcard_parser_reset	This is function bt_vcard_parser_reset.

Macros

	Name	Description
	VCARD_CALL_TYPE_DIALED	This is macro VCARD_CALL_TYPE_DIALED.
	VCARD_CALL_TYPE_MISSED	This is macro VCARD_CALL_TYPE_MISSED.
	VCARD_CALL_TYPE_RECEIVED	This is macro VCARD_CALL_TYPE_RECEIVED.
	VCARD_EVT_PROPERTY_PARAM	This is macro VCARD_EVT_PROPERTY_PARAM.
	VCARD_EVT_PROPERTY_STARTED	This is macro VCARD_EVT_PROPERTY_STARTED.
	VCARD_EVT_PROPERTY_VALUE	This is macro VCARD_EVT_PROPERTY_VALUE.
	VCARD_EVT_VCARD_ENDED	This is macro VCARD_EVT_VCARD_ENDED.
	VCARD_EVT_VCARD_STARTED	This is macro VCARD_EVT_VCARD_STARTED.
	VCARD_PARAM_CALL_TYPE	This is macro VCARD_PARAM_CALL_TYPE.
	VCARD_PARAM_CELL	This is macro VCARD_PARAM_CELL.
	VCARD_PARAM_DIALED	This is macro VCARD_PARAM_DIALED.
	VCARD_PARAM_ENCODING	This is macro VCARD_PARAM_ENCODING.
	VCARD_PARAM_HOME	This is macro VCARD_PARAM_HOME.
	VCARD_PARAM_LANGUAGE	This is macro VCARD_PARAM_LANGUAGE.
	VCARD_PARAM_MISSED	This is macro VCARD_PARAM_MISSED.
	VCARD_PARAM_RECEIVED	This is macro VCARD_PARAM_RECEIVED.
	VCARD_PARAM_TYPE	This is macro VCARD_PARAM_TYPE.
	VCARD_PARAM_VALUE	This is macro VCARD_PARAM_VALUE.
	VCARD_PARAM_WORK	This is macro VCARD_PARAM_WORK.
	VCARD_PARSER_STATE_READ_PARAM_NAME	This is macro VCARD_PARSER_STATE_READ_PARAM_NAME.
	VCARD_PARSER_STATE_READ_PARAM_VALUE	This is macro VCARD_PARSER_STATE_READ_PARAM_VALUE.
	VCARD_PARSER_STATE_READ_TYPE_NAME	This is macro VCARD_PARSER_STATE_READ_TYPE_NAME.
	VCARD_PARSER_STATE_READ_TYPE_VALUE	This is macro VCARD_PARSER_STATE_READ_TYPE_VALUE.
	VCARD_PARSER_STATE_SKIP_PARAM	This is macro VCARD_PARSER_STATE_SKIP_PARAM.
	VCARD_PARSER_STATE_SKIP_TYPE	This is macro VCARD_PARSER_STATE_SKIP_TYPE.
	VCARD_PROPERTY_ADR	Delivery Address
	VCARD_PROPERTY_AGENT	vCard of Person Representing
	VCARD_PROPERTY_BDAY	Birthday
	VCARD_PROPERTY_BEGIN	This is macro VCARD_PROPERTY_BEGIN.
	VCARD_PROPERTY_CATEGORIES	Categories
	VCARD_PROPERTY_CLASS	Class information
	VCARD_PROPERTY_EMAIL	Electronic Mail Address
	VCARD_PROPERTY_END	This is macro VCARD_PROPERTY_END.
	VCARD_PROPERTY_FN	Formatted Name
	VCARD_PROPERTY_GEO	Geographic Position
	VCARD_PROPERTY_KEY	Public Encryption Key

	VCARD_PROPERTY_LABEL	Delivery
	VCARD_PROPERTY_LOGO	Organization Logo
	VCARD_PROPERTY_MAILER	Electronic Mail
	VCARD_PROPERTY_N	Structured Presentation of Name
	VCARD_PROPERTY_NICKNAME	Nickname
	VCARD_PROPERTY_NOTE	Comments
	VCARD_PROPERTY_ORG	Name of Organization
	VCARD_PROPERTY_PHOTO	Associated Image or Photo
	VCARD_PROPERTY_PROID	Product ID
	VCARD_PROPERTY_REV	Revision
	VCARD_PROPERTY_ROLE	Role within the Organization
	VCARD_PROPERTY_SORT_STRING	String used for sorting operations
	VCARD_PROPERTY_SOUND	Pronunciation of Name
	VCARD_PROPERTY_TEL	Telephone Number
	VCARD_PROPERTY_TITLE	Job
	VCARD_PROPERTY_TZ	Time Zone
	VCARD_PROPERTY_UID	Unique ID
	VCARD_PROPERTY_URL	Uniform Resource Locator
	VCARD_PROPERTY_VERSION	vCard Version
	VCARD_PROPERTY_X_IRMC_CALL_DATETIME	Time stamp

Structures

	Name	Description
	_bt_vcard_evt_prop_param_t	This is type bt_vcard_evt_prop_param_t.
	_bt_vcard_evt_prop_t	This is type bt_vcard_evt_prop_t.
	_bt_vcard_parser_t	This is type bt_vcard_parser_t.
	bt_vcard_evt_prop_param_t	This is type bt_vcard_evt_prop_param_t.
	bt_vcard_evt_prop_t	This is type bt_vcard_evt_prop_t.
	bt_vcard_parser_t	This is type bt_vcard_parser_t.

Types

	Name	Description
	bt_vcard_parser_callback_fp	This is type bt_vcard_parser_callback_fp.

Description

Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.

SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements.

Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.

xml_parser.h

Functions

	Name	Description
	bt_xml_parse_fragment	This is function bt_xml_parse_fragment.
	bt_xml_parser_reset	This is function bt_xml_parser_reset.

Macros

	Name	Description
	XML_EVT_ATTR_NAME	This is macro XML_EVT_ATTR_NAME.
	XML_EVT_ATTR_VALUE	This is macro XML_EVT_ATTR_VALUE.
	XML_EVT_TAG_ENDED	This is macro XML_EVT_TAG_ENDED.
	XML_EVT_TAG_STARTED	This is macro XML_EVT_TAG_STARTED.
	XML_PARSER_STATE_READ_ATTR_NAME	This is macro XML_PARSER_STATE_READ_ATTR_NAME.

	XML_PARSER_STATE_READ_ATTR_VALUE	This is macro XML_PARSER_STATE_READ_ATTR_VALUE.
	XML_PARSER_STATE_READ_TAG_NAME	This is macro XML_PARSER_STATE_READ_TAG_NAME.
	XML_PARSER_STATE_READ_TAG_VALUE	This is macro XML_PARSER_STATE_READ_TAG_VALUE.
	XML_PARSER_STATE_SKIP_ATTR	This is macro XML_PARSER_STATE_SKIP_ATTR.
	XML_PARSER_STATE_SKIP_COMMENTS	This is macro XML_PARSER_STATE_SKIP_COMMENTS.
	XML_PARSER_STATE_SKIP_PROC_INST	This is macro XML_PARSER_STATE_SKIP_PROC_INST.
	XML_PARSER_STATE_SKIP_TAG	This is macro XML_PARSER_STATE_SKIP_TAG.
	XML_PARSER_STATE_WAIT_ATTR_NAME	This is macro XML_PARSER_STATE_WAIT_ATTR_NAME.
	XML_PARSER_STATE_WAIT_ATTR_VALUE	This is macro XML_PARSER_STATE_WAIT_ATTR_VALUE.
	XML_PARSER_STATE_WAIT_TAG_CLOSE	This is macro XML_PARSER_STATE_WAIT_TAG_CLOSE.
	XML_PARSER_STATE_WAIT_TAG_NAME	This is macro XML_PARSER_STATE_WAIT_TAG_NAME.

Structures

	Name	Description
	_bt_xml_evt_attribute_t	This is type bt_xml_evt_attribute_t.
	_bt_xml_evt_attribute_value_t	This is type bt_xml_evt_attribute_value_t.
	_bt_xml_evt_tag_started_t	This is type bt_xml_evt_tag_started_t.
	_bt_xml_parser_t	This is type bt_xml_parser_t.
	bt_xml_evt_attribute_t	This is type bt_xml_evt_attribute_t.
	bt_xml_evt_attribute_value_t	This is type bt_xml_evt_attribute_value_t.
	bt_xml_evt_tag_started_t	This is type bt_xml_evt_tag_started_t.
	bt_xml_parser_t	This is type bt_xml_parser_t.

Types

	Name	Description
	bt_xml_parser_callback_pf	This is type bt_xml_parser_callback_pf.

Description

Contains proprietary and confidential information of SEARAN LLC. May not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2011-2016 SEARAN LLC. All Rights Reserved.

SEARAN LLC is the exclusive licensee and developer of dotstack with all its modifications and enhancements.

Contains proprietary and confidential information of CandleDragon and may not be used or disclosed to any other party except in accordance with a license from SEARAN LLC. Copyright (c) 2009, 2010, 2011 CandleDragon. All Rights Reserved.

Index

_

__A2DP_CODEC_AAC_H macro 234
__A2DP_CODEC_MPEG_H macro 234
__A2DP_CODEC_SBC_H macro 234
__A2DP_H macro 235
__A2DP_PRIVATE_H macro 235
__AVCTP_CONFIG_H macro 21
__AVCTP_H macro 287
__AVCTP_PACKET_H macro 287
__AVCTP_PRIVATE_H macro 287
__AVDTP_CONFIG_H macro 340
__AVDTP_CONTROL_H macro 340
__AVDTP_H macro 341
__AVDTP_PRIVATE_H macro 341
__AVRCP_COMMAND_H macro 416
__AVRCP_CONFIG_EVENT_HANDLERS_H macro 512
__AVRCP_CONFIG_H macro 28
__AVRCP_H macro 416
__AVRCP_PRIVATE_H macro 416
__BT_APP_CONFIG_H macro 20
__HCI_CONFIG_EVENT_HANDLERS_H macro 38
__HCI_CONFIG_H macro 38
__L2CAP_CONFIG_H macro 46
__L2CAP_CONFIG_HANDLERS_H macro 47
__RFCOMM_CONFIG_H macro 50
__SBC_H macro 834
__SDP_CONFIG_H macro 52
__SPP_CONFIG_H macro 54
_bt_a2dp_aac_config_t structure 230
_bt_a2dp_avdtp_mgr_callback function 224
_bt_a2dp_event_u union 228
_bt_a2dp_evt_open_and_start_stream_completed_s structure 231
_bt_a2dp_mgr_t structure 230
_bt_a2dp_mpeg_config_t structure 233
_bt_a2dp_sbc_config_t structure 233
_bt_a2dp_sbc_packet_info_t structure 234
_bt_at_parser_t structure 996
_bt_av_add_to_now_playing_s structure 397
_bt_av_battery_status_of_ct_s structure 397
_bt_av_capability_company_id_s structure 398
_bt_av_capability_event_id_s structure 398
_bt_av_command_t structure 395
_bt_av_displayable_character_set_s structure 399
_bt_av_element_attribute_s structure 399
_bt_av_element_attributes_s structure 400
_bt_av_element_id_s structure 400
_bt_av_get_element_attributes_s structure 400
_bt_av_notification_addressed_player_changed_s structure 401
_bt_av_notification_app_setting_changed_s structure 401
_bt_av_notification_battery_status_s structure 401
_bt_av_notification_playback_pos_changed_s structure 402
_bt_av_notification_playback_status_changed_s structure 402
_bt_av_notification_s structure 403
_bt_av_notification_system_status_changed_s structure 402
_bt_av_notification_track_changed_s structure 404
_bt_av_notification_uids_changed_s structure 404
_bt_av_notification_volume_changed_s structure 404
_bt_av_play_item_s structure 405
_bt_av_play_status_s structure 405
_bt_av_player_setting_current_values_s structure 405
_bt_av_player_setting_values_s structure 406
_bt_av_player_setting_values_text_s structure 406
_bt_av_player_settings_s structure 407
_bt_av_player_settings_text_s structure 407
_bt_av_player_text_s structure 408
_bt_av_register_notification_t structure 408
_bt_av_response_t structure 408
_bt_av_set_absolute_volume_s structure 409
_bt_av_set_addressed_player_s structure 409
_bt_avctp_allocate_channel function 272
_bt_avctp_allocate_message function 272
_bt_avctp_allocate_transport function 272
_bt_avctp_channel_t structure 280
_bt_avctp_event_u union 281
_bt_avctp_evt_channel_connected_t structure 282
_bt_avctp_evt_channel_disconnected_t structure 282
_bt_avctp_evt_command_cancelled_t structure 283
_bt_avctp_evt_command_received_t structure 283
_bt_avctp_evt_command_sent_t structure 283
_bt_avctp_evt_connection_failed_t structure 284
_bt_avctp_evt_response_cancelled_t structure 284
_bt_avctp_evt_response_received_t structure 284
_bt_avctp_evt_response_sent_t structure 285
_bt_avctp_find_channel function 272
_bt_avctp_find_transport function 273
_bt_avctp_free_channel function 273
_bt_avctp_free_message function 273
_bt_avctp_free_transport function 273
_bt_avctp_init_message_buffers function 273
_bt_avctp_init_signal function 274
_bt_avctp_l2cap_read_data_callback function 274
_bt_avctp_message_t structure 280
_bt_avctp_mgr_t structure 280
_bt_avctp_packet_assembler function 274
_bt_avctp_packet_t structure 286
_bt_avctp_send_ipid function 274
_bt_avctp_set_signal function 274
_bt_avctp_transport_t structure 281
_bt_avdtp_add_param_byte function 294
_bt_avdtp_add_param_uint function 294
_bt_avdtp_add_param_uintn function 313
_bt_avdtp_allocate_buffers function 294
_bt_avdtp_allocate_cmd function 295
_bt_avdtp_allocate_sep function 295
_bt_avdtp_allocate_sep_config function 295
_bt_avdtp_allocate_stream function 295
_bt_avdtp_allocate_transport_channel function 295
_bt_avdtp_allocate_transport_session_id function 296
_bt_avdtp_begin_tc_channel_operation function 296
_bt_avdtp_codec_op_decode_t structure 318
_bt_avdtp_codec_op_encode_t structure 318
_bt_avdtp_codec_op_param_u union 316
_bt_avdtp_codec_op_parse_config_t structure 318

_bt_avdtp_codec_op_parse_packet_t structure 319
_bt_avdtp_codec_op_serialize_config_t structure 319
_bt_avdtp_codec_t structure 320
_bt_avdtp_commit_tc_channel_operation function 296
_bt_avdtp_control_channel_accept_handler function 296
_bt_avdtp_control_channel_cmd_handler function 296
_bt_avdtp_control_channel_event_handler function 297
_bt_avdtp_control_channel_reject_handler function 297
_bt_avdtp_control_channel_t structure 320
_bt_avdtp_control_cmd_t structure 320
_bt_avdtp_ctrl_evt_data_received_t structure 321
_bt_avdtp_event_u union 321
_bt_avdtp_evt_abort_stream_requested_t structure 323
_bt_avdtp_evt_close_stream_completed_t structure 323
_bt_avdtp_evt_close_stream_requested_t structure 323
_bt_avdtp_evt_ctrl_channel_connected_s structure 324
_bt_avdtp_evt_ctrl_channel_disconnected_s structure 324
_bt_avdtp_evt_ctrl_connection_failed_s structure 369
_bt_avdtp_evt_delay_report_completed_t structure 368
_bt_avdtp_evt_discover_completed_t structure 325
_bt_avdtp_evt_get_sep_capabilities_completed_t structure 325
_bt_avdtp_evt_get_stream_configuration_completed_t structure 325
_bt_avdtp_evt_media_packet_received_t structure 326
_bt_avdtp_evt_media_packet_send_failed_t structure 326
_bt_avdtp_evt_media_packet_sent_t structure 326
_bt_avdtp_evt_open_stream_completed_t structure 327
_bt_avdtp_evt_open_stream_requested_t structure 327
_bt_avdtp_evt_reconfigure_stream_requested_t structure 328
_bt_avdtp_evt_sep_capabilities_received_t structure 328
_bt_avdtp_evt_sep_info_received_t structure 328
_bt_avdtp_evt_set_stream_configuration_completed_t structure 329
_bt_avdtp_evt_set_stream_configuration_requested_t structure 329
_bt_avdtp_evt_set_stream_configuration_t structure 370
_bt_avdtp_evt_start_stream_completed_t structure 330
_bt_avdtp_evt_start_stream_requested_t structure 330
_bt_avdtp_evt_stream_aborted_t structure 331
_bt_avdtp_evt_stream_closed_t structure 331
_bt_avdtp_evt_stream_configured_t structure 331
_bt_avdtp_evt_stream_opened_t structure 332
_bt_avdtp_evt_stream_reconfigure_completed_t structure 332
_bt_avdtp_evt_stream_reconfigured_t structure 332
_bt_avdtp_evt_stream_security_control_completed_t structure 333
_bt_avdtp_evt_stream_started_t structure 333
_bt_avdtp_evt_stream_suspended_t structure 333
_bt_avdtp_evt_suspend_stream_completed_t structure 334
_bt_avdtp_evt_suspend_stream_requested_t structure 334
_bt_avdtp_execute_tc_channel_operation function 297
_bt_avdtp_find_listening_stream function 297
_bt_avdtp_find_stream function 297
_bt_avdtp_find_stream_by_remote_sep_id function 298
_bt_avdtp_find_stream_by_sep_id function 298
_bt_avdtp_free_cmd function 298
_bt_avdtp_free_sep_config function 298
_bt_avdtp_free_stream function 298
_bt_avdtp_free_transport_channel function 299
_bt_avdtp_get_control_channel function 299
_bt_avdtp_init_cmd_buffers function 299
_bt_avdtp_init_sep_config_buffers function 299
_bt_avdtp_init_signal function 299
_bt_avdtp_is_sep_inuse function 300
_bt_avdtp_mgr_t structure 336
_bt_avdtp_open_control_channel_ex function 300
_bt_avdtp_read_caps function 314
_bt_avdtp_register_transport_channel_for_operation function 300
_bt_avdtp_send_command function 300
_bt_avdtp_send_media_packet function 301
_bt_avdtp_sep_capabilities_t structure 337
_bt_avdtp_sep_t structure 316
_bt_avdtp_set_signal function 301
_bt_avdtp_stream_t structure 338
_bt_avdtp_transport_channel_t structure 339
_bt_avdtp_transport_l2cap_read_data_callback function 301
_bt_avdtp_transport_l2cap_state_changed_callback function 301
_bt_avdtp_transport_session_t structure 340
_bt_avdtp_write_caps function 301
_bt_avrcp_allocate_browsing_cmd function 370
_bt_avrcp_allocate_browsing_response function 394
_bt_avrcp_allocate_bt_specific_cmd function 370
_bt_avrcp_allocate_bt_specific_response function 370
_bt_avrcp_allocate_channel function 371
_bt_avrcp_allocate_cmd function 371
_bt_avrcp_allocate_response function 371
_bt_avrcp_allocate_simple_panel_cmd function 371
_bt_avrcp_allocate_simple_panel_response function 371
_bt_avrcp_channel_t structure 396
_bt_avrcp_device_s structure 410
_bt_avrcp_event_u union 410
_bt_avrcp_evt_channel_connected_t structure 411
_bt_avrcp_evt_channel_disconnected_t structure 412
_bt_avrcp_evt_connection_failed_t structure 412
_bt_avrcp_evt_panel_command_received_t structure 412
_bt_avrcp_evt_panel_response_received_t structure 413
_bt_avrcp_evt_register_events_completed_t structure 413
_bt_avrcp_evt_search_completed_s structure 414
_bt_avrcp_find_channel function 372
_bt_avrcp_free_channel function 372
_bt_avrcp_free_cmd function 372
_bt_avrcp_general_reject function 394
_bt_avrcp_get_tick_count function 372
_bt_avrcp_handle_add_to_now_playing function 372
_bt_avrcp_handle_command function 373
_bt_avrcp_handle_command_sent function 373
_bt_avrcp_handle_get_capabilities function 373
_bt_avrcp_handle_get_current_player_application_setting_value function 373
_bt_avrcp_handle_get_element_attributes function 373
_bt_avrcp_handle_get_play_status function 374
_bt_avrcp_handle_get_player_application_setting_attribute_text function 374
_bt_avrcp_handle_get_player_application_setting_value_text function 374
_bt_avrcp_handle_inform_battery_status_of_ct function 374
_bt_avrcp_handle_inform_displayable_character_set function 374
_bt_avrcp_handle_list_player_application_setting_attributes function 375
_bt_avrcp_handle_list_player_application_setting_values function 375
_bt_avrcp_handle_play_item function 375
_bt_avrcp_handle_register_notification function 375

_bt_avrcp_handle_request_continuing_response function 375
_bt_avrcp_handle_response function 376
_bt_avrcp_handle_response_sent function 376
_bt_avrcp_handle_set_absolute_volume function 376
_bt_avrcp_handle_set_addressed_player function 376
_bt_avrcp_handle_set_player_application_setting_value function 376
_bt_avrcp_init_cmd_buffers function 275
_bt_avrcp_init_signal function 377
_bt_avrcp_init_timer function 377
_bt_avrcp_mgr_t structure 396
_bt_avrcp_register_next_notification function 377
_bt_avrcp_register_pending_notification function 377
_bt_avrcp_send_notifications function 377
_bt_avrcp_send_rejected_response function 395
_bt_avrcp_set_signal function 378
_bt_avrcp_start_timer function 378
_bt_avrcp_tg_handle_get_capabilities function 378
_bt_avrcp_tg_handle_get_element_attributes function 378
_bt_avrcp_tg_handle_get_play_status function 378
_bt_avrcp_tg_handle_inform_battery_status_of_ct function 379
_bt_avrcp_tg_handle_inform_displayable_character_set function 379
_bt_avrcp_tg_handle_register_notification function 379
_bt_avrcp_tg_handle_set_absolute_volume function 379
_bt_avrcp_write_command_header function 379
_bt_bdaddr_s structure 217
_bt_buffer_header_t structure 997
_bt_buffer_mgr_t structure 997
_bt_cp_header_s structure 368
_bt_device_t structure 514
_bt_hci_command_s structure 678
_bt_hci_conn_state_s structure 693
_bt_hci_ctrl_listener_t structure 694
_bt_hci_ctrl_notify_data_listeners function 556
_bt_hci_ctrl_notify_listeners function 556
_bt_hci_ctrl_state_s structure 679
_bt_hci_data_buffer_s structure 680
_bt_hci_data_s structure 681
_bt_hci_event_e union 682
_bt_hci_event_s structure 683
_bt_hci_evt_authentication_complete_t structure 683
_bt_hci_evt_command_complete_t structure 683
_bt_hci_evt_command_status_t structure 683
_bt_hci_evt_connection_complete_t structure 684
_bt_hci_evt_connection_request_t structure 684
_bt_hci_evt_disconnection_complete_t structure 684
_bt_hci_evt_encryption_change_s structure 684
_bt_hci_evt_mode_change_t structure 685
_bt_hci_evt_role_change_t structure 685
_bt_hci_get_tick_count function 556
_bt_hci_init_signal function 556
_bt_hci_init_timer function 556
_bt_hci_init_transport function 557
_bt_hci_inquiry_response_t structure 686
_bt_hci_le_advertising_report_t structure 694
_bt_hci_le_command_complete_handler function 557
_bt_hci_le_conn_state_t structure 686
_bt_hci_le_connect_parameters_t structure 686
_bt_hci_le_ctrl_state_t structure 687
_bt_hci_le_evt_connection_updated_t structure 687
_bt_hci_le_evt_read_remote_used_features_completed_t structure 688
_bt_hci_le_evt_read_support_params_t structure 688
_bt_hci_link_key_s structure 688
_bt_hci_listener_t structure 694
_bt_hci_notify_listeners function 557
_bt_hci_set_init_flags function 561
_bt_hci_set_signal function 557
_bt_l2cap_cfg_option_t structure 766
_bt_l2cap_channel_ext_t structure 782
_bt_l2cap_channel_t structure 765
_bt_l2cap_clear_channel_cmd_queue function 714
_bt_l2cap_cmd_config_req_t structure 767
_bt_l2cap_cmd_config_res_t structure 767
_bt_l2cap_cmd_conn_param_update_req_t structure 768
_bt_l2cap_cmd_conn_param_update_res_t structure 768
_bt_l2cap_cmd_connection_req_t structure 768
_bt_l2cap_cmd_connection_res_t structure 769
_bt_l2cap_cmd_disconnection_req_t structure 769
_bt_l2cap_cmd_echo_req_t structure 770
_bt_l2cap_cmd_echo_res_t structure 770
_bt_l2cap_cmd_header_t structure 771
_bt_l2cap_cmd_info_req_t structure 772
_bt_l2cap_cmd_info_res_t structure 772
_bt_l2cap_cmd_reject_param_t union 773
_bt_l2cap_cmd_reject_t structure 773
_bt_l2cap_command_t union 774
_bt_l2cap_connect_params_s structure 783
_bt_l2cap_eretr_handle_xmit_event function 714
_bt_l2cap_eretr_pack_config_request function 725
_bt_l2cap_eretr_recv function 725
_bt_l2cap_eretr_retr_frames function 714
_bt_l2cap_eretr_send_data function 725
_bt_l2cap_eretr_send_pending_frames function 725
_bt_l2cap_eretr_send_smart_data function 725
_bt_l2cap_eretr_xmit_event_e enumeration 774
_bt_l2cap_fcs function 715
_bt_l2cap_fixed_channel_s structure 774
_bt_l2cap_frame_desc_s structure 775
_bt_l2cap_get_tick_count function 715
_bt_l2cap_init_signal function 715
_bt_l2cap_init_timer function 715
_bt_l2cap_mgr_s structure 776
_bt_l2cap_notify_and_remove function 715
_bt_l2cap_option_flash_timeout_t structure 776
_bt_l2cap_option_max_mtu_t structure 777
_bt_l2cap_option_qos_t structure 777
_bt_l2cap_option_rfc_t structure 777
_bt_l2cap_option_unknown_t structure 777
_bt_l2cap_packet_t structure 778
_bt_l2cap_process_connect_signal function 728
_bt_l2cap_psm_s structure 778
_bt_l2cap_recv_req_seq_and_fbit function 726
_bt_l2cap_send_ack function 726
_bt_l2cap_send_commands_from_queue function 716
_bt_l2cap_send_i_or_rr_or_rnr function 726
_bt_l2cap_send_rej function 726
_bt_l2cap_send_rnr function 726

_bt_l2cap_send_rr function 727
_bt_l2cap_send_rr_or_rnr function 727
_bt_l2cap_set_signal function 716
_bt_l2cap_start_monitor_timer macro 763
_bt_l2cap_start_monitor_timer_if_not_running macro 764
_bt_l2cap_start_retr_timer macro 764
_bt_l2cap_start_retr_timer_if_not_running macro 764
_bt_l2cap_stop_monitor_timer macro 764
_bt_l2cap_stop_retr_timer macro 764
_bt_l2cap_xmit_event_param_t structure 779
_bt_le_evt_handler function 560
_bt_linkkey_notification_t structure 219
_bt_linkkey_request_t structure 219
_bt_linkkey_t structure 219
_bt_media_packet_t structure 316
_bt_memcpy function 977
_bt_packet_t structure 998
_bt_queue_element_t structure 998
_bt_rfcomm_allocate_channel function 797
_bt_rfcomm_clear_queue function 797
_bt_rfcomm_command_t structure 826
_bt_rfcomm_ctl_msg_t structure 827
_bt_rfcomm_dlc_t structure 828
_bt_rfcomm_find_channel function 798
_bt_rfcomm_get_mgr function 798
_bt_rfcomm_init_signal function 798
_bt_rfcomm_mgr_allocate_session function 798
_bt_rfcomm_mgr_l2cap_listen_callback function 798
_bt_rfcomm_mgr_notify_listeners function 799
_bt_rfcomm_mgr_t structure 828
_bt_rfcomm_server_channel_t structure 829
_bt_rfcomm_session_listener_t structure 829
_bt_rfcomm_session_t structure 830
_bt_rfcomm_set_signal function 799
_bt_sdp_client_init function 843
_bt_sdp_data_element_t structure 877
_bt_sdp_found_attr_list_t structure 878
_bt_sdp_packet_t structure 878
_bt_sdp_sequence_t structure 879
_bt_sdp_serialization_state_t structure 880
_bt_sdp_server_attribute_t structure 880
_bt_sdp_server_data_element_t structure 880
_bt_sdp_server_record_t structure 880
_bt_sdp_service_transaction_t structure 881
_bt_sdp_transaction_t structure 882
_bt_signal_t structure 216
_bt_spp_client_init function 900
_bt_spp_find_port function 891
_bt_spp_handle_rx function 891
_bt_spp_handle_tx function 891
_bt_spp_port_event_e enumeration 894
_bt_spp_port_state_e enumeration 895
_bt_spp_port_t structure 902
_bt_spp_rfcomm_read_data_callback function 891
_bt_spp_rfcomm_send_data_callback function 891
_bt_spp_send_status_e enumeration 896
_bt_ssp_io_capability structure 903
_bt_ssp_keypress_notification structure 903
_bt_ssp_oob_data structure 903
_bt_ssp_simple_pairing_complete structure 904
_bt_ssp_user_confirmation_request structure 904
_bt_ssp_user_passkey_notification structure 904
_bt_ssp_user_passkey_request structure 904
_bt_timer_id_enum enumeration 222
_bt_uuid_s structure 223
_bt_vcard_evt_prop_param_t structure 998
_bt_vcard_evt_prop_t structure 999
_bt_vcard_parser_t structure 999
_bt_xml_evt_attribute_t structure 999
_bt_xml_evt_attribute_value_t structure 1000
_bt_xml_evt_tag_started_t structure 1000
_bt_xml_parser_t structure 1000
_btx_csr_autobaud_buffer_t structure 944
_btx_csr_bccmd_header_s structure 946
_btx_csr_bccmd_listener_t structure 967
_btx_csr_cached_temperature_s structure 946
_btx_csr_create_operator_c_s structure 965
_btx_csr_exec_hq_script_buffer_t structure 967
_btx_csr_exec_script_buffer_t structure 944
_btx_csr_pio_direction_mask_s structure 947
_btx_csr_pio_protection_mask_s structure 947
_btx_csr_pio_s structure 948
_btx_csr_rssi_acl_s structure 948
_btx_csr_script_t structure 948
_btx_csr_set_ps_vars_buffer_t structure 945
_btx_csr_strm_connect_s structure 966
_btx_csr_strm_get_sink_s structure 949
_btx_csr_strm_get_source_s structure 949
_btx_csr_var_u union 949
_btx_ti_codec_config_s structure 966
_btx_ti_exec_script_buffer_t structure 945
_btx_ti_exec_script_oem_buffer_t structure 945
_btx_ti_script_t structure 951
_calc_fcs function 799
_cmd_disconnection_res structure 780
_compact_buffer function 972
_expand_buffer function 972
_hci_allocate_buffers function 557
_hci_allocate_cmd function 558
_hci_free_cmd function 558
_hci_init_cmd_buffers function 558
_hci_init_cmd_queues function 558
_hci_receive_start function 558
_hci_recv_sco_data_packet function 559
_hci_send_commands_from_queue function 559
_hci_send_data function 559
_hci_send_data_fragment function 559
_hci_send_data_from_queue function 559
_hci_transport_t structure 692
_hcitr_tih4_power_event_e enumeration 692
_is_empty_str function 977
_l2cap_allocate_buffers function 716
_l2cap_data_receive_callback function 716
_pack_cmd_reject function 716
_pack_config_request function 717
_pack_config_response function 717

_pack_conn_param_update_request function 717
_pack_conn_param_update_response function 717
_pack_conn_request function 717
_pack_conn_response function 718
_pack_dconn_request function 718
_pack_dconn_response function 718
_pack_echo_request function 718
_pack_echo_response function 718
_pack_info_request function 719
_pack_info_response function 719
_process_config_req function 719
_process_config_res function 719
_process_conn_param_update_req function 719
_process_conn_param_update_res function 720
_process_conn_req function 720
_process_conn_res function 720
_process_dconn_req function 720
_process_dconn_res function 720
_process_echo_req function 721
_process_echo_res function 721
_process_info_req function 721
_process_info_res function 721
_process_reject function 721
_process_unknown_req function 722
_process_unknown_res function 722
_read_bdaddr function 972
_read_cmd_reject function 722
_read_config_request function 722
_read_config_response function 722
_read_conn_param_update_request function 723
_read_conn_param_update_response function 723
_read_conn_request function 723
_read_conn_response function 723
_read_dconn_request function 723
_read_dconn_response function 724
_read_echo_request function 724
_read_echo_response function 724
_read_info_request function 724
_read_info_response function 724
_readb function 973
_readbn macro 995
_readi function 973
_readin function 973
_readl function 973
_readln function 973
_readui function 974
_readui macro 995
_readuin function 974
_readuin macro 995
_readul function 974
_readul macro 996
_readuln function 974
_readuln macro 996
_rfcomm_alloc_cmd_buffer function 790
_rfcomm_allocate_buffers function 790
_rfcomm_allocate_mx_cmd function 790
_rfcomm_free_cmd_buffer function 790
_rfcomm_get_tick_count function 791
_rfcomm_init_cmd_buffers function 791
_rfcomm_init_mgr function 791
_rfcomm_init_sessions function 791
_rfcomm_init_timer function 791
_rfcomm_l2cap_read_data_callback function 792
_rfcomm_l2cap_state_changed_callback function 799
_rfcomm_mx_process_fc function 792
_rfcomm_mx_process_pn function 792
_rfcomm_mx_process_rpn function 792
_rfcomm_process_cmd_frame_disc function 792
_rfcomm_process_cmd_frame_dm function 793
_rfcomm_process_cmd_frame_sabm function 793
_rfcomm_process_cmd_frame_ua function 793
_rfcomm_process_cmd_frame_uih function 793
_rfcomm_process_mx_fc_response function 793
_rfcomm_process_mx_msc_response function 794
_rfcomm_process_mx_pn_response function 794
_rfcomm_process_mx_rls_response function 794
_rfcomm_process_res_frame_disc function 794
_rfcomm_process_res_frame_dm function 794
_rfcomm_process_res_frame_sabm function 795
_rfcomm_process_res_frame_ua function 795
_rfcomm_process_res_frame_uih function 795
_rfcomm_send_command function 795
_rfcomm_send_dm_response function 795
_rfcomm_send_mx_fc_cmd function 796
_rfcomm_send_mx_msc_response function 796
_rfcomm_send_mx_nsc_response function 796
_rfcomm_send_mx_rls_cmd function 796
_rfcomm_send_mx_rls_response function 796
_rfcomm_send_mx_test_response function 797
_rfcomm_send_sabm_cmd function 797
_rfcomm_send_ua_response function 797
_sdp_alloc_svc_tran_buffer function 840
_sdp_alloc_tran_buffer function 840
_sdp_find_svc_transaction function 841
_sdp_find_transaction function 841
_sdp_free_svc_tran_buffer function 841
_sdp_free_tran_buffer function 841
_sdp_get_de_data_len function 841
_sdp_get_de_hdr_len function 842
_sdp_init_tran_buffers function 842
_sdp_read_de_header function 842
_sdp_write_data_element function 842
_SSP_AUTHENTICATION_REQUIREMENTS enumeration 904
_SSP_EVENT enumeration 906
_SSP_IO_CAPABILITY enumeration 905
_SSP_KEYPRESS_NOTIFICATION_TYPE enumeration 905
_SSP_MODE enumeration 905
_SSP_OOB_DATA_PRESENT enumeration 906
_str2ulong function 977
_str2ulong_dec function 974
_to_lower_case function 975
_ulong2str function 975
_ulong2str_dec function 975
_W macro 944
_write_bdaddr function 975
_writeb function 975

_writebn macro 996
_writei function 976
_writein function 976
_writel function 976
_writeln function 976
_writes function 976
_writesx function 977
_zero_memory function 977

A

a2dp.h 1012
a2dp_codec_aac.h 1022
a2dp_codec_mpeg.h 1023
a2dp_codec_sbc.h 1024
A2DP_EVT_ABORT_STREAM_COMPLETED macro 235
A2DP_EVT_ABORT_STREAM_REQUESTED macro 235
A2DP_EVT_CLOSE_STREAM_COMPLETED macro 235
A2DP_EVT_CLOSE_STREAM_REQUESTED macro 236
A2DP_EVT_CTRL_CHANNEL_CONNECTED macro 236
A2DP_EVT_CTRL_CHANNEL_DISCONNECTED macro 236
A2DP_EVT_CTRL_CONNECTION_FAILED macro 236
A2DP_EVT_DISCOVER_SEP_COMPLETED macro 236
A2DP_EVT_GET_SEP_CAPABILITIES_COMPLETED macro 237
A2DP_EVT_GET_STREAM_CONFIGURATION_COMPLETED macro 237
A2DP_EVT_MEDIA_PACKET_RECEIVED macro 237
A2DP_EVT_MEDIA_PACKET_SEND_FAILED macro 237
A2DP_EVT_MEDIA_PACKET_SENT macro 237
A2DP_EVT_NOTHING macro 238
A2DP_EVT_OPEN_AND_START_STREAM_COMPLETED macro 238
A2DP_EVT_OPEN_STREAM_COMPLETED macro 238
A2DP_EVT_OPEN_STREAM_REQUESTED macro 238
A2DP_EVT_RECONFIGURE_STREAM_COMPLETED macro 238
A2DP_EVT_RECONFIGURE_STREAM_REQUESTED macro 239
A2DP_EVT_SEP_CAPABILITIES_RECEIVED macro 239
A2DP_EVT_SEP_INFO_RECEIVED macro 239
A2DP_EVT_SET_STREAM_CONFIGURATION macro 272
A2DP_EVT_SET_STREAM_CONFIGURATION_COMPLETED macro 239
A2DP_EVT_SET_STREAM_CONFIGURATION_REQUESTED macro 240
A2DP_EVT_START_STREAM_COMPLETED macro 240
A2DP_EVT_START_STREAM_REQUESTED macro 240
A2DP_EVT_STREAM_ABORTED macro 240
A2DP_EVT_STREAM_CLOSED macro 240
A2DP_EVT_STREAM_CONFIGURATION_RECEIVED macro 241
A2DP_EVT_STREAM_CONFIGURED macro 241
A2DP_EVT_STREAM_OPENED macro 241
A2DP_EVT_STREAM_RECONFIGURED macro 241
A2DP_EVT_STREAM_SECURITY_CONTROL_COMPLETED macro 242
A2DP_EVT_STREAM_STARTED macro 242
A2DP_EVT_STREAM_SUSPENDED macro 242
A2DP_EVT_SUSPEND_STREAM_COMPLETED macro 242
A2DP_EVT_SUSPEND_STREAM_REQUESTED macro 242
A2DP_MANAGER_STATE_CONNECTING macro 243
A2DP_MANAGER_STATE_IDLE macro 243
a2dp_private.h 1025
A2DP_SINK_FEATURE_AMPLIFIER macro 243
A2DP_SINK_FEATURE_HEADPHONE macro 243
A2DP_SINK_FEATURE_RECORDER macro 243
A2DP_SINK_FEATURE_SPEAKER macro 244
A2DP_SOURCE_FEATURE_MICROPHONE macro 244
A2DP_SOURCE_FEATURE_MIXER macro 244
A2DP_SOURCE_FEATURE_PLAYER macro 244
A2DP_SOURCE_FEATURE_TUNER macro 244
AAC_CHANNELS_1 macro 245
AAC_CHANNELS_2 macro 245
AAC_CHANNELS_ALL macro 245
AAC_OBJECT_TYPE_MPEG_2_LC macro 245
AAC_OBJECT_TYPE_MPEG_4_LC macro 245
AAC_OBJECT_TYPE_MPEG_4_LTP macro 246
AAC_OBJECT_TYPE_MPEG_4_SCALABLE macro 246
AAC_SAMPLING_FREQUENCY_11025 macro 246
AAC_SAMPLING_FREQUENCY_12000 macro 246
AAC_SAMPLING_FREQUENCY_16000 macro 246
AAC_SAMPLING_FREQUENCY_22050 macro 247
AAC_SAMPLING_FREQUENCY_24000 macro 247
AAC_SAMPLING_FREQUENCY_32000 macro 247
AAC_SAMPLING_FREQUENCY_44100 macro 247
AAC_SAMPLING_FREQUENCY_48000 macro 247
AAC_SAMPLING_FREQUENCY_64000 macro 248
AAC_SAMPLING_FREQUENCY_8000 macro 248
AAC_SAMPLING_FREQUENCY_88200 macro 248
AAC_SAMPLING_FREQUENCY_96000 macro 248
AAC_SAMPLING_FREQUENCY_ALL macro 248
AAC_VBR_NOT_SUPPORTED macro 249
AAC_VBR_SUPPORTED macro 249
ARG_NOT_USED macro 206
AT_EVT_CMD_CODE macro 978
AT_EVT_CMD_COMPLETED macro 978
AT_EVT_CMD_PARAM macro 978
AT_EVT_CMD_READ_CODE macro 978
AT_EVT_ERROR macro 978
AT_EVT_OK macro 979
AT_EVT_RING macro 979
at_parser.h 1083
ATCMD_BUFFER_LEN macro 979
ATT_ALLOCATE_BUFFERS function 15
ATT_CLIENT_ALLOCATE_BUFFERS function 15
AVC_BATTERY_STATUS_CRITICAL macro 416
AVC_BATTERY_STATUS_EXTERNAL macro 417
AVC_BATTERY_STATUS_FULL_CHARGE macro 417
AVC_BATTERY_STATUS_NORMAL macro 417
AVC_BATTERY_STATUS_WARNING macro 417
AVC_CAPABILITY_COMPANY_ID macro 417
AVC_CAPABILITY_EVENTS_SUPPORTED macro 418
AVC_CMD_GENERAL_POWER macro 418
AVC_CMD_GENERAL_RESERVE macro 418
AVC_CMD_GENERAL_SUBUNIT_INFO macro 418
AVC_CMD_GENERAL_UNIT_INFO macro 418
AVC_CMD_GENERAL_VENDOR_DEPENDENT macro 419
AVC_CMD_GENERAL_VERSION macro 419
AVC_CMD_PANEL_PASS_THROUGH macro 419
AVC_CTYPE_CONTROL macro 419
AVC_CTYPE_GENERAL_INQUORY macro 419
AVC_CTYPE_NOTIFY macro 420

AVC_CTYPE_SPECIFIC_IQUIRY macro 420
AVC_CTYPE_STATUS macro 420
AVC_EVENT_ADDRESSED_PLAYER_CHANGED macro 420
AVC_EVENT_AVAILABLE_PLAYERS_CHANGED macro 420
AVC_EVENT_BATT_STATUS_CHANGED macro 421
AVC_EVENT_FLAG_ADDRESSED_PLAYER_CHANGED macro 421
AVC_EVENT_FLAG_ALL macro 421
AVC_EVENT_FLAG_AVAILABLE_PLAYERS_CHANGED macro 421
AVC_EVENT_FLAG_BATT_STATUS_CHANGED macro 421
AVC_EVENT_FLAG_NOW_PLAYING_CONTENT_CHANGED macro 422
AVC_EVENT_FLAG_PLAYBACK_POS_CHANGED macro 422
AVC_EVENT_FLAG_PLAYBACK_STATUS_CHANGED macro 422
AVC_EVENT_FLAG_PLAYER_APPLICATION_SETTING_CHANGED macro 422
AVC_EVENT_FLAG_SYSTEM_STATUS_CHANGED macro 422
AVC_EVENT_FLAG_TRACK_CHANGED macro 423
AVC_EVENT_FLAG_TRACK_REACHED_END macro 423
AVC_EVENT_FLAG_TRACK_REACHED_START macro 423
AVC_EVENT_FLAG_UIDS_CHANGED macro 423
AVC_EVENT_FLAG_VOLUME_CHANGED macro 423
AVC_EVENT_NOW_PLAYING_CONTENT_CHANGED macro 424
AVC_EVENT_PLAYBACK_POS_CHANGED macro 424
AVC_EVENT_PLAYBACK_STATUS_CHANGED macro 424
AVC_EVENT_PLAYER_APPLICATION_SETTING_CHANGED macro 424
AVC_EVENT_SYSTEM_STATUS_CHANGED macro 424
AVC_EVENT_TRACK_CHANGED macro 425
AVC_EVENT_TRACK_REACHED_END macro 425
AVC_EVENT_TRACK_REACHED_START macro 425
AVC_EVENT_UIDS_CHANGED macro 425
AVC_EVENT_VOLUME_CHANGED macro 425
AVC_FLAG_BROWSING_CMD macro 426
AVC_FLAG_PANEL_CLICK macro 426
AVC_FLAG_RESPONSE macro 426
AVC_MAXEVENTS macro 426
AVC_MEDIA_ATTR_FLAG_ALBUM macro 426
AVC_MEDIA_ATTR_FLAG_ALL macro 427
AVC_MEDIA_ATTR_FLAG_ARTIST macro 427
AVC_MEDIA_ATTR_FLAG_GENRE macro 427
AVC_MEDIA_ATTR_FLAG_NUMBER macro 427
AVC_MEDIA_ATTR_FLAG_PLAYING_TIME macro 427
AVC_MEDIA_ATTR_FLAG_TITLE macro 428
AVC_MEDIA_ATTR_FLAG_TOTAL_NUMBER macro 428
AVC_MEDIA_ATTR_ID_ALBUM macro 428
AVC_MEDIA_ATTR_ID_ARTIST macro 428
AVC_MEDIA_ATTR_ID_GENRE macro 428
AVC_MEDIA_ATTR_ID_NUMBER macro 429
AVC_MEDIA_ATTR_ID_PLAYING_TIME macro 429
AVC_MEDIA_ATTR_ID_TITLE macro 429
AVC_MEDIA_ATTR_ID_TOTAL_NUMBER macro 429
AVC_MEDIA_PLAYER_VIRTUAL_FILESYSTEM macro 429
AVC_NOW_PLAYING macro 430
AVC_PACKET_TYPE_CONTINUE macro 430
AVC_PACKET_TYPE_END macro 430
AVC_PACKET_TYPE_SINGLE macro 430
AVC_PACKET_TYPE_START macro 430
AVC_PANEL_BUTTON_PRESSED macro 431
AVC_PANEL_BUTTON_RELEASED macro 431
AVC_PANEL_OPID_0 macro 431
AVC_PANEL_OPID_1 macro 431
AVC_PANEL_OPID_2 macro 431
AVC_PANEL_OPID_3 macro 432
AVC_PANEL_OPID_4 macro 432
AVC_PANEL_OPID_5 macro 432
AVC_PANEL_OPID_6 macro 432
AVC_PANEL_OPID_7 macro 432
AVC_PANEL_OPID_8 macro 433
AVC_PANEL_OPID_9 macro 433
AVC_PANEL_OPID_A macro 433
AVC_PANEL_OPID_ANGLE macro 433
AVC_PANEL_OPID_APPS_MENU macro 433
AVC_PANEL_OPID_B macro 434
AVC_PANEL_OPID_BACKWARD macro 434
AVC_PANEL_OPID_C macro 434
AVC_PANEL_OPID_CHANNEL_DOWN macro 434
AVC_PANEL_OPID_CHANNEL_UP macro 434
AVC_PANEL_OPID_CLEAR macro 435
AVC_PANEL_OPID_CONTENTS_MENU macro 435
AVC_PANEL_OPID_D macro 435
AVC_PANEL_OPID_DISPLAY_INFORMATION macro 435
AVC_PANEL_OPID_DOT macro 435
AVC_PANEL_OPID_DOWN macro 436
AVC_PANEL_OPID_EJECT macro 436
AVC_PANEL_OPID_ENTER macro 436
AVC_PANEL_OPID_EXIT macro 436
AVC_PANEL_OPID_F1 macro 436
AVC_PANEL_OPID_F2 macro 437
AVC_PANEL_OPID_F3 macro 437
AVC_PANEL_OPID_F4 macro 437
AVC_PANEL_OPID_F5 macro 437
AVC_PANEL_OPID_F6 macro 437
AVC_PANEL_OPID_F7 macro 438
AVC_PANEL_OPID_F8 macro 438
AVC_PANEL_OPID_F9 macro 438
AVC_PANEL_OPID_FAST_FORWARD macro 438
AVC_PANEL_OPID_FAVORITE_MENU macro 438
AVC_PANEL_OPID_FORWARD macro 439
AVC_PANEL_OPID_HELP macro 439
AVC_PANEL_OPID_INPUT_SELECT macro 439
AVC_PANEL_OPID_KEYBORD_FUNCTION macro 439
AVC_PANEL_OPID_LEFT macro 439
AVC_PANEL_OPID_LEFT_DOWN macro 440
AVC_PANEL_OPID_LEFT_UP macro 440
AVC_PANEL_OPID_LINKED_CONTENT macro 440
AVC_PANEL_OPID_LIST macro 440
AVC_PANEL_OPID_LIVE_TV macro 440
AVC_PANEL_OPID_LOCK macro 441
AVC_PANEL_OPID_MUTE macro 441
AVC_PANEL_OPID_MUTE_FUNCTION macro 441
AVC_PANEL_OPID_NEXT_DAY macro 441
AVC_PANEL_OPID_ON_DEMAND_MENU macro 441
AVC_PANEL_OPID_PAGE_DOWN macro 442
AVC_PANEL_OPID_PAGE_UP macro 442
AVC_PANEL_OPID_PAUSE macro 442
AVC_PANEL_OPID_PAUSE_PLAY_FUNCTION macro 442
AVC_PANEL_OPID_PAUSE_RECORD_FUNCTION macro 442

AVC_PANEL_OPID_PIP_DOWN macro 443
AVC_PANEL_OPID_PIP_MOVE macro 443
AVC_PANEL_OPID_PIP_UP macro 443
AVC_PANEL_OPID_PLAY macro 443
AVC_PANEL_OPID_PLAY_FUNCTION macro 443
AVC_PANEL_OPID_POWER_STATE_FUNCTION macro 444
AVC_PANEL_OPID_POWER_TOGGLE macro 444
AVC_PANEL_OPID_PREVIOUS_CHANNEL macro 444
AVC_PANEL_OPID_PREVIOUS_DAY macro 444
AVC_PANEL_OPID_RECORD macro 444
AVC_PANEL_OPID_RECORD_FUNCTION macro 445
AVC_PANEL_OPID_RESTORE_FOLUME_FUNCTION macro 445
AVC_PANEL_OPID_REWIND macro 445
AVC_PANEL_OPID_RF_BYPASS macro 445
AVC_PANEL_OPID_RIGHT macro 445
AVC_PANEL_OPID_RIGHT_DOWN macro 446
AVC_PANEL_OPID_RIGHT_UP macro 446
AVC_PANEL_OPID_ROOT_MENU macro 446
AVC_PANEL_OPID_SELECT macro 446
AVC_PANEL_OPID_SELECT_AUDIO_INPUT_FUNCTION macro 446
AVC_PANEL_OPID_SELECT_AV_INPUT_FUNCTION macro 447
AVC_PANEL_OPID_SELECT_DISK_FUNCTION macro 447
AVC_PANEL_OPID_SETUP_MENU macro 447
AVC_PANEL_OPID_SKIP macro 447
AVC_PANEL_OPID_SOUND_SELECT macro 447
AVC_PANEL_OPID_STOP macro 448
AVC_PANEL_OPID_STOP_FUNCTION macro 448
AVC_PANEL_OPID_SUBPICTURE macro 448
AVC_PANEL_OPID_TUNE_FUNCTION macro 448
AVC_PANEL_OPID_UP macro 448
AVC_PANEL_OPID_VENDOR_UNIQUE macro 449
AVC_PANEL_OPID_VOLUME_DOWN macro 449
AVC_PANEL_OPID_VOLUME_UP macro 449
AVC_PANEL_OPID_ZOOM macro 449
AVC_PDUID_ABORT_CONTINUING_RESPONSE macro 449
AVC_PDUID_ADD_TO_NOW_PLAYING macro 450
AVC_PDUID_CHANGE_PATH macro 450
AVC_PDUID_GENERAL_REJECT macro 450
AVC_PDUID_GET_CURRENT_PLAYER_APPLICATION_SETTING_VALUE
macro 450
AVC_PDUID_GET_ELEMENT_ATTRIBUTES macro 450
AVC_PDUID_GET_FOLDER_ITEMS macro 451
AVC_PDUID_GET_ITEM_ATTRIBUTES macro 451
AVC_PDUID_GET_PLAY_STATUS macro 451
AVC_PDUID_GET_PLAYER_APPLICATION_SETTING_ATTRIBUTE_TEXT
macro 451
AVC_PDUID_GET_PLAYER_APPLICATION_SETTING_VALUE_TEXT macro 451
AVC_PDUID_GETCAPABILITIES macro 452
AVC_PDUID_INFORM_BATTERY_STATUS_OF_CT macro 452
AVC_PDUID_INFORM_DISPLAYABLE_CHARACTER_SET macro 452
AVC_PDUID_LIST_PLAYER_APPLICATION_SETTING_ATTRIBUTES macro 452
AVC_PDUID_LIST_PLAYER_APPLICATION_SETTING_VALUES macro 452
AVC_PDUID_PLAY_ITEM macro 453
AVC_PDUID_REGISTER_NOTIFICATION macro 453
AVC_PDUID_REQUEST_CONTINUING_RESPONSE macro 453
AVC_PDUID_SEARCH macro 453
AVC_PDUID_SET_ABSOLUTE_VOLUME macro 453
AVC_PDUID_SET_ADDRESSED_PLAYER macro 454
AVC_PDUID_SET_BROWSED_PLAYER macro 454
AVC_PDUID_SET_PLAYER_APPLICATION_SETTING_VALUE macro 454
AVC_PLAY_STATUS_ERROR macro 454
AVC_PLAY_STATUS_FW_SEEK macro 454
AVC_PLAY_STATUS_PAUSED macro 455
AVC_PLAY_STATUS_PLAYING macro 455
AVC_PLAY_STATUS_REV_SEEK macro 455
AVC_PLAY_STATUS_STOPPED macro 455
AVC_PLAYER_SETTING_EQUALIZER_OFF macro 455
AVC_PLAYER_SETTING_EQUALIZER_ON macro 456
AVC_PLAYER_SETTING_EQUALIZER_STATUS macro 456
AVC_PLAYER_SETTING_REPEAT_ALL_TRACKS macro 456
AVC_PLAYER_SETTING_REPEAT_GROUP macro 456
AVC_PLAYER_SETTING_REPEAT_MODE_OFF macro 456
AVC_PLAYER_SETTING_REPEAT_MODE_STATUS macro 457
AVC_PLAYER_SETTING_REPEAT_SINGLE_TRACK macro 457
AVC_PLAYER_SETTING_SCAN_ALL_TRACKS macro 457
AVC_PLAYER_SETTING_SCAN_GROUP macro 457
AVC_PLAYER_SETTING_SCAN_OFF macro 457
AVC_PLAYER_SETTING_SCAN_STATUS macro 458
AVC_PLAYER_SETTING_SHUFFLE_ALL_TRACKS macro 458
AVC_PLAYER_SETTING_SHUFFLE_GROUP macro 458
AVC_PLAYER_SETTING_SHUFFLE_OFF macro 458
AVC_PLAYER_SETTING_SHUFFLE_STATUS macro 458
AVC_RESPONSE_ACCEPTED macro 459
AVC_RESPONSE_CHANGED macro 459
AVC_RESPONSE_IMPLEMENTED macro 459
AVC_RESPONSE_IN_TRANSITION macro 459
AVC_RESPONSE_INTERIM macro 459
AVC_RESPONSE_NOT_IMPLEMENTED macro 460
AVC_RESPONSE_REJECTED macro 460
AVC_RESPONSE_STABLE macro 460
AVC_RESPONSE_TIMEOUT macro 460
AVC_SCOPE_MEDIA_PLAYER_LIST macro 460
AVC_SEARCH macro 461
AVC_SUBUNIT_ID_EXTENDED_TO_NEXT_BYTE macro 461
AVC_SUBUNIT_ID_IGNORE macro 461
AVC_SUBUNIT_TYPE_AUDIO macro 461
AVC_SUBUNIT_TYPE_BULLETIN_BOARD macro 461
AVC_SUBUNIT_TYPE_CA macro 462
AVC_SUBUNIT_TYPE_CAMERA macro 462
AVC_SUBUNIT_TYPE_CAMERA_STORAGE macro 462
AVC_SUBUNIT_TYPE_DISC macro 462
AVC_SUBUNIT_TYPE_EXTENDED_TO_NEXT_BYTE macro 462
AVC_SUBUNIT_TYPE_MONITOR macro 463
AVC_SUBUNIT_TYPE_PANEL macro 463
AVC_SUBUNIT_TYPE_PRINTER macro 463
AVC_SUBUNIT_TYPE_TAPE_RECORDER_PLAYER macro 463
AVC_SUBUNIT_TYPE_TUNER macro 463
AVC_SUBUNIT_TYPE_UNIT macro 464
AVC_SUBUNIT_TYPE_VENDOR_UNIQUE macro 464
AVC_VOLUME_MAX macro 464
AVC_VOLUME_MIN macro 464

avctp.h 1025
AVCTP_ALLOCATE_BUFFERS_RAM_SIZE_VAR macro 21
AVCTP_ALLOCATE_BUFFERS_VARS macro 22
AVCTP_CHANNEL_FLAG_LISTENING macro 287
AVCTP_CHANNEL_FLAG_SENDING macro 287
AVCTP_CHANNEL_STATE_CONNECTED macro 288
AVCTP_CHANNEL_STATE_CONNECTING macro 288
AVCTP_CHANNEL_STATE_DISCONNECTING macro 288
AVCTP_CHANNEL_STATE_FREE macro 288
AVCTP_CHANNEL_STATE_IDLE macro 288
avctp_config.h 1029
AVCTP_ERROR_BAD_STATE macro 289
AVCTP_ERROR_SUCCESS macro 289
AVCTP_EVT_CHANNEL_CONNECTED macro 289
AVCTP_EVT_CHANNEL_DISCONNECTED macro 289
AVCTP_EVT_COMMAND_CANCELLED macro 289
AVCTP_EVT_COMMAND_RECEIVED macro 290
AVCTP_EVT_COMMAND_SENT macro 290
AVCTP_EVT_CONNECTION_FAILED macro 290
AVCTP_EVT_NOTHING macro 290
AVCTP_EVT_RESPONE_RECEIVED macro 290
AVCTP_EVT_RESPONSE_CANCELLED macro 291
AVCTP_EVT_RESPONSE_SENT macro 291
AVCTP_MANAGER_STATE_IDLE macro 291
AVCTP_MAX_CHANNELS macro 22
AVCTP_MAX_TRANSPORT_CHANNELS macro 23
AVCTP_MESSAGE_PACKET_TYPE_CONTINUE macro 291
AVCTP_MESSAGE_PACKET_TYPE_END macro 291
AVCTP_MESSAGE_PACKET_TYPE_SINGLE macro 292
AVCTP_MESSAGE_PACKET_TYPE_START macro 292
AVCTP_MESSAGE_TYPE_COMMAND macro 292
AVCTP_MESSAGE_TYPE_RESPONSE macro 292
avctp_packet.h 1030
avctp_private.h 1031
AVCTP_TRANSPORT_FLAG_RX_MESSAGE_STARTED macro 292
AVCTP_TRANSPORT_FLAG_SENDING macro 293
AVCTP_TRANSPORT_STATE_CONNECTED macro 293
AVCTP_TRANSPORT_STATE_CONNECTING macro 293
AVCTP_TRANSPORT_STATE_DISCONNECTING macro 293
AVCTP_TRANSPORT_STATE_FREE macro 293
AVCTP_TRANSPORT_STATE_IDLE macro 294
avdtp.h 1031
AVDTP_ALLOCATE_BUFFERS_FUNCTION macro 23
AVDTP_ALLOCATE_BUFFERS_RAM_SIZE_VAR macro 23
AVDTP_ALLOCATE_BUFFERS_VARS macro 24
AVDTP_CMD_ABORT macro 24
AVDTP_CMD_CLOSE macro 24
AVDTP_CMD_DELAYREPORT macro 366
AVDTP_CMD_DISCOVER macro 25
AVDTP_CMD_GET_ALL_CAPABILITIES macro 367
AVDTP_CMD_GET_CAPABILITIES macro 25
AVDTP_CMD_GET_CONFIGURATION macro 25
AVDTP_CMD_OPEN macro 25
AVDTP_CMD_RECONFIGURE macro 25
AVDTP_CMD_SECURITY_CONTROL macro 26
AVDTP_CMD_SET_CONFIGURATION macro 26
AVDTP_CMD_START macro 26
AVDTP_CMD_SUSPEND macro 26
AVDTP_CODEC_CONFIG_BUFFER_LEN macro 26
AVDTP_CODEC_OPCODE_PARSE_CONFIG macro 341
AVDTP_CODEC_OPCODE_PARSE_PACKET macro 341
AVDTP_CODEC_OPCODE_SERIALIZE_CONFIG macro 341
AVDTP_CODEC_TYPE_ATRAC macro 342
AVDTP_CODEC_TYPE_MPEG1_2_AUDIO macro 342
AVDTP_CODEC_TYPE_MPEG2_4_AAC macro 342
AVDTP_CODEC_TYPE_NON_A2DP macro 342
AVDTP_CODEC_TYPE_SBC macro 342
avdtp_config.h 1049
AVDTP_CONTENT_PROTECTION_METHOD_SCMS_T macro 367
avdtp_control.h 1051
AVDTP_CTLR_CHANNEL_EVT_CONNECTED macro 343
AVDTP_CTLR_CHANNEL_EVT_CONNECTION_FAILED macro 343
AVDTP_CTLR_CHANNEL_EVT_DISCONNECTED macro 343
AVDTP_CTLR_CHANNEL_EVT_NOTHING macro 343
AVDTP_CTRL_CHANNEL_EVT_DATA_RECEIVED macro 343
AVDTP_CTRL_CHANNEL_STATE_CONNECTED macro 344
AVDTP_CTRL_CHANNEL_STATE_CONNECTING macro 344
AVDTP_CTRL_CHANNEL_STATE_DISCONNECTED macro 344
AVDTP_CTRL_MESSAGE_TYPE_ACCEPT macro 344
AVDTP_CTRL_MESSAGE_TYPE_COMMAND macro 344
AVDTP_CTRL_MESSAGE_TYPE_FLD macro 345
AVDTP_CTRL_MESSAGE_TYPE_GENERAL_REJECT macro 367
AVDTP_CTRL_MESSAGE_TYPE_REJECT macro 345
AVDTP_CTRL_PACKET_TYPE_CONTINUE macro 345
AVDTP_CTRL_PACKET_TYPE_END macro 345
AVDTP_CTRL_PACKET_TYPE_FLD macro 345
AVDTP_CTRL_PACKET_TYPE_SIGNLE macro 346
AVDTP_CTRL_PACKET_TYPE_START macro 346
AVDTP_ERROR_BAD_ACP_SEID macro 346
AVDTP_ERROR_BAD_CP_FORMAT macro 346
AVDTP_ERROR_BAD_HEADER_FORMAT macro 346
AVDTP_ERROR_BAD_LENGTH macro 347
AVDTP_ERROR_BAD_MEDIA_TRANSPORT_FORMAT macro 347
AVDTP_ERROR_BAD_MULTIPLEXING_FORMAT macro 347
AVDTP_ERROR_BAD_PAYLOAD_FORMAT macro 347
AVDTP_ERROR_BAD_RECOVERY_FORMAT macro 347
AVDTP_ERROR_BAD_RECOVERY_TYPE macro 348
AVDTP_ERROR_BAD_ROHC_FORMAT macro 348
AVDTP_ERROR_BAD_SERV_CATEGORY macro 348
AVDTP_ERROR_BAD_STATE macro 348
AVDTP_ERROR_FAILED_TO_CONNECT_CONTROL macro 348
AVDTP_ERROR_FAILED_TO_CONNECT_TRANSPORT macro 349
AVDTP_ERROR_INVALID_CAPABILITIES macro 349
AVDTP_ERROR_NOT_SUPPORTED_COMMAND macro 349
AVDTP_ERROR_SEP_IN_USE macro 349
AVDTP_ERROR_SEP_NOT_IN_USE macro 349
AVDTP_ERROR_SUCCESS macro 350
AVDTP_ERROR_UNSUPPORTED_CONFIGURAION macro 350
AVDTP_EVT_ABORT_STREAM_COMPLETED macro 350
AVDTP_EVT_ABORT_STREAM_REQUESTED macro 350
AVDTP_EVT_CLOSE_STREAM_COMPLETED macro 350
AVDTP_EVT_CLOSE_STREAM_REQUESTED macro 351
AVDTP_EVT_CTRL_CHANNEL_CONNECTED macro 351
AVDTP_EVT_CTRL_CHANNEL_DISCONNECTED macro 351
AVDTP_EVT_CTRL_CONNECTION_FAILED macro 351
AVDTP_EVT_DELAYREPORT_COMPLETED macro 368

AVDTP_EVT_DISCOVER_COMPLETED macro 351
AVDTP_EVT_GET_SEP_CAPABILITIES_COMPLETED macro 352
AVDTP_EVT_GET_STREAM_CONFIGURATION_COMPLETED macro 352
AVDTP_EVT_LAST macro 352
AVDTP_EVT_MEDIA_PACKET_RECEIVED macro 352
AVDTP_EVT_MEDIA_PACKET_SEND_FAILED macro 352
AVDTP_EVT_MEDIA_PACKET_SENT macro 353
AVDTP_EVT_NULL macro 353
AVDTP_EVT_OPEN_STREAM_COMPLETED macro 353
AVDTP_EVT_OPEN_STREAM_REQUESTED macro 353
AVDTP_EVT_RECONFIGURE_STREAM_REQUESTED macro 353
AVDTP_EVT_SEP_CAPABILITIES_RECEIVED macro 354
AVDTP_EVT_SEP_INFO_RECEIVED macro 354
AVDTP_EVT_SET_STREAM_CONFIGURATION macro 369
AVDTP_EVT_SET_STREAM_CONFIGURATION_COMPLETED macro 354
AVDTP_EVT_SET_STREAM_CONFIGURATION_REQUESTED macro 354
AVDTP_EVT_START_STREAM_COMPLETED macro 354
AVDTP_EVT_START_STREAM_REQUESTED macro 355
AVDTP_EVT_STREAM_ABORTED macro 355
AVDTP_EVT_STREAM_CLOSED macro 355
AVDTP_EVT_STREAM_CONFIGURATION_RECEIVED macro 355
AVDTP_EVT_STREAM_CONFIGURED macro 355
AVDTP_EVT_STREAM_OPENED macro 356
AVDTP_EVT_STREAM_RECONFIGURE_COMPLETED macro 356
AVDTP_EVT_STREAM_RECONFIGURED macro 356
AVDTP_EVT_STREAM_SECURITY_CONTROL_COMPLETED macro 356
AVDTP_EVT_STREAM_STARTED macro 357
AVDTP_EVT_STREAM_SUSPENDED macro 357
AVDTP_EVT_SUSPEND_STREAM_COMPLETED macro 357
AVDTP_EVT_SUSPEND_STREAM_REQUESTED macro 357
AVDTP_MANAGER_FLAG_SENDING_MEDIA_PACKET macro 294
AVDTP_MANAGER_STATE_CONNECTING macro 357
AVDTP_MANAGER_STATE_IDLE macro 358
AVDTP_MAX_CMD_BUFFERS macro 27
AVDTP_MAX_CMD_PARAM_LEN macro 27
AVDTP_MAX_REMOTE_DEVICES macro 27
AVDTP_MAX_SEP macro 27
AVDTP_MAX_STREAM_TRANSPORT_SESSION macro 358
AVDTP_MAX_STREAMS macro 28
AVDTP_MAX_TRANSPORT_CHANNELS macro 28
AVDTP_MAX_TX_BUFFER_LEN macro 28
AVDTP_MEDIA_TYPE_AUDIO macro 358
AVDTP_MEDIA_TYPE_MULTIMEDIA macro 358
AVDTP_MEDIA_TYPE_VIDEO macro 358
avdtp_private.h 1052
AVDTP_SCMS_T_CP_BIT macro 367
AVDTP_SCMS_T_L_BIT macro 367
AVDTP_SEP_CAPABILITY_FLAG_CONTENT_PROTECTION macro 359
AVDTP_SEP_CAPABILITY_FLAG_DELAY_REPORTING macro 369
AVDTP_SEP_CAPABILITY_FLAG_HEADER_COMPRESSION macro 359
AVDTP_SEP_CAPABILITY_FLAG_MEDIA_CODEC macro 359
AVDTP_SEP_CAPABILITY_FLAG_MEDIA_TRANSPORT macro 359
AVDTP_SEP_CAPABILITY_FLAG_MULTIPLEXING macro 359
AVDTP_SEP_CAPABILITY_FLAG_RECOVERY macro 360
AVDTP_SEP_CAPABILITY_FLAG_REPORTING macro 360
AVDTP_SEP_SERVICE_CAPABILITY_CONTENT_PROTECTION macro 360
AVDTP_SEP_SERVICE_CAPABILITY_DELAY_REPORTING macro 369
AVDTP_SEP_SERVICE_CAPABILITY_HEADER_COMPRESSION macro 360
AVDTP_SEP_SERVICE_CAPABILITY_MEDIA_CODEC macro 360
AVDTP_SEP_SERVICE_CAPABILITY_MEDIA_TRANSPORT macro 361
AVDTP_SEP_SERVICE_CAPABILITY_MULTIPLEXING macro 361
AVDTP_SEP_SERVICE_CAPABILITY_RECOVERY macro 361
AVDTP_SEP_SERVICE_CAPABILITY_REPORTING macro 361
AVDTP_SEP_STATE_FREE macro 361
AVDTP_SEP_STATE_IDLE macro 362
AVDTP_SEP_TYPE_SINK macro 362
AVDTP_SEP_TYPE_SOURCE macro 362
AVDTP_STREAM_CLOSING_TRANSPORT_CHANNELS macro 362
AVDTP_STREAM_FLAG_LISTENING macro 362
AVDTP_STREAM_OPENING_TRANSPORT_CHANNELS macro 363
AVDTP_STREAM_STATE_ABORTING macro 363
AVDTP_STREAM_STATE_CLOSING macro 363
AVDTP_STREAM_STATE_CONFIGURED macro 363
AVDTP_STREAM_STATE_IDLE macro 363
AVDTP_STREAM_STATE_OPEN macro 364
AVDTP_STREAM_STATE_STREAMING macro 364
AVDTP_TC_OPCODE_CONNECT macro 364
AVDTP_TC_OPCODE_DISCONNECT macro 364
AVDTP_TC_STATUS_ERROR macro 364
AVDTP_TC_STATUS_SUCCESS macro 365
AVDTP_TRANSPORT_CHANNEL_TYPE_DEDICATED macro 365
AVDTP_TRANSPORT_CHANNEL_TYPE_SHARED macro 365
AVDTP_TRANSPORT_SESSION_TYPE_MEDIA macro 365
AVDTP_TRANSPORT_SESSION_TYPE_RECOVERY macro 365
AVDTP_TRANSPORT_SESSION_TYPE_REPORTING macro 366
avrcp.h 1053
AVRCP_ALLOCATE_BUFFERS_RAM_SIZE_VAR macro 29
AVRCP_ALLOCATE_BUFFERS_VARS macro 29
AVRCP_BTSIG_COMPANY_ID macro 464
AVRCP_CHANNEL_FLAG_LISTENING macro 465
AVRCP_CHANNEL_FLAG_PLAY_STATUS_REQUESTED macro 465
AVRCP_CHANNEL_FLAG_REGISTERING_NOTIFICATIONS macro 465
AVRCP_CHANNEL_FLAG_SENDING macro 465
AVRCP_CHANNEL_STATE_CONNECTED macro 465
AVRCP_CHANNEL_STATE_CONNECTING macro 466
AVRCP_CHANNEL_STATE_DISCONNECTING macro 466
AVRCP_CHANNEL_STATE_FREE macro 466
AVRCP_CHANNEL_STATE_IDLE macro 466
AVRCP_CMD_TIMEOUT macro 30
avrcp_command.h 1064
AVRCP_COMMAND_HANDLER macro 513
AVRCP_COMMAND_SENT_HANDLER macro 513
AVRCP_COMMAND_TYPE_BROWSING macro 466
AVRCP_COMMAND_TYPE_CONTROL macro 467
avrcp_config.h 1080
avrcp_config_event_handlers.h 1081
AVRCP_EVT_ADD_TO_NOW_PLAYING_COMPLETED macro 467
AVRCP_EVT_ADDRESSED_PLAYER_CHANGED macro 467

AVRCP_EVT_AVAILABLE_PLAYERS_CHANGED macro 467
AVRCP_EVT_BATT_STATUS_CHANGED macro 467
AVRCP_EVT_BATTERY_STATUS_OF_CT_RECEIVED macro 468
AVRCP_EVT_BROWSING_CHANNEL_CONNECTED macro 468
AVRCP_EVT_BROWSING_CHANNEL_DISCONNECTED macro 468
AVRCP_EVT_BROWSING_CONNECTION_FAILED macro 468
AVRCP_EVT_COMPANY_ID_LIST_RECEIVED macro 468
AVRCP_EVT_CONTROL_CHANNEL_CONNECTED macro 469
AVRCP_EVT_CONTROL_CHANNEL_DISCONNECTED macro 469
AVRCP_EVT_CONTROL_CONNECTION_FAILED macro 469
AVRCP_EVT_DISPLAYABLE_CHARACTER_SET_RECEIVED macro 469
AVRCP_EVT_ELEMENT_ATTRIBUTES_REQUESTED macro 469
AVRCP_EVT_EVENT_ID_LIST_RECEIVED macro 470
AVRCP_EVT_GET_ELEMENT_ATTRIBUTES_RECEIVED macro 470
AVRCP_EVT_GET_PLAY_STATUS_RECEIVED macro 470
AVRCP_EVT_INFORM_BATTERY_STATUS_OF_CT_COMPLETED macro 470
AVRCP_EVT_INFORM_DISPLAYABLE_CHARACTER_SET_COMPLETED macro 470
AVRCP_EVT_NOTHING macro 471
AVRCP_EVT_NOW_PLAYING_CONTENT_CHANGED macro 471
AVRCP_EVT_PANEL_COMMAND_RECEIVED macro 471
AVRCP_EVT_PANEL_RESPONSE_RECEIVED macro 471
AVRCP_EVT_PLAY_ITEM_COMPLETED macro 471
AVRCP_EVT_PLAYBACK_POS_CHANGED macro 472
AVRCP_EVT_PLAYBACK_STATUS_CHANGED macro 472
AVRCP_EVT_PLAYER_APPLICATION_SETTING_CHANGED macro 472
AVRCP_EVT_PLAYER_CURRENT_SETTING_VALUES_RECEIVED macro 472
AVRCP_EVT_PLAYER_SETTING_ATTRIBUTES_RECEIVED macro 472
AVRCP_EVT_PLAYER_SETTING_ATTRIBUTES_TEXT_RECEIVED macro 473
AVRCP_EVT_PLAYER_SETTING_VALUES_RECEIVED macro 473
AVRCP_EVT_PLAYER_SETTING_VALUES_TEXT_RECEIVED macro 473
AVRCP_EVT_REGISTER_NOTIFICATION_REQUESTED macro 473
AVRCP_EVT_REGISTER_NOTIFICATIONS_COMPLETED macro 473
AVRCP_EVT_SEARCH_COMPLETED macro 474
AVRCP_EVT_SET_ABSOLUTE_VOLUME_COMPLETED macro 474
AVRCP_EVT_SET_ABSOLUTE_VOLUME_REQUESTED macro 474
AVRCP_EVT_SET_ADDRESSED_PLAYER_COMPLETED macro 474
AVRCP_EVT_SET_PLAYER_SETTING_VALUES_COMPLETED macro 474
AVRCP_EVT_SYSTEM_STATUS_CHANGED macro 475
AVRCP_EVT_TRACK_CHANGED macro 475
AVRCP_EVT_TRACK_REACHED_END macro 475
AVRCP_EVT_TRACK_REACHED_START macro 475
AVRCP_EVT_UIDS_CHANGED macro 475
AVRCP_EVT_VOLUME_CHANGED macro 476
AVRCP_MANAGER_FLAG_SEARCHING macro 476
AVRCP_MANAGER_STATE_IDLE macro 476
AVRCP_MAX_CHANNELS macro 30
AVRCP_MAX_CMD_BUFFERS macro 30
AVRCP_MAX_CMD_PARAM_LEN macro 30
AVRCP_MAX_DEVICE_NAME_LEN macro 31
AVRCP_MAX_ELEMENT_ATTRIBUTES macro 476
AVRCP_MAX_SEARCH_RESULTS macro 31
avrcp_private.h 1082
AVRCP_RESPONSE_HANDLER macro 513
AVRCP_RESPONSE_SENT_HANDLER macro 513

B

baseband.h 1084
BEGIN_DE_SEQUENCE macro 843
Bluetooth Stack Library Help 2
Bluetooth Stack Licensing 4
bt_a2dp_aac_codec_handler function 224
bt_a2dp_aac_config_t structure 230
bt_a2dp_abort_stream macro 249
bt_a2dp_add_media_rx_buffer macro 249
bt_a2dp_add_media_tx_buffer macro 250
bt_a2dp_call_codec macro 250
bt_a2dp_cancel_listen macro 250
bt_a2dp_clear_media_tx_queue macro 270
bt_a2dp_close_stream macro 250
bt_a2dp_connect macro 251
bt_a2dp_connect_ex macro 251
bt_a2dp_create_stream macro 251
bt_a2dp_destroy_stream macro 252
bt_a2dp_disconnect macro 252
bt_a2dp_discover macro 252
bt_a2dp_event_t type 231
bt_a2dp_evt_open_and_start_stream_completed_t structure 231
bt_a2dp_find_codec macro 253
bt_a2dp_find_server function 224
bt_a2dp_find_server_callback_fp type 231
bt_a2dp_find_sink function 224
bt_a2dp_find_source function 225
bt_a2dp_get_all_capabilities macro 270
bt_a2dp_get_capabilities macro 253
bt_a2dp_get_configuration macro 270
bt_a2dp_get_hci_connection macro 253
bt_a2dp_get_mgr function 225
bt_a2dp_get_stream_codec_config macro 254
bt_a2dp_get_stream_codec_type macro 254
bt_a2dp_get_stream_config macro 271
bt_a2dp_get_stream_direction macro 271
bt_a2dp_get_stream_local_sep_id macro 254
bt_a2dp_get_stream_remote_address macro 255
bt_a2dp_get_stream_remote_sep_id macro 255
bt_a2dp_get_stream_state macro 255
bt_a2dp_init function 225
bt_a2dp_listen macro 255
bt_a2dp_mgr_callback_fp type 232
bt_a2dp_mgr_t type 233
bt_a2dp_mpeg_codec_handler function 225
bt_a2dp_mpeg_config_t structure 233
bt_a2dp_open_and_start_stream function 226
bt_a2dp_open_stream macro 270
bt_a2dp_reconfigure_stream macro 256
bt_a2dp_register_aac_codec function 226
bt_a2dp_register_callback function 226
bt_a2dp_register_mpeg_codec function 227
bt_a2dp_register_sink macro 256

bt_a2dp_register_source macro 256
bt_a2dp_remove_media_rx_buffer macro 257
bt_a2dp_remove_media_tx_buffer macro 257
bt_a2dp_report_delay macro 271
bt_a2dp_sbc_codec_handler function 227
bt_a2dp_sbc_config_t structure 233
bt_a2dp_sbc_packet_info_t structure 234
bt_a2dp_set_configuration function 228
bt_a2dp_set_media_tx_queue_limit macro 271
bt_a2dp_start function 228
bt_a2dp_start_stream macro 257
bt_a2dp_suspend_stream macro 258
bt_alloc_buffer function 968
BT_ASSERT macro 206
bt_at_parse_fragment function 968
bt_at_parser_callback_pf type 996
bt_at_parser_reset function 968
bt_at_parser_t structure 996
bt_av_add_to_now_playing_t structure 397
bt_av_battery_status_of_ct_t structure 397
bt_av_capability_company_id_t structure 398
bt_av_capability_event_id_t structure 398
bt_av_command_t type 399
bt_av_displayable_character_set_t structure 399
bt_av_element_attribute_t structure 399
bt_av_element_attributes_t structure 400
bt_av_element_id_t structure 400
bt_av_get_element_attributes_t structure 400
bt_av_notification_addressed_player_changed_t structure 401
bt_av_notification_app_setting_changed_t structure 401
bt_av_notification_battery_status_t structure 401
bt_av_notification_playback_pos_changed_t structure 402
bt_av_notification_playback_status_changed_t structure 402
bt_av_notification_system_status_changed_t structure 402
bt_av_notification_t structure 403
bt_av_notification_track_changed_t structure 404
bt_av_notification_uids_changed_t structure 404
bt_av_notification_volume_changed_t structure 404
bt_av_play_item_t structure 405
bt_av_play_status_t structure 405
bt_av_player_setting_current_values_t structure 405
bt_av_player_setting_values_t structure 406
bt_av_player_setting_values_text_t structure 406
bt_av_player_settings_t structure 407
bt_av_player_settings_text_t structure 407
bt_av_player_text_t structure 408
bt_av_register_notification_t structure 408
bt_av_response_t structure 408
bt_av_set_absolute_volume_t structure 409
bt_av_set_addressed_player_t structure 409
bt_avctp_cancel_command function 275
bt_avctp_cancel_listen function 275
bt_avctp_cancel_response function 275
bt_avctp_channel_t type 281
bt_avctp_connect function 276
bt_avctp_create_channel function 276
bt_avctp_create_outgoing_channel function 276
bt_avctp_destroy_channel function 276
bt_avctp_disconnect function 277
bt_avctp_event_t union 281
bt_avctp_evt_channel_connected_t structure 282
bt_avctp_evt_channel_disconnected_t structure 282
bt_avctp_evt_command_cancelled_t structure 283
bt_avctp_evt_command_received_t structure 283
bt_avctp_evt_command_sent_t structure 283
bt_avctp_evt_connection_failed_t structure 284
bt_avctp_evt_response_cancelled_t structure 284
bt_avctp_evt_response_received_t structure 284
bt_avctp_evt_response_sent_t structure 285
bt_avctp_get_channel_remote_address function 277
bt_avctp_get_channel_state function 277
bt_avctp_get_hci_connection function 277
bt_avctp_get_mgr function 278
bt_avctp_init function 278
bt_avctp_listen function 278
bt_avctp_message_t type 285
bt_avctp_mgr_callback_fp type 285
bt_avctp_mgr_t type 286
bt_avctp_packet_t structure 286
bt_avctp_send_command function 278
bt_avctp_send_response function 279
bt_avctp_set_callback function 279
bt_avctp_start function 279
bt_avctp_transport_t type 286
bt_avdtp_abort_stream function 302
bt_avdtp_add_media_rx_buffer function 302
bt_avdtp_add_media_tx_buffer function 302
bt_avdtp_cancel_listen function 303
bt_avdtp_clear_media_tx_queue function 314
bt_avdtp_close_stream function 303
bt_avdtp_codec_handler_fp type 317
bt_avdtp_codec_op_decode_t structure 318
bt_avdtp_codec_op_encode_t structure 318
bt_avdtp_codec_op_param_t type 318
bt_avdtp_codec_op_parse_config_t structure 318
bt_avdtp_codec_op_parse_packet_t structure 319
bt_avdtp_codec_op_serialize_config_t structure 319
bt_avdtp_codec_t structure 320
bt_avdtp_connect macro 366
bt_avdtp_connect_ex macro 366
bt_avdtp_control_channel_t structure 320
bt_avdtp_control_cmd_t structure 320
bt_avdtp_create_stream function 303
bt_avdtp_ctrl_evt_data_received_t structure 321
bt_avdtp_destroy_stream function 303
bt_avdtp_disconnect function 304
bt_avdtp_discover function 304
bt_avdtp_event_t union 321
bt_avdtp_evt_abort_stream_requested_t structure 323
bt_avdtp_evt_close_stream_completed_t structure 323
bt_avdtp_evt_close_stream_requested_t structure 323
bt_avdtp_evt_ctrl_channel_connected_t structure 324
bt_avdtp_evt_ctrl_channel_disconnected_t structure 324
bt_avdtp_evt_ctrl_connection_failed_t structure 369
bt_avdtp_evt_delay_report_completed_t structure 368
bt_avdtp_evt_discover_completed_t structure 325

bt_avdtp_evt_get_sep_capabilities_completed_t structure 325
bt_avdtp_evt_get_stream_configuration_completed_t structure 325
bt_avdtp_evt_media_packet_received_t structure 326
bt_avdtp_evt_media_packet_send_failed_t structure 326
bt_avdtp_evt_media_packet_sent_t structure 326
bt_avdtp_evt_open_stream_completed_t structure 327
bt_avdtp_evt_open_stream_requested_t structure 327
bt_avdtp_evt_reconfigure_stream_requested_t structure 328
bt_avdtp_evt_sep_capabilities_received_t structure 328
bt_avdtp_evt_sep_info_received_t structure 328
bt_avdtp_evt_set_stream_configuration_completed_t structure 329
bt_avdtp_evt_set_stream_configuration_requested_t structure 329
bt_avdtp_evt_set_stream_configuration_t structure 370
bt_avdtp_evt_start_stream_completed_t structure 330
bt_avdtp_evt_start_stream_requested_t structure 330
bt_avdtp_evt_stream_aborted_t structure 331
bt_avdtp_evt_stream_closed_t structure 331
bt_avdtp_evt_stream_configured_t structure 331
bt_avdtp_evt_stream_opened_t structure 332
bt_avdtp_evt_stream_reconfigure_completed_t structure 332
bt_avdtp_evt_stream_reconfigured_t structure 332
bt_avdtp_evt_stream_security_control_completed_t structure 333
bt_avdtp_evt_stream_started_t structure 333
bt_avdtp_evt_stream_suspended_t structure 333
bt_avdtp_evt_suspend_stream_completed_t structure 334
bt_avdtp_evt_suspend_stream_requested_t structure 334
bt_avdtp_find_codec function 304
bt_avdtp_get_all_capabilities function 314
bt_avdtp_get_capabilities function 305
bt_avdtp_get_configuration function 305
bt_avdtp_get_hci_connection function 305
bt_avdtp_get_l2cap_channel function 314
bt_avdtp_get_mgr function 306
bt_avdtp_get_sep function 306
bt_avdtp_get_stream_codec_config function 306
bt_avdtp_get_stream_codec_type function 306
bt_avdtp_get_stream_config function 307
bt_avdtp_get_stream_direction function 315
bt_avdtp_get_stream_local_sep_id function 307
bt_avdtp_get_stream_remote_address function 307
bt_avdtp_get_stream_remote_sep_id function 308
bt_avdtp_get_stream_state function 308
bt_avdtp_init function 308
bt_avdtp_listen function 308
bt_avdtp_mgr_callback_fp type 334
bt_avdtp_mgr_t structure 336
bt_avdtp_open_stream function 309
bt_avdtp_reconfigure_stream function 309
bt_avdtp_register_callback function 309
bt_avdtp_register_codec function 310
bt_avdtp_register_sep function 310
bt_avdtp_remove_media_rx_buffer function 311
bt_avdtp_remove_media_tx_buffer function 311
bt_avdtp_report_delay function 315
bt_avdtp_security_control function 311
bt_avdtp_sep_capabilities_t structure 337
bt_avdtp_sep_t type 338
bt_avdtp_set_configuration function 312
bt_avdtp_set_media_tx_queue_limit function 315
bt_avdtp_start function 312
bt_avdtp_start_stream function 312
bt_avdtp_stream_t structure 338
bt_avdtp_suspend_stream function 313
bt_avdtp_transport_channel_t structure 339
bt_avdtp_transport_op_callback_fp type 339
bt_avdtp_transport_session_t structure 340
bt_avdtp_unregister_codec function 313
bt_avrcp_0_click macro 476
bt_avrcp_0_press macro 477
bt_avrcp_0_release macro 500
bt_avrcp_1_click macro 477
bt_avrcp_1_press macro 477
bt_avrcp_1_release macro 501
bt_avrcp_2_click macro 477
bt_avrcp_2_press macro 477
bt_avrcp_2_release macro 501
bt_avrcp_3_click macro 478
bt_avrcp_3_press macro 478
bt_avrcp_3_release macro 501
bt_avrcp_4_click macro 478
bt_avrcp_4_press macro 478
bt_avrcp_4_release macro 501
bt_avrcp_5_click macro 478
bt_avrcp_5_press macro 479
bt_avrcp_5_release macro 501
bt_avrcp_6_click macro 479
bt_avrcp_6_press macro 479
bt_avrcp_6_release macro 502
bt_avrcp_7_click macro 479
bt_avrcp_7_press macro 479
bt_avrcp_7_release macro 502
bt_avrcp_8_click macro 480
bt_avrcp_8_press macro 480
bt_avrcp_8_release macro 502
bt_avrcp_9_click macro 480
bt_avrcp_9_press macro 480
bt_avrcp_9_release macro 502
bt_avrcp_abort_continuing_response function 380
bt_avrcp_add_to_now_playing function 380
bt_avrcp_angle_click macro 480
bt_avrcp_angle_press macro 481
bt_avrcp_angle_release macro 502
bt_avrcp_avrcpt_request_continuing_response function 380
bt_avrcp_backward_click macro 481
bt_avrcp_backward_press macro 481
bt_avrcp_backward_release macro 503
bt_avrcp_cancel_find function 380
bt_avrcp_cancel_listen function 380
bt_avrcp_channel_down_click macro 481
bt_avrcp_channel_down_press macro 481
bt_avrcp_channel_down_release macro 503
bt_avrcp_channel_t type 409
bt_avrcp_channel_up_click macro 482
bt_avrcp_channel_up_press macro 482
bt_avrcp_channel_up_release macro 503
bt_avrcp_clear_click macro 482

bt_avrcp_clear_press macro 482
bt_avrcp_clear_release macro 503
bt_avrcp_connect function 381
bt_avrcp_content_menu_click macro 482
bt_avrcp_content_menu_release macro 503
bt_avrcp_contents_menu_press macro 483
bt_avrcp_create_channel function 381
bt_avrcp_create_outgoing_channel function 381
bt_avrcp_destroy_channel function 382
bt_avrcp_device_t structure 410
bt_avrcp_disconnect function 382
bt_avrcp_display_info_click macro 483
bt_avrcp_display_info_press macro 483
bt_avrcp_display_info_release macro 504
bt_avrcp_dot_click macro 483
bt_avrcp_dot_press macro 483
bt_avrcp_dot_release macro 504
bt_avrcp_down_click macro 484
bt_avrcp_down_press macro 484
bt_avrcp_down_release macro 504
bt_avrcp_eject_click macro 484
bt_avrcp_eject_press macro 484
bt_avrcp_eject_release macro 504
bt_avrcp_enter_click macro 484
bt_avrcp_enter_press macro 485
bt_avrcp_enter_release macro 504
bt_avrcp_event_t union 410
bt_avrcp_evt_channel_connected_t structure 411
bt_avrcp_evt_channel_disconnected_t structure 412
bt_avrcp_evt_connection_failed_t structure 412
bt_avrcp_evt_panel_command_received_t structure 412
bt_avrcp_evt_panel_response_received_t structure 413
bt_avrcp_evt_register_events_completed_t structure 413
bt_avrcp_evt_search_completed_t structure 414
bt_avrcp_exit_click macro 485
bt_avrcp_exit_press macro 485
bt_avrcp_exit_release macro 505
bt_avrcp_f1_click macro 485
bt_avrcp_f1_press macro 485
bt_avrcp_f1_release macro 505
bt_avrcp_f2_click macro 486
bt_avrcp_f2_press macro 486
bt_avrcp_f2_release macro 505
bt_avrcp_f3_click macro 486
bt_avrcp_f3_press macro 486
bt_avrcp_f3_release macro 505
bt_avrcp_f4_click macro 486
bt_avrcp_f4_press macro 487
bt_avrcp_f4_release macro 505
bt_avrcp_f5_click macro 487
bt_avrcp_f5_press macro 487
bt_avrcp_f5_release macro 506
bt_avrcp_f6_click macro 487
bt_avrcp_f6_press macro 487
bt_avrcp_f6_release macro 506
bt_avrcp_f7_click macro 488
bt_avrcp_f7_press macro 488
bt_avrcp_f7_release macro 506
bt_avrcp_f8_click macro 488
bt_avrcp_f8_press macro 488
bt_avrcp_f8_release macro 506
bt_avrcp_f9_click macro 488
bt_avrcp_f9_press macro 489
bt_avrcp_f9_release macro 506
bt_avrcp_fast_forward_click macro 489
bt_avrcp_fast_forward_press macro 489
bt_avrcp_fast_forward_release macro 507
bt_avrcp_favorite_menu_click macro 489
bt_avrcp_favorite_menu_release macro 507
bt_avrcp_favorive_menu_press macro 489
bt_avrcp_find_callback_fp type 414
bt_avrcp_find_controller function 382
bt_avrcp_find_target function 382
bt_avrcp_find_targets function 383
bt_avrcp_forward_click macro 490
bt_avrcp_forward_press macro 490
bt_avrcp_forward_release macro 507
bt_avrcp_get_browsing_channel_state function 383
bt_avrcp_get_channel_remote_address function 383
bt_avrcp_get_company_id_list function 384
bt_avrcp_get_control_channel_state function 384
bt_avrcp_get_current_player_application_setting_value function 384
bt_avrcp_get_element_attributes function 384
bt_avrcp_get_hci_connection function 385
bt_avrcp_get_mgr function 385
bt_avrcp_get_play_status function 385
bt_avrcp_get_player_application_setting_attr_text function 385
bt_avrcp_get_player_application_setting_value_text function 386
bt_avrcp_get_subuint_info function 386
bt_avrcp_get_supported_event_id_list function 386
bt_avrcp_get_unit_info function 387
bt_avrcp_help_click macro 490
bt_avrcp_help_press macro 490
bt_avrcp_help_release macro 507
bt_avrcp_inform_battery_status function 387
bt_avrcp_inform_displayable_character_set function 387
bt_avrcp_init_controller function 387
bt_avrcp_init_target function 388
bt_avrcp_input_select_click macro 490
bt_avrcp_input_select_press macro 491
bt_avrcp_input_select_release macro 507
bt_avrcp_left_click macro 491
bt_avrcp_left_down_click macro 491
bt_avrcp_left_down_press macro 491
bt_avrcp_left_down_release macro 508
bt_avrcp_left_press macro 491
bt_avrcp_left_release macro 508
bt_avrcp_left_up_click macro 492
bt_avrcp_left_up_press macro 492
bt_avrcp_left_up_release macro 508
bt_avrcp_list_player_application_setting_attributes function 388
bt_avrcp_list_player_application_setting_values function 388
bt_avrcp_listen function 389
bt_avrcp_mgr_callback_fp type 414
bt_avrcp_mgr_t type 416
bt_avrcp_mute_click macro 492

bt_avrcp_mute_press macro 492
bt_avrcp_mute_release macro 508
bt_avrcp_page_down_click macro 492
bt_avrcp_page_down_press macro 493
bt_avrcp_page_down_release macro 508
bt_avrcp_page_up_click macro 493
bt_avrcp_page_up_press macro 493
bt_avrcp_page_up_release macro 509
bt_avrcp_pause_click macro 493
bt_avrcp_pause_press macro 493
bt_avrcp_pause_release macro 509
bt_avrcp_play_click macro 494
bt_avrcp_play_item function 389
bt_avrcp_play_press macro 494
bt_avrcp_play_release macro 509
bt_avrcp_power_click macro 494
bt_avrcp_power_press macro 494
bt_avrcp_power_release macro 509
bt_avrcp_previous_channel_click macro 494
bt_avrcp_previous_channel_press macro 495
bt_avrcp_previous_channel_release macro 509
bt_avrcp_record_click macro 495
bt_avrcp_record_press macro 495
bt_avrcp_record_release macro 510
bt_avrcp_register_notification function 389
bt_avrcp_register_notifications function 389
bt_avrcp_rewind_click macro 495
bt_avrcp_rewind_press macro 495
bt_avrcp_rewind_release macro 510
bt_avrcp_right_click macro 496
bt_avrcp_right_down_click macro 496
bt_avrcp_right_down_press macro 496
bt_avrcp_right_down_release macro 510
bt_avrcp_right_press macro 496
bt_avrcp_right_release macro 510
bt_avrcp_right_up_click macro 496
bt_avrcp_right_up_press macro 497
bt_avrcp_right_up_release macro 510
bt_avrcp_root_menu_click macro 497
bt_avrcp_root_menu_press macro 497
bt_avrcp_root_menu_release macro 511
bt_avrcp_select_click macro 497
bt_avrcp_select_press macro 497
bt_avrcp_select_release macro 511
bt_avrcp_send_button_click function 390
bt_avrcp_send_cmd function 390
bt_avrcp_send_panel_control function 390
bt_avrcp_send_press_panel_control function 391
bt_avrcp_send_release_panel_control function 395
bt_avrcp_send_simple_panel_cmd function 391
bt_avrcp_set_absolute_volume function 391
bt_avrcp_set_addressed_player function 391
bt_avrcp_set_player_application_setting_value function 392
bt_avrcp_setup_menu_click macro 498
bt_avrcp_setup_menu_press macro 498
bt_avrcp_setup_menu_release macro 511
bt_avrcp_sound_select_click macro 498
bt_avrcp_sound_select_press macro 498
bt_avrcp_sound_select_release macro 511
bt_avrcp_start function 392
bt_avrcp_stop_click macro 498
bt_avrcp_stop_press macro 499
bt_avrcp_stop_release macro 511
bt_avrcp_subpicture_click macro 499
bt_avrcp_subpicture_press macro 499
bt_avrcp_subpicture_release macro 512
bt_avrcp_tg_send_element_attributes function 392
bt_avrcp_tg_set_absolute_volume function 393
bt_avrcp_tg_set_battery_status function 393
bt_avrcp_tg_set_channel_absolute_volume function 393
bt_avrcp_tg_set_current_track function 393
bt_avrcp_tg_set_play_status function 394
bt_avrcp_tg_set_system_status function 394
bt_avrcp_up_click macro 499
bt_avrcp_up_press macro 499
bt_avrcp_up_release macro 512
bt_avrcp_volume_down_click macro 500
bt_avrcp_volume_down_press macro 500
bt_avrcp_volume_down_release macro 512
bt_avrcp_volume_up_click macro 500
bt_avrcp_volume_up_press macro 500
bt_avrcp_volume_up_release macro 512
bt_bdaddr.h 1084
bt_bdaddr_cp type 216
bt_bdaddr_is_null function 197
bt_bdaddr_p type 216
bt_bdaddr_t structure 217
bt_bdaddrs_are_equal function 197
bt_bool type 217
bt_bt_buffer_header_p structure 997
bt_buffer_header_t structure 997
bt_buffer_mgr_p structure 997
bt_buffer_mgr_t structure 997
bt_byte type 784
bt_byte_cp type 217
bt_byte_p type 217
bt_char type 217
bt_char_cp type 218
bt_char_p type 218
bt_config.h 1086
bt_cp_header_t type 368
bt_device_t structure 514
BT_ENABLE_BLE macro 19
BT_ENABLE_SCO macro 31
BT_FALSE macro 207
bt_find_devices_callback_fp type 514
bt_free_buffer function 969
bt_gap_find_devices function 513
bt_get_buffer function 969
bt_get_buffer_header function 969
bt_get_buffer_index function 969
bt_hci_add_param_bdaddr function 514
bt_hci_add_param_byte function 514
bt_hci_add_param_cod function 515
bt_hci_add_param_hconn macro 563
bt_hci_add_param_int function 515

bt_hci_add_param_lap macro 563
bt_hci_add_param_linkkey function 515
bt_hci_add_param_long function 515
bt_hci_add_param_string function 515
bt_hci_add_param_uint macro 564
bt_hci_add_param_ulong macro 564
bt_hci_alloc_canned_command function 516
bt_hci_alloc_command function 516
bt_hci_alloc_data_buffer function 516
bt_hci_allocate_write_eir_command function 516
bt_hci_authenticate macro 564
bt_hci_authenticate_ex function 516
bt_hci_cancel_find_devices function 517
bt_hci_cancel_find_devices_ex function 561
bt_hci_cancel_request_remote_name function 517
bt_hci_cancel_send_acl_data function 517
bt_hci_cmd_callback_fp type 677
bt_hci_cmd_listener_fp type 677
bt_hci_command_p structure 678
bt_hci_command_t structure 678
bt_hci_conn_state_p type 678
bt_hci_conn_state_t type 678
bt_hci_connect function 517
bt_hci_connect_callback_fp type 679
bt_hci_connect_sco function 517
bt_hci_ctrl_listener_t type 679
bt_hci_ctrl_register_data_listener function 518
bt_hci_ctrl_register_listener function 518
bt_hci_ctrl_state_t structure 679
bt_hci_ctrl_unregister_listener function 518
bt_hci_data_buffer_p structure 680
bt_hci_data_buffer_t structure 680
bt_hci_data_callback_fp type 680
bt_hci_data_listener_fp type 680
bt_hci_data_p structure 681
bt_hci_data_t structure 681
bt_hci_disconnect function 518
bt_hci_disconnect_callback_fp type 681
bt_hci_event_e union 682
bt_hci_event_handler_ex_fp type 682
bt_hci_event_handler_fp type 682
bt_hci_event_listener_fp type 682
bt_hci_event_p structure 682
bt_hci_event_t structure 683
bt_hci_evt_authentication_complete_handler function 518
BT_HCI_EVT_AUTHENTICATION_COMPLETE_HANDLER macro 31
bt_hci_evt_authentication_complete_t structure 683
bt_hci_evt_change_conn_link_complete_handler function 519
bt_hci_evt_command_complete_handler function 519
bt_hci_evt_command_complete_t structure 683
bt_hci_evt_command_status_handler function 519
bt_hci_evt_command_status_t structure 683
bt_hci_evt_conn_packet_type_changed_handler function 519
bt_hci_evt_connection_complete_handler function 519
BT_HCI_EVT_CONNECTION_COMPLETE_HANDLER macro 32
bt_hci_evt_connection_complete_t structure 684
bt_hci_evt_connection_request_handler function 520
BT_HCI_EVT_CONNECTION_REQUEST_HANDLER macro 32
bt_hci_evt_connection_request_t structure 684
bt_hci_evt_data_buffer_overflow_handler function 520
bt_hci_evt_default_handler function 520
bt_hci_evt_disconnection_complete_handler function 520
bt_hci_evt_disconnection_complete_t structure 684
bt_hci_evt_encryption_change_handler function 520
BT_HCI_EVT_ENCRYPTION_CHANGE_HANDLER macro 32
bt_hci_evt_encryption_change_t structure 684
bt_hci_evt_extended_inquiry_result_handler function 521
BT_HCI_EVT_EXTENDED_INQUIRY_RESULT_HANDLER macro 32
bt_hci_evt_flow_specification_complete_handler function 521
bt_hci_evt_flush_occured_handler function 521
bt_hci_evt_hardware_error_handler function 521
bt_hci_evt_inquiry_complete_handler function 521
BT_HCI_EVT_INQUIRY_COMPLETE_HANDLER macro 32
bt_hci_evt_inquiry_result_handler function 522
BT_HCI_EVT_INQUIRY_RESULT_HANDLER macro 33
bt_hci_evt_inquiry_result_with_rssi_handler function 522
BT_HCI_EVT_INQUIRY_RESULT_WITH_RSSI_HANDLER macro 33
bt_hci_evt_link_key_notification_handler function 522
BT_HCI_EVT_LINK_KEY_NOTIFICATION_HANDLER macro 33
bt_hci_evt_link_key_request_handler function 522
BT_HCI_EVT_LINK_KEY_REQUEST_HANDLER macro 33
bt_hci_evt_loopback_command_handler function 522
bt_hci_evt_master_link_key_complete_handler function 523
bt_hci_evt_max_slots_change_handler function 523
bt_hci_evt_mode_change_handler function 523
BT_HCI_EVT_MODE_CHANGE_HANDLER macro 33
bt_hci_evt_mode_change_t structure 685
bt_hci_evt_num_of_completed_packets_handler function 523
bt_hci_evt_page_scan_repet_mode_change_handler function 523
bt_hci_evt_pin_code_request_handler function 524
BT_HCI_EVT_PIN_CODE_REQUEST_HANDLER macro 34
bt_hci_evt_qos_setup_complete_handler function 524
bt_hci_evt_qos_violation_handler function 524
bt_hci_evt_read_clock_offset_complete_handler function 524
bt_hci_evt_read_rmt_ext_features_comp_handler function 524
bt_hci_evt_read_rmt_sup_features_comp_handler function 525
bt_hci_evt_read_rmt_version_info_comp_handler function 525
bt_hci_evt_remote_name_request_complete_handler function 525
BT_HCI_EVT_REMOTE_NAME_REQUEST_COMPLETE_HANDLER macro 34
bt_hci_evt_return_link_keys_handler function 525
bt_hci_evt_role_change_handler function 525
BT_HCI_EVT_ROLE_CHANGE_HANDLER macro 34
bt_hci_evt_role_change_t structure 685
bt_hci_evt_synch_connection_changed_handler function 526
bt_hci_evt_synch_connection_complete_handler function 526
BT_HCI_EVT_SYNCH_CONNECTION_COMPLETE_HANDLER macro 34
bt_hci_exit_park_state function 526
bt_hci_exit_sniff_mode macro 564
bt_hci_exit_sniff_mode_ex function 526
bt_hci_find_devices function 526
bt_hci_find_devices_ex function 527
bt_hci_free_command function 527
bt_hci_free_data_buffer function 527
bt_hci_get_evt_param_bdaddr function 527

bt_hci_get_evt_param_byte function 527
bt_hci_get_evt_param_devclass function 528
bt_hci_get_evt_param_hconn macro 564
bt_hci_get_evt_param_int function 528
bt_hci_get_evt_param_linkkey function 528
bt_hci_get_evt_param_long function 528
bt_hci_get_evt_param_uint function 528
bt_hci_get_evt_param_uint macro 565
bt_hci_get_evt_param_ulong function 529
bt_hci_get_evt_param_ulong macro 565
bt_hci_get_inquiry_response_tx_power_level function 562
bt_hci_get_last_cmd_status function 529
bt_hci_get_param_bdaddr function 529
bt_hci_get_param_byte function 529
bt_hci_get_param_hconn macro 565
bt_hci_get_param_int function 529
bt_hci_get_param_linkkey function 530
bt_hci_get_param_long function 530
bt_hci_get_recv_buffer function 530
bt_hci_get_recv_buffer_len function 530
bt_hci_get_send_buffer function 530
bt_hci_get_send_buffer_len function 531
bt_hci_hconn_p type 685
bt_hci_hconn_t type 685
bt_hci_init function 531
bt_hci_init_data_buffers function 531
bt_hci_init_data_queues function 531
bt_hci_init_ex function 562
bt_hci_init_linkkey_buffers function 531
bt_hci_inquiry_callback_fp type 685
bt_hci_inquiry_response_t structure 686
bt_hci_le_add_device_to_white_list function 532
bt_hci_le_advertising_add function 532
bt_hci_le_advertising_add_local_name function 532
bt_hci_le_advertising_device_id_add function 532
bt_hci_le_advertising_flags_add function 532
bt_hci_le_advertising_get_local_name function 533
bt_hci_le_advertising_report_t type 686
bt_hci_le_advertising_tx_power_level_add function 533
bt_hci_le_advertising_uuid128_add function 533
bt_hci_le_advertising_uuid16_add function 533
bt_hci_le_advertising_uuid32_add function 533
bt_hci_le_advertising_vendor_add function 534
bt_hci_le_allocate_set_advertising_data_command function 534
bt_hci_le_allocate_set_scan_response_data_command function 560
bt_hci_le_cancel_connect macro 702
bt_hci_le_cancel_connect_ex function 560
bt_hci_le_cancel_find_devices function 534
bt_hci_le_clear_white_list function 534
bt_hci_le_conn_state_t structure 686
bt_hci_le_connect_ex function 534
bt_hci_le_connect_parameters_t structure 686
bt_hci_le_ctrl_state_t structure 687
bt_hci_le_enable function 535
bt_hci_le_encrypt function 535
bt_hci_le_evt_connection_updated_t structure 687
bt_hci_le_evt_read_remote_used_features_completed_t structure 688
bt_hci_le_evt_read_support_params_t structure 688
bt_hci_le_find_devices function 535
bt_hci_le_get_connect_parameters function 535
bt_hci_le_ibeacon_add function 560
bt_hci_le_init function 535
bt_hci_le_ltk_negative_reply function 536
bt_hci_le_ltk_reply function 536
bt_hci_le_rand function 536
bt_hci_le_read_channel_map function 536
bt_hci_le_read_remote_used_features function 536
bt_hci_le_read_support function 537
bt_hci_le_read_white_list_size function 537
bt_hci_le_receiver_test function 537
bt_hci_le_remove_device_from_white_list function 537
bt_hci_le_scan_callback_fp type 688
bt_hci_le_set_adevrtising_enable macro 565
bt_hci_le_set_adevrtising_enable_ex function 537
bt_hci_le_set_advertising_parameters function 538
bt_hci_le_set_connect_parameters function 538
bt_hci_le_set_host_channel_classification function 538
bt_hci_le_set_random_address function 538
bt_hci_le_set_scan_enable function 538
bt_hci_le_set_scan_parameters function 539
bt_hci_le_start_encryption function 539
bt_hci_le_supported function 539
bt_hci_le_test_end function 539
bt_hci_le_transmitter_test function 539
bt_hci_le_update_connection function 540
bt_hci_le_write_support function 540
bt_hci_link_key_t structure 688
bt_hci_listen function 540
bt_hci_listen_sco function 540
bt_hci_listener_t type 689
bt_hci_param_eir_add function 540
bt_hci_param_eir_device_id_add function 541
bt_hci_param_eir_local_name_add function 541
bt_hci_param_eir_uuid128_add function 541
bt_hci_param_eir_uuid16_add function 541
bt_hci_param_eir_uuid32_add function 541
bt_hci_param_eir_vendor_add function 542
bt_hci_param_tx_power_level_add function 542
bt_hci_park_state function 542
bt_hci_read_inquiry_mode function 542
bt_hci_read_inquiry_mode_callback_fp type 689
bt_hci_read_inquiry_scan_activity function 542
bt_hci_read_inquiry_scan_activity_callback_fp type 689
bt_hci_read_inquiry_scan_type function 543
bt_hci_read_inquiry_scan_type_callback_fp type 689
bt_hci_read_page_scan_activity function 543
bt_hci_read_page_scan_activity_callback_fp type 689
bt_hci_read_page_scan_period_mode function 543
bt_hci_read_page_scan_period_mode_callback_fp type 690
bt_hci_read_page_scan_type function 543
bt_hci_read_page_scan_type_callback_fp type 690
bt_hci_read_page_timeout function 543
bt_hci_read_page_timeout_callback_fp type 690
bt_hci_register_listener function 544
bt_hci_reject_pin_code function 544
bt_hci_request_remote_name function 544

bt_hci_request_remote_name_callback_fp type 690
bt_hci_reset function 562
bt_hci_role_change macro 565
bt_hci_role_change_ex function 544
bt_hci_sco_read_data_callback_fp type 690
bt_hci_send_acl_data function 544
bt_hci_send_cmd function 545
bt_hci_send_linkkey function 545
bt_hci_send_pin_code macro 703
bt_hci_send_pin_code_ex function 562
bt_hci_send_sco_data function 545
bt_hci_set_encryption macro 566
bt_hci_set_encryption_ex function 545
bt_hci_set_event_listener function 545
bt_hci_set_incoming_connection_role function 546
bt_hci_set_scan function 546
bt_hci_set_scan_ex function 563
bt_hci_set_vendor_specific_event_handler function 560
bt_hci_sniff_mode macro 566
bt_hci_sniff_mode_ex function 546
bt_hci_sniff_subrating macro 566
bt_hci_sniff_subrating_ex function 546
bt_hci_start function 546
bt_hci_start_callback_fp type 691
bt_hci_start_no_init function 547
bt_hci_stop function 547
bt_hci_stop_callback_fp type 691
bt_hci_transport_recv_packet function 547
bt_hci_transport_recv_packet_callback_fp type 691
bt_hci_transport_send_cmd function 548
bt_hci_transport_send_data function 548
bt_hci_transport_send_packet function 548
bt_hci_transport_send_packet_callback_fp type 691
bt_hci_transport_set_transport function 548
bt_hci_unregister_listener function 548
bt_hci_write_default_link_policy_settings function 549
bt_hci_write_eir function 549
bt_hci_write_inquiry_mode function 549
bt_hci_write_inquiry_scan_activity function 549
bt_hci_write_inquiry_scan_type function 549
bt_hci_write_link_policy_settings function 563
bt_hci_write_local_name function 550
bt_hci_write_local_name_ex function 563
bt_hci_write_page_scan_activity function 550
bt_hci_write_page_scan_period_mode function 550
bt_hci_write_page_scan_type function 550
bt_hci_write_page_timeout function 550
bt_hcitr.h 1086
bt_hcitr_3wire_cancel_recv_packet function 206
bt_hcitr_3wire_init macro 566
bt_hcitr_3wire_init_ex function 561
bt_hcitr_3wire_reset macro 566
bt_hcitr_3wire_reset_ex function 561
bt_hcitr_3wire_start function 551
bt_hcitr_bcsp_init macro 567
bt_hcitr_bcsp_init_ex function 551
bt_hcitr_bcsp_reset macro 567
bt_hcitr_bcsp_reset_ex function 551
bt_hcitr_bcsp_start function 551
bt_hcitr_packet_init function 551
bt_hcitr_packet_reset function 552
bt_hcitr_packet_start function 552
bt_hcitr_tih4_init function 552
bt_hcitr_tih4_power_callback_fp type 691
bt_hcitr_tih4_power_event_e enumeration 692
bt_hcitr_tih4_reset function 552
bt_hcitr_tih4_start function 552
bt_hcitr_tih4_wake_up function 553
bt_hcitr_uart_init function 553
bt_hcitr_uart_reset function 553
bt_hcitr_uart_start function 553
bt_id type 218
BT_INCLUDE_IAP macro 20
BT_INCLUDE_IAP2 macro 20
BT_INCLUDE_RFCOMM macro 19
bt_init_buffer_mgr function 969
bt_int type 784
bt_int_cp type 218
bt_int_p type 218
bt_l2cap_alloc_cmd_buffer function 704
bt_l2cap_alloc_cmd_config_req function 704
bt_l2cap_alloc_cmd_config_res function 704
bt_l2cap_alloc_cmd_conn_param_update_req function 704
bt_l2cap_alloc_cmd_conn_param_update_res function 704
bt_l2cap_alloc_cmd_connection_req function 705
bt_l2cap_alloc_cmd_connection_res function 705
bt_l2cap_alloc_cmd_disconnection_req function 705
bt_l2cap_alloc_cmd_disconnection_res function 705
bt_l2cap_alloc_cmd_echo_req function 705
bt_l2cap_alloc_cmd_echo_res function 706
bt_l2cap_alloc_cmd_info_req function 706
bt_l2cap_alloc_cmd_info_res function 706
bt_l2cap_alloc_cmd_reject function 706
bt_l2cap_alloc_frame_buffer function 706
bt_l2cap_allocate_channel function 707
bt_l2cap_allocate_fixed_channel function 707
bt_l2cap_allocate_mgr function 707
bt_l2cap_allocate_psm function 707
bt_l2cap_cfg_option_p structure 765
bt_l2cap_cfg_option_t structure 766
bt_l2cap_channel_ext_t type 782
bt_l2cap_channel_t type 766
bt_l2cap_cmd_assembler_fp type 766
bt_l2cap_cmd_config_req_p structure 766
bt_l2cap_cmd_config_req_t structure 767
bt_l2cap_cmd_config_res_p structure 767
bt_l2cap_cmd_config_res_t structure 767
bt_l2cap_cmd_conn_param_update_req_t structure 768
bt_l2cap_cmd_conn_param_update_res_t structure 768
bt_l2cap_cmd_connection_req_p structure 768
bt_l2cap_cmd_connection_req_t structure 768
bt_l2cap_cmd_connection_res_p structure 769
bt_l2cap_cmd_connection_res_t structure 769
bt_l2cap_cmd_disconnection_req_p structure 769
bt_l2cap_cmd_disconnection_req_t structure 769
bt_l2cap_cmd_echo_req_p structure 770

bt_l2cap_cmd_echo_req_t structure 770
bt_l2cap_cmd_echo_res_p structure 770
bt_l2cap_cmd_echo_res_t structure 770
bt_l2cap_cmd_header_p structure 771
bt_l2cap_cmd_header_t structure 771
bt_l2cap_cmd_info_req_p structure 771
bt_l2cap_cmd_info_req_t structure 772
bt_l2cap_cmd_info_res_p structure 772
bt_l2cap_cmd_info_res_t structure 772
bt_l2cap_cmd_parser_fp type 772
bt_l2cap_cmd_reject_p structure 773
bt_l2cap_cmd_reject_param_t union 773
bt_l2cap_cmd_reject_t structure 773
bt_l2cap_command_t union 774
bt_l2cap_connect macro 728
bt_l2cap_connect_callback_fp type 774
bt_l2cap_connect_ext function 707
bt_l2cap_connect_fixed_channel function 708
bt_l2cap_connect_params_t type 783
bt_l2cap_disconnect function 708
bt_l2cap_disconnect_ex function 727
bt_l2cap_echo function 708
bt_l2cap_eretr_xmit_event_e enumeration 774
bt_l2cap_find_fixed_channel function 708
bt_l2cap_find_psm function 709
bt_l2cap_fixed_channel_t structure 774
bt_l2cap_frame_desc_t structure 775
bt_l2cap_free_channel function 727
bt_l2cap_free_cmd_buffer function 709
bt_l2cap_free_fixed_channel function 709
bt_l2cap_free_frame_buffer function 709
bt_l2cap_free_mgr function 709
bt_l2cap_free_psm function 710
bt_l2cap_get_channel function 710
bt_l2cap_get_channel_by_bdaddr_cid function 710
bt_l2cap_get_channel_by_hconn_cid function 710
bt_l2cap_get_channel_by_hconn_dest_cid function 710
bt_l2cap_get_channel_by_psm function 711
bt_l2cap_get_mgr function 711
bt_l2cap_hci_has_open_channels function 727
bt_l2cap_init function 711
bt_l2cap_init_channels function 711
bt_l2cap_init_cmd_buffers function 711
bt_l2cap_init_frame_buffers function 712
bt_l2cap_init_psms function 712
bt_l2cap_is_channel_open function 728
bt_l2cap_listen macro 729
bt_l2cap_listen_callback_fp type 775
bt_l2cap_listen_ext function 712
bt_l2cap_listen_fixed_channel function 712
bt_l2cap_mgr_p structure 775
bt_l2cap_mgr_t structure 776
bt_l2cap_option_flash_timeout_t structure 776
bt_l2cap_option_max_mtu_t structure 777
bt_l2cap_option_qos_t structure 777
bt_l2cap_option_rfc_t structure 777
bt_l2cap_option_unknown_t structure 777
bt_l2cap_packet_cmd_assembler function 712
bt_l2cap_packet_data_assembler function 713
bt_l2cap_packet_t structure 778
bt_l2cap_psm_t structure 778
bt_l2cap_read_data function 713
bt_l2cap_read_data_callback_fp type 778
bt_l2cap_reject function 713
bt_l2cap_request_handler_fp type 778
bt_l2cap_response_handler_fp type 779
bt_l2cap_send_cmd function 713
bt_l2cap_send_config function 728
bt_l2cap_send_data function 713
bt_l2cap_send_data_callback_fp type 779
bt_l2cap_send_smart_data function 728
bt_l2cap_state_changed_callback_fp type 779
bt_l2cap_test_enable_local_config macro 729
bt_l2cap_test_enable_remote_config macro 729
bt_l2cap_update_conn_parameters function 714
bt_l2cap_xmit_event_param_t structure 779
BT_LE_EVT_HANDLER macro 34
bt_le_evt_handler type 692
bt_linkkey_cp type 219
BT_LINKKEY_LENGTH macro 207
bt_linkkey_notification_t structure 219
bt_linkkey_p type 219
bt_linkkey_request_t structure 219
bt_linkkey_t structure 219
BT_LOG macro 207
bt_log.h 1087
bt_log_bdaddr function 197
BT_LOG_EX macro 207
bt_log_int function 197
BT_LOG_LEVEL_A2DP_PAKCET macro 964
BT_LOG_LEVEL_ALL macro 207
BT_LOG_LEVEL_DEBUG macro 208
BT_LOG_LEVEL_ERROR macro 208
BT_LOG_LEVEL_INFO macro 208
BT_LOG_LEVEL_MAX macro 19
BT_LOG_LEVEL_MIN macro 19
BT_LOG_LEVEL_OFF macro 208
bt_log_linkkey function 198
bt_log_memory function 198
bt_log_msg function 198
BT_LOGADDR macro 208
BT_LOGADDR_EX macro 209
BT_LOGINT macro 209
BT_LOGINT_EX macro 209
BT_LOGLINKKEY macro 209
BT_LOGLINKKEY_EX macro 209
BT_LOGMEMORY macro 210
BT_LOGMEMORY_EX macro 210
BT_LOGWRITE macro 210
bt_long type 784
bt_long_cp type 220
bt_long_p type 220
BT_MAKE_BDADDR macro 210
BT_MAKE_BDADDR_LE macro 210
bt_max macro 979
bt_media_packet_t type 340

bt_min macro 979
BT_NO macro 211
bt_oem.h 1088
bt_oem_assert function 198
bt_oem_config.h 1088
bt_oem_get_device_class function 198
bt_oem_get_device_name function 199
bt_oem_get_pin_code function 199
bt_oem_linkkey_notification function 199
bt_oem_linkkey_request function 199
bt_oem_log_write function 199
bt_oem_recv function 200
bt_oem_recv_callback_fp type 220
bt_oem_schedule_signals function 200
bt_oem_send function 200
bt_oem_send_callback_fp type 220
bt_oem_ssp_callback function 205
bt_oem_storage_get_capacity function 200
bt_oem_storage_read function 201
bt_oem_storage_start function 201
bt_oem_storage_stop function 201
bt_oem_storage_write function 201
bt_oem_timer_clear function 202
bt_oem_timer_set function 202
bt_packet_assembler_fp type 998
bt_packet_t structure 998
bt_private.h 1089
bt_q_add function 970
bt_q_contains function 206
bt_q_get function 970
bt_q_get_head function 970
bt_q_get_length function 970
bt_q_get_next function 970
bt_q_push function 971
bt_q_remove function 971
bt_q_remove_by_idx function 971
bt_queue_element_t structure 998
bt_rfcomm_allocate_dlc function 785
bt_rfcomm_allocate_session function 785
bt_rfcomm_cancel_listen function 800
bt_rfcomm_close_dlc function 785
bt_rfcomm_cmd_callback_fp type 826
bt_rfcomm_command_p structure 826
bt_rfcomm_command_t structure 826
bt_rfcomm_connect function 786
bt_rfcomm_ctl_msg_p structure 827
bt_rfcomm_ctl_msg_t structure 827
bt_rfcomm_dlc_p structure 827
bt_rfcomm_dlc_state_callback_fp type 828
bt_rfcomm_dlc_t structure 828
bt_rfcomm_find_dlc function 786
bt_rfcomm_free_dlc function 786
bt_rfcomm_free_session function 786
bt_rfcomm_get_frame_length function 800
bt_rfcomm_init function 787
bt_rfcomm_listen function 787
bt_rfcomm_mgr_t structure 828
bt_rfcomm_open_dlc function 787
bt_rfcomm_read_data_callback_fp type 828
bt_rfcomm_register_listener function 787
bt_rfcomm_send_credit function 788
bt_rfcomm_send_data function 788
bt_rfcomm_send_data_callback_fp type 829
bt_rfcomm_server_channel_t structure 829
bt_rfcomm_session_listener_t structure 829
bt_rfcomm_session_p structure 829
bt_rfcomm_session_t structure 830
bt_rfcomm_state_callback_fp type 830
bt_rfcomm_unregister_listener function 788
bt_sdp_client_callback_fp type 877
bt_sdp_client_evt_connected_t structure 876
bt_sdp_client_evt_disconnected_t structure 876
bt_sdp_data_element_cp type 877
bt_sdp_data_element_p type 877
bt_sdp_data_element_t structure 877
bt_sdp_de_to_uuid function 839
bt_sdp_found_attr_list_t structure 878
bt_sdp_packet_assembler function 839
bt_sdp_packet_t structure 878
bt_sdp_read_attribute function 839
bt_sdp_read_de_callback_ftp type 878
bt_sdp_request_service_attribute function 842
bt_sdp_request_service_search function 843
bt_sdp_sequence_cp type 879
bt_sdp_sequence_p type 879
bt_sdp_sequence_t structure 879
bt_sdp_serialization_state_p structure 879
bt_sdp_serialization_state_t structure 880
bt_sdp_server_attribute_t structure 880
bt_sdp_server_data_element_t structure 880
bt_sdp_server_record_t structure 880
bt_sdp_service_attribute_callback_fp type 881
bt_sdp_service_search_callback_fp type 881
bt_sdp_service_transaction_p structure 881
bt_sdp_service_transaction_t structure 881
bt_sdp_start function 839
bt_sdp_transaction_t structure 882
bt_signal.h 1090
bt_signal_handler_fp type 220
bt_signal_init function 205
bt_signal_process_pending function 202
bt_signal_register function 202
bt_signal_set function 202
bt_signal_t type 221
bt_signal_unregister function 203
bt_spp_allocate function 885
bt_spp_cancel_listen function 901
bt_spp_cancel_receive function 885
bt_spp_cancel_send function 885
bt_spp_clr_port_options function 886
bt_spp_connect function 886
bt_spp_deallocate function 886
bt_spp_disconnect function 886
bt_spp_find_server function 887
bt_spp_find_server_callback_fp type 894
bt_spp_find_server_ex function 901

bt_spp_get_frame_length function 887
bt_spp_get_hci_connection function 887
bt_spp_get_local_modem_status function 887
bt_spp_get_remote_address function 888
bt_spp_get_remote_modem_status function 888
bt_spp_init function 888
bt_spp_listen function 888
bt_spp_port_event_e enumeration 894
bt_spp_port_state_e enumeration 895
bt_spp_port_t type 896
bt_spp_read_local_oob_data_callback_fp type 903
bt_spp_receive function 889
bt_spp_receive_callback_fp type 896
bt_spp_send function 889
bt_spp_send_callback_fp type 896
bt_spp_send_status_e enumeration 896
bt_spp_set_dtr function 889
bt_spp_set_local_modem_status function 890
bt_spp_set_port_options function 890
bt_spp_set_rts function 890
bt_spp_state_callback_fp type 897
bt_sr_handle_p type 882
bt_sr_handle_t type 882
bt_ssp_evt_handler function 897
BT_SSP_EVT_HANDLER macro 35
bt_ssp_init function 897
bt_ssp_io_capability structure 903
bt_ssp_keypress_notification structure 903
bt_ssp_oob_data structure 903
bt_ssp_read_local_oob_data function 897
bt_ssp_send_keypress_notification function 898
bt_ssp_send_user_confirmation function 898
bt_ssp_send_user_passkey function 898
bt_ssp_set_io_capabilities function 898
bt_ssp_set_mode function 898
bt_ssp_set_oob_data function 899
bt_ssp_simple_pairing_complete structure 904
bt_ssp_user_confirmation_request structure 904
bt_ssp_user_passkey_notification structure 904
bt_ssp_user_passkey_request structure 904
bt_std.h 1090
bt_storage.h 1090
bt_storage_callback_fp type 221
bt_sys_callback_fp type 221
bt_sys_get_connectable function 203
bt_sys_get_discoverable function 203
bt_sys_get_l2cap_manager function 203
bt_sys_init function 203
bt_sys_init_ex function 205
bt_sys_set_modes function 204
bt_sys_start function 204
bt_system.h 1091
bt_timer.h 1092
BT_TIMER_3WIRE_T0 enumeration member 222
BT_TIMER_3WIRE_T1 enumeration member 222
BT_TIMER_ATT enumeration member 222
BT_TIMER_ATT_CLIENT enumeration member 222
BT_TIMER_AVRCP enumeration member 222
bt_timer_callback_fp type 221
BT_TIMER_HCI enumeration member 222
BT_TIMER_HCRP enumeration member 222
BT_TIMER_HFP_AG enumeration member 222
BT_TIMER_HSP_AG enumeration member 222
BT_TIMER_IAP enumeration member 222
BT_TIMER_IAP2 enumeration member 222
bt_timer_id enumeration 222
BT_TIMER_L2CAP enumeration member 222
BT_TIMER_MAX enumeration member 222
BT_TIMER_RFCOMM enumeration member 222
BT_TIMER_SMP enumeration member 222
BT_TIMER_TEST enumeration member 222
BT_TIMER_WAKEUP_ACK enumeration member 222
BT_TRUE macro 211
bt_types.h 1093
bt_uint type 784
bt_uint_cp type 222
bt_uint_p type 222
bt_ulong type 785
bt_ulong_cp type 222
bt_ulong_p type 223
bt_uuid_cp type 223
bt_uuid_p type 223
bt_uuid_t structure 223
bt_uuid16 type 223
bt_uuid32 type 224
bt_vcard_evt_prop_param_t structure 998
bt_vcard_evt_prop_t structure 999
bt_vcard_parse_fragment function 971
bt_vcard_parser_callback_fp type 999
bt_vcard_parser_reset function 971
bt_vcard_parser_t structure 999
bt_xml_evt_attribute_t structure 999
bt_xml_evt_attribute_value_t structure 1000
bt_xml_evt_tag_started_t structure 1000
bt_xml_parse_fragment function 972
bt_xml_parser_callback_pf type 1000
bt_xml_parser_reset function 972
bt_xml_parser_t structure 1000
BT_YES macro 211
btx_csr_alloc_bccmd_getreq function 906
btx_csr_alloc_bccmd_setreq function 906
btx_csr_autobaud function 907
btx_csr_autobaud_buffer_t type 945
btx_csr_autobaud_callback_fp type 946
btx_csr_bc7_sel_host_interface_h4 function 907
btx_csr_bccmd_callback_fp type 965
btx_csr_bccmd_header_t structure 946
btx_csr_bccmd_listener_t type 965
btx_csr_cached_temperature_t structure 946
btx_csr_create_operator_c_t structure 965
btx_csr_enable_tx function 907
btx_csr_exec_hq_script function 925
btx_csr_exec_hq_script_buffer_t type 966
btx_csr_exec_hq_script_callback_fp type 966
btx_csr_exec_script function 907
btx_csr_exec_script_buffer_t type 946

btx_csr_exec_script_callback_fp type 947
btx_csr_get_cached_temperature function 908
btx_csr_get_ps_var function 908
btx_csr_get_ps_var_callback_fp type 947
btx_csr_get_ps_var_ex function 908
btx_csr_get_rssi_acl function 908
btx_csr_get_script__dsp_script__PB_109_DSP_rev8 function 919
btx_csr_get_script__PB_101_CSR8811_CSP28_UART function 919
btx_csr_get_script__PB_109_CSR8811_REV16 function 919
btx_csr_get_script__PB_173_CSR8X11_REV1 function 919
btx_csr_get_script__PB_27_R20_BC6ROM_A04 function 908
btx_csr_get_script__PB_90_REV6 function 909
btx_csr_get_script_fp type 966
btx_csr_get_var function 909
btx_csr_get_var_callback_fp type 947
btx_csr_init function 920
btx_csr_init_hq_script function 920
btx_csr_patch_controller function 920
btx_csr_pio_direction_mask_t structure 947
btx_csr_pio_protection_mask_t structure 947
btx_csr_pio_t structure 948
btx_csr_register_bccmd_listener function 920
btx_csr_rssi_acl_t structure 948
btx_csr_script_t structure 948
btx_csr_send_dsp_config_data function 921
btx_csr_send_next_hq_script_packet function 921
btx_csr_set_ps_var function 909
btx_csr_set_ps_var_ex function 909
btx_csr_set_ps_vars function 909
btx_csr_set_ps_vars_buffer_t type 948
btx_csr_set_ps_vars_callback_fp type 949
btx_csr_set_ps_vars_ex function 910
btx_csr_set_var function 910
btx_csr_set_var_callback_fp type 949
btx_csr_strm_connect_t structure 966
btx_csr_strm_get_sink_t structure 949
btx_csr_strm_get_source_t structure 949
btx_csr_unregister_bccmd_listener function 921
btx_csr_var_t union 949
btx_csr_warm_reset function 910
btx_csr_warm_reset_ex function 910
BTX_TI_A3DP_ROLE_SINK macro 951
BTX_TI_A3DP_ROLE_SOURCE macro 951
btx_ti_a3dp_sink_close_stream function 921
btx_ti_a3dp_sink_codec_config function 921
btx_ti_a3dp_sink_open_stream function 922
btx_ti_a3dp_sink_start_stream function 922
btx_ti_a3dp_sink_stop_stream function 922
btx_ti_avpr_debug function 922
BTX_TI_AVPR_DISABLE macro 951
BTX_TI_AVPR_DO_NOT_LOAD_A3DP_CODE macro 951
btx_ti_avpr_enable function 922
BTX_TI_AVPR_ENABLE macro 952
BTX_TI_AVPR_LOAD_A3DP_CODE macro 952
btx_ti_codec_config_t structure 966
btx_ti_completion_callback_fp type 950
btx_ti_drpb_enable_rf_calibration function 911
btx_ti_drpb_set_power_vector function 911
btx_ti_drpb_tester_con_tx function 911
btx_ti_enable_deep_sleep function 911
btx_ti_enable_fast_clock_crystal function 912
btx_ti_enable_low_power_scan function 912
btx_ti_enable_low_power_scan_default function 912
btx_ti_exec_script function 912
btx_ti_exec_script_buffer_t type 950
btx_ti_exec_script_oem function 912
btx_ti_exec_script_oem_buffer_t type 950
btx_ti_exec_script_oem_callback_fp type 950
BTX_TI_EXEC_SCRIPT_OEM_RX_BUFFER_SIZE macro 940
btx_ti_get_script__BL6450_2_0_BT_Service_Pack_2_36 function 913
btx_ti_get_script__BL6450_2_0_BT_Service_Pack_2_44 function 913
btx_ti_get_script__BL6450L_BT_Service_Pack_2_10 function 913
btx_ti_get_script__BL6450L_BT_Service_Pack_2_10_BLE_AddOn function 913
btx_ti_get_script__BL6450L_BT_Service_Pack_2_12 function 913
btx_ti_get_script__BL6450L_BT_Service_Pack_2_12_BLE_AddOn function 914
btx_ti_get_script__BL6450L_BT_Service_Pack_2_14 function 923
btx_ti_get_script__BL6450L_BT_Service_Pack_2_14_BLE_AddOn function 923
btx_ti_get_script__BL6450L_BT_Service_Pack_2_7 function 914
btx_ti_get_script__BL6450L_BT_Service_Pack_2_8_AVPR_AddOn function 914
btx_ti_get_script__BL6450L_BT_Service_Pack_2_8_BLE_AddOn function 914
btx_ti_get_script__BL6450L_BT_Service_Pack_2_8_Short function 914
btx_ti_get_script__BL6450x_BT_Service_Pack_2_7_AVPR_AddOn function 915
btx_ti_get_script__BL6450x_BT_Service_Pack_2_7_BLE_AddOn function 915
btx_ti_get_script__CC2560_ServicePack macro 940
btx_ti_get_script__CC2564_BLE_Init macro 940
btx_ti_get_script__CC2564_ServicePack macro 940
btx_ti_get_script__CC2564B_AVPR_Init macro 952
btx_ti_get_script__CC2564B_BLE_Init macro 941
btx_ti_get_script__CC2564B_BT_Service_Pack_0_1 function 915
btx_ti_get_script__CC2564B_BT_Service_Pack_0_1_BLE_AddOn function 915
btx_ti_get_script__CC2564B_BT_Service_Pack_0_2 function 915
btx_ti_get_script__CC2564B_BT_Service_Pack_0_2_BLE_AddOn function 916
btx_ti_get_script__CC2564B_BT_Service_Pack_1_0 function 923
btx_ti_get_script__CC2564B_BT_Service_Pack_1_0_BLE_AddOn function 923
btx_ti_get_script__CC2564B_BT_Service_Pack_1_1 function 923
btx_ti_get_script__CC2564B_BT_Service_Pack_1_1_BLE_AddOn function 924
btx_ti_get_script__CC2564B_BT_Service_Pack_1_2 function 924
btx_ti_get_script__CC2564B_BT_Service_Pack_1_2_AVPR_AddOn function 924
btx_ti_get_script__CC2564B_BT_Service_Pack_1_2_BLE_AddOn function 924
btx_ti_get_script__CC2564B_ServicePack macro 941
btx_ti_get_script__WL127xL_BT_Service_Pack_2_3 function 916
btx_ti_get_script__WL127xL_BT_Service_Pack_2_3_AVPR_AddOn function 916
btx_ti_get_script__WL127xL_BT_Service_Pack_2_3_BLE_AddOn function 916
btx_ti_get_script__WL127xL_BT_Service_Pack_2_3_DC2DC_AddOn function 916

btx_ti_get_script__WL127xL_BT_Service_Pack_2_4 function 917
btx_ti_get_script__WL127xL_BT_Service_Pack_2_4_AVPR_AddOn function 917
btx_ti_get_script__WL127xL_BT_Service_Pack_2_4_BLE_AddOn function 917
btx_ti_get_script__WL127xL_BT_Service_Pack_2_4_DC2DC_AddOn function 917
btx_ti_get_script__XWL1271L1_BT_ServicePack_1_3 function 917
btx_ti_get_script__XWL1271L1_BT_ServicePack_1_3_BLE_Init function 918
btx_ti_init_ble_controller function 918
btx_ti_init_controller function 918
btx_ti_le_enable function 924
BTX_TI_MODULATION_SCHEME_8_DPSK macro 941
BTX_TI_MODULATION_SCHEME_CW macro 941
BTX_TI_MODULATION_SCHEME_GFSK macro 941
BTX_TI_MODULATION_SCHEME_P4_DPSK macro 942
BTX_TI_MODULATION_TYPE_EDR2 macro 942
BTX_TI_MODULATION_TYPE_EDR3 macro 942
BTX_TI_MODULATION_TYPE_GFSK macro 942
BTX_TI_SBC_ALLOCATION_METHOD_LOUDNESS macro 952
BTX_TI_SBC_ALLOCATION_METHOD_SNR macro 952
BTX_TI_SBC_BLOCKS_12 macro 953
BTX_TI_SBC_BLOCKS_16 macro 953
BTX_TI_SBC_BLOCKS_4 macro 953
BTX_TI_SBC_BLOCKS_8 macro 953
BTX_TI_SBC_CHANNEL_MODE_DUAL_CHANNEL macro 953
BTX_TI_SBC_CHANNEL_MODE_JOINT_STEREO macro 954
BTX_TI_SBC_CHANNEL_MODE_MONO macro 954
BTX_TI_SBC_CHANNEL_MODE_STEREO macro 954
BTX_TI_SBC_SAMPLE_FREQUENCY_16000 macro 954
BTX_TI_SBC_SAMPLE_FREQUENCY_32000 macro 954
BTX_TI_SBC_SAMPLE_FREQUENCY_44100 macro 955
BTX_TI_SBC_SAMPLE_FREQUENCY_48000 macro 955
btx_ti_script_t structure 951
btx_ti_set_afh_mode function 925
btx_ti_set_uart_baud_rate function 918
BTX_TI_TEST_PATTERN_ALL_0 macro 942
BTX_TI_TEST_PATTERN_ALL_1 macro 943
BTX_TI_TEST_PATTERN_F0F0 macro 943
BTX_TI_TEST_PATTERN_FF00 macro 943
BTX_TI_TEST_PATTERN_PN15 macro 943
BTX_TI_TEST_PATTERN_PN9 macro 943
BTX_TI_TEST_PATTERN_USER macro 944
BTX_TI_TEST_PATTERN_Z0Z0 macro 944
btx_ti_write_bdaddr function 918
btx_ti_write_codec_config function 925
btx_ti_write_hardware_register function 919
buffer.h 1094
BUFFER_HDR_LEN macro 980
BUFFER_STATE_FREE macro 980
BUFFER_STATE_USED macro 980
bufferutils.h 1094
Building the Library 55
    PIC32 Bluetooth Stack Library 55
BYTE_SIZE macro 784

C

CDS_LAST_DEVICE_ADDR macro 211
CDS_SIGNATURE macro 211
CDS_SIGNATURE_ADDR macro 212
channel.h 1095
CHANNEL_SIGNAL_CMD_DISCONNECT_FIXED macro 729
check_fcs function 788
chmanager.h 1097
CID_ATT macro 729
CID_LE_SIG macro 730
CID_MAX macro 730
CID_MAX_FIXED macro 730
CID_NULL macro 730
CID_RECV macro 730
CID_SIG macro 731
CID_SM macro 731
cmd_disconnection_res structure 780
cmdbuffer.h 1097
CMODE_BASIC macro 731
CMODE_ERETR macro 731
CMODE_FLOW macro 731
CMODE_RETR macro 732
CMODE_STRM macro 732
COD_MAJOR_AUDIO macro 567
COD_MAJOR_COMPUTER macro 567
COD_MAJOR_HEALTH macro 567
COD_MAJOR_IMAGING macro 568
COD_MAJOR_MISC macro 568
COD_MAJOR_NET_ACCESS_POINT macro 568
COD_MAJOR_PERIPHERAL macro 568
COD_MAJOR_PHONE macro 568
COD_MAJOR_TOY macro 569
COD_MAJOR_UNCATEGORIZED macro 569
COD_MAJOR_WEARABLE macro 569
COD_MINOR_AV_CAMCORDER macro 569
COD_MINOR_AV_CAR_AUDIO macro 569
COD_MINOR_AV_GAMING macro 570
COD_MINOR_AV_HANDSFREE macro 570
COD_MINOR_AV_HEADPHONES macro 570
COD_MINOR_AV_HEADSET macro 570
COD_MINOR_AV_HIFI_AUDIO macro 570
COD_MINOR_AV_LOUDSPEAKER macro 571
COD_MINOR_AV_MICROPHONE macro 571
COD_MINOR_AV_PORTABLE_AUDIO macro 571
COD_MINOR_AV_RESERVED macro 571
COD_MINOR_AV_RESERVERD2 macro 571
COD_MINOR_AV_SET_TOP_BOX macro 572
COD_MINOR_AV_UNCATEGORIZED macro 572
COD_MINOR_AV_VCR macro 572
COD_MINOR_AV_VIDE_DISPLAY_AND_LOUDSPEAKER macro 572
COD_MINOR_AV_VIDEO_CAMERA macro 572
COD_MINOR_AV_VIDEO_CONFERENCING macro 573
COD_MINOR_AV_VIDEO_MONITOR macro 573
COD_MINOR_COMPUTER_DESKTOP macro 573
COD_MINOR_COMPUTER_HANDHELD macro 573
COD_MINOR_COMPUTER_LAPTOP macro 573
COD_MINOR_COMPUTER_PALMSIZED macro 574
COD_MINOR_COMPUTER_SERVER macro 574
COD_MINOR_COMPUTER_UNCATEGORIZED macro 574
COD_MINOR_COMPUTER_WEARABLE macro 574

COD_MINOR_HEALTH_BPM macro 574
COD_MINOR_HEALTH_DATA_DISPLAY macro 575
COD_MINOR_HEALTH_GLUCOSE_METER macro 575
COD_MINOR_HEALTH_HEART_MONITOR macro 575
COD_MINOR_HEALTH_PULSE_OXIMETER macro 575
COD_MINOR_HEALTH_THERMOMETER macro 575
COD_MINOR_HEALTH_UNCATEGORIZED macro 576
COD_MINOR_HEALTH_WEIGHING_SCALE macro 576
COD_MINOR_IMAGING_CAMERA macro 576
COD_MINOR_IMAGING_DISPLAY macro 576
COD_MINOR_IMAGING_PRINTER macro 576
COD_MINOR_IMAGING_SCANNER macro 577
COD_MINOR_IMAGING_UNCATEGORIZED macro 577
COD_MINOR_LAN_01_17 macro 577
COD_MINOR_LAN_17_33 macro 577
COD_MINOR_LAN_33_50 macro 577
COD_MINOR_LAN_50_67 macro 578
COD_MINOR_LAN_67_83 macro 578
COD_MINOR_LAN_83_99 macro 578
COD_MINOR_LAN_FULLY_AVAILABLE macro 578
COD_MINOR_LAN_NO_SERVICE macro 578
COD_MINOR_LAN_UNCATEGORIZED macro 579
COD_MINOR_PERIPHERAL_CARD_READER macro 579
COD_MINOR_PERIPHERAL_COMBO macro 579
COD_MINOR_PERIPHERAL_DIGITIZER macro 579
COD_MINOR_PERIPHERAL_GAMEPAD macro 579
COD_MINOR_PERIPHERAL_JOYSTICK macro 580
COD_MINOR_PERIPHERAL_KEYBOARD macro 580
COD_MINOR_PERIPHERAL_MOUSE macro 580
COD_MINOR_PERIPHERAL_OTHER macro 580
COD_MINOR_PERIPHERAL_REMOTE macro 580
COD_MINOR_PERIPHERAL_SENSING macro 581
COD_MINOR_PERIPHERAL_UNCATEGORIZED macro 581
COD_MINOR_PHONE_CELLULAR macro 581
COD_MINOR_PHONE_CORDLESS macro 581
COD_MINOR_PHONE_ISDN macro 581
COD_MINOR_PHONE_SMART macro 582
COD_MINOR_PHONE_UNCATEGORIZED macro 582
COD_MINOR_PHONE_WIREDMODEM macro 582
COD_MINOR_TOY_CONTROLLER macro 582
COD_MINOR_TOY_DOLL macro 582
COD_MINOR_TOY_GAME macro 583
COD_MINOR_TOY_ROBOT macro 583
COD_MINOR_TOY_UNCATEGORIZED macro 583
COD_MINOR_TOY_VEHICLE macro 583
COD_MINOR_WEARABLE_GLASSES macro 583
COD_MINOR_WEARABLE_HELMET macro 584
COD_MINOR_WEARABLE_JACKET macro 584
COD_MINOR_WEARABLE_PAGER macro 584
COD_MINOR_WEARABLE_UNCATEGORIZED macro 584
COD_MINOR_WEARABLE_WATCH macro 584
command.h 1098
config.h 1101
Configuring the Library 7
    PIC32 Bluetooth Stack Library 7
COS_AUDIO macro 585
COS_CAPTURING macro 585
COS_INFORMATION macro 585
COS_LIMITED_DISCOVERABLE_MODE macro 585
COS_NETWORKING macro 585
COS_OBJECTTRANSFER macro 586
COS_POSITIONING macro 586
COS_RENDERING macro 586
COS_TELEPHONY macro 586
csr.h 1101
CSR_I2S_STREAM_CFG_KEY_AUDIO_ATTEN macro 955
CSR_I2S_STREAM_CFG_KEY_AUDIO_ATTEN_ENABLE macro 955
CSR_I2S_STREAM_CFG_KEY_BITS_PER_SAMPLE macro 955
CSR_I2S_STREAM_CFG_KEY_CROP_ENABLE macro 956
CSR_I2S_STREAM_CFG_KEY_JUSTIFY_DELAY macro 956
CSR_I2S_STREAM_CFG_KEY_JUSTIFY_FORMAT macro 956
CSR_I2S_STREAM_CFG_KEY_JUSTIFY_RESOLUTION macro 956
CSR_I2S_STREAM_CFG_KEY_MASTER macro 956
CSR_I2S_STREAM_CFG_KEY_MCLK macro 957
CSR_I2S_STREAM_CFG_KEY_POLARITY macro 957
CSR_I2S_STREAM_CFG_KEY_RX_START_SAMPLE macro 957
CSR_I2S_STREAM_CFG_KEY_SYNC_RATE macro 957
CSR_I2S_STREAM_CFG_KEY_TX_START_SAMPLE macro 957
CSR_MAX_HQ_PACKET_LEN macro 958
CSR_MSG_TYPE_GETREQ macro 958
CSR_MSG_TYPE_GETRESP macro 958
CSR_MSG_TYPE_SETREQ macro 958
CSR_PCM_STREAM_CFG_KEY_LSB_FIRST macro 958
CSR_PCM_STREAM_CFG_KEY_MANCHESTER macro 959
CSR_PCM_STREAM_CFG_KEY_MANCHESTER_SLAVE macro 959
CSR_PCM_STREAM_CFG_KEY_MASTER macro 959
CSR_PCM_STREAM_CFG_KEY_MCLK macro 959
CSR_PCM_STREAM_CFG_KEY_SHORT_SYNC macro 959
CSR_PCM_STREAM_CFG_KEY_SIGN_EXTEND macro 960
CSR_PCM_STREAM_CFG_KEY_SLOT_COUNT macro 960
CSR_PCM_STREAM_CFG_KEY_SYNC_RATE macro 960
CSR_PCM_STREAM_CFG_KEY_TX_TRISTATE macro 960
CSR_SNK_ADC macro 925
CSR_SNK_FASTPIPE macro 925
CSR_SNK_FM macro 926
CSR_SNK_I2S macro 926
CSR_SNK_L2CAP macro 926
CSR_SNK_PCM macro 926
CSR_SNK_SCO macro 926
CSR_SNK_SPDIF macro 927
CSR_SRC_ADC macro 927
CSR_SRC_FASTPIPE macro 927
CSR_SRC_FM macro 927
CSR_SRC_I2S macro 927
CSR_SRC_L2CAP macro 928
CSR_SRC_MIC macro 928
CSR_SRC_PCM macro 928
CSR_SRC_SCO macro 928
CSR_SRC_SPDIF macro 928
CSR_VARID_CACHED_TEMPERATURE macro 929
CSR_VARID_CAPABILITY_DOWNLOAD_INDICATION macro 960
CSR_VARID_CREATE_OPERATOR_C macro 961
CSR_VARID_CREATE_OPERATOR_P macro 961
CSR_VARID_DESTROY_OPERATORS macro 961
CSR_VARID_DSPMANAGER_CONFIG_REQUEST macro 961
CSR_VARID_ENABLE_SCO_STREAMS macro 929

CSR_VARID_MAP_SCO_AUDIO macro 929
CSR_VARID_MESSAGE_FROM_OPERATOR macro 961
CSR_VARID_PIO macro 929
CSR_VARID_PIO_DIRECTION_MASK macro 929
CSR_VARID_PIO_PROTECT_MASK macro 930
CSR_VARID_RSSI_ACL macro 930
CSR_VARID_START_OPERATORS macro 962
CSR_VARID_STOP_OPERATORS macro 962
CSR_VARID_STREAM_CLOSE_SINK macro 930
CSR_VARID_STREAM_CLOSE_SOURCE macro 930
CSR_VARID_STREAM_CONFIGURE macro 930
CSR_VARID_STREAM_CONNECT macro 931
CSR_VARID_STREAM_DRAINED_NOTIFICATION macro 962
CSR_VARID_STREAM_GET_SINK macro 931
CSR_VARID_STREAM_GET_SOURCE macro 931
CSR_VARID_STREAM_TRANSFORM_DISCONNECT macro 962
CSTATE_CLOSED macro 732
CSTATE_FREE macro 732
CSTATE_OPEN macro 732
CSTATE_WAIT_CONFIG macro 733
CSTATE_WAIT_CONFIG_REQ macro 733
CSTATE_WAIT_CONFIG_RSP macro 733
CSTATE_WAIT_CONNECT macro 733
CSTATE_WAIT_CONNECT_RSP macro 733
CSTATE_WAIT_DISCONNECT macro 734
CTYPE_CL macro 734
CTYPE_CO macro 734

D

DCRH macro 965
DE_BOOL macro 844
DE_INT macro 844
DE_STRING macro 845
DE_STRING2 macro 845
DE_UINT macro 845
DE_UINT16 macro 845
DE_URL macro 846
DE_UUID128 macro 846
DE_UUID16 macro 846
DE_UUID32 macro 846

E

END_DE_SEQUENCE macro 847

F

FALSE macro 212
Files 1002
    PIC32 Bluetooth Stack Library 1002
frame_buffer.h 1106
FTP_ALLOCATE_BUFFERS function 15

G

gap.h 1107
GATT_CLIENT_ALLOCATE_BUFFERS function 16
GET_F_BIT macro 734
GET_FRAME_TYPE macro 734
GET_P_BIT macro 735
GET_REQ_SEQ macro 735
GET_S_FUNCTION macro 735
GET_SAR macro 735
GET_TX_SEQ macro 735
GETRESP_BAD_REQ macro 931
GETRESP_ERROR macro 931
GETRESP_NO_ACCESS macro 932
GETRESP_NO_SUCH_VARID macro 932
GETRESP_NO_VALUE macro 932
GETRESP_OK macro 932
GETRESP_PERMISSION_DENIED macro 932
GETRESP_READ_ONLY macro 933
GETRESP_TOO_BIG macro 933
GETRESP_WRITE_ONLY macro 933

H

H macro 933
hci.h 1107
HCI_ACCEPT_CONNECTION_REQUEST macro 586
HCI_ACCEPT_SYNCH_CONNECTION_REQUEST macro 587
HCI_ACL_DATA_HEADER_LEN macro 587
HCI_ALLOCATE_BUFFERS function 16
HCI_ALLOCATE_BUFFERS_FUNCTION macro 35
HCI_ALLOCATE_BUFFERS_RAM_SIZE_VAR macro 35
HCI_ALLOCATE_BUFFERS_VARS macro 35
hci_allocate_conn_state function 553
HCI_AUTHENTICATION_REQUESTED macro 587
HCI_BB_PACKET_TYPE_DH1 macro 587
HCI_BB_PACKET_TYPE_DH3 macro 587
HCI_BB_PACKET_TYPE_DH5 macro 588
HCI_BB_PACKET_TYPE_DM1 macro 588
HCI_BB_PACKET_TYPE_DM3 macro 588
HCI_BB_PACKET_TYPE_DM5 macro 588
HCI_BB_PACKET_TYPE_NO_2_DH1 macro 588
HCI_BB_PACKET_TYPE_NO_2_DH3 macro 589
HCI_BB_PACKET_TYPE_NO_2_DH5 macro 589
HCI_BB_PACKET_TYPE_NO_3_DH1 macro 589
HCI_BB_PACKET_TYPE_NO_3_DH3 macro 589
HCI_BB_PACKET_TYPE_NO_3_DH5 macro 589
HCI_BDADDR_LEN macro 212
HCI_C2H_BROADCAST_NOT_PARCKED_SLAVE macro 590
HCI_C2H_BROADCAST_P2P macro 590
HCI_C2H_BROADCAST_PARCKED_SLAVE macro 590
HCI_C2H_BROADCAST_RESERVED macro 590
HCI_CHANGE_CONNECTION_LINK_KEY macro 590
HCI_CHANGE_CONNECTION_PACKET_TYPE macro 591
hci_check_aux_info function 554
hci_cmd_buffer.h 1121
HCI_CMD_HEADER_LEN macro 591
hci_cmd_queue.h 1121
HCI_CMD_STATUS_BEING_SENT macro 591
HCI_CMD_STATUS_PENDING macro 591
HCI_CMD_STATUS_WAITING_RESPONSE macro 591
hci_config.h 1121
HCI_CONFIG_BECOME_MASTER macro 592
HCI_CONFIG_ENABLE_AUTHENTICATION macro 592
HCI_CONFIG_ENABLE_ENCRYPTION macro 592
hci_config_event_handlers.h 1122
HCI_CONN_ROLE_MASTER macro 592
HCI_CONN_ROLE_SLAVE macro 592

hci_conn_state.h 1123
HCI_CONN_STATE_AUTHENTICATING macro 593
HCI_CONN_STATE_CLOSED macro 593
HCI_CONN_STATE_OPEN macro 593
HCI_CONN_TYPE_ACL macro 593
HCI_CONN_TYPE_ESCO macro 593
HCI_CONN_TYPE_SCO macro 594
HCI_CONNECTABLE macro 594
HCI_CONTROLLER macro 594
hci_cq_find_by_bdaddr_and_opcode function 554
hci_cq_find_by_hconn function 554
hci_cq_find_by_hconn_and_opcode function 554
hci_cq_find_by_opcode function 554
HCI_CREATE_CONNECTION macro 594
HCI_CREATE_CONNECTION_CANCEL macro 594
HCI_CREATE_NEW_UNIT_KEY macro 595
HCI_CTRL_LISTENER_ACL_DATA macro 595
HCI_CTRL_LISTENER_EVENT macro 595
HCI_CTRL_LISTENER_SCO_DATA macro 595
hci_ctrl_state.h 1123
HCI_CTRL_STATE_CLOSED macro 595
HCI_CTRL_STATE_CONNECTABLE macro 596
HCI_CTRL_STATE_DISCOVERABLE macro 596
HCI_CTRL_STATE_INIT macro 596
HCI_CTRL_STATE_READY macro 596
HCI_CTRL_STATE_SLEEP macro 596
HCI_CTRL_STATE_WAKING_UP macro 597
hci_data_buffer.h 1124
HCI_DATA_BUFFER_STATE_FREE macro 597
HCI_DATA_BUFFER_STATE_USED macro 597
hci_data_queue.h 1125
HCI_DATA_STATUS_BEING_SENT macro 597
HCI_DATA_STATUS_PENDING macro 597
HCI_DECLARE_LE_CONN_STATES macro 36
HCI_DECLARE_LE_CTRL_STATE macro 36
HCI_DEFAULT_ACL_CONFIG macro 598
HCI_DELETE_STORED_LINK_KEY macro 598
HCI_DISCONNECT macro 598
HCI_DISCOVERABLE macro 598
HCI_DISCOVERABLE_MODE_GENERAL macro 694
HCI_DISCOVERABLE_MODE_LIMITED macro 695
hci_eir.h 1125
HCI_EIR_FEC_NOT_REQUIRED macro 598
HCI_EIR_FEC_REQUIRED macro 599
HCI_EIR_TYPE_3D_Information_Data macro 695
HCI_EIR_TYPE_ADVERTISING_INTERVAL macro 695
HCI_EIR_TYPE_APPEARANCE macro 695
HCI_EIR_TYPE_DEVICE_ID macro 599
HCI_EIR_TYPE_FLAGS macro 599
HCI_EIR_TYPE_LE_BLUETOOTH_DEVICE_ADDRESS macro 695
HCI_EIR_TYPE_LE_ROLE macro 696
HCI_EIR_TYPE_LE_SECURE_CONNECTIONS_CONFIRMATION_VALUE macro 696
HCI_EIR_TYPE_LE_SECURE_CONNECTIONS_RANDOM_VALUE macro 696
HCI_EIR_TYPE_LOCAL_NAME_COMPLETE macro 599
HCI_EIR_TYPE_LOCAL_NAME_SHORTENED macro 599
HCI_EIR_TYPE_MANUFACTURER_SPECIFIC macro 600
HCI_EIR_TYPE_OOB_COD macro 600
HCI_EIR_TYPE_OOB_HASH macro 600
HCI_EIR_TYPE_OOB_RANDOMIZER macro 600
HCI_EIR_TYPE_PUBLIC_TARGET_ADDRESS macro 696
HCI_EIR_TYPE_RANDOM_TARGET_ADDRESS macro 696
HCI_EIR_TYPE_SERVICE_DATA macro 697
HCI_EIR_TYPE_SERVICE_DATA_UUID128 macro 697
HCI_EIR_TYPE_SERVICE_DATA_UUID16 macro 697
HCI_EIR_TYPE_SERVICE_DATA_UUID32 macro 697
HCI_EIR_TYPE_SIMPLE_PAIRING_HASH_C_256 macro 697
HCI_EIR_TYPE_SIMPLE_PAIRING_RANDOMIZER_R_256 macro 698
HCI_EIR_TYPE_SLAVE_CONN_INTERVAL_RANGE macro 698
HCI_EIR_TYPE_SM_OOB_FLAGS macro 698
HCI_EIR_TYPE_SM_TK_VALUE macro 698
HCI_EIR_TYPE_SOLICITATION_UUID128_LIST macro 698
HCI_EIR_TYPE_SOLICITATION_UUID16_LIST macro 699
HCI_EIR_TYPE_SOLICITATION_UUID32_LIST macro 699
HCI_EIR_TYPE_TX_POWER_LEVEL macro 600
HCI_EIR_TYPE_UUID128_LIST_COMPLETE macro 601
HCI_EIR_TYPE_UUID128_LIST_MORE_AVAILABLE macro 601
HCI_EIR_TYPE_UUID16_LIST_COMPLETE macro 601
HCI_EIR_TYPE_UUID16_LIST_MORE_AVAILABLE macro 601
HCI_EIR_TYPE_UUID32_LIST_COMPLETE macro 601
HCI_EIR_TYPE_UUID32_LIST_MORE_AVAILABLE macro 602
HCI_ENABLE_CTRL_TO_HOST_FLOW_CONTROL macro 36
HCI_ENABLE_DEVICE_UNDER_TEST_MODE macro 602
HCI_ENCRYPTION_OFF macro 602
HCI_ENCRYPTION_ON macro 602
HCI_ENHANCED_FLUSH macro 602
HCI_ERR_ACL_CONN_ALREADY_EXISTS macro 603
HCI_ERR_AUTHENTICATION_FAILURE macro 603
HCI_ERR_CONN_REJECT_LIMITED_RESOURCES macro 603
HCI_ERR_CONNECTION_TIMEOUT macro 699
HCI_ERR_INVALID_PARAMETERS macro 603
HCI_ERR_MEMORY_CAPACITY_EXCEEDED macro 603
HCI_ERR_PAIRING_NOT_ALLOWED macro 703
HCI_ERR_PIN_OR_KEY_MISSING macro 703
HCI_ERR_SCO_CONN_LIMIT_EXCEEDED macro 604
HCI_ERR_SIMPLE_PAIRING_NOT_SUPPORTED macro 604
HCI_ERR_SUCCESS macro 604
HCI_ERR_UNSPECIFIED macro 604
hci_errors.h 1126
HCI_EVT_ALL_HCI_EVENTS macro 604
HCI_EVT_AUTHENTICATION_COMPLETE macro 605
HCI_EVT_CHANGE_CONN_LINK_COMPLETE macro 605
HCI_EVT_CMD_SEND_FINISHED macro 605
HCI_EVT_CMD_SEND_STARTED macro 605
HCI_EVT_COMMAND_COMPLETE macro 605
HCI_EVT_COMMAND_COMPLETE_PARAM_LEN macro 606
HCI_EVT_COMMAND_STATUS macro 606
HCI_EVT_CONN_PACKET_TYPE_CHANGED macro 606
HCI_EVT_CONNECTION_COMPLETE macro 606
HCI_EVT_CONNECTION_REQUEST macro 606
HCI_EVT_DATA_BUFFER_OVERFLOW macro 607
HCI_EVT_DISCONNECTION_COMPLETE macro 607
HCI_EVT_ENCRYPTION_CHANGE macro 607
HCI_EVT_ENCRYPTION_KEY_REFRESH_COMPLETE macro 607
HCI_EVT_ENHANCED_FLUSH_COMPLETE macro 607

HCI_EVT_EXTENDED_INQUIRY_RESULT macro 608
HCI_EVT_FIRST macro 608
HCI_EVT_FLOW_SPECIFICATION_COMPLETE macro 608
HCI_EVT_FLUSH_OCCURED macro 608
hci_evt_handlers.h 1127
HCI_EVT_HARDWARE_ERROR macro 608
HCI_EVT_INQUIRY_COMPLETE macro 609
HCI_EVT_INQUIRY_RESULT macro 609
HCI_EVT_INQUIRY_RESULT_WITH_RSSI macro 609
HCI_EVT_IO_CAPABILITY_REQUEST macro 609
HCI_EVT_IO_CAPABILITY_RESPONSE macro 609
HCI_EVT_KEYPRESS_NOTIFICATION macro 610
HCI_EVT_LAST macro 610
HCI_EVT_LE_META_EVENT macro 610
HCI_EVT_LINK_IS_BUSY macro 610
HCI_EVT_LINK_IS_IDLE macro 610
HCI_EVT_LINK_KEY_NOTIFICATION macro 611
HCI_EVT_LINK_KEY_REQUEST macro 611
HCI_EVT_LINK_SUPERVISION_TO_CHANGED macro 611
HCI_EVT_LOOPBACK_COMMAND macro 611
HCI_EVT_MASTER_LINK_KEY_COMPLETE macro 611
HCI_EVT_MAX_SLOTS_CHANGE macro 612
HCI_EVT_MODE_CHANGE macro 612
HCI_EVT_NUM_OF_COMPLETED_PACKETS macro 612
HCI_EVT_PAGE_SCAN_REPET_MODE_CHANGE macro 612
HCI_EVT_PIN_CODE_REQUEST macro 612
HCI_EVT_QOS_SETUP_COMPLETE macro 613
HCI_EVT_QOS_VIOLATION macro 613
HCI_EVT_READ_CLOCK_OFFSET_COMPLETE macro 613
HCI_EVT_READ_RMT_EXT_FEATURES_COMP macro 613
HCI_EVT_READ_RMT_SUP_FEATURES_COMP macro 613
HCI_EVT_READ_RMT_VERSION_INFO_COMP macro 614
HCI_EVT_REMOTE_NAME_REQUEST_COMPLETE macro 614
HCI_EVT_REMOTE_OOB_DATA_REQUEST macro 614
HCI_EVT_RETURN_LINK_KEYS macro 614
HCI_EVT_RMT_HOST_SUPP_FEATURES_NTF macro 614
HCI_EVT_ROLE_CHANGE macro 615
HCI_EVT_SIMPLE_PAIRING_COMPLETE macro 615
HCI_EVT_SNIFF_SUBRATING macro 615
HCI_EVT_SYNCH_CONNECTION_CHANGED macro 615
HCI_EVT_SYNCH_CONNECTION_COMPLETE macro 615
HCI_EVT_USER_CONFIRMATION_REQUEST macro 616
HCI_EVT_USER_PASSKEY_NOTIFICATION macro 616
HCI_EVT_USER_PASSKEY_REQUEST macro 616
HCI_EXIT_PARK_STATE macro 616
HCI_EXIT_PERIODIC_INQUIRY_MODE macro 616
HCI_EXIT_SNIFF_MODE macro 617
HCI_FLOW_SPECIFICATION macro 617
HCI_FLUSH macro 617
hci_get_conn_state function 555
hci_get_conn_state_by_bdaddr function 561
HCI_H2C_BROADCAST_ACTIVE_SLAVE macro 617
HCI_H2C_BROADCAST_NO_BROADCATS macro 617
HCI_H2C_BROADCAST_PARCKED_SLAVE macro 618
HCI_H2C_BROADCAST_RESERVED macro 618
HCI_HOLD_MODE macro 618
HCI_HOST_BUFFER_SIZE macro 618
HCI_HOST_NUM_OF_COMPLETED_PACKETS macro 618
HCI_INIT_FLAG_IGNORE_TOTAL_NUM_ACL_DATA_PACKETS macro 699
HCI_INIT_FLAG_SEND_HCI_RESET macro 699
HCI_INIT_LE_CONN_STATES macro 37
HCI_INIT_LE_CTRL_STATE macro 37
HCI_INIT_SCO_HANDLERS macro 37
HCI_INIT_SSP_HANDLERS macro 37
HCI_INQUIRY macro 619
HCI_INQUIRY_CANCEL macro 619
HCI_INQUIRY_MODE_EXTENDED macro 619
HCI_INQUIRY_MODE_STANDARD macro 619
HCI_INQUIRY_MODE_WITH_RSSI macro 619
HCI_IO_CAPABILITY_REQUEST_NEGATIVE_REPLY macro 620
HCI_IO_CAPABILITY_REQUEST_REPLY macro 620
HCI_L2CAP_BUFFER_LEN macro 37
hci_le.h 1128
HCI_LE_ADD_DEVICE_TO_WHITE_LIST macro 620
HCI_LE_ADDRESS_TYPE_PUBLIC macro 620
HCI_LE_ADDRESS_TYPE_RANDOM macro 620
HCI_LE_ADV_CHANNEL_MAP_ENABLE_37 macro 700
HCI_LE_ADV_CHANNEL_MAP_ENABLE_38 macro 700
HCI_LE_ADV_CHANNEL_MAP_ENABLE_39 macro 700
HCI_LE_ADV_CHANNEL_MAP_ENABLE_ALL macro 700
HCI_LE_ADV_CHANNEL_MAP_RESERVED macro 700
HCI_LE_ADV_TYPE_CONN_UNDIRECT macro 701
HCI_LE_ADV_TYPE_DIRECT_HIGH macro 701
HCI_LE_ADV_TYPE_DIRECT_LOW macro 701
HCI_LE_ADV_TYPE_NONCONN macro 701
HCI_LE_ADV_TYPE_SCAN macro 701
HCI_LE_ADVERTISING_FLAG_BREDR_NOT_SUPPORTED macro 621
HCI_LE_ADVERTISING_FLAG_GENERAL_DISCOVERABLE_MODE macro 621
HCI_LE_ADVERTISING_FLAG_LIMITED_DISCOVERABLE_MODE macro 621
HCI_LE_ADVERTISING_FLAG_SIMULTANEOUS_LE_BREDR_CONTROLLER macro 621
HCI_LE_ADVERTISING_FLAG_SIMULTANEOUS_LE_BREDR_HOST macro 621
HCI_LE_AVERTISING_DISABLED macro 622
HCI_LE_AVERTISING_ENABLED macro 622
HCI_LE_CLEAR_WHITE_LIST macro 622
HCI_LE_CONNECTION_UPDATE macro 622
HCI_LE_CREATE_CONNECTION macro 622
HCI_LE_CREATE_CONNECTION_CANCEL macro 623
HCI_LE_DISABLED macro 623
HCI_LE_DUPLICATE_FILTERING_DISABLED macro 623
HCI_LE_DUPLICATE_FILTERING_ENABLED macro 623
HCI_LE_ENABLED macro 623
HCI_LE_ENCRYPT macro 624
HCI_LE_EVT_ADVERTISING_REPORT macro 624
HCI_LE_EVT_CONNECTION_COMPLETE macro 624
HCI_LE_EVT_CONNECTION_UPDATE_COMPLETE macro 624
HCI_LE_EVT_LONG_TERM_KEY_REQUEST macro 624
HCI_LE_EVT_READ_REMOTE_USED_FEATURES_COMPLETE macro 625
HCI_LE_FILTER_POLICY_NOT_USED macro 625
HCI_LE_FILTER_POLICY_WHITE_LIST macro 625
HCI_LE_FLAG_SHARED_ACL_BUFFERS macro 625

HCI_LE_FLAG_SIMULTANEOUS_LE_BREDR macro 625
HCI_LE_LONG_TERM_KEY_REQUEST_NEGATIVE_REPLY macro 626
HCI_LE_LONG_TERM_KEY_REQUEST_REPLY macro 626
HCI_LE_MAX_AD_LEN macro 626
HCI_LE_RAND macro 626
HCI_LE_RANDOM_ADDRESS_TYPE_NON_RESOLVABLE macro 702
HCI_LE_RANDOM_ADDRESS_TYPE_RESOLVABLE macro 702
HCI_LE_RANDOM_ADDRESS_TYPE_STATIC macro 702
HCI_LE_READ_ADVERTISING_CHANNEL_TX_POWER macro 626
HCI_LE_READ_BUFFER_SIZE macro 627
HCI_LE_READ_CHANNEL_MAP macro 627
HCI_LE_READ_LOCAL_SUPPORTED_FEATURES macro 627
HCI_LE_READ_REMOTE_USED_FEATURES macro 627
HCI_LE_READ_SUPPORTED_STATES macro 627
HCI_LE_READ_WHITE_LIST_SIZE macro 628
HCI_LE_RECEIVE_TEST macro 628
HCI_LE_REMOVE_DEVICE_FROM_WHITE_LIST macro 628
HCI_LE_SCAN_DISABLED macro 628
HCI_LE_SCAN_ENABLED macro 628
HCI_LE_SCAN_EVT_CANCEL_FAILED macro 629
HCI_LE_SCAN_EVT_CANCELLED macro 629
HCI_LE_SCAN_EVT_DEVICE_FOUND macro 629
HCI_LE_SCAN_EVT_START_FAILED macro 629
HCI_LE_SCAN_EVT_STARTED macro 629
HCI_LE_SCAN_FILTER_POLICY_ALL macro 630
HCI_LE_SCAN_FILTER_POLICY_WHITE_LIST macro 630
HCI_LE_SCAN_TYPE_ACTIVE macro 630
HCI_LE_SCAN_TYPE_PASSIVE macro 630
HCI_LE_SET_ADVERTISE_ENABLE macro 630
HCI_LE_SET_ADVERTISING_DATA macro 631
HCI_LE_SET_ADVERTISING_PARAMETERS macro 631
HCI_LE_SET_EVENT_MASK macro 631
HCI_LE_SET_HOST_CHANNEL_CLASSIFICATION macro 631
HCI_LE_SET_RANDOM_ADDRESS macro 631
HCI_LE_SET_SCAN_ENABLE macro 632
HCI_LE_SET_SCAN_PARAMETERS macro 632
HCI_LE_SET_SCAN_RESPONSE_DATA macro 632
HCI_LE_SIMULTANEOUS_DISABLED macro 632
HCI_LE_SIMULTANEOUS_ENABLED macro 632
HCI_LE_START_ENCRYPTION macro 633
HCI_LE_TEST_END macro 633
HCI_LE_TRANSMITTER_TEST macro 633
HCI_LINK_KEY_LEN macro 633
HCI_LINK_KEY_REQUEST_NEGATIVE_REPLY macro 633
HCI_LINK_KEY_REQUEST_REPLY macro 634
HCI_LINK_KEY_TYPE_COMBINATION macro 634
HCI_LINK_KEY_TYPE_LOCAL_UNIT macro 634
HCI_LINK_KEY_TYPE_REMOTE_UNIT macro 634
HCI_LINK_POLICY_ENABLE_ALL macro 702
HCI_LINK_POLICY_ENABLE_HOLD_MODE macro 634
HCI_LINK_POLICY_ENABLE_PARK_STATE macro 635
HCI_LINK_POLICY_ENABLE_ROLE_SWITCH macro 635
HCI_LINK_POLICY_ENABLE_SNIFF_MODE macro 635
HCI_LINK_TYPE_BD_EDR macro 635
HCI_LINK_TYPE_LE macro 635
hci_linkkey_buffer.h 1131
HCI_MASTER_LINK_KEY macro 636
HCI_MAX_CONNECT_ATTEMPTS macro 39
HCI_MAX_DATA_BUFFERS macro 636
HCI_MAX_EVENT_PARAM_LEN macro 636
HCI_MAX_PARAM_LEN macro 636
HCI_MAX_PIN_LENGTH macro 636
HCI_OPCODE macro 637
HCI_PACKET_BOUNDARY_COMPLETE macro 703
HCI_PACKET_BOUNDARY_CONTINUE macro 637
HCI_PACKET_BOUNDARY_FIRST macro 637
HCI_PACKET_BOUNDARY_FIRST_AUTO_FLUSH macro 637
HCI_PACKET_BOUNDARY_FIRST_NO_AUTO_FLUSH macro 637
HCI_PACKET_TYPE_ACL_DATA macro 638
HCI_PACKET_TYPE_COMMAND macro 638
HCI_PACKET_TYPE_EVENT macro 638
HCI_PACKET_TYPE_NONE macro 638
HCI_PACKET_TYPE_SCO_DATA macro 638
HCI_PAGE_SCAN_REPETITION_MODE_R0 macro 639
HCI_PAGE_SCAN_REPETITION_MODE_R1 macro 639
HCI_PAGE_SCAN_REPETITION_MODE_R2 macro 639
HCI_PARAM_LEN_BD_ADDR macro 639
HCI_PARAM_LEN_BYTE macro 639
HCI_PARAM_LEN_DEV_CLASS macro 640
HCI_PARAM_LEN_HANDLE macro 640
HCI_PARAM_LEN_INT macro 640
HCI_PARAM_LEN_LONG macro 640
HCI_PARAM_TYPE_BD_ADDR macro 640
HCI_PARAM_TYPE_BYTE macro 641
HCI_PARAM_TYPE_DEV_CLASS macro 641
HCI_PARAM_TYPE_HANDLE macro 641
HCI_PARAM_TYPE_INT macro 641
HCI_PARAM_TYPE_LONG macro 641
HCI_PARAM_TYPE_STRING macro 642
HCI_PARK_STATE macro 642
HCI_PERIODIC_INQUIRY_MODE macro 642
HCI_PIN_CODE_REQUEST_NEGATIVE_REPLY macro 642
HCI_PIN_CODE_REQUEST_REPLY macro 642
HCI_POWER_MODE_ACTIVE macro 643
HCI_POWER_MODE_HOLD macro 643
HCI_POWER_MODE_PARK macro 643
HCI_POWER_MODE_SNIFF macro 643
hci_private.h 1132
HCI_QOS_SETUP macro 643
HCI_READ_AFH_CHANNEL_ASSESSMENT_MODE macro 644
HCI_READ_AFH_CHANNEL_MAP macro 644
HCI_READ_AUTHENTICATION_ENABLE macro 644
HCI_READ_AUTOMATIC_FLASH_TIMEOUT macro 644
HCI_READ_BD_ADDR macro 644
HCI_READ_BUFFER_SIZE macro 645
HCI_READ_CLASS_OF_DEVICE macro 645
HCI_READ_CLOCK_COMMAND macro 645
HCI_READ_CLOCK_OFFSET macro 645
HCI_READ_CONNECTION_ACCEPT_TIMEOUT macro 645
HCI_READ_CURRENT_IAC_LAP macro 646
HCI_READ_DEFAULT_ERRONEOUS_DATA_REPORTING macro 646
HCI_READ_DEFAULT_POLICY_SETTINGS macro 646
HCI_READ_ENCRYPTION_MODE macro 646
HCI_READ_EXTENDED_INQUIRY_RESPONSE macro 646
HCI_READ_FAILED_CONTACT_COUNTER macro 647

HCI_READ_HOLD_MODE_ACTIVITY macro 647
HCI_READ_INQUIRY_MODE macro 647
HCI_READ_INQUIRY_RESPONSE_TX_POWER_LEVEL macro 647
HCI_READ_INQUIRY_SCAN_ACTIVITY macro 647
HCI_READ_INQUIRY_SCAN_TYPE macro 648
HCI_READ_LE_HOST_SUPPORT macro 648
HCI_READ_LINK_POLICY_SETTINGS macro 648
HCI_READ_LINK_QUALITY macro 648
HCI_READ_LINK_SUPERVISION_TIMEOUT macro 648
HCI_READ_LMP_HANDLE macro 649
HCI_READ_LOCAL_EXTENDED_FEATURES macro 649
HCI_READ_LOCAL_NAME macro 649
HCI_READ_LOCAL_OOB_DATA macro 649
HCI_READ_LOCAL_SUPPORTED_COMMANDS macro 649
HCI_READ_LOCAL_SUPPORTED_FEATURES macro 650
HCI_READ_LOCAL_VERSION_INFORMATION macro 650
HCI_READ_LOOPBACK_MODE macro 650
HCI_READ_NUM_BROADCST_RETR macro 650
HCI_READ_NUM_OF_SUPPORTED_IAC macro 650
HCI_READ_PAGE_SCAN_ACTIVITY macro 651
HCI_READ_PAGE_SCAN_PERIOD_MODE macro 651
HCI_READ_PAGE_SCAN_TYPE macro 651
HCI_READ_PAGE_TIMEOUT macro 651
HCI_READ_PIN_TYPE macro 651
HCI_READ_REFRESH_ENCRYPTION_KEY macro 652
HCI_READ_REMOTE_EXTENDED_FEATURES macro 652
HCI_READ_REMOTE_SUPPORTED_FEATURES macro 652
HCI_READ_REMOTE_VERSION_INFORMATION macro 652
HCI_READ_RSSI macro 652
HCI_READ_SCAN_ENABLE macro 653
HCI_READ_SIMPLE_PAIRING_MODE macro 653
HCI_READ_STORED_LINK_KEY macro 653
HCI_READ_SYNC_FLOW_CONTROL_ENABLE macro 653
HCI_READ_TRANSMIT_POWER_LEVEL macro 653
HCI_READ_VOICE_SETTING macro 654
HCI_REJECT_CONNECTION_REQUEST macro 654
HCI_REJECT_SYNCH_CONNECTION_REQUEST macro 654
HCI_REMOTE_NAME_REQUEST macro 654
HCI_REMOTE_NAME_REQUEST_CANCEL macro 654
HCI_REMOTE_OOB_DATA_REQUEST_NEGATIVE_REPLY macro 655
HCI_REMOTE_OOB_DATA_REQUEST_REPLY macro 655
HCI_RESET macro 655
HCI_RESET_FAILED_CONTACT_COUNTER macro 655
HCI_ROLE_DISCOVERY macro 655
HCI_ROLE_SWITCH_ALLOW macro 656
HCI_ROLE_SWITCH_DISALLOW macro 656
HCI_SCAN_INQUIRY macro 656
HCI_SCAN_PAGE macro 656
HCI_SCO_CONTENT_FORMAT_AIR_CODING_A_LAW macro 656
HCI_SCO_CONTENT_FORMAT_AIR_CODING_CSVD macro 657
HCI_SCO_CONTENT_FORMAT_AIR_CODING_N_LAW macro 657
HCI_SCO_CONTENT_FORMAT_AIR_CODING_TRANSPARENT macro 657
HCI_SCO_CONTENT_FORMAT_INPUT_CODING_A_LAW macro 657
HCI_SCO_CONTENT_FORMAT_INPUT_CODING_LINEAR macro 657
HCI_SCO_CONTENT_FORMAT_INPUT_CODING_N_LAW macro 658
HCI_SCO_CONTENT_FORMAT_INPUT_FORMAT_1_COMPLEMENT macro 658
HCI_SCO_CONTENT_FORMAT_INPUT_FORMAT_2_COMPLEMENT macro 658
HCI_SCO_CONTENT_FORMAT_INPUT_FORMAT_SIGN_MAGNITUDE macro 658
HCI_SCO_CONTENT_FORMAT_INPUT_FORMAT_UNSIGNED macro 658
HCI_SCO_CONTENT_FORMAT_INPUT_SAMPLE_SIZE_16 macro 659
HCI_SCO_CONTENT_FORMAT_INPUT_SAMPLE_SIZE_8 macro 659
HCI_SCO_DATA_HEADER_LEN macro 659
HCI_SCO_MAX_DATA_LEN macro 659
HCI_SCO_MAX_LATENCY_DONTCARE macro 659
HCI_SCO_PACKET_TYPE_ALL macro 660
HCI_SCO_PACKET_TYPE_EV3 macro 660
HCI_SCO_PACKET_TYPE_EV4 macro 660
HCI_SCO_PACKET_TYPE_EV5 macro 660
HCI_SCO_PACKET_TYPE_HV1 macro 660
HCI_SCO_PACKET_TYPE_HV2 macro 661
HCI_SCO_PACKET_TYPE_HV3 macro 661
HCI_SCO_PACKET_TYPE_NO_2_EV3 macro 661
HCI_SCO_PACKET_TYPE_NO_2_EV5 macro 661
HCI_SCO_PACKET_TYPE_NO_3_EV3 macro 661
HCI_SCO_PACKET_TYPE_NO_3_EV5 macro 662
HCI_SCO_RTX_EFFORT_DONTCARE macro 662
HCI_SCO_RTX_EFFORT_NO_RETRANSMISSION macro 662
HCI_SCO_RTX_EFFORT_OPTIMIZE_LINK_QUALITY macro 662
HCI_SCO_RTX_EFFORT_OPTIMIZE_POWER_CONSUMPTION macro 662
HCI_SCO_RX_BANDWIDTH_DONTCARE macro 663
HCI_SCO_TX_BANDWIDTH_DONTCARE macro 663
HCI_SEND_DATA_STATUS_INTERRUPTED macro 663
HCI_SEND_DATA_STATUS_SUCCESS macro 663
HCI_SEND_KEY_PRESS_NOTIFICATION macro 663
HCI_SET_AFH_HOST_CHANNEL_CLASSIFICATION macro 664
HCI_SET_CONNECTION_ENCRYPTION macro 664
HCI_SET_CTRL_TO_HOST_FLOW_CONTROL macro 664
HCI_SET_EVENT_FILTER macro 664
HCI_SET_EVENT_MASK macro 664
HCI_SETUP_SYNCHRONOUS_CONNECTION macro 665
hci_signal.h 1132
HCI_SIZEOF_LE_CONN_STATES macro 38
HCI_SIZEOF_LE_CTRL_STATE macro 38
hci_sleep function 555
HCI_SNIFF_MODE macro 665
HCI_SNIFF_SUBRATING macro 665
HCI_SUCCESS macro 703
HCI_SWITCH_ROLE macro 665
hci_transport.h 1133
HCI_TRANSPORT_HEADER_LEN macro 665
hci_transport_t structure 692
HCI_TX_BUFFER_LEN macro 38
HCI_UART_PACKET_TYPE_ACL_DATA macro 666
HCI_UART_PACKET_TYPE_COMMAND macro 666
HCI_UART_PACKET_TYPE_EVENT macro 666
HCI_UART_PACKET_TYPE_SCO_DATA macro 666
HCI_USER_CONFIRMATION_REQ_NEGATIVE_REPLY macro 666
HCI_USER_CONFIRMATION_REQUEST_REPLY macro 667
HCI_USER_PASSKEY_REQUEST_NEGATIVE_REPLY macro 667
HCI_USER_PASSKEY_REQUEST_REPLY macro 667

hci_utils.h 1133
hci_wakeup function 555
HCI_WRITE_AFH_CHANNEL_ASSESSMENT_MODE macro 667
HCI_WRITE_AUTHENTICATION_ENABLE macro 667
HCI_WRITE_AUTOMATIC_FLASH_TIMEOUT macro 668
HCI_WRITE_CLASS_OF_DEVICE macro 668
HCI_WRITE_CONNECTION_ACCEPT_TIMEOUT macro 668
HCI_WRITE_CURRENT_IAC_LAP macro 668
HCI_WRITE_DEFAULT_ERRONEOUS_DATA_REPORTING macro 668
HCI_WRITE_DEFAULT_POLICY_SETTINGS macro 669
HCI_WRITE_ENCRYPTION_MODE macro 669
HCI_WRITE_EXTENDED_INQUIRY_RESPONSE macro 669
HCI_WRITE_EXTENDED_INQUIRY_RESPONSE_PARAM_LEN macro 669
HCI_WRITE_HOLD_MODE_ACTIVITY macro 669
HCI_WRITE_INQUIRY_MODE macro 670
HCI_WRITE_INQUIRY_SCAN_ACTIVITY macro 670
HCI_WRITE_INQUIRY_SCAN_TYPE macro 670
HCI_WRITE_INQUIRY_TX_POWER_LEVEL macro 670
HCI_WRITE_LE_HOST_SUPPORT macro 670
HCI_WRITE_LINK_POLICY_SETTINGS macro 671
HCI_WRITE_LINK_SUPERVISION_TIMEOUT macro 671
HCI_WRITE_LOCAL_NAME macro 671
HCI_WRITE_LOCAL_NAME_PARAM_LEN macro 671
HCI_WRITE_LOOPBACK_MODE macro 671
HCI_WRITE_NUM_BROADCST_RETR macro 672
HCI_WRITE_PAGE_SCAN_ACTIVITY macro 672
HCI_WRITE_PAGE_SCAN_PERIOD_MODE macro 672
HCI_WRITE_PAGE_SCAN_TYPE macro 672
HCI_WRITE_PAGE_TIMEOUT macro 672
HCI_WRITE_PIN_TYPE macro 673
HCI_WRITE_SCAN_ENABLE macro 673
HCI_WRITE_SIMPLE_PAIRING_DEBUG_MODE macro 673
HCI_WRITE_SIMPLE_PAIRING_MODE macro 673
HCI_WRITE_STORED_LINK_KEY macro 673
HCI_WRITE_SYNC_FLOW_CONTROL_ENABLE macro 674
HCI_WRITE_VOICE_SETTING macro 674
hcitr_3wire.h 1133
HCITR_3WIRE_DEFAULT_ACK_TIMEOUT macro 674
hcitr_bcsp.h 1134
HCITR_BCSP_ALLOCATE_BUFFERS function 204
HCITR_BCSP_DEFAULT_ACK_TIMEOUT macro 674
hcitr_packet.h 1134
hcitr_tih4.h 1134
HCITR_TIH4_POWER_EVENT_AWAKE enumeration member 692
HCITR_TIH4_POWER_EVENT_PREPARE_TO_SLEEP enumeration member 692
HCITR_TIH4_POWER_EVENT_SLEEP enumeration member 692
HCITR_TIH4_POWER_EVENT_WAKE_UP enumeration member 692
hcitr_uart.h 1135
HCRP_ALLOCATE_BUFFERS function 205
HFP_ALLOCATE_BUFFERS function 16
HID_ALLOCATE_BUFFERS function 16
HIDS_LAST_DEVICE_ADDR macro 212
HIDS_SIGNATURE macro 212
HIDS_SIGNATURE_ADDR macro 213
HSP_AG_ALLOCATE_BUFFERS function 16
HSP_ALLOCATE_BUFFERS function 17

I

IAP_ALLOCATE_BUFFERS function 17
IAP_BT_ALLOCATE_BUFFERS function 17
IAP_BT_MAX_TRANSPORTS macro 20
IAP_MAX_SESSIONS macro 20
IAP_RX_BUFFER_SIZE macro 21
IAP2_ALLOCATE_BUFFERS function 17
IAP2_MAX_PACKET_SIZE macro 21
IAP2_MAX_SESSIONS macro 21
IAPEA_ALLOCATE_BUFFERS function 204
INIT_DE_SEQUENCE macro 847
INP_BUF_SIZE macro 834
Introduction 3
    PIC32 Bluetooth Stack Library 3
is_bdaddr_command function 555
is_hconn_command function 555

L

l2cap.h 1136
L2CAP_ALLOCATE_BUFFERS function 17
L2CAP_ALLOCATE_BUFFERS_FUNCTION macro 39
L2CAP_ALLOCATE_BUFFERS_RAM_SIZE_VAR macro 40
L2CAP_ALLOCATE_BUFFERS_VARS macro 40
L2CAP_CHANNEL_FLAG_FORCE_HCI_DISCONNECT macro 780
L2CAP_CHANNEL_FLAG_INCOMING macro 736
L2CAP_CHANNEL_FLAG_SENDING macro 736
L2CAP_CMD_CONFIG_REQUEST macro 736
L2CAP_CMD_CONFIG_RESPONSE macro 736
L2CAP_CMD_CONN_PARAM_UPDATE_REQUEST macro 736
L2CAP_CMD_CONN_PARAM_UPDATE_RESPONSE macro 737
L2CAP_CMD_CONN_REQUEST macro 737
L2CAP_CMD_CONN_RESPONSE macro 737
L2CAP_CMD_DATA_LEN_CMD_REJECT macro 737
L2CAP_CMD_DATA_LEN_CONFIG_REQUEST macro 737
L2CAP_CMD_DATA_LEN_CONFIG_RESPONSE macro 738
L2CAP_CMD_DATA_LEN_CONN_REQUEST macro 738
L2CAP_CMD_DATA_LEN_CONN_RESPONSE macro 738
L2CAP_CMD_DATA_LEN_DCONN_REQUEST macro 738
L2CAP_CMD_DATA_LEN_DCONN_RESPONSE macro 738
L2CAP_CMD_DATA_LEN_ECHO_REQUEST macro 739
L2CAP_CMD_DATA_LEN_ECHO_RESPONSE macro 739
L2CAP_CMD_DATA_LEN_INFO_REQUEST macro 739
L2CAP_CMD_DATA_LEN_INFO_RESPONSE macro 739
L2CAP_CMD_DCONN_REQUEST macro 739
L2CAP_CMD_DCONN_RESPONSE macro 740
L2CAP_CMD_ECHO_REQUEST macro 740
L2CAP_CMD_ECHO_RESPONSE macro 740
L2CAP_CMD_HEADER_LEN macro 740
L2CAP_CMD_INFO_REQUEST macro 740
L2CAP_CMD_INFO_RESPONSE macro 741
L2CAP_CMD_LAST macro 741
L2CAP_CMD_REJECT macro 741
L2CAP_CMD_RESERVED macro 741
L2CAP_CMD_STATUS_BEING_SENT macro 741
L2CAP_CMD_STATUS_PENDING macro 742
L2CAP_CMD_STATUS_WAITING_RESPONSE macro 742
l2cap_command_queue.h 1138

l2cap_config.h 1138
l2cap_config_handlers.h 1139
L2CAP_CONFIG_RESULT_REJECTED macro 742
L2CAP_CONFIG_RESULT_SUCCESS macro 742
L2CAP_CONFIG_RESULT_UNACCEPTABLE_PARAMETER macro 742
L2CAP_CONFIG_RESULT_UNKNOWN_OPTION macro 743
L2CAP_CONN_REQ_RESULT_INVALID_PSM macro 743
L2CAP_CONN_REQ_RESULT_INVALID_SOURCE_CID macro 783
L2CAP_CONN_REQ_RESULT_NO_RESOURCES macro 743
L2CAP_CONN_REQ_RESULT_PENDING macro 743
L2CAP_CONN_REQ_RESULT_SECURITY_BLOCK macro 743
L2CAP_CONN_REQ_RESULT_SRC_CID_ALREADY_ALLOCATED macro 783
L2CAP_CONN_REQ_RESULT_SUCCESS macro 744
L2CAP_CONN_REQ_STATUS_AUTHENTICATION_PENDING macro 744
L2CAP_CONN_REQ_STATUS_AUTHORIZATION_PENDING macro 744
L2CAP_CONN_REQ_STATUS_NO_INFO macro 744
L2CAP_DECL_ERETR_FUNCTIONS macro 47
L2CAP_DEFAULT_ERTX macro 744
L2CAP_DEFAULT_RTX macro 745
L2CAP_ECHO_MAX_DATA_LEN macro 745
l2cap_eretr.h 1140
L2CAP_ERETR_EVENT_DATA_REQUEST enumeration member 774
L2CAP_ERETR_EVENT_LOCAL_BUSY_CLEAR enumeration member 774
L2CAP_ERETR_EVENT_LOCAL_BUSY_DETECTED enumeration member 774
L2CAP_ERETR_EVENT_MONITOR_TIMER_EXPIRED enumeration member 774
L2CAP_ERETR_EVENT_RECV_FBIT enumeration member 774
L2CAP_ERETR_EVENT_RECV_REQSEQ_AND_FBIT enumeration member 774
L2CAP_ERETR_EVENT_RETRANSMIT_TIMER_EXPIRED enumeration member 774
L2CAP_ERETR_RECV_STATE_RECV macro 745
L2CAP_ERETR_RECV_STATE_REJ_SENT macro 745
L2CAP_ERETR_RECV_STATE_SREJ_SENT macro 745
L2CAP_ERETR_XMIT_STATE_WAIT_ACK macro 746
L2CAP_ERETR_XMIT_STATE_WAIT_F macro 746
L2CAP_ERETR_XMIT_STATE_XMIT macro 746
L2CAP_EXT_BI_QOS macro 746
L2CAP_EXT_ENHANCED_RETRANSMISSION macro 746
L2CAP_EXT_EXT_FLOW_SPEC_BR_EDR macro 781
L2CAP_EXT_EXT_WINDOW_SIZE macro 781
L2CAP_EXT_FCS_OPTION macro 747
L2CAP_EXT_FEATURES_ENABLED macro 781
L2CAP_EXT_FIXED_CHANNELS macro 781
L2CAP_EXT_FLOW_CONTROL macro 747
L2CAP_EXT_RETRANSMISSION macro 747
L2CAP_EXT_STREAMING macro 747
L2CAP_EXT_UNICAST_CONNLESS_DATA_RCPT macro 781
l2cap_fixed_channel.h 1141
L2CAP_FIXED_CHANNELS_DECL macro 40
L2CAP_FRAME_TYPE_I macro 747
L2CAP_FRAME_TYPE_S macro 748
L2CAP_HCI_CONNECT_PACKET_TYPE macro 748
L2CAP_HCI_CONNECT_PAGE_SCAN_REPETITION_MODE macro 748
L2CAP_HCI_CONNECT_ROLE_SWITCH macro 748
L2CAP_HCI_PACKET_TYPE macro 40
L2CAP_HCI_PAGE_SCAN_REPETITION_MODE macro 41
L2CAP_HCI_ROLE_SWITCH macro 41
L2CAP_HEADER_LEN macro 748
L2CAP_IDLE_CONNECTION_TIMEOUT macro 41
L2CAP_IDLE_HCI_CONNECTION_TIMEOUT macro 749
L2CAP_INFO_NOT_SUPPORTED macro 749
L2CAP_INFO_RESULT_SUCCESS macro 749
L2CAP_INFO_TYPE_CONNECTIONLESS_MTU macro 749
L2CAP_INFO_TYPE_EXTENDED_SUPPORT macro 749
L2CAP_INFO_TYPE_FIXED_CHANNELS macro 782
L2CAP_LINK_TYPE_BD_EDR macro 750
L2CAP_LINK_TYPE_LE macro 750
L2CAP_MAX_CMD_QUEUE_LEN macro 750
L2CAP_MAX_FIXED_CHANNELS macro 47
L2CAP_MAX_FRAME_BUFFERS macro 750
L2CAP_MAX_MANAGERS macro 750
L2CAP_MAX_MTU macro 751
L2CAP_MAX_OPTIONS macro 751
L2CAP_MAX_RTX macro 751
L2CAP_MGR_STATE_FREE macro 751
L2CAP_MGR_STATE_LISTENING_L2CAP macro 751
L2CAP_MGR_STATE_USED macro 752
L2CAP_MIN_MTU macro 752
L2CAP_MONITOR_TIMEOUT macro 752
L2CAP_OPTION_LEN_FLASH_QOS macro 752
L2CAP_OPTION_LEN_FLASH_RFC macro 752
L2CAP_OPTION_LEN_FLASH_TIMEOUT macro 753
L2CAP_OPTION_LEN_MAX_MTU macro 753
L2CAP_OPTION_LEN_UNKNOWN macro 753
L2CAP_OPTION_LEN_UNKNOWN_DATA macro 753
L2CAP_OPTION_TYPE_FLASH_QOS macro 753
L2CAP_OPTION_TYPE_FLASH_QOS_FLAG macro 754
L2CAP_OPTION_TYPE_FLASH_RFC macro 754
L2CAP_OPTION_TYPE_FLASH_RFC_FLAG macro 754
L2CAP_OPTION_TYPE_FLASH_TIMEOUT macro 754
L2CAP_OPTION_TYPE_FLASH_TIMEOUT_FLAG macro 754
L2CAP_OPTION_TYPE_MAX_MTU macro 755
L2CAP_OPTION_TYPE_MAX_MTU_FLAG macro 755
L2CAP_OPTION_TYPE_UNKNOWN_FLAG macro 755
l2cap_packet.h 1141
L2CAP_PACKET_DATA_TYPE_RAW macro 755
L2CAP_PACKET_DATA_TYPE_SMART macro 755
l2cap_private.h 1141
l2cap_psm.h 1143
L2CAP_REJECT_REASON_INVALID_CHANNEL macro 756
L2CAP_REJECT_REASON_MTU_EXCEEDED macro 756
L2CAP_REJECT_REASON_NOT_UNDERSTOOD macro 756
L2CAP_RETR_TIMEOUT macro 756
L2CAP_RFC_BASIC macro 756
L2CAP_RFC_ENHANCED_RETRANSMISSION macro 757
L2CAP_RFC_ERETR_TIMEOUT macro 757
L2CAP_RFC_ERETR_TX_WINDOW macro 757
L2CAP_RFC_FLOW_CONTROL macro 757
L2CAP_RFC_MONITOR_TIMEOUT macro 757
L2CAP_RFC_RETRANSMISSION macro 758
L2CAP_RFC_STREAMING macro 758

L2CAP_SAR_SDU_CONTINUE macro 758
L2CAP_SAR_SDU_START macro 758
L2CAP_SAR_SDU_STOP macro 758
L2CAP_SAR_UNSEGMENTED macro 759
L2CAP_SERVICE_TYPE_BEST_EFFORT macro 759
L2CAP_SERVICE_TYPE_GUARANTEED macro 759
L2CAP_SERVICE_TYPE_NO_TRAFFIC macro 759
L2CAP_SFUNCTION_REJ macro 759
L2CAP_SFUNCTION_RNR macro 760
L2CAP_SFUNCTION_RR macro 760
L2CAP_SFUNCTION_SREJ macro 760
l2cap_signal.h 1143
l2cap_test.h 1143
l2cap_timer.h 1143
Library Interface 56
    PIC32 Bluetooth Stack Library 56
Library Overview 5
    PIC32 Bluetooth Stack Library 5
LKS_FIRST_KEY_ADDR macro 213
LKS_MAX_LINK_KEYS macro 213
LKS_SIGNATURE macro 213
LKS_SIGNATURE_ADDR macro 213
lm.h 1135
LM_PACKET_TYPE_DH1 macro 674
LM_PACKET_TYPE_DH3 macro 675
LM_PACKET_TYPE_DH5 macro 675
LM_PACKET_TYPE_DM1 macro 675
LM_PACKET_TYPE_DM3 macro 675
LM_PACKET_TYPE_DM5 macro 675
LOG macro 214
LOG_EX macro 214
LOGADDR macro 214
LOGADDR_EX macro 214
LOGCLEAR macro 214
LOGINT macro 215
LOGINT_EX macro 215
LOGMEMORY macro 215
LOGMEMORY_EX macro 215
LOGWRITE macro 215

M

MAP_ALLOCATE_BUFFERS function 206
max macro 980
MAX_SBC_DEC_STATE_SIZE macro 835
min macro 980
MK_CMD_ADDRESS macro 800
MK_DLCI macro 800
MPEG_BITRATE_0000 macro 258
MPEG_BITRATE_0001 macro 258
MPEG_BITRATE_0010 macro 258
MPEG_BITRATE_0011 macro 258
MPEG_BITRATE_0100 macro 259
MPEG_BITRATE_0101 macro 259
MPEG_BITRATE_0110 macro 259
MPEG_BITRATE_0111 macro 259
MPEG_BITRATE_1000 macro 259
MPEG_BITRATE_1001 macro 260
MPEG_BITRATE_1010 macro 260
MPEG_BITRATE_1011 macro 260
MPEG_BITRATE_1100 macro 260
MPEG_BITRATE_1101 macro 260
MPEG_BITRATE_1110 macro 261
MPEG_BITRATE_ALL macro 261
MPEG_CHANNEL_MODE_ALL macro 261
MPEG_CHANNEL_MODE_DUAL_CHANNEL macro 261
MPEG_CHANNEL_MODE_JOINT_STEREO macro 261
MPEG_CHANNEL_MODE_MONO macro 262
MPEG_CHANNEL_MODE_STEREO macro 262
MPEG_CRC_PROTECTION_NOT_SUPPORTED macro 262
MPEG_CRC_PROTECTION_SUPPORTED macro 262
MPEG_LAYER_1 macro 262
MPEG_LAYER_2 macro 263
MPEG_LAYER_3 macro 263
MPEG_LAYER_ALL macro 263
MPEG_MPF_1 macro 263
MPEG_MPF_2 macro 263
MPEG_SAMPLING_FREQUENCY_16000 macro 264
MPEG_SAMPLING_FREQUENCY_22050 macro 264
MPEG_SAMPLING_FREQUENCY_24000 macro 264
MPEG_SAMPLING_FREQUENCY_32000 macro 264
MPEG_SAMPLING_FREQUENCY_44100 macro 264
MPEG_SAMPLING_FREQUENCY_48000 macro 265
MPEG_SAMPLING_FREQUENCY_ALL macro 265
MPEG_VBR_NOT_SUPPORTED macro 265
MPEG_VBR_SUPPORTED macro 265

N

NULL macro 216

O

OBEX_ALLOCATE_BUFFERS function 18
OGF_CTRL_BASEBAND macro 676
OGF_INFORMATION macro 676
OGF_LE macro 676
OGF_LINK_CONTROL macro 676
OGF_LINK_POLICY macro 676
OGF_STATUS macro 677
OGF_TESTING macro 677
OGF_VENDOR macro 677
OOB_DATA_HASH_LENGTH macro 901
OOB_DATA_RANDOMIZER_LENGTH macro 901
OUT_BUF_SIZE macro 835

P

PACK_CONFIG_REQUEST macro 42
PACK_CONFIG_RESPONSE macro 42
PACK_CONN_REQUEST macro 42
PACK_CONN_RESPONSE macro 42
PACK_DCONN_REQUEST macro 42
PACK_DCONN_RESPONSE macro 43
PACK_INFO_REQUEST macro 43
PACK_INFO_RESPONSE macro 43
packet.h 1144
PASSKEY_CLEARED enumeration member 905
PASSKEY_DIGIT_ENTERED enumeration member 905
PASSKEY_DIGIT_ERASED enumeration member 905
PASSKEY_ENTRY_COMPLETED enumeration member 905

PASSKEY_ENTRY_STYARTED enumeration member 905
patch.h 1144
PBAP_ALLOCATE_BUFFERS function 18
pcmd_disconnection_res structure 780
pf_hci_sleep_callback type 692
pf_hci_wakeup_callback type 693
pf_l2cap_cmd_callback type 780
pf_l2cap_receive_callback type 693
PROCESS_CONFIG_REQ macro 43
PROCESS_CONFIG_RES macro 43
PROCESS_CONN_REQ macro 44
PROCESS_CONN_RES macro 44
PROCESS_DCONN_REQ macro 44
PROCESS_DCONN_RES macro 44
PROCESS_INFO_REQ macro 44
PROCESS_INFO_RES macro 45
PS_DEFAULT macro 934
PS_F macro 934
PS_I macro 934
PS_RAM macro 934
PS_ROM macro 934
PSKEY_ANA_FREQ macro 935
PSKEY_BDADDR macro 935
PSKEY_BLE_DEFAULT_TX_POWER macro 962
PSKEY_CLOCK_REQUEST_ENABLE macro 935
PSKEY_DEEP_SLEEP_CLEAR_RTS macro 935
PSKEY_DEEP_SLEEP_EXTERNAL_CLOCK_SOURCE_PIO macro 963
PSKEY_DEEP_SLEEP_STATE macro 935
PSKEY_DEEP_SLEEP_USE_EXTERNAL_CLOCK macro 936
PSKEY_DEEP_SLEEP_WAKE_CTS macro 936
PSKEY_DIGITAL_AUDIO_BITS_PER_SAMPLE macro 936
PSKEY_DIGITAL_AUDIO_CONFIG macro 936
PSKEY_H_HC_FC_MAX_SCO_PKT_LEN macro 963
PSKEY_H_HC_FC_MAX_SCO_PKTS macro 963
PSKEY_HCI_NOP_DISABLE macro 936
PSKEY_HOST_INTERFACE macro 937
PSKEY_HOSTIO_MAP_SCO_PCM macro 937
PSKEY_HOSTIO_UART_RESET_TIMEOUT macro 937
PSKEY_LC_DEFAULT_TX_POWER macro 967
PSKEY_LC_MAX_TX_POWER macro 968
PSKEY_LC_MAX_TX_POWER_NO_RSSI macro 968
PSKEY_PCM_CLOCK_RATE macro 963
PSKEY_PCM_CONFIG32 macro 963
PSKEY_PCM_FORMAT macro 964
PSKEY_PCM_PULL_CONTROL macro 937
PSKEY_PCM_SYNC_RATE macro 964
PSKEY_PCM_USE_LOW_JITTER_MODE macro 964
PSKEY_PIO_DEEP_SLEEP_EITHER_LEVEL macro 937
PSKEY_UART_BAUDRATE macro 938
PSKEY_UART_BITRATE macro 938
PSKEY_UART_CONFIG_BCSP macro 938
PSKEY_UART_CONFIG_H4 macro 938
PSKEY_UART_CONFIG_H4DS macro 964
PSKEY_UART_CONFIG_H5 macro 938
PSKEY_UART_TX_WINDOW_SIZE macro 939
PSKEY_VM_DISABLE macro 939
PSM_ATT macro 760
PSM_AVCTP macro 760
PSM_AVCTP_Browsing macro 761
PSM_AVDTP macro 761
PSM_BNEP macro 761
PSM_HID_Control macro 761
PSM_HID_Interrupt macro 761
PSM_RFCOMM macro 762
PSM_SDP macro 762

Q

queue.h 1144

R

read_command function 788
READ_CONFIG_REQUEST macro 45
READ_CONFIG_RESPONSE macro 45
READ_CONN_REQUEST macro 45
READ_CONN_RESPONSE macro 45
READ_DCONN_REQUEST macro 46
READ_DCONN_RESPONSE macro 46
READ_INFO_REQUEST macro 46
READ_INFO_RESPONSE macro 46
rfcomm.h 1145
RFCOMM_ALLOCATE_BUFFERS function 18
RFCOMM_ALLOCATE_BUFFERS_FUNCTION macro 47
RFCOMM_ALLOCATE_BUFFERS_RAM_SIZE_VAR macro 48
RFCOMM_ALLOCATE_BUFFERS_VARS macro 48
RFCOMM_BUFFER_SIZE macro 48
RFCOMM_CFC_ENABLED macro 801
RFCOMM_CFC_LOCAL_CREDIT macro 801
RFCOMM_CFC_MAX_INITIAL_CREDIT macro 801
rfcomm_cmd_queue.h 1148
RFCOMM_CMD_STATUS_FC_PENDING macro 801
RFCOMM_CMD_STATUS_PENDING macro 801
RFCOMM_CMD_STATUS_WAITING_RESPONSE macro 802
RFCOMM_COMMAND macro 802
rfcomm_config.h 1149
rfcomm_cq_ack_cmd function 789
rfcomm_cq_ack_mx_cmd function 789
rfcomm_cq_find_failed_pn function 800
RFCOMM_CTL_MSG_CLD macro 802
RFCOMM_CTL_MSG_FCOFF macro 802
RFCOMM_CTL_MSG_FCON macro 802
RFCOMM_CTL_MSG_MSC macro 803
RFCOMM_CTL_MSG_NSC macro 803
RFCOMM_CTL_MSG_PN macro 803
RFCOMM_CTL_MSG_PSC macro 803
RFCOMM_CTL_MSG_RLS macro 803
RFCOMM_CTL_MSG_RPN macro 804
RFCOMM_CTL_MSG_SNC macro 804
RFCOMM_CTL_MSG_TEST macro 804
RFCOMM_DLC_CHANGED_CONN_STATE macro 804
RFCOMM_DLC_CHANGED_REMOTE_MSC macro 804
RFCOMM_DLC_CONNECTION_FAILED macro 805
RFCOMM_DLC_STATE_CLOSED macro 805
RFCOMM_DLC_STATE_OPEN macro 805
RFCOMM_DLCI_CONTROL macro 805
RFCOMM_DLCI_FREE macro 805
RFCOMM_ENABLE_MULTIDEVICE_CHANNELS macro 48

RFCOMM_ERR_DM macro 806
RFCOMM_ERR_INTERRUPTED macro 830
RFCOMM_ERR_SUCCESS macro 806
RFCOMM_ERR_TIMEOUT macro 806
RFCOMM_FC_TYPE_AGREGATE macro 806
RFCOMM_FC_TYPE_CREDIT macro 806
rfcomm_find_session function 799
RFCOMM_FLAG_CR macro 807
RFCOMM_FLAG_EA macro 807
RFCOMM_FLAG_PF macro 807
RFCOMM_FRAME_HEADER_LEN macro 807
RFCOMM_FRAME_TYPE_DISC macro 807
RFCOMM_FRAME_TYPE_DM macro 808
RFCOMM_FRAME_TYPE_SABM macro 808
RFCOMM_FRAME_TYPE_UA macro 808
RFCOMM_FRAME_TYPE_UI macro 808
RFCOMM_FRAME_TYPE_UIH macro 808
RFCOMM_INFO_LEN macro 49
RFCOMM_LINE_STATUS_FRAMING macro 809
RFCOMM_LINE_STATUS_OVERRUN macro 809
RFCOMM_LINE_STATUS_PARITY macro 809
RFCOMM_LOCAL_CREDIT macro 49
RFCOMM_LOCAL_CREDIT_SEND_THRESHOLD_DECL macro 50
RFCOMM_MAX_CMD_BUFFERS macro 49
RFCOMM_MAX_DLCS macro 49
RFCOMM_MAX_INFO_LEN macro 809
RFCOMM_MAX_SERVER_CHANNELS macro 50
RFCOMM_MAX_SESSIONS macro 50
rfcomm_mgr.h 1150
RFCOMM_MIX_INFO_LEN macro 830
RFCOMM_MODEM_STATUS_DV macro 809
RFCOMM_MODEM_STATUS_FC macro 810
RFCOMM_MODEM_STATUS_IC macro 810
RFCOMM_MODEM_STATUS_RTC macro 810
RFCOMM_MODEM_STATUS_RTR macro 810
rfcomm_mx.h 1150
RFCOMM_MX_MSG_FCOFF macro 810
RFCOMM_MX_MSG_FCON macro 811
RFCOMM_MX_MSG_MAX_DATA_LEN macro 831
RFCOMM_MX_MSG_MSC macro 811
RFCOMM_MX_MSG_NSC macro 811
RFCOMM_MX_MSG_PN macro 811
RFCOMM_MX_MSG_RLS macro 811
RFCOMM_MX_MSG_RPN macro 812
RFCOMM_MX_MSG_TEST macro 812
rfcomm_private.h 1151
RFCOMM_RESPONSE macro 812
RFCOMM_ROLE_INITIATOR macro 812
RFCOMM_ROLE_RESPONDER macro 812
RFCOMM_RPN_BAUD_RATE_1152 macro 813
RFCOMM_RPN_BAUD_RATE_192 macro 813
RFCOMM_RPN_BAUD_RATE_2304 macro 813
RFCOMM_RPN_BAUD_RATE_24 macro 813
RFCOMM_RPN_BAUD_RATE_384 macro 813
RFCOMM_RPN_BAUD_RATE_48 macro 814
RFCOMM_RPN_BAUD_RATE_576 macro 814
RFCOMM_RPN_BAUD_RATE_72 macro 814
RFCOMM_RPN_BAUD_RATE_96 macro 814
RFCOMM_RPN_DATA_BIT_5 macro 814
RFCOMM_RPN_DATA_BIT_6 macro 815
RFCOMM_RPN_DATA_BIT_7 macro 815
RFCOMM_RPN_DATA_BIT_8 macro 815
RFCOMM_RPN_FLC_N macro 815
RFCOMM_RPN_FLC_RTC_INPUT macro 815
RFCOMM_RPN_FLC_RTC_OUTPUT macro 816
RFCOMM_RPN_FLC_RTR_INPUT macro 816
RFCOMM_RPN_FLC_RTR_OUTPUT macro 816
RFCOMM_RPN_FLC_XONOFF_INPUT macro 816
RFCOMM_RPN_FLC_XONOFF_OUTPUT macro 816
RFCOMM_RPN_PARITY_EVEN macro 817
RFCOMM_RPN_PARITY_MARK macro 817
RFCOMM_RPN_PARITY_N macro 817
RFCOMM_RPN_PARITY_ODD macro 817
RFCOMM_RPN_PARITY_SPACE macro 817
RFCOMM_RPN_PARITY_Y macro 818
RFCOMM_RPN_STOP_BIT_1 macro 818
RFCOMM_RPN_STOP_BIT_1_5 macro 818
RFCOMM_RPN_XOFF_DEFAULT macro 818
RFCOMM_RPN_XON_DEFAULT macro 818
rfcomm_send_cmd function 789
rfcomm_send_commands_from_queue function 789
rfcomm_send_mx_msc_cmd function 789
rfcomm_send_mx_pn_cmd function 790
RFCOMM_SERIAL_PORT_CH_1 macro 819
RFCOMM_SERIAL_PORT_CH_10 macro 819
RFCOMM_SERIAL_PORT_CH_11 macro 819
RFCOMM_SERIAL_PORT_CH_12 macro 819
RFCOMM_SERIAL_PORT_CH_13 macro 819
RFCOMM_SERIAL_PORT_CH_14 macro 820
RFCOMM_SERIAL_PORT_CH_15 macro 820
RFCOMM_SERIAL_PORT_CH_16 macro 820
RFCOMM_SERIAL_PORT_CH_17 macro 820
RFCOMM_SERIAL_PORT_CH_18 macro 820
RFCOMM_SERIAL_PORT_CH_19 macro 821
RFCOMM_SERIAL_PORT_CH_2 macro 821
RFCOMM_SERIAL_PORT_CH_20 macro 821
RFCOMM_SERIAL_PORT_CH_21 macro 821
RFCOMM_SERIAL_PORT_CH_22 macro 821
RFCOMM_SERIAL_PORT_CH_23 macro 822
RFCOMM_SERIAL_PORT_CH_24 macro 822
RFCOMM_SERIAL_PORT_CH_25 macro 822
RFCOMM_SERIAL_PORT_CH_26 macro 822
RFCOMM_SERIAL_PORT_CH_27 macro 822
RFCOMM_SERIAL_PORT_CH_28 macro 823
RFCOMM_SERIAL_PORT_CH_29 macro 823
RFCOMM_SERIAL_PORT_CH_3 macro 823
RFCOMM_SERIAL_PORT_CH_30 macro 823
RFCOMM_SERIAL_PORT_CH_4 macro 823
RFCOMM_SERIAL_PORT_CH_5 macro 824
RFCOMM_SERIAL_PORT_CH_6 macro 824
RFCOMM_SERIAL_PORT_CH_7 macro 824
RFCOMM_SERIAL_PORT_CH_8 macro 824
RFCOMM_SERIAL_PORT_CH_9 macro 824
RFCOMM_SESSION_CHANGED_AFC macro 825
RFCOMM_SESSION_CHANGED_CONN_STATE macro 825
RFCOMM_SESSION_STATE_CONNECTED macro 825

RFCOMM_SESSION_STATE_DISCONNECTED macro 825
RFCOMM_SESSION_STATE_FREE macro 825
rfcomm_signal.h 1152
RFCOMM_TIMEOUT macro 826
rfcomm_timer.h 1152

S

sbc.h 1152
SBC_ALLOCATION_METHOD_ALL macro 265
SBC_ALLOCATION_METHOD_LOUDNESS macro 266
SBC_ALLOCATION_METHOD_SNR macro 266
SBC_AM_LOUDNESS macro 835
SBC_AM_SNR macro 835
SBC_BE macro 835
SBC_BLK_12 macro 836
SBC_BLK_16 macro 836
SBC_BLK_4 macro 836
SBC_BLK_8 macro 836
SBC_BLOCK_LENGTH_12 macro 266
SBC_BLOCK_LENGTH_16 macro 266
SBC_BLOCK_LENGTH_4 macro 266
SBC_BLOCK_LENGTH_8 macro 267
SBC_BLOCK_LENGTH_ALL macro 267
SBC_CHANNEL_MODE_ALL macro 267
SBC_CHANNEL_MODE_DUAL_CHANNEL macro 267
SBC_CHANNEL_MODE_JOINT_STEREO macro 267
SBC_CHANNEL_MODE_MONO macro 268
SBC_CHANNEL_MODE_STEREO macro 268
sbc_decode function 831
SBC_FREQ_16000 macro 836
SBC_FREQ_32000 macro 837
SBC_FREQ_44100 macro 837
SBC_FREQ_48000 macro 837
sbc_get_codesize function 831
sbc_get_frame_duration function 832
sbc_get_frame_length function 832
sbc_get_state_size function 833
sbc_init function 833
SBC_LE macro 837
SBC_MODE_DUAL_CHANNEL macro 837
SBC_MODE_JOINT_STEREO macro 838
SBC_MODE_MONO macro 838
SBC_MODE_STEREO macro 838
SBC_SAMPLING_FREQUENCY_16000 macro 268
SBC_SAMPLING_FREQUENCY_32000 macro 268
SBC_SAMPLING_FREQUENCY_44100 macro 268
SBC_SAMPLING_FREQUENCY_48000 macro 269
SBC_SAMPLING_FREQUENCY_ALL macro 269
SBC_SB_4 macro 838
SBC_SB_8 macro 838
sbc_struct structure 833
SBC_SUBBANDS_4 macro 269
SBC_SUBBANDS_8 macro 269
SBC_SUBBANDS_ALL macro 269
sbc_t type 834
sdp.h 1153
SDP_ALLOCATE_BUFFERS function 18
SDP_ALLOCATE_BUFFERS_RAM_SIZE_VAR macro 50
SDP_ALLOCATE_BUFFERS_VARS macro 51
SDP_ATTRID_AdditionalProtocolDescriptorLists macro 847
SDP_ATTRID_BluetoothProfileDescriptorList macro 847
SDP_ATTRID_BrowseGroupList macro 847
SDP_ATTRID_ClientExecutableURL macro 848
SDP_ATTRID_DIPrimaryRecord macro 848
SDP_ATTRID_DIProductId macro 848
SDP_ATTRID_DISpecificationId macro 848
SDP_ATTRID_DIVendorId macro 848
SDP_ATTRID_DIVendorIdSource macro 849
SDP_ATTRID_DIVersion macro 849
SDP_ATTRID_DocumentationURL macro 849
SDP_ATTRID_GAPRemoteAudioVolumeControl macro 849
SDP_ATTRID_GroupID macro 849
SDP_ATTRID_HCRP_1284ID macro 883
SDP_ATTRID_HCRP_DeviceLocation macro 883
SDP_ATTRID_HCRP_DeviceName macro 883
SDP_ATTRID_HCRP_FriendlyName macro 883
SDP_ATTRID_HDPDataExchangeSpecification macro 850
SDP_ATTRID_HDPMCAPSupportedProcedures macro 850
SDP_ATTRID_HDPSuportedFeatures macro 850
SDP_ATTRID_HFPAGNetwork macro 850
SDP_ATTRID_HFPSupportedFeatures macro 850
SDP_ATTRID_HIDBatteryPower macro 851
SDP_ATTRID_HIDBootDevice macro 851
SDP_ATTRID_HIDCountryCode macro 851
SDP_ATTRID_HIDDescriptorList macro 851
SDP_ATTRID_HIDDeviceReleaseNumber macro 851
SDP_ATTRID_HIDDeviceSubclass macro 852
SDP_ATTRID_HIDLANGIDBaseList macro 852
SDP_ATTRID_HIDNormallyConnectable macro 852
SDP_ATTRID_HIDParserVersion macro 852
SDP_ATTRID_HIDProfileVersion macro 852
SDP_ATTRID_HIDReconnectInitiate macro 853
SDP_ATTRID_HIDRemoteWake macro 853
SDP_ATTRID_HIDSDPDisable macro 853
SDP_ATTRID_HIDSupervisionTimeout macro 853
SDP_ATTRID_HIDVirtualCable macro 853
SDP_ATTRID_IconURL macro 854
SDP_ATTRID_INVALID macro 854
SDP_ATTRID_LanguageBaseAttributeIDList macro 854
SDP_ATTRID_OFFSET_ProviderName macro 854
SDP_ATTRID_OFFSET_ServiceDescription macro 854
SDP_ATTRID_OFFSET_ServiceName macro 855
SDP_ATTRID_PrimaryLanguageBaseId macro 855
SDP_ATTRID_ProtocolDescriptorList macro 855
SDP_ATTRID_ServiceAvailability macro 855
SDP_ATTRID_ServiceClassIDList macro 855
SDP_ATTRID_ServiceDatabaseState macro 856
SDP_ATTRID_ServiceID macro 856
SDP_ATTRID_ServiceInfoTimeToLive macro 856
SDP_ATTRID_ServiceRecordHandle macro 856
SDP_ATTRID_ServiceRecordState macro 856
SDP_ATTRID_SupportedFeatures macro 857
SDP_ATTRID_VersionNumberList macro 857
sdp_client.h 1159
SDP_CLIENT_EVT_CONNECTED macro 857
SDP_CLIENT_EVT_CONNECTION_FAILED macro 882

SDP_CLIENT_EVT_DISCONNECTED macro 857
SDP_CLIENT_EVT_NULL macro 857
SDP_CLSID_ADVANCED_AUDIO_DISTRIBUTION macro 858
SDP_CLSID_AUDIO_SINK macro 858
SDP_CLSID_AUDIO_SOURCE macro 858
SDP_CLSID_AV_REMOTE_CONTROL macro 858
SDP_CLSID_AV_REMOTE_CONTROL_CONTROLLER macro 858
SDP_CLSID_AV_REMOTE_CONTROL_PROFILE_ID macro 859
SDP_CLSID_AV_REMOTE_CONTROL_TARGET macro 859
SDP_CLSID_AVCTP macro 859
SDP_CLSID_AVDTP macro 859
SDP_CLSID_BrowseGroupDescriptorServiceClassID macro 859
SDP_CLSID_DialupNetworking macro 860
SDP_CLSID_GENERIC_AUDIO macro 860
SDP_CLSID_HARD_COPY_CABLE_REPLACEMENT macro 883
SDP_CLSID_HARD_COPY_CONTROL_CHANNEL macro 884
SDP_CLSID_HARD_COPY_DATA_CHANNEL macro 884
SDP_CLSID_HARD_COPY_NOTIFICATION macro 884
SDP_CLSID_HCR_PRINT macro 884
SDP_CLSID_HCR_SCAN macro 884
SDP_CLSID_HDP macro 860
SDP_CLSID_HDP_SINK macro 860
SDP_CLSID_HDP_SOURCE macro 860
SDP_CLSID_HFP macro 861
SDP_CLSID_HFP_AG macro 861
SDP_CLSID_HID macro 861
SDP_CLSID_HIDProtocol macro 861
SDP_CLSID_HSP macro 861
SDP_CLSID_HSP_AG macro 862
SDP_CLSID_HSP_HS macro 862
SDP_CLSID_L2CAP macro 862
SDP_CLSID_MCAP_CONTROL macro 862
SDP_CLSID_MCAP_DATA macro 862
SDP_CLSID_OBEXFileTransfer macro 863
SDP_CLSID_OBEXObjectPush macro 863
SDP_CLSID_PBAP_PCE macro 863
SDP_CLSID_PBAP_PSE macro 863
SDP_CLSID_PNPInformation macro 863
SDP_CLSID_PublicBrowseGroup macro 864
SDP_CLSID_RFCOMM macro 864
SDP_CLSID_SerialPort macro 864
SDP_CLSID_ServiceDiscoveryServerServiceClassID macro 864
sdp_compare_uuid_de function 840
sdp_config.h 1160
SDP_DATA_TYPE_ALTERNATIVE macro 864
SDP_DATA_TYPE_BOOL macro 865
SDP_DATA_TYPE_INT macro 865
SDP_DATA_TYPE_INT128 macro 865
SDP_DATA_TYPE_INT16 macro 865
SDP_DATA_TYPE_INT32 macro 865
SDP_DATA_TYPE_INT64 macro 866
SDP_DATA_TYPE_INT8 macro 866
SDP_DATA_TYPE_NIL macro 866
SDP_DATA_TYPE_SEQUENCE macro 866
SDP_DATA_TYPE_STRING macro 866
SDP_DATA_TYPE_UINT macro 867
SDP_DATA_TYPE_UINT128 macro 867
SDP_DATA_TYPE_UINT16 macro 867
SDP_DATA_TYPE_UINT32 macro 867
SDP_DATA_TYPE_UINT64 macro 867
SDP_DATA_TYPE_UINT8 macro 868
SDP_DATA_TYPE_URL macro 868
SDP_DATA_TYPE_UUID macro 868
SDP_DATA_TYPE_UUID128 macro 868
SDP_DATA_TYPE_UUID16 macro 868
SDP_DATA_TYPE_UUID32 macro 869
SDP_ERROR_INSUFFICIENT_RESOURCE macro 869
SDP_ERROR_INVALID_CONTINUATION_STATE macro 869
SDP_ERROR_INVALID_PDU_SIZE macro 869
SDP_ERROR_INVALID_REQUEST_SYNTAX macro 869
SDP_ERROR_INVALID_SDP_VERSION macro 870
SDP_ERROR_INVALID_SR_HANDLE macro 870
SDP_ERROR_RESERVED macro 870
SDP_ErrorResponse macro 870
sdp_find_attributes function 840
sdp_find_service_records2 function 840
SDP_FTP_SERVICE_ID macro 870
SDP_HID_SERVICE_ID macro 871
SDP_HSP_AG_SERVICE_ID macro 871
SDP_HSP_HS_SERVICE_ID macro 871
SDP_MAX_ATTRIBUTE_PATTERN_LEN macro 871
SDP_MAX_ATTRIBUTE_RESULT_LEN macro 52
SDP_MAX_DATA_ELEMENT_LEN macro 871
SDP_MAX_DATA_ELEMENTS macro 872
SDP_MAX_PDU_BUFFERS macro 52
SDP_MAX_SEARCH_PATTERN_LEN macro 872
SDP_MAX_SEARCH_RESULT_LEN macro 52
SDP_MAX_TRANSACTIONS macro 872
sdp_packet.h 1160
SDP_PDU_HEADER_LEN macro 872
sdp_private.h 1160
SDP_RFCOMM_SERVICE_ID macro 872
SDP_ServiceAttributeRequest macro 873
SDP_ServiceAttributeResponse macro 873
SDP_ServiceSearchAttributeRequest macro 873
SDP_ServiceSearchAttributeResponse macro 873
SDP_ServiceSearchRequest macro 873
SDP_ServiceSearchResponse macro 874
SDP_SR_HANDLE_HDP_SINK macro 874
SDP_SR_HANDLE_HDP_SOURCE macro 874
SDP_SR_HANDLE_HFP_HF macro 874
SDP_SR_HANDLE_HID macro 874
SDP_SR_HANDLE_HID_KEYBOARD macro 875
SDP_SR_HANDLE_HSP_HS macro 875
SDP_SR_HANDLE_OBEXFileTransfer macro 875
SDP_SR_HANDLE_OBEXObjectPush macro 875
SDP_SR_HANDLE_PNPINFORMATION macro 875
SDP_SR_HANDLE_RFCOMM macro 876
SDP_SR_HANDLE_SERVER macro 876
SDP_SR_HANDLE_TEST macro 876
sdp_utils.h 1161
SET_F_BIT macro 762
SET_FRAME_TYPE macro 762
SET_P_BIT macro 762
SET_PS_VALUE_BDADDR macro 939
SET_PS_VALUE_UINT16 macro 939

SET_PS_VALUE_UINT32 macro 939
SET_REQ_SEQ macro 763
SET_S_FUNCTION macro 763
SET_SAR macro 763
SET_TX_SEQ macro 763
SMP_ALLOCATE_BUFFERS function 18
spp.h 1161
SPP_ALLOCATE_BUFFERS function 19
SPP_ALLOCATE_BUFFERS_RAM_SIZE_VAR macro 52
SPP_ALLOCATE_BUFFERS_VARS macro 53
spp_config.h 1165
SPP_DECLARE_FRAME_BUFFERS macro 53
SPP_DISABLE_BUFFERING macro 53
SPP_FRAME_BUFFERS_RAM_SIZE macro 53
SPP_FRAME_BUFFERS_SIZE macro 53
SPP_MAX_PORTS macro 54
SPP_PORT_EVENT_CONNECTED enumeration member 894
SPP_PORT_EVENT_CONNECTION_FAILED enumeration member 894
SPP_PORT_EVENT_DISCONNECTED enumeration member 894
SPP_PORT_EVENT_LOCAL_MODEM_STATUS_CHANGE_FAILED enumeration member 894
SPP_PORT_EVENT_LOCAL_MODEM_STATUS_CHANGED enumeration member 894
SPP_PORT_EVENT_REMOTE_MODEM_STATUS_CHANGED enumeration member 894
SPP_PORT_EVENT_SEND_PROGRESS enumeration member 894
SPP_PORT_OPTION_ENCRYPTED macro 892
SPP_PORT_OPTION_MASTER macro 892
SPP_PORT_OPTION_SECURE macro 892
SPP_PORT_STATE_CONNECTED enumeration member 895
SPP_PORT_STATE_CONNECTING enumeration member 895
SPP_PORT_STATE_DISCONNECTED enumeration member 895
SPP_PORT_STATE_DISCONNECTING enumeration member 895
SPP_PORT_STATE_FREE enumeration member 895
SPP_PORT_TYPE_INCOMING macro 892
SPP_PORT_TYPE_OUTGOING macro 892
spp_private.h 1166
SPP_RS232_CTS macro 893
SPP_RS232_DCD macro 893
SPP_RS232_DSR macro 893
SPP_RS232_DTR macro 893
SPP_RS232_RI macro 894
SPP_RS232_RTS macro 894
SPP_SEND_STATUS_CANCELED enumeration member 896
SPP_SEND_STATUS_INTERRUPTED enumeration member 896
SPP_SEND_STATUS_NOT_ENOUGH_RESOURCES enumeration member 896
SPP_SEND_STATUS_SUCCESS enumeration member 896
SPP_SEND_STATUS_TIMEOUT enumeration member 896
ssize_t macro 839
ssp.h 1166
SSP_AUTHENTICATION_REQUIREMENTS enumeration 904
SSP_EVENT enumeration 906
ssp_event.h 1167
ssp_event_handler.h 1168
SSP_EVENT_IO_CAPABILITY_REQUEST enumeration member 906
SSP_EVENT_IO_CAPABILITY_RESPONSE enumeration member 906
SSP_EVENT_KEYPRESS_NOTIFICATION enumeration member 906
SSP_EVENT_OOB_DATA_REQUEST enumeration member 906
SSP_EVENT_SIMPLE_PAIRING_COMPLETE enumeration member 906
SSP_EVENT_USER_CONFIRMATION_REQUEST enumeration member 906
SSP_EVENT_USER_PASSKEY_NOTIFICATION enumeration member 906
SSP_EVENT_USER_PASSKEY_REQUEST enumeration member 906
ssp_evt_io_capability_request function 899
ssp_evt_io_capability_response function 899
ssp_evt_keypress_notification function 899
ssp_evt_oob_data_request function 899
ssp_evt_ssp_complete function 900
ssp_evt_user_confirmation_request function 900
ssp_evt_user_passkey_notification function 900
ssp_evt_user_passkey_request function 900
SSP_IO_CAPABILITY enumeration 905
SSP_IO_CAPABILITY_DISPLAY_ONLY enumeration member 905
SSP_IO_CAPABILITY_DISPLAY_YESNO enumeration member 905
SSP_IO_CAPABILITY_KEYBOARD_ONLY enumeration member 905
SSP_IO_CAPABILITY_NO_INPUT_NO_OUTPUT enumeration member 905
SSP_KEYPRESS_NOTIFICATION_TYPE enumeration 905
SSP_MAX_MANAGERS macro 901
SSP_MITM_NOT_REQUIRED_DEDICATED_BONDING enumeration member 904
SSP_MITM_NOT_REQUIRED_GENERAL_BONDING enumeration member 904
SSP_MITM_NOT_REQUIRED_NO_BONDING enumeration member 904
SSP_MITM_REQUIRED_DEDICATED_BONDING enumeration member 904
SSP_MITM_REQUIRED_GENERAL_BONDING enumeration member 904
SSP_MITM_REQUIRED_NO_BONDING enumeration member 904
SSP_MODE enumeration 905
SSP_MODE_DISABLED enumeration member 905
SSP_MODE_ENABLED enumeration member 905
SSP_OOB_DATA_NOT_PRESENT enumeration member 906
SSP_OOB_DATA_PRESENT enumeration 906
SSP_OOB_DATA_REMOTE_DEVICE_DATA_PRESENT enumeration member 906

T

ti.h 1168
ti_private.h 1171
TRUE macro 216
types.h 1171

U

Using the Library 5
    PIC32 Bluetooth Stack Library 5

V

VCARD_CALL_TYPE_DIALED macro 981
VCARD_CALL_TYPE_MISSED macro 981
VCARD_CALL_TYPE_RECEIVED macro 981
VCARD_EVT_PROPERTY_PARAM macro 981
VCARD_EVT_PROPERTY_STARTED macro 981
VCARD_EVT_PROPERTY_VALUE macro 982
VCARD_EVT_VCARD_ENDED macro 982
VCARD_EVT_VCARD_STARTED macro 982
VCARD_PARAM_CALL_TYPE macro 982

VCARD_PARAM_CELL macro 982
VCARD_PARAM_DIALED macro 983
VCARD_PARAM_ENCODING macro 983
VCARD_PARAM_HOME macro 983
VCARD_PARAM_LANGUAGE macro 983
VCARD_PARAM_MISSED macro 983
VCARD_PARAM_RECEIVED macro 984
VCARD_PARAM_TYPE macro 984
VCARD_PARAM_VALUE macro 984
VCARD_PARAM_WORK macro 984
vcard_parser.h 1171
VCARD_PARSER_STATE_READ_PARAM_NAME macro 984
VCARD_PARSER_STATE_READ_PARAM_VALUE macro 985
VCARD_PARSER_STATE_READ_TYPE_NAME macro 985
VCARD_PARSER_STATE_READ_TYPE_VALUE macro 985
VCARD_PARSER_STATE_SKIP_PARAM macro 985
VCARD_PARSER_STATE_SKIP_TYPE macro 985
VCARD_PROPERTY_ADR macro 986
VCARD_PROPERTY_AGENT macro 986
VCARD_PROPERTY_BDAY macro 986
VCARD_PROPERTY_BEGIN macro 986
VCARD_PROPERTY_CATEGORIES macro 986
VCARD_PROPERTY_CLASS macro 987
VCARD_PROPERTY_EMAIL macro 987
VCARD_PROPERTY_END macro 987
VCARD_PROPERTY_FN macro 987
VCARD_PROPERTY_GEO macro 987
VCARD_PROPERTY_KEY macro 988
VCARD_PROPERTY_LABEL macro 988
VCARD_PROPERTY_LOGO macro 988
VCARD_PROPERTY_MAILER macro 988
VCARD_PROPERTY_N macro 988
VCARD_PROPERTY_NICKNAME macro 989
VCARD_PROPERTY_NOTE macro 989
VCARD_PROPERTY_ORG macro 989
VCARD_PROPERTY_PHOTO macro 989
VCARD_PROPERTY_PROID macro 989
VCARD_PROPERTY_REV macro 990
VCARD_PROPERTY_ROLE macro 990
VCARD_PROPERTY_SORT_STRING macro 990
VCARD_PROPERTY_SOUND macro 990
VCARD_PROPERTY_TEL macro 990
VCARD_PROPERTY_TITLE macro 991
VCARD_PROPERTY_TZ macro 991
VCARD_PROPERTY_UID macro 991
VCARD_PROPERTY_URL macro 991
VCARD_PROPERTY_VERSION macro 991
VCARD_PROPERTY_X_IRMC_CALL_DATETIME macro 992

W

W macro 940

X

XML_EVT_ATTR_NAME macro 992
XML_EVT_ATTR_VALUE macro 992
XML_EVT_TAG_ENDED macro 992
XML_EVT_TAG_STARTED macro 992
xml_parser.h 1173
XML_PARSER_STATE_READ_ATTR_NAME macro 993
XML_PARSER_STATE_READ_ATTR_VALUE macro 993
XML_PARSER_STATE_READ_TAG_NAME macro 993
XML_PARSER_STATE_READ_TAG_VALUE macro 993
XML_PARSER_STATE_SKIP_ATTR macro 993
XML_PARSER_STATE_SKIP_COMMENTS macro 994
XML_PARSER_STATE_SKIP_PROC_INST macro 994
XML_PARSER_STATE_SKIP_TAG macro 994
XML_PARSER_STATE_WAIT_ATTR_NAME macro 994
XML_PARSER_STATE_WAIT_ATTR_VALUE macro 994
XML_PARSER_STATE_WAIT_TAG_CLOSE macro 995
XML_PARSER_STATE_WAIT_TAG_NAME macro 995