CySmart™ API Reference Guide

Document No.: 002-11435 Rev *A

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709
Phone (USA): 800.858.1810
Phone (Intnl): +1.408.943.2600
www.cypress.com

Copyrights

Contents

Introduction **7**

CySmart Subsystem Overview 7

CySmart APIs 8

Dongle Communicator 8

Dongle Transport Channel 8

Development Platform 8

API Design **9**

ICySmartDongleCommunicator 10

ICyBleMgr 10

ICyBleDevice 10

ICyGattClient 11

ICyL2CapMgr 11

ICyL2CapChannel 11

ICyBleDeviceList 11

ICyBleSecurityMgr 12

ICyBleDeviceAddressMgr 12

API Reference **13**

Enumeration Type Documentation 13

CyAddressResolutionControlInfo 30

CyAddToResolvingListInfo 31

CyAdvertisementData 32

CyAdvertisementDataItem 33

CyApiErr 38

CyAuthenticationKeys 41

CyBleBdAddress 43

CyBleConnectionSettings 45

CyBleDeviceCallback 50

CyBleMgrCallback 54

CyBleScanSettings 59

CyBondListDevice 64

CyChannelClassificationInfo 64

CyCharacteristicChangedInfo 68
CyConnectInfo 68
CyConnectionParameters 69
CyConnectResult 71
CyConvertOctetToTimeInfo 72
CyConvertOctetToTimeResult 73
CyCurrentConnectionParameters 73
CyCurrentDataLength 74
CyDataLengthInfo 75
CyDefaultDataLengthResult 76
CyDeviceAddressMgrCallback 78
CyDeviceListCallback 80
CyDiscoverAllServicesResult 83
CyDiscoverCharacteristicsByUUIDInfo 84
CyDiscoverCharacteristicsCallback 85
CyDiscoverCharacteristicsInfo 86
CyDiscoverCharacteristicsResult 87
CyDiscoverDescriptorsCallback 88
CyDiscoverDescriptorsInfo 88
CyDiscoverDescriptorsResult 90
CyDiscoverPrimaryServiceCallback 90
CyDiscoverPrimaryServicesByUUIDInfo 91
CyDiscoverPrimaryServicesResult 92
CyDongleInfo 92
CyEstablishL2CapChannelInfo 94
CyFindIncludedServicesCallback 96
CyFindIncludedServicesInfo 97
CyFindIncludedServicesResult 98
CyGattAttribute 98
CyGattCharacteristic 100
CyGattClientCallback 101
CyGattDescriptor 105
CyGattExchangeMtuInfo 106
CyGattExchangeMtuResult 107
CyGattIncludedService 108
CyGattReadInfo 109
CyGattReadResult 111
CyGattService 112

CyGattWriteInfo 114
CyGattWriteResult 117
CyGenerateBdAddressInfo 117
CyGenerateBdAddressResult 118
CyGenerateSecureConnectionOobDataInfo 119
CyGetPeerDeviceAuthenticationKeyInfo 120
CyGetPeerDeviceAuthenticationKeyResult 121
CyL2CapConnectionResponseInfo 121
CyL2CapDataReceivedInfo 123
CyL2CapDisconnectConfirmation 124
CyL2CapDisconnectIndicationInfo 125
CyL2CapMgrCallback 126
CyL2CapReceiveCreditLowInfo 129
CyL2CapSendCreditsInfo 130
CyL2CapSendDataInfo 131
CyL2CapTransmitCreditInfo 132
CyNumericComparisonResponse 133
CyOobData 133
CyPairSettings 135
CyPasskeyDisplayInfo 136
CyPasskeyEntryResponse 137
CyReadCharacteristicByUUIDInfo 138
CyReadCharacteristicByUUIDResult 139
CyReadMultipleCharacteristicInfo 139
CyReadMultipleCharacteristicResult 141
CyRegisteredPsm 141
CyRegisterPsmInfo 142
CyReliableWriteInfo 143
CyResolvableAddressResult 144
CyResolvableAddressTimeoutInfo 144
CyResolvePeerDeviceInfo 145
CyResolvingListDevice 146
CyScanCallback 147
CyScanRecord 149
CyScanResult 150
CySecureConnectionOobDataResult 150
CySecurityMgrCallback 152
CySetSuggestedDataLengthInfo 154

CySmartDongleMgr 156
CyTxPowerInfo 157
CyUUID 158
CyWhitelistDevice 160
CyWriteBufferFullResponse 160
ICyBleDevice 161
ICyBleDeviceAddressMgr 166
ICyBleDeviceList 169
ICyBleMgr 172
ICyBleSecurityMgr 180
ICyGattClient 183
ICyL2CapChannel 192
ICyL2CapMgr 194
ICySmartDongleCommunicator 196
Revision History 198
Document Revision History 198

Introduction

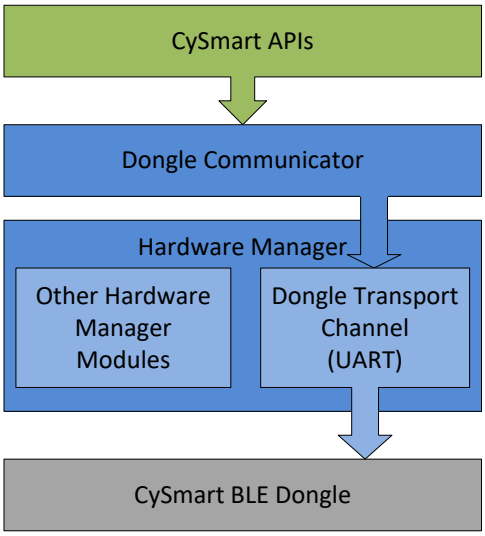
CySmart™ is a Bluetooth® LE (BLE) host emulation tool for Windows PCs. The tool uses the CySmart BLE dongle to emulate a Generic Access Profile (GAP) central device. The tool provides an easy-to-use Graphical User Interface (GUI) to enable customers to test and debug their Bluetooth LE peripheral applications. In addition to the GUI, the tool also provides C# APIs to communicate with the dongle. These APIs allow custom GAP central applications to be developed.

This document serves as the reference guide for the APIs exposed by the tool. The document also captures the high level API design, development platform and general usage guidelines for the APIs.

CySmart Subsystem Overview

Figure 1 shows an overview of the CySmart APIs module and the other CySmart modules that the API module interacts. The following sections provide a brief description of these modules.

Figure 1: Subsystem Overview



CySmart APIs

The CySmart APIs module exposes the APIs to build custom GAP central applications. The module is essentially a wrapper over the ‘Dongle Communicator’ module and abstracts the complexities of the dongle communicator module. See API Design section for details about the design.

Dongle Communicator

This module understands the CySmart BLE protocol and implements the protocol communication state machine. This module is designed to be independent of the transport channel to the CySmart BLE dongle.

Dongle Transport Channel

This module is part of the ‘Hardware Manager’ module and implements the transport channel specific interfaces and methods. At present, only UART transport channel is supported by the CySmart BLE dongle.

Development Platform

The CySmart APIs are developed in C# and built for .NET framework 2.0. To use the CySmart APIs, add reference to the followings CySmart DLLs in your C# project. The DLLs can be located in the CySmart installation directory.

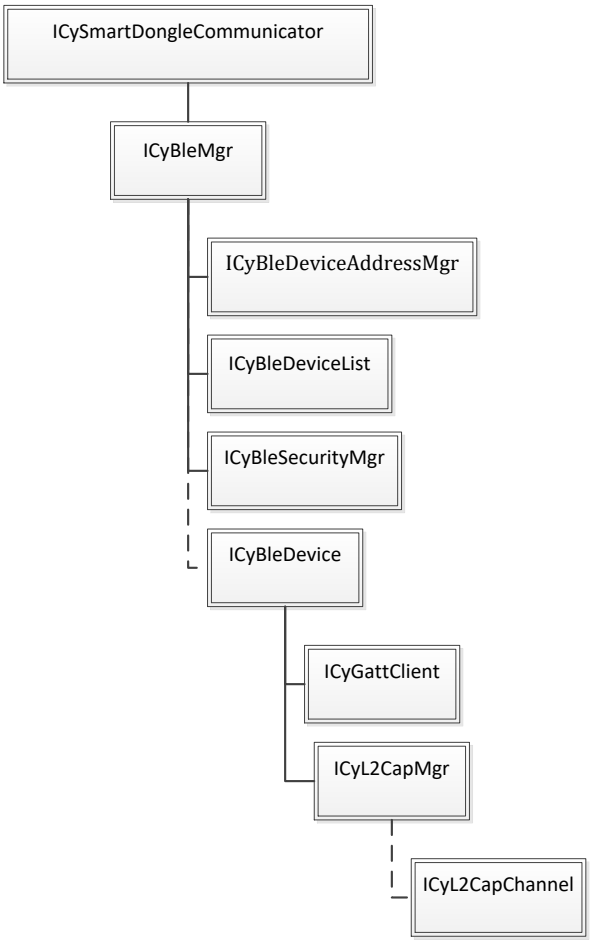
- cybledonglecommunicator.dll
- cybleautobase.dll
- cyblecommonbase.dll

All CySmart APIs are defined under the CySmart.DongleCommunicator.API namespace in the cybledonglecommunicator.dll. The other two DLLs contain data structure and utility methods used by the API implementation.

API Design

CySmart APIs communicates with the CySmart dongle to perform a BLE operation. Due to the nature of communication (interface with hardware), the APIs are asynchronous and uses callback mechanism to report the result and status of an operation. Depending on the purpose of the APIs, the APIs are categorized into multiple modules.

Each module defines its own callback interface and provides a mechanism to register the callbacks. The callbacks need to be implemented by the API client. When implementing the callback method, ensure that long running operations are not performed on the thread in which the callback is invoked. If long running operations needs to performed, execute them on a different thread.



ICySmartDongleCommunicator

This interface provides the following information about the CySmart BLE dongle and provides access to the ICyBleMgr interface:

- Device ID – Identifies whether the dongle supports BLE version 4.1 or 4.2
- BLE stack version
- Firmware version
- Hardware version

Use CySmartDongleMgr class methods to get an instance of this interface.

ICyBleMgr

This interface represents the BLE manager and provides APIs to perform the following BLE GAP central operations:

- Scan
- Connect
- Set device IO capabilities
- Register and Unregister L2CAP PSMs
- Set Tx power level for advertisement and connection channel
- Set host channel map
- Set data length (BLE version 4.2 only)
- Enable / Disable Privacy 1.2 (BLE version 4.2 only)

The interface also provides access to the following interfaces

- ICyBleDevice
- ICyBleDeviceList
- ICyBleSecurityMgr
- ICyBleDeviceAddressMgr

ICyBleDevice

An instance of this interface is created when a connection with a peer device is established. This interface represents an active connection with a peer device. This interface provides the following information about the peer device

- Device address
- Connection handle

The interface provides APIs to perform the following operations:

- Pair
- Update data length (BLE version 4.2 only)
- Update connection parameters

The following interfaces can be accessed from this interface

- ICyGattClient
- ICyL2CapMgr

ICyGattClient

This represents the GATT client interface of the connected peer BLE device. This interface provides APIs to perform the following operations

- Negotiate GATT MTU size
- Discover services and characteristics
- Read and write characteristics
- Read and write characteristic descriptors
- Receive notifications and indications

ICyL2CapMgr

This represents the L2CAP Credit Based Flow Control (CBFC) channel manager interface for the connected peer BLE device. This interface provides APIs to perform the following operations:

- Create new L2CAP channel
- Accept L2CAP channel request from peer device
- Disconnect L2CAP channels
- Send and receive data
- Send and receive flow control credits

This interface provides access to the ICyL2CapChannel interface.

ICyL2CapChannel

An instance of this interface is created when an L2CAP CBFC channel is created. This represents an active L2CAP channel. This interface provides the following information about the channel

- Channel ID
- Local MTU and MPS size
- Remote MTU and MPS size

ICyBleDeviceList

This interface represents the device list manager and provides APIs to manage the following lists in the CySmart dongle:

- Whitelist
    - Add, remove and clear whitelist
    - Get devices in the whitelist
- Bond list

    - Get devices in the bond list
- Resolving list (BLE version 4.2 only)
    - Add, remove and clear resolving list
    - Get devices in the resolving list

ICyBleSecurityMgr

This interface represents the security manager of the CySmart dongle and provides APIs to perform the following operations

- Get and set security keys (LTK, IRK, CSRK)
- Generate new security keys
- Get security keys of previously bonded devices
- Generate ECDH keys (BLE version 4.2 only)
- Generate OOB data for secure connection pairing (BLE version 4.2 only)

ICyBleDeviceAddressMgr

This interface provides APIs to manage the CySmart dongle address. The following operations can be performed using the APIs

- Get and set dongle address
- Generate address based on address type
- Set dongle identity address

Enumeration Type Documentation

enum CyConnectionParametersResponse : ushort [strong]

Enumeration of response for connection parameter update request from remote device

Enumerator

ACCEPT Accept the new connection parameters requested by the remote device

REJECT Reject the new connection parameters requested by the remote device

enum CySecurityLevel [strong]

Security level enumeration

Enumerator

NO_SECURITY No security (no authentication and no encryption)

UNAUTHENTICATED_PAIRING_WITH_ENCRYPTION Unauthenticated pairing with encryption (Just works)

AUTHENTICATED_PAIRING_WITH_ENCRYPTION Authenticated pairing with encryption

AUTHENTICATED_LE_SECURE_CONNECTION_PAIRING_WITH_ENCRYPTION Authenticated LE secure connection pairing with encryption

UNAUTHENTICATED_PAIRING_WITH_DATA_SIGNING Unauthenticated pairing with data signing

AUTHENTICATED_PAIRING_WITH_DATA_SIGNING Authenticated pairing with data signing

enum CyPairingProperties : byte [strong]

Pairing properties

Enumerator

NONE None

USE_MITM Use MITM (applicable only for secure connection)

USE_KEYPRESS_NOTIFICATION Use keypress notification (applicable only for secure connection)

enum CyKeyPressNotification : byte [strong]

Key press notification

Enumerator

DIGIT_ENTERED A passkey digit is entered by the user

DIGIT_ERASED A passkey digit is erased by the user

PASSKEY_CLEARED Passkey entry is cleared by the user

enum CyPairingResponseCode : byte [strong]

Pairing response code enumeration

Enumerator

REJECT Reject request

ACCEPT Accept request

enum CyGattCharacteristicProperty : byte [strong]

Enumeration of possible characteristic properties

Enumerator

NONE Unknown This is the initial state for a characteristic

BROADCAST Broadcast

READ Read

WRITE_WITHOUT_RESPONSE Write without response

WRITE Write

NOTIFY Notify

INDICATE Indicate

AUTHENTICATED_SIGNED_WRITES Authenticated signed writes

EXTENDED_PROPERTIES Extended properties

enum CyL2CapConnectionResponseCode : ushort [strong]

Enumeration of responses

Enumerator

CONNECTION_SUCCESSFUL Connection successful

CONNECTION_REFUSED_PSM_NOT_SUPPORTED Connection refused - PSM is not supported

CONNECTION_REFUSED_NO_RESOURCE Connection refused - No resource available

CONNECTION_REFUSED_INSUFFICIENT_AUTHENTICATION Connection refused - Insufficient authentication

CONNECTION_REFUSED_INSUFFICIENT_AUTHORIZATION Connection refused - Insufficient authorization

CONNECTION_REFUSED_INSUFFICIENT_ENCRYPTION_KEY_SIZE Connection refused - Insufficient encryption key size

enum CyL2CapResultCode : ushort [strong]

Enumeration of L2CAP results

Enumerator

SUCCESS Success

COMMAND_TIMEOUT Command timeout. If the L2CAP signaling timeout occurs, the application should disconnect

INCORRECT_SDU_LENGTH Invalid SDU length

NOT_ENOUGH_CREDITS Not enough credits to perform an operation

CREDIT_OVERFLOW Credit overflow. Total credit exceeded 65535 (maximum)

UNACCEPTABLE_CREDIT_VALUE Invalid credit value. The received credit is zero

enum CyScanStatus [strong]

Scan status

Enumerator

IN_PROGRESS Scan is in-progress

STOPPED Scan stopped

ERROR Error starting or stopping scan

enum CyInitiatorAddrType : byte [strong]

Initiator address type enumeration

Enumerator

PUBLIC Use public address

RANDOM Use random address

RESOLVABLE_ADDRESS_OR_PUBLIC Use resolvable private address, if controller address generation is enabled; otherwise use public address

RESOLVABLE_ADDRESS_OR_RANDOM Use resolvable private address, if controller address generation is enabled; otherwise use random address

enum CyScanType : byte [strong]

Enumeration of supported device scan types

Enumerator

PASSIVE_SCAN Passive scan

ACTIVE_SCAN Active scan

enum CyScanInitiatorFilterPolicy : byte [strong]

Scan initiator filter policy

Enumerator

ACCEPT_ALL_ADV Accept all advertisement packets except directed advertising packets not addressed to this device

ACCEPT_ADV_FROM_WHITELIST Accept only advertisement packets from devices where the advertiser's address is in the White list. Directed advertising packets which are not addressed for this device shall be ignored.

ACCEPT_DIRECTED_RPA_ADV Accept all undirected advertisement packets, and directed advertising packets where the initiator address is a resolvable private address, and directed advertising packets addressed to this device.

ACCEPT_WHITELIST_DIRECTED_RPA_ADV Accept all advertisement packets from devices where the advertiser's address is in the White list, and directed advertising packets where the initiator address is a resolvable private address, and directed advertising packets addressed to this device.

enum CyScanDuplicateFilterPolicy : byte [strong]

Scan advertisement duplicate filtering policy

Enumerator

DISABLE_DUPLICATE_FILTERING Disable duplicate filtering

ENABLE_DUPLICATE_FILTERING Enable duplicate filtering

enum CyDiscoveryType : byte [strong]

Enumeration of supported device discovery procedures

Enumerator

OBSERVATION_PROCEDURE Observation procedure - This procedure needs to be used to discover directed advertisements

LIMITED_DISCOVERY Limited discovery - Discovers BLE devices in limited discoverable mode

GENERAL_DISCOVERY General discovery Discovers BLE devices in either limited or general discoverable mode

enum CyConnectionInitiatorFilterPolicy : byte [strong]

Connection initiator filter policy enumeration

Enumerator

USE_PEER_DEVICE_ADDRESS Use the peer device address specified in the connection request. Whitelist is ignored

USE_WHITE_LIST Whitelist is used to determine which advertiser to connect to. Peer device address in the connection request is ignored

enum CyBleDeviceIoCapabilities : byte [strong]

Enumerator

DISPLAY_ONLY Display only

DISPLAY_YES_NO Display and Yes / No input

KEYBOARD_ONLY Keyboard only

NO_INPUT_AND_NO_OUTPUT No input and no output

KEYBOARD_AND_DISPLAY Display and keyboard

UNKNOWN Unknown

enum CyBleBdAddressType : byte [strong]

Bluetooth device address type enumeration

Enumerator

PUBLIC_ADDRESS Public address

RANDOM_ADDRESS Random Address

PUBLIC_ID_ADDRESS Public Identity Address

RANDOM_STATIC_ID_ADDRESS Random (Static) Identity Address

enum CyExpandedBdAddrType : byte [strong]

Enumeration of the expanded Bluetooth device address type

Enumerator

RANDOM_NON_RESOLVABLE Random non-resolvable Bluetooth device address

RANDOM_RESOLVABLE Random resolvable Bluetooth device address

PUBLIC Public Bluetooth device address

RANDOM_STATIC Random static Bluetooth device address

enum CyChannelGroup : byte [strong]

Enumeration of link layer channel groups

Enumerator

ADV_CHANNEL Advertisement channel

CONNECTION_CHANNEL Connection channel

enum CyPowerLevel : byte [strong]

Enumeration of power level modes supported by the system

Enumerator

PWR_LEVEL_NEG_18_DBM -18 dBm

PWR_LEVEL_NEG_12_DBM -12 dBm

PWR_LEVEL_NEG_6_DBM -6 dBm

PWR_LEVEL_NEG_3_DBM -3 dBm

PWR_LEVEL_NEG_2_DBM -2 dBm

PWR_LEVEL_NEG_1_DBM -1 dBm

PWR_LEVEL_0_DBM 0 dBm

PWR_LEVEL_3_DBM 3 dBm

PWR_LEVEL_MAX Max possible

enum CyPhyType : byte [strong]

Physical layer type enumeration

Enumerator

PHY_1MBPS 1 Mbps physical layer

enum CyAuthenticationKeyFlags : byte [strong]

Authentication key distribution flags

Enumerator

NONE No keys are exchanged

INITIATOR_SHARES_ENCRYPTION_INFORMATION Initiator exchanges LTK and Master Identification keys

INITIATOR_SHARES_IDENTITY_INFORMATION Initiator exchanges IRK and Identification address information

INITIATOR_SHARES_SIGNATURE_KEY Initiator exchanges the CSRK information used for data signing

RESPONDER_SHARES_ENCRYPTION_INFORMATION Requests responder to distribute LTK and Master Identity keys

RESPONDER_SHARES_IDENTITY_INFORMATION Requests responder to distribute IRK and Identification address information

RESPONDER_SHARES_SIGNATURE_KEY Requests responder to distribute the CSRK information used for data signing

INITIATOR_SHARE_ALL Initiator distributes all the keys

RESPONDER_SHARE_ALL Request responder to distribute all the keys

SHARE_ALL Initiator distributes all the keys and also requests the responder to distribute all keys

enum CyResolveAddressResult : byte [strong]

Result of private address resolution triggered by the client application

Enumerator

RESOLVED Peer device address was resolved and matches with an existing bonded device

NOT_RESOLVED Peer device address resolution failed or did not match with a device in the bond list

enum CyDongleID : uint[strong]

Dongle ID enumeration

Enumerator

UNKNOWN Unknown dongle

DONGLE_BLE_VERSION_4_1 CySmart dongle running BLE version 4.1

DONGLE_BLE_VERSION_4_2 CySmart dongle running BLE version 4.2

enum CyStatus : ushort[strong]

Enumeration of status and error codes

Enumerator

BLE_STATUS_OK Status OK

BLE_ERROR_INVALID_PARAMETER At least one of the input parameters is invalid

BLE_ERROR_INVALID_OPERATION Operation is not permitted

BLE_ERROR_MEMORY_ALLOCATION_FAILED An internal error occurred in the stack

BLE_ERROR_INSUFFICIENT_RESOURCES Insufficient resources to perform requested operation

BLE_ERROR_OOB_NOT_AVAILABLE OOB data not available

BLE_ERROR_NO_CONNECTION Connection is required to perform requested operation. Connection not present

BLE_ERROR_NO_DEVICE_ENTITY No device entity to perform requested operation

BLE_ERROR_REPEATED_ATTEMTS Attempted repeat operation is not allowed

BLE_ERROR_GAP_ROLE GAP role is incorrect

BLE_ERROR_TX_POWER_READ Error reading TC power

BLE_ERROR_BT_ON_NOT_COMPLETED BLE Initialization failed

BLE_ERROR_SECURITY_FAILED Security operation failed

BLE_ERROR_L2CAP_PSM_WRONG_ENCODING L2CAP PSM encoding is incorrect

BLE_ERROR_L2CAP_PSM_ALREADY_REGISTERED L2CAP PSM has already been registered

BLE_ERROR_L2CAP_PSM_NOT_REGISTERED L2CAP PSM has not been registered

BLE_ERROR_L2CAP_CONNECTION_ENTITY_NOT_FOUND L2CAP connection entity not found

BLE_ERROR_L2CAP_CHANNEL_NOT_FOUND L2CAP channel not found

BLE_ERROR_L2CAP_PSM_NOT_IN_RANGE Specified PSM is out of range

BLE_ERROR_DEVICE_ALREADY_EXISTS Device cannot be added to whitelist as it has already been added

BLE_ERROR_FLASH_WRITE_NOT_PERMITTED Write to flash is not permitted

BLE_ERROR_MIC_AUTH_FAILED MIC Authentication failure

BLE_ERROR_HARDWARE_FAILURE Hardware failure

BLE_ERROR_UNSUPPORTED_FEATURE_OR_PARAMETER_VALUE Unsupported feature or parameter value

BLE_ERROR_FLASH_WRITE Error in flash write

BLE_ERROR_CONTROLLER_BUSY Controller busy

BLE_ERROR_MAX_FOR_HOST_STACK Maximum Error Code for Host/Device

BLE_ERROR_NOTIFICATION_DISABLED Characteristic notification is disabled

BLE_ERROR_INDICATION_DISABLED Characteristic indication is disabled

BLE_ERROR_INVALID_STATE The state is not valid for the current operation

BLE_ERROR_HCI_UNKNOWN_HCI_COMMAND Unknown HCI command

BLE_ERROR_HCI_UNKNOWN_CONNECTION_IDENTIFIER Unknown Connection Identifier

BLE_ERROR_HCI_HARDWRE_FAILURE Hardware Failure

BLE_ERROR_HCI_PAGE_TIMEOUT Page Timeout

BLE_ERROR_HCI_AUTHENTICATION_FAILURE Authentication failure

BLE_ERROR_HCI_PIN_OR_KEY_MISSING PIN or Key Missing

BLE_ERROR_HCI_MEMORY_CAPACITY_EXCEEDED Memory Capacity Exceeded

BLE_ERROR_HCI_CONNECTION_TIMEOUT Connection Timeout

BLE_ERROR_HCI_CONNECTION_LIMIT_EXCEEDED Connection Limit Exceeded

BLE_ERROR_HCI_SYNCHRONOUS_CONNECTION_LIMIT_TO_A_DEVICE_EXCEEDED
Synchronous Connection Limit to a Device Exceeded

BLE_ERROR_HCI_ACL_CONNECTION_ALREADY_EXISTS ACL Connection Already Exists

BLE_ERROR_HCI_COMMAND_DISALLOWED Command Disallowed

BLE_ERROR_HCI_CONNECTION_REJECTED_DUE_TO_LIMITED_RESOURCES Connection Rejected Due to Limited Resources

BLE_ERROR_HCI_CONNECTION_REJECTED_DUE_TO_SECURITY_REASONS Connection Rejected Due to Security Reasons

BLE_ERROR_HCI_CONNECTION_REJECTED_DUE_TO_UNACCEPTABLE_BD_ADDR
Connection Rejected Due to Unacceptable Bluetooth device Address

BLE_ERROR_HCI_CONNECTION_ACCEPT_TIMEOUT_EXCEEDED Connection Accept Timeout Exceeded

BLE_ERROR_HCI_UNSUPPORTED_FEATURE_OR_PARAMETER_VALUE Unsupported Feature or Parameter Value

BLE_ERROR_HCI_INVALID_HCI_COMMAND_PARAMETERS Invalid HCI Command Parameters

BLE_ERROR_HCI_REMOTE_USER_TERMINATED_CONNECTION Remote User Terminated Connection

BLE_ERROR_HCI_REMOTE_DEVICE_TERMINATED_CONNECTION_DUE_TO_LOW_RESOURCES Remote Device Terminated Connection Due to Low Resources

BLE_ERROR_HCI_REMOTE_DEVICE_TERMINATED_CONNECTION_DUE_TO_POWER_OFF Remote Device Terminated Connection Due to Power Off

BLE_ERROR_HCI_CONNECTION_TERMINATED_BY_LOCAL_HOST Connection Terminated by Local Host

BLE_ERROR_HCI_REPEATED_ATTEMPTS Repeated Attempts

BLE_ERROR_HCI_PAIRING_NOT_ALLOWED Pairing Not Allowed

BLE_ERROR_HCI_UNKNOWN_LMP_PDU Unknown LMP PDU

BLE_ERROR_HCI_UNSUPPORTED_REMOTE_FEATURE Unsupported Remote Feature / Unsupported LMP Feature

BLE_ERROR_HCI_SCO_OFFSET_REJECTED SCO Offset Rejected

BLE_ERROR_HCI_SCO_INTERVAL_REJECTED SCO Interval Rejected

BLE_ERROR_HCI_SCO_AIR_MODE_REJECTED SCO Air Mode Rejected

BLE_ERROR_HCI_INVALID_LL_PARAMETERS Invalid LMP Parameters / Invalid LL Parameters

BLE_ERROR_HCI_UNSPECIFIED_ERROR Unspecified Error

BLE_ERROR_HCI_UNSUPPORTED_LL_PARAMETER_VALUE Unsupported LMP Parameter Value / Unsupported LL Parameter Value

BLE_ERROR_HCI_ROLE_CHANGE_NOT_ALLOWED Role Change Not Allowed

BLE_ERROR_HCI_LL_RESPONSE_TIMEOUT LMP Response Timeout / LL Response Timeout

BLE_ERROR_HCI_LMP_ERROR_TRANSACTION_COLLISION LMP Error Transaction Collision

BLE_ERROR_HCI_LMP_PDU_NOT_ALLOWED LMP_PDU_NOT_ALLOWED

BLE_ERROR_HCI_ENCRYPTION_MODE_NOT_ACCEPTABLE Encryption Mode Not Acceptable

BLE_ERROR_HCI_LINK_KEY_CANNOT_BE_CHANGED Link Key Cannot be Changed

BLE_ERROR_HCI_REQUESTED_QoS_NOT_SUPPORTED Requested QoS Not Supported

BLE_ERROR_HCI_INSTANT_PASSED Instant Passed

BLE_ERROR_HCI_PAIRING_WITH_UNIT_KEY_NOT_SUPPORTED Pairing with Unit Key Not Supported

BLE_ERROR_HCI_DIFFERENT_TRANSACTION_COLLISION Different Transaction Collision

BLE_ERROR_HCI_QoS_UNACCEPTABLE_PARAMETER QoS Unacceptable Parameter

BLE_ERROR_HCI_QoS_REJECTED QoS Rejected

BLE_ERROR_HCI_CHANNEL_CLASSIFICATION_NOT_SUPPORTED Channel Classification Not Supported

BLE_ERROR_HCI_INSUFFICIENT_SECURITY Insufficient Security

BLE_ERROR_HCI_PARAMETER_OUT_OF_MANDATORY_RANGE Parameter Out of Mandatory Range

BLE_ERROR_HCI_ROLE_SWITCH_PENDING Role Switch Pending

BLE_ERROR_HCI_RESERVED_SLOT_VIOLATION Reserved Slot Violation

BLE_ERROR_HCI_ROLE_SWITCH_FAILED Role Switch Failed

BLE_ERROR_HCI_EXTENDED_INQUIRY_RESPONSE_TOO_LARGE Extended Inquiry Response Too Large

BLE_ERROR_HCI_SECURE_SIMPLE_PAIRING_NOT_SUPPORTED_BY_HOST Secure Simple Pairing Not Supported by Host

BLE_ERROR_HCI_HOST_BUSY_PAIRING Host Busy - Pairing

BLE_ERROR_HCI_CONNECTION_REJECTED_DUE_TO_NO_SUITABLE_CHANNEL_FOUND Connection Rejected due to No Suitable Channel Found

BLE_ERROR_HCI_CONTROLLER_BUSY Controller Busy

BLE_ERROR_HCI_UNACCEPTABLE_CONNECTION_PARAMETERS Unacceptable Connection Parameters

BLE_ERROR_HCI_DIRECTED_ADVERTISING_TIMEOUT Directed Advertising Timeout

BLE_ERROR_HCI_CONNECTION_TERMINATED_DUE_TO_MIC_FAILURE Connection Terminated due to MIC Failure

BLE_ERROR_HCI_CONNECTION_FAILED_TO_BE_ESTABLISHED Connection Failed to be Established

BLE_ERROR_HCI_MAC_CONNECTION_FAILED MAC Connection Failed

BLE_ERROR_HCI_COARSE_CLOCK_ADJUSTMENT_REJECTED Coarse Clock Adjustment Rejected but Will Try to Adjust Using Clock Dragging

FW_ERROR_INSUFFICIENT_RESOURCES Dongle firmware resources are insufficient to complete the operation

BLE_ERROR_GATT_INVALID_HANDLE The attribute handle given was not valid on this server

BLE_ERROR_GATT_READ_NOT_PERMITTED The attribute cannot be read

BLE_ERROR_GATT_WRITE_NOT_PERMITTED The attribute cannot be written

BLE_ERROR_GATT_INVALID_PDU Format of the attribute PDU was invalid

BLE_ERROR_GATT_INSUFFICIENT_AUTHENTICATION An access to an attribute is attempted on an un-authenticated link. The attribute requires authentication before it can be read or written

BLE_ERROR_GATT_REQUEST_NOT_SUPPORTED Attribute server does not support the request received from the client

BLE_ERROR_GATT_INVALID_OFFSET Offset specified was past the end of the attribute

BLE_ERROR_GATT_INSUFFICIENT_AUTHORIZATION Attribute server does not authorize the client. The attribute requires authorization before it can be read or written

BLE_ERROR_GATT_PREPARE_WRITE_QUEUE_FULL Prepare write queue on the server is full. Server cannot accept any more prepare writes from the client

BLE_ERROR_GATT_ATTRIBUTE_NOT_FOUND No attributes of the specified attribute type was found in the given attribute handle range

BLE_ERROR_GATT_ATTRIBUTE_NOT_LONG The attribute cannot be read or written using the Read Long or Write Long request

BLE_ERROR_GATT_INSUFFICIENT_ENC_KEY_SIZE The Encryption Key Size used for encrypting this link is insufficient

BLE_ERROR_GATT_INVALID_ATTRIBUTE_LEN The attribute value length is invalid for the operation

BLE_ERROR_GATT_UNLIKELY_ERROR The attribute request that was requested has encountered an error that was unlikely, and therefore could not be completed as requested

BLE_ERROR_GATT_INSUFFICIENT_ENCRYPTION The attribute requires encryption before it can be read or written

BLE_ERROR_GATT_UNSUPPORTED_GROUP_TYPE The attribute type is not a supported grouping attribute as defined by a higher layer specification

BLE_ERROR_GATT_INSUFFICIENT_RESOURCE Insufficient Resources to complete the request

BLE_ERROR_GATT_HRS_CPT_NOT_SUPPORTED

BLE_ERROR_GATT_CCD_IMPROPERLY_CONFIGURED

BLE_ERROR_GATT_PROCEDURE_ALREADY_IN_PROGRESS

BLE_ERROR_GATT_OUT_OF_RANGE

BLE_ERROR_GATT_OPERATION_TIMEOUT

BLE_ERROR_OPERATION_CANCELLED

BLE_STATUS_GAP_AUTH_OK Error None

BLE_ERROR_GAP_AUTH_PASSKEY_ENTRY_FAILED User input of passkey failed, for example, the user cancelled the operation

BLE_ERROR_GAP_AUTH_ERROR_OOB_DATA_NOT_AVAILABLE Out Of Band data is not available, applicable if NFC is supported

BLE_ERROR_GAP_AUTH_AUTHENTICATION_REQ_NOT_MET Pairing procedure cannot be performed as authentication* requirements cannot be met due to IO capabilities of one or both devices

BLE_ERROR_GAP_AUTH_CONFIRM_VALUE_MISMATCH Confirm value does not match the calculated compare value

BLE_ERROR_GAP_AUTH_PAIRING_NOT_SUPPORTED Pairing is not supported by the device

BLE_ERROR_GAP_AUTH_INSUFFICIENT_ENCRYPTION_KEY_SIZE Insufficient key size for the security requirements of this device

BLE_ERROR_GAP_AUTH_COMMAND_NOT_SUPPORTED command received is not supported

BLE_ERROR_GAP_AUTH_UNSPECIFIED_REASON Pairing failed due to an unspecified reason

BLE_ERROR_GAP_AUTH_REPEATED_ATTEMTS Pairing or authentication procedure is disallowed because too little time * has elapsed since last pairing request or security request

BLE_ERROR_GAP_AUTH_INVALID_PARAMETERS Invalid Parameters in Request - Invalid Command length and Parameter value outside range

BLE_ERROR_GAP_AUTH_DHKEY_CHECK_FAILED DHKey check value received doesn't match with the calculated value

BLE_ERROR_GAP_AUTH_NUMERIC_COMPARISON_FAILED Numeric comparison confirm values do not match

BLE_ERROR_GAP_AUTH_BR_EDR_PAIRING_IN_PROGRESS Pairing over LE link failed due to an on-going pairing over BR/EDR link

BLE_ERROR_GAP_AUTH_CROSS_TRANSPORT_KEY_GEN_DER_NOT_ALLOWED Link Key generated on the BR/EDR transport cannot be used to derive and distribute keys for LE transport

BLE_ERROR_GAP_AUTH_AUTHENTICATION_TIMEOUT Authentication process timeout

BLE_ERROR_GAP_AUTH_LINK_DISCONNECTED Link disconnected

BLE_ERROR_STACK_BUSY BLE stack is busy

HARDWARE_ERROR BLE hardware error

BLE_ERROR_L2CAP_COMMAND_NOT_UNDERSTOOD L2CAP command not understood

BLE_ERROR_L2CAP_SIGNALLING_MTU_EXCEEDED L2CAP signaling MTU exceeded

BLE_ERROR_L2CAP_INVALID_CID_IN_REQUEST Invalid Connection Identifier (CID) in request

BLE_ERROR_L2CAP_CONNECTION_REFUSED_PSM_NOT_SUPPORTED Connection refused - PSM is not supported

BLE_ERROR_L2CAP_CONNECTION_REFUSED_NO_RESOURCE Connection refused - No resource available

BLE_ERROR_L2CAP_CONNECTION_REFUSED_INSUFFICIENT_AUTHENTICATION Connection refused - Insufficient authentication

BLE_ERROR_L2CAP_CONNECTION_REFUSED_INSUFFICIENT_AUTHORIZATION Connection refused - Insufficient authorization

BLE_ERROR_L2CAP_CONNECTION_REFUSED_INSUFFICIENT_ENCRYPTION_KEY_SIZE Connection refused - Insufficient encryption key size

BLE_ERROR_L2CAP_COMMAND_TIMEOUT Command timeout. If the L2CAP signaling timeout occurs, the application should disconnect

BLE_ERROR_L2CAP_INCORRECT_SDU_LENGTH Invalid SDU length

BLE_ERROR_L2CAP_NOT_ENOUGH_CREDITS Not enough credits to perform an operation

BLE_ERROR_L2CAP_CREDIT_OVERFLOW Credit overflow. Total credit exceeded 65535 (maximum)

BLE_ERROR_L2CAP_UNACCEPTABLE_CREDIT_VALUE Invalid credit value. The received credit is zero

BLE_ERROR_TOOL_REQUEST_TIMEDOUT Tool request timed out

BLE_ERROR_TOOL_INVALID_RESPONSE_FORMAT Format of the response received from the dongle is invalid

BLE_ERROR_MAX Error maximum

CyAddressResolutionControlInfo

Holds the information necessary to enable / disable address resolution in controller

Public Member Functions

- CyAddressResolutionControlInfo (bool enable)
  Creates the information necessary to control the address resolution

Properties

- bool Enable `[get]`
  Gets whether the address resolution in controller should be enabled or disabled

---

Detailed Description

Holds the information necessary to enable / disable address resolution in controller

Constructor & Destructor Documentation

CyAddressResolutionControlInfo (bool *enable*)

Creates the information necessary to control the address resolution

Parameters:

enable	True to enable address resolution; False to disable address resolution

---

Property Documentation

bool Enable [get]

Gets whether the address resolution in controller should be enabled or disabled

---

CyAddToResolvingListInfo

Holds the information necessary to add a device to the resolving list

Public Member Functions

- CyAddToResolvingListInfo (CyBleBdAddress peerIdAddress, byte[] peerIRK, byte[] localIRK)
  Creates the information necessary to add a device to the resolving list

Properties

- CyBleBdAddress PeerIdAddress [get]
  Gets the ID address of the peer device to be added to the resolving list
- byte[] PeerIRK [get]
  Gets the peer device IRK
- byte[] LocalIRK [get]
  Gets the local device IRK

Detailed Description

Holds the information necessary to add a device to the resolving list

Constructor & Destructor Documentation

CyAddToResolvingListInfo (CyBleBdAddress *peerIdAddress*, byte[] *peerIRK*, byte[] *localIRK*)

Creates the information necessary to add a device to the resolving list

Parameters:

peerIdAddress	Peer device ID address

peerIRK	Peer device IRK
localIRK	Local device IRK

Property Documentation

CyBleBdAddress PeerIdAddress [get]

Gets the ID address of the peer device to be added to the resolving list

byte [] PeerIRK [get]

Gets the peer device IRK

byte [] LocalIRK [get]

Gets the local device IRK

CyAdvertisementData

Represents advertisement or scan response data

Properties

- CyBleAdvEventType AdvertisementType [get]
  Gets the advertisement type
- List< CyAdvertisementDataItem > Items [get]
  Gets the advertisement data items
- byte[] RawData [get]
  Gets the raw advertisement or scan data

Detailed Description

Represents advertisement or scan response data

Property Documentation

CyBleAdvEventType AdvertisementType [get]

Gets the advertisement type

List<CyAdvertisementDataItem> Items [get]

Gets the advertisement data items

byte [] RawData [get]

Gets the raw advertisement or scan data

CyAdvertisementDataItem

Represents an item in the advertisement data

Public Attributes

- const byte FLAGS = 0x01
  Flags
- const byte INCOMPLETE_16_BIT_SERVICE_UUID = 0x02
  Incomplete List of 16-bit Service Class UUIDs
- const byte COMPLETE_16_BIT_SERVICE_UUID = 0x03
  Complete List of 16-bit Service Class UUIDs
- const byte INCOMPLETE_32_BIT_SERVICE_UUID = 0x04
  Incomplete List of 32-bit Service Class UUIDs
- const byte COMPLETE_32_BIT_SERVICE_UUID = 0x05
  Complete List of 32-bit Service Class UUIDs
- const byte INCOMPLETE_128_BIT_SERVICE_UUID = 0x06
  Incomplete List of 128-bit Service Class UUIDs
- const byte COMPLETE_128_BIT_SERVICE_UUID = 0x07
  Complete List of 128-bit Service Class UUIDs
- const byte SHORTENED_LOCAL_NAME = 0x08
  Shortened Local Name
- const byte COMPLETE_LOCAL_NAME = 0x09
  Complete Local Name
- const byte TX_POWER_LEVEL = 0x0A

Tx Power Level

- const byte DEVICE_ID = 0x10
  Device ID
- const byte SLAVE_CONNECTION_INTERVAL_RANGE = 0x12
  Slave Connection Interval Range
- const byte SERVICE_SOLICITATION_16_BIT_UUID = 0x14
  List of 16-bit Service Solicitation UUIDs
- const byte SERVICE_SOLICITATION_32_BIT_UUID = 0x1F
  List of 32-bit Service Solicitation UUIDs
- const byte SERVICE_SOLICITATION_128_BIT_UUID = 0x15
  List of 128-bit Service Solicitation UUIDs
- const byte SERVICE_DATA_16_BIT_UUID = 0x16
  Service Data - 16-bit UUID
- const byte SERVICE_DATA_32_BIT_UUID = 0x20
  Service Data - 32-bit UUID
- const byte SERVICE_DATA_128_BIT_UUID = 0x21
  Service Data - 128-bit UUID
- const byte PUBLIC_TARGET_ADDRESS = 0x17
  Public Target Address
- const byte RANDOM_ADDRESS = 0x18
  Random Target Address
- const byte APPEARANCE = 0x19
  Appearance
- const byte ADVERTISING_INTERVAL = 0x1A
  Advertising Interval
- const byte LE_BLUETOOTH_DEVICE_ADDRESS = 0x1B
  LE Bluetooth Device Address
- const byte LE_ROLE = 0x1C
  LE Role
- const byte URI = 0x24
  URI
- const byte MANUFACTURER_SPECIFIC_DATA = 0xFF
  Manufacturer Specific Data

Properties

- int Type `[get]`
  Gets the advertisement item type
- int Length `[get]`
  Gets the advertisement data item length
- List< byte > Data `[get]`
  Gets the advertisement item data

Detailed Description

Represents an item in the advertisement data

Member Data Documentation

const byte FLAGS = 0x01

Flags

const byte INCOMPLETE_16_BIT_SERVICE_UUID = 0x02

Incomplete List of 16-bit Service Class UUIDs

const byte COMPLETE_16_BIT_SERVICE_UUID = 0x03

Complete List of 16-bit Service Class UUIDs

const byte INCOMPLETE_32_BIT_SERVICE_UUID = 0x04

Incomplete List of 32-bit Service Class UUIDs

const byte COMPLETE_32_BIT_SERVICE_UUID = 0x05

Complete List of 32-bit Service Class UUIDs

const byte INCOMPLETE_128_BIT_SERVICE_UUID = 0x06

Incomplete List of 128-bit Service Class UUIDs

const byte COMPLETE_128_BIT_SERVICE_UUID = 0x07

Complete List of 128-bit Service Class UUIDs

const byte SHORTENED_LOCAL_NAME = 0x08

Shortened Local Name

const byte COMPLETE_LOCAL_NAME = 0x09

Complete Local Name

const byte TX_POWER_LEVEL = 0x0A

Tx Power Level

const byte DEVICE_ID = 0x10

Device ID

const byte SLAVE_CONNECTION_INTERVAL_RANGE = 0x12

Slave Connection Interval Range

const byte SERVICE_SOLICITATION_16_BIT_UUID = 0x14

List of 16-bit Service Solicitation UUIDs

const byte SERVICE_SOLICITATION_32_BIT_UUID = 0x1F

List of 32-bit Service Solicitation UUIDs

const byte SERVICE_SOLICITATION_128_BIT_UUID = 0x15

List of 128-bit Service Solicitation UUIDs

const byte SERVICE_DATA_16_BIT_UUID = 0x16

Service Data - 16-bit UUID

const byte SERVICE_DATA_32_BIT_UUID = 0x20

Service Data - 32-bit UUID

const byte SERVICE_DATA_128_BIT_UUID = 0x21

Service Data - 128-bit UUID

const byte PUBLIC_TARGET_ADDRESS = 0x17

Public Target Address

const byte RANDOM_ADDRESS = 0x18

Random Target Address

const byte APPEARANCE = 0x19

Appearance

const byte ADVERTISING_INTERVAL = 0x1A

Advertising Interval

const byte LE_BLUETOOTH_DEVICE_ADDRESS = 0x1B

LE Bluetooth Device Address

const byte LE_ROLE = 0x1C

LE Role

const byte URI = 0x24

URI

const byte MANUFACTURER_SPECIFIC_DATA = 0xFF

Manufacturer Specific Data

Property Documentation

int Type [get]

Gets the advertisement item type

int Length [get]

Gets the advertisement data item length

List<byte> Data [get]

Gets the advertisement item data

CyApiErr

Indicate success or failure, and if failure, what was the cause.

Public Member Functions

- override string ToString ()
  Stringify a CyApiErr.
- override bool **Equals** (object obj)
- override int **GetHashCode** ()

Static Public Member Functions

- static bool **operator==** (CyApiErr a, CyApiErr b)
- static bool **operator!=** (CyApiErr a, CyApiErr b)

Public Attributes

- const int **ID_OK** = 0
- const int **ID_FAIL** = -1

Properties

- StackTrace `[get]`
  Get the stack trace at the point where the CyApiErr was created.
- bool HasStackTrace `[get]`
  Return true if the CyApiErr captured the stack.
- Exception WrappedException `[get]`
  Get the wrapped exception. Will return null if no exception was wrapped.
- bool HasWrappedException `[get]`
  Returns true if the CyApiErr is wrapping an exception.
- int **ErrorId** `[get]`
- string Message `[get]`
  Get the message from a CyApiErr.
- static CyApiErr Ok `[get]`
  The cannonical "Ok" CyApiErr. When reporting success, always return CyApiErr.Ok.
- static CyApiErr OK `[get]`
  An alias for "Ok".
- bool IsOk `[get]`
  Check to see if the CyApiErr is the cannonical "Ok" instance. Used to indicate success.
- bool IsOK `[get]`
  An alias for "IsOk".
- bool IsNotOk `[get]`
  Check to see if the CyApiErr is not the canonical "Ok" instance. Used to indicate failure.
- bool IsNotOK `[get]`
  An alias for "IsNotOk".

Detailed Description

Indicate success or failure, and if failure, what was the cause.

CyApiErr may contain an exception, a string message, and an integer ID. A CyApiErr instances has been pre-created to represent Ok. It is available as CyApiErr.Ok.

CyApiErr can also wrap exceptions. If no message is explicitly specified with the error, the exceptions message is adopted by the wrapping CyApiErr instance.

Member Function Documentation

override string ToString ()

Stringify a CyApiErr.

Returns:

Returns message of the CyApiErr

Property Documentation

StackTrace StackTrace [get]

Get the stack trace at the point where the CyApiErr was created.

bool HasStackTrace [get]

Return true if the CyApiErr captured the stack.

Exception WrappedException [get]

Get the wrapped exception. Will return null if no exception was wrapped.

bool HasWrappedException [get]

Returns true if the CyApiErr is wrapping an exception.

string Message [get]

Get the message from a CyApiErr.

CyApiErr Ok [static], [get]

The cannonical "Ok" CyApiErr. When reporting success, always return CyApiErr.Ok.

CyApiErr OK [static], [get]

An alias for "Ok".

bool IsOk [get]

Check to see if the CyApiErr is the cannonical "Ok" instance. Used to indicate success.

bool IsOK [get]

An alias for "IsOk".

bool IsNotOk [get]

Check to see if the CyApiErr is not the canonical "Ok" instance. Used to indicate failure.

bool IsNotOK [get]

An alias for "IsNotOk".

CyAuthenticationKeys

Holds the authentication keys

Public Member Functions

- CyAuthenticationKeys (CyAuthenticationKeyFlags keyFlags, byte[] ltk, byte[] masterIdKeys, byte[] irk, byte[] csrk)
  Creates an instance of the authentication keys

Properties

- CyAuthenticationKeyFlags KeyFlags [get]
  Keys to be exchanged between the pairing initiator and responder
- byte[] LTK [get]
  Gets the 16-byte Long Term Key (LTK)
- byte[] MasterIDKeys [get]
  Gets the 10-byte Master Identification Keys - Encryption Diversifier (EDIV) and a Random number
- byte[] IRK [get]
  Gets the 16-byte Identity Resolution Key (IRK)
- byte[] CSRK [get]
  Gets the 16-byte Connection data Signature Resolution Key (CSRK)

Detailed Description

Holds the authentication keys

Constructor & Destructor Documentation

CyAuthenticationKeys (CyAuthenticationKeyFlags *keyFlags*, byte[] *ltk*, byte[] *masterIdKeys*, byte[] *irk*, byte[] *csrk*)

Creates an instance of the authentication keys

Parameters:

keyFlags	Keys to be exchanged between initiator and responder during pairing
ltk	Long Term Key (LTK)
masterIdKeys	Master Identification Keys - Encryption Diversifier (EDIV) and a Random number
irk	Identity Resolution Key (IRK)
csrk	Connection data Signature Resolution Key (CSRK)

Property Documentation

CyAuthenticationKeyFlags KeyFlags [get]

Keys to be exchanged between the pairing initiator and responder

byte [] LTK [get]

Gets the 16-byte Long Term Key (LTK)

byte [] MasterIDKeys [get]

Gets the 10-byte Master Identification Keys - Encryption Diversifier (EDIV) and a Random number

byte [] IRK [get]

Gets the 16-byte Identity Resolution Key (IRK)

byte [] CSRK [get]

Gets the 16-byte Connection data Signature Resolution Key (CSRK)

CyBleBdAddress

Represents the Bluetooth device address

Public Member Functions

- CyBleBdAddress (ulong address, CyBleBdAddressType addressType)
  Creates an instance of the Bluetooth device address class
- override bool Equals (object obj)
  Check if the addresses are equal
- override int GetHashCode ()
  Gets the hash code for the address

Public Attributes

- const ulong MAX_ADDRESS = (ulong.MaxValue & ADDR_MASK)
  Maximum value for the address
- const ulong MIN_ADDRESS = ulong.MinValue
  Minimum value for the address

Properties

- CyBleBdAddressType AddressType [get]
  Gets the address type
- ulong Address [get]
  Gets the address

Detailed Description

Represents the Bluetooth device address

Constructor & Destructor Documentation

CyBleBdAddress (ulong *address*, CyBleBdAddressType *addressType*)

Creates an instance of the Bluetooth device address class

Parameters:

address	Bluetooth device address The address should be within 0x000000000000 and 0xFFFFFFFFFFFF (inclusive)
addressType	Device address type

Member Function Documentation

override bool Equals (object *obj*)

Check if the addresses are equal

Parameters:

obj	Bluetooth device address

Returns:

True if equal; otherwise, false

override int GetHashCode ()

Gets the hash code for the address

Returns:

Hash code

Member Data Documentation

const ulong MAX_ADDRESS = (ulong.MaxValue & ADDR_MASK)

Maximum value for the address

const ulong MIN_ADDRESS = ulong.MinValue

Minimum value for the address

Property Documentation

CyBleBdAddressType AddressType [get]

Gets the address type

ulong Address [get]

Gets the address

CyBleConnectionSettings

Holds the settings to be used for connection establishment

Public Member Functions

- CyBleConnectionSettings ()
  Creates a connection parameter info object with the default settings
- CyBleConnectionSettings (ushort minConnectionInterval, ushort maxConnectionInterval, ushort minCeLength, ushort maxCeLength, CyConnectionInitiatorFilterPolicy connectionInitiatorFilterPolicy, CyInitiatorAddrType connectionOwnBdAddrType, ushort slaveLatency, ushort connectionSuperVisionTimeout, ushort connectionScanInterval, ushort connectionScanWindow)
  Creates a connection parameter info object

Public Attributes

- const ushort DEFAULT_MINIMUM_CONNECTION_INTERVAL = 0x0006
  Default minimum connection interval
- const ushort DEFAULT_MAXIMUM_CONNECTION_INTERVAL = 0x0006
  Default maximum connection interval
- const ushort DEFAULT_SLAVE_LATENCY = 0x0000
  Default slave latency
- const ushort DEFAULT_SUPERVISION_TIMEOUT = 0x000A
  Default supervision timeout
- const CyConnectionInitiatorFilterPolicy DEFAULT_INITIATOR_FILTER_POLICY

Default initiator filter policy

- const CyInitiatorAddrType DEFAULT_INITIATOR_ADDRESS_TYPE = CyInitiatorAddrType.PUBLIC
  Default initiator address type
- const ushort DEFAULT_MINIMUM_CE_LENGTH = 0x0000
  Default minimum CE length
- const ushort DEFAULT_MAXIMUM_CE_LENGTH = 0x0000
  Default maximum CE length
- const ushort DEFAULT_SCAN_INTERVAL = 0x0010
  Default scan interval
- const ushort DEFAULT_SCAN_WINDOW = 0x0010
  Default scan window

Properties

- ushort MinimumConnectionInterval `[get]`
  Gets the minimum connection interval
- ushort MaximumConnectionInterval `[get]`
  Gets the maximum connection interval
- ushort SlaveLatency `[get]`
  Gets the slave latency
- ushort ConnectionSupervisionTimeout `[get]`
  Gets the connection supervision timeout
- CyConnectionInitiatorFilterPolicy ConnectionInitiatorFilterPolicy `[get]`
  Gets the connection initiator filter policy
- CyInitiatorAddrType ConnectionInitiatorBdAddrType `[get]`
  Gets the connection initiator Bluetooth address type
- ushort MinimumCeLength `[get]`
  Gets the minimum CE length
- ushort MaximumCeLength `[get]`
  Gets the maximum CE length
- ushort ConnectionScanInterval `[get]`
  Gets the connection scan interval
- ushort ConnectionScanWindow `[get]`
  Gets the connection scan window

Detailed Description

Holds the settings to be used for connection establishment

Constructor & Destructor Documentation

CyBleConnectionSettings ()

Creates a connection parameter info object with the default settings

CyBleConnectionSettings (ushort *minConnectionInterval*, ushort *maxConnectionInterval*, ushort *minCeLength*, ushort *maxCeLength*, CyConnectionInitiatorFilterPolicy *connectionInitiatorFilterPolicy*, CyInitiatorAddrType *connectionOwnBdAddrType*, ushort *slaveLatency*, ushort *connectionSuperVisionTimeout*, ushort *connectionScanInterval*, ushort *connectionScanWindow*)

Creates a connection parameter info object

Parameters:

minConnectionInterval	minimum connection interval
maxConnectionInterval	maximum connection interval
minCeLength	minimum CE length
maxCeLength	maximum CE length
connectionInitiatorFilterPolicy	connection initiator filter policy
connectionOwnBdAddrType	connection initiator Bluetooth address type
slaveLatency	slave latency
connectionSuperVisionTimeout	connection supervision timeout

Member Data Documentation

const ushort DEFAULT_MINIMUM_CONNECTION_INTERVAL = 0x0006

Default minimum connection interval

const ushort DEFAULT_MAXIMUM_CONNECTION_INTERVAL = 0x0006

Default maximum connection interval

const ushort DEFAULT_SLAVE_LATENCY = 0x0000

Default slave latency

const ushort DEFAULT_SUPERVISION_TIMEOUT = 0x000A

Default supervision timeout

const CyConnectionInitiatorFilterPolicy DEFAULT_INITIATOR_FILTER_POLICY

```
Initial value:=
            CyConnectionInitiatorFilterPolicy.USE_PEER_DEVICE_ADDRESS
```

Default initiator filter policy

const CyInitiatorAddrType DEFAULT_INITIATOR_ADDRESS_TYPE = CyInitiatorAddrType.PUBLIC

Default initiator address type

const ushort DEFAULT_MINIMUM_CE_LENGTH = 0x0000

Default minimum CE length

const ushort DEFAULT_MAXIMUM_CE_LENGTH = 0x0000

Default maximum CE length

const ushort DEFAULT_SCAN_INTERVAL = 0x0010

Default scan interval

const ushort DEFAULT_SCAN_WINDOW = 0x0010

Default scan window

Property Documentation

ushort MinimumConnectionInterval [get]

Gets the minimum connection interval

ushort MaximumConnectionInterval [get]

Gets the maximum connection interval

ushort SlaveLatency [get]

Gets the slave latency

ushort ConnectionSupervisionTimeout [get]

Gets the connection supervision timeout

CyConnectionInitiatorFilterPolicy ConnectionInitiatorFilterPolicy [get]

Gets the connection initiator filter policy

CyInitiatorAddrType ConnectionInitiatorBdAddrType [get]

Gets the connection initiator Bluetooth address type

ushort MinimumCeLength [get]

Gets the minimum CE length

ushort MaximumCeLength [get]

Gets the maximum CE length

ushort ConnectionScanInterval [get]

Gets the connection scan interval

ushort ConnectionScanWindow [get]

Gets the connection scan window

CyBleDeviceCallback

Device callback Defines device specific method callbacks

Public Member Functions

- virtual void OnUpdateConnectionParameterRequestReceived (CyConnectionParameters request)
  Connection parameter update request received from remote device
- virtual void OnUpdateConnectionParameter (CyStatus status)
  Reports the status of ICyBleDevice.UpdateConnectionParameter
- virtual void OnConnectionParameterChanged (CyCurrentConnectionParameters connectionParameters)
  Reports the updated connection parameter values
- virtual void OnSetDataLength (CyStatus status)
  Reports the status of the ICyBleDevice.SetDataLength
- virtual void OnDataLengthChanged (CyCurrentDataLength dataLength)
  Reports the updated data length for the connection
- virtual void OnPairRequestReceived (CyPairSettings settings)
  Reports the pairing request received from the remote device
- virtual void OnPairingSettingsNegotiated (CyPairSettings settings)
  Reports the negotiated pairing settings
- virtual void OnPasskeyEntryRequest (CyPasskeyEntryResponse response)
  Passkey entry request
- virtual void OnPasskeyDisplayRequest (CyPasskeyDisplayInfo info)
  Passkey display request
- virtual void OnNumericComparisonRequest (CyNumericComparisonResponse response)
  Secure connection numeric comparison request
- virtual void OnSendKeyPressNotification (CyStatus status)

Reports the status of ICyBleDevice.SendKeyPressNotification

- virtual void OnPairingCompleted (CyStatus status)
  Reports the status of ICyBleDevice.Pair
- virtual void OnSetOob (CyStatus status)
  Reports the status of ICyBleDevice.SetOob

Detailed Description

Device callback Defines device specific method callbacks

Member Function Documentation

virtual void OnUpdateConnectionParameterRequestReceived (CyConnectionParameters *request*) [virtual]

Connection parameter update request received from remote device

Parameters:

request	New connection parameters requested by the remote device

Use ICyBleDevice.SendConnectionParametersResponse method to send the response

virtual void OnUpdateConnectionParameter (CyStatus *status*) [virtual]

Reports the status of ICyBleDevice.UpdateConnectionParameter

Parameters:

status	Status of ICyBleDevice.UpdateConnectionParameter

virtual void OnConnectionParameterChanged (CyCurrentConnectionParameters *connectionParameters*) [virtual]

Reports the updated connection parameter values

Parameters:

connectionParameters	Current connection parameters

virtual void OnSetDataLength (CyStatus *status*) [virtual]

Reports the status of the ICyBleDevice.SetDataLength

Parameters:

status	Status of ICyBleDevice.SetDataLength

virtual void OnDataLengthChanged (CyCurrentDataLength *dataLength*) [virtual]

Reports the updated data length for the connection

Parameters:

dataLength	Current data length for the connection

virtual void OnPairRequestReceived (CyPairSettings *settings*) [virtual]

Reports the pairing request received from the remote device

Parameters:

settings	Remote device pairing settings

virtual void OnPairingSettingsNegotiated (CyPairSettings *settings*) [virtual]

Reports the negotiated pairing settings

Parameters:

settings	Negotiated pair settings

virtual void OnPasskeyEntryRequest (CyPasskeyEntryResponse *response*) [virtual]

Passkey entry request

Parameters:

response	Contains the response for passkey entry request

Use ICyBleDevice.SendPasskeyResponse to send the response

virtual void OnPasskeyDisplayRequest (CyPasskeyDisplayInfo *info*) [virtual]

Passkey display request

Parameters:

info	Passkey to be displayed

virtual void OnNumericComparisonRequest (CyNumericComparisonResponse *response*) [virtual]

Secure connection numeric comparison request

Parameters:

response	Contains the numeric comparison request details and the response for the request

Use ICyBleDevice.SendNumericComparisonResponse to send the response

virtual void OnSendKeyPressNotification (CyStatus *status*) [virtual]

Reports the status of ICyBleDevice.SendKeyPressNotification

Parameters:

status	Status of ICyBleDevice.SendKeyPressNotification

virtual void OnPairingCompleted (CyStatus *status*) [virtual]

Reports the status of ICyBleDevice.Pair

Parameters:

status	Status of ICyBleDevice.Pair

virtual void OnSetOob (CyStatus *status*) [virtual]

Reports the status of ICyBleDevice.SetOob

Parameters:

status	Status of ICyBleDevice.SetOob

CyBleMgrCallback

BLE manager callback The interface defines callback methods which needs to be implemented by the API client

Public Member Functions

- virtual void OnSetDeviceIoCapabilities (CyStatus status)
  This callback reports the status of the ICyBleMgr.SetDeviceIoCapabilities method
- virtual void OnGetDeviceIoCapabilities (CyBleDeviceIoCapabilities ioCapabilities, CyStatus status)
  This callback reports the IO capabilities of the local devices
- virtual void OnGetRssi (sbyte rssi, CyStatus status)
  This callback reports the RSSI and the status of the ICyBleMgr.GetRSSI method
- virtual void OnConnected (CyConnectResult result, CyStatus status)
  This callback reports the result and status of the ICyBleMgr.Connect method
- virtual void OnCancelled (CyStatus status)
  This callback reports the status of the ICyBleMgr.CancelConnection method
- virtual void OnDisconnected (CyBleBdAddress deviceAddress, CyStatus status)
  This callback reports the status of the ICyBleMgr.Disconnect method
- virtual void OnRegisterPsm (CyRegisteredPsm psm, CyStatus status)
  Reports the registered PSM
- virtual void OnUnregisterPsm (CyStatus status)
  Reports the status of the ICyBleMgr.UnregisterPsm method
- virtual void OnSetHostChannelClassification (CyStatus status)
  Reports the status of the ICyBleMgr.SetHostChannelClassification method
- virtual void OnSetTxPower (CyStatus status)
  Reports the status of the ICyBleMgr.SetTxPower method
- virtual void OnGetTxPower (CyTxPowerInfo result, CyStatus status)

Reports the channel transmission power

- virtual void OnGetDefaultDataLength (CyDefaultDataLengthResult result, CyStatus status)
  Reports the default data length
- virtual void OnSetSuggestedDataLength (CyStatus status)
  Reports the status of the ICyBleMgr.SetSuggestedDataLength method
- virtual void OnConvertDataLengthOctetToTime (CyConvertOctetToTimeResult result, CyStatus status)
  Reports the result of the octet to time conversion
- virtual void OnSetResolvableAddressTimeout (CyStatus status)
  Reports the status of ICyBleMgr.SetResolvableAddressTimeout
- virtual void OnSetAddressResolutionControl (CyStatus status)
  Reports the status of ICyBleMgr.SetAddressResolutionControl

---

Detailed Description

BLE manager callback The interface defines callback methods which needs to be implemented by the API client

Member Function Documentation

virtual void OnSetDeviceIoCapabilities (CyStatus *status*) [virtual]

This callback reports the status of the ICyBleMgr.SetDeviceIoCapabilities method

Parameters:

status	Status of the ICyBleMgr.SetDeviceIoCapabilities method execution

virtual void OnGetDeviceIoCapabilities (CyBleDeviceIoCapabilities *ioCapabilities*, CyStatus *status*) [virtual]

This callback reports the IO capabilities of the local devices

Parameters:

ioCapabilities	IO capabilities of the local device
status	Status of the ICyBleMgr.GetDeviceIoCapabilities method execution

virtual void OnGetRssi (sbyte *rssi*, CyStatus *status*) [virtual]

This callback reports the RSSI and the status of the ICyBleMgr.GetRSSI method

Parameters:

rssi	RSSI of the last received packet This field is valid only if the *status* is OK
status	Status of ICyBleMgr.GetRSSI method execution

virtual void OnConnected (CyConnectResult *result*, CyStatus *status*) [virtual]

This callback reports the result and status of the ICyBleMgr.Connect method

Parameters:

result	Contains the connection result if the *status* is OK; otherwise is null
status	Status of the ICyBleMgr.Connect method execution

virtual void OnCancelled (CyStatus *status*) [virtual]

This callback reports the status of the ICyBleMgr.CancelConnection method

Parameters:

status	Status of the ICyBleMgr.CancelConnection method execution

virtual void OnDisconnected (CyBleBdAddress *deviceAddress*, CyStatus *status*) [virtual]

This callback reports the status of the ICyBleMgr.Disconnect method

Parameters:

deviceAddress	Contains the address of the disconnected device if the *status* is OK; otherwise is null
status	Status of the ICyBleMgr.Disconnect method execution

virtual void OnRegisterPsm (CyRegisteredPsm *psm*, CyStatus *status*) [virtual]

Reports the registered PSM

Parameters:

psm	Registered PSM
status	Status of the ICyBleMgr.RegisterPsm method execution

virtual void OnUnregisterPsm (CyStatus *status*) [virtual]

Reports the status of the ICyBleMgr.UnregisterPsm method

Parameters:

status	Status of the ICyBleMgr.UnregisterPsm method execution

virtual void OnSetHostChannelClassification (CyStatus *status*) [virtual]

Reports the status of the ICyBleMgr.SetHostChannelClassification method

Parameters:

status	Status of the ICyBleMgr.SetHostChannelClassification method execution

virtual void OnSetTxPower (CyStatus *status*) [virtual]

Reports the status of the ICyBleMgr.SetTxPower method

Parameters:

status	Status of the ICyBleMgr.SetTxPower method execution

virtual void OnGetTxPower (CyTxPowerInfo *result*, CyStatus *status*) [virtual]

Reports the channel transmission power

Parameters:

result	Channel power level
status	Status of the ICyBleMgr.GetTxPower method execution

virtual void OnGetDefaultDataLength (CyDefaultDataLengthResult *result*, CyStatus *status*) [virtual]

Reports the default data length

Parameters:

result	Default data length of the local device (dongle)
status	Status of the ICyBleMgr.GetDefaultDataLength method execution

virtual void OnSetSuggestedDataLength (CyStatus *status*) [virtual]

Reports the status of the ICyBleMgr.SetSuggestedDataLength method

Parameters:

status	Status of the ICyBleMgr.SetSuggestedDataLength method execution

virtual void OnConvertDataLengthOctetToTime (CyConvertOctetToTimeResult *result*, CyStatus *status*) [virtual]

Reports the result of the octet to time conversion

Parameters:

result	Contains the conversion result if *status* is OK; Otherwise is null
status	Status of ICyBleMgr.ConvertDataLengthOctetToTime method execution

virtual void OnSetResolvableAddressTimeout (CyStatus *status*) [virtual]

Reports the status of ICyBleMgr.SetResolvableAddressTimeout

Parameters:

status	Status of ICyBleMgr.SetResolvableAddressTimeout

virtual void OnSetAddressResolutionControl (CyStatus *status*) [virtual]

Reports the status of ICyBleMgr.SetAddressResolutionControl

Parameters:

status	Status of ICyBleMgr.SetAddressResolutionControl

CyBleScanSettings

Holds the scan settings

Public Member Functions

- CyBleScanSettings ()
  Creates an instance of the scan parameters with default settings
- CyBleScanSettings (CyScanType scanType, ushort scanInterval, ushort scanWindow, CyInitiatorAddrType scanOwnBdAddrType, CyScanInitiatorFilterPolicy scanFilterPolicy, CyDiscoveryType scanProcedureType, ushort scanTimeout, CyScanDuplicateFilterPolicy scanDuplicateFilterPolicy)
  Creates an instance of scan parameters info

Public Attributes

- const CyScanType DEFAULT_SCAN_TYPE = CyScanType.ACTIVE_SCAN
  Default scan type
- const ushort DEFAULT_SCAN_INTERVAL = 0x0010
  Default scan interval
- const ushort DEFAULT_SCAN_WINDOW = 0x0010
  Default scan window
- const CyInitiatorAddrType DEFAULT_SCAN_OWN_ADDRESS_TYPE = CyInitiatorAddrType.PUBLIC
  Default own Bluetooth address type
- const CyScanInitiatorFilterPolicy DEFAULT_INITIATOR_FILTER_POLICY = CyScanInitiatorFilterPolicy.ACCEPT_ALL_ADV
  Default initiator filter policy
- const CyDiscoveryType DEFAULT_DISCOVERY_TYPE = CyDiscoveryType.OBSERVATION_PROCEDURE
  Default discovery procedure
- const ushort DEFAULT_SCAN_TIMEOUT = 0
  Default scan timeout
- const CyScanDuplicateFilterPolicy DEFAULT_DUPLICATE_FILTER_POLICY
  Default duplicate filter policy

Properties

- CyScanType ScanType `[get]`
  Gets the scan type
- ushort ScanInterval `[get]`
  Gets the scan interval
- ushort ScanWindow `[get]`
  Gets the scan window
- CyInitiatorAddrType ScanOwnBdAddrType `[get]`
  Gets the own bluetooth address type to be used for scan
- CyScanInitiatorFilterPolicy InitiatorFilterPolicy `[get]`
  Gets the scan initiator filter policy
- CyDiscoveryType ScanProcedureType `[get]`
  Gets the scan procedure type
- ushort ScanTimeout `[get]`
  Gets the scan timeout in seconds
- CyScanDuplicateFilterPolicy DuplicateFilterPolicy `[get]`
  Gets the scan duplicate filtering policy

Detailed Description

Holds the scan settings

Constructor & Destructor Documentation

CyBleScanSettings ()

Creates an instance of the scan parameters with default settings

CyBleScanSettings (CyScanType *scanType*, ushort *scanInterval*, ushort *scanWindow*, CyInitiatorAddrType *scanOwnBdAddrType*, CyScanInitiatorFilterPolicy *scanFilterPolicy*, CyDiscoveryType *scanProcedureType*, ushort *scanTimeout*, CyScanDuplicateFilterPolicy *scanDuplicateFilterPolicy*)

Creates an instance of scan parameters info

Parameters:

scanType	scan type
scanInterval	scan interval
scanWindow	scan window
scanOwnBdAddrType	Own bluetooth address type to be used for scan
scanFilterPolicy	scan initiator filter policy
scanProcedureType	scan procedure type
scanTimeout	scan timeout in seconds Specify zero for continuous scan
scanDuplicateFilterPolicy	Specify the scan duplicate filter policy

---

Member Data Documentation

const CyScanType DEFAULT_SCAN_TYPE = CyScanType.ACTIVE_SCAN

Default scan type

const ushort DEFAULT_SCAN_INTERVAL = 0x0010

Default scan interval

const ushort DEFAULT_SCAN_WINDOW = 0x0010

Default scan window

const CyInitiatorAddrType DEFAULT_SCAN_OWN_ADDRESS_TYPE = CyInitiatorAddrType.PUBLIC

Default own bluetooth address type

const CyScanInitiatorFilterPolicy DEFAULT_INITIATOR_FILTER_POLICY = CyScanInitiatorFilterPolicy.ACCEPT_ALL_ADV

Default initiator filter policy

const CyDiscoveryType DEFAULT_DISCOVERY_TYPE = CyDiscoveryType.OBSERVATION_PROCEDURE

Default discovery procedure

const ushort DEFAULT_SCAN_TIMEOUT = 0

Default scan timeout

const CyScanDuplicateFilterPolicy DEFAULT_DUPLICATE_FILTER_POLICY

```
Initial value:=
            CyScanDuplicateFilterPolicy.ENABLE_DUPLICATE_FILTERING
```

Default duplicate filter policy

Property Documentation

CyScanType ScanType [get]

Gets the scan type

ushort ScanInterval [get]

Gets the scan interval

ushort ScanWindow [get]

Gets the scan window

CyInitiatorAddrType ScanOwnBdAddrType [get]

Gets the own bluetooth address type to be used for scan

CyScanInitiatorFilterPolicy InitiatorFilterPolicy [get]

Gets the scan initiator filter policy

CyDiscoveryType ScanProcedureType [get]

Gets the scan procedure type

ushort ScanTimeout [get]

Gets the scan timeout in seconds

CyScanDuplicateFilterPolicy DuplicateFilterPolicy [get]

Gets the scan duplicate filtering policy

CyBondListDevice

Represents a device in the bond list

Properties

- CyBleBdAddress DeviceAddress [get]
  Gets the device address
- byte DeviceHandle [get]
  Gets the device handle

Detailed Description

Represents a device in the bond list

Property Documentation

CyBleBdAddress DeviceAddress [get]

Gets the device address

byte DeviceHandle [get]

Gets the device handle

CyChannelClassificationInfo

Holds the information necessary to modify the host channel classification

Public Types

- enum Channels : ulong { NONE = 0x0000000000, CHANNEL_00 = 0x0000000001, CHANNEL_01 = 0x0000000002, CHANNEL_02 = 0x0000000004, CHANNEL_03 = 0x0000000008, CHANNEL_04 = 0x0000000010, CHANNEL_05 = 0x0000000020, CHANNEL_06 = 0x0000000040, CHANNEL_07 = 0x0000000080, CHANNEL_08 = 0x0000000100, CHANNEL_09 = 0x0000000200, CHANNEL_10 = 0x0000000400, CHANNEL_11 = 0x0000000800, CHANNEL_12 = 0x0000001000, CHANNEL_13 = 0x0000002000, CHANNEL_14 = 0x0000004000, CHANNEL_15 = 0x0000008000, CHANNEL_16 = 0x0000010000, CHANNEL_17 = 0x0000020000, CHANNEL_18 = 0x0000040000, CHANNEL_19 = 0x0000080000, CHANNEL_20 = 0x0000100000, CHANNEL_21 = 0x0000200000, CHANNEL_22 = 0x0000400000, CHANNEL_23 = 0x0000800000, CHANNEL_24 = 0x0001000000, CHANNEL_25 = 0x0002000000, CHANNEL_26 = 0x0004000000, CHANNEL_27 = 0x0008000000, CHANNEL_28 = 0x0010000000, CHANNEL_29 = 0x0020000000, CHANNEL_30 = 0x0040000000, CHANNEL_31 = 0x0080000000, CHANNEL_32 = 0x0100000000, CHANNEL_33 =

0x0200000000, CHANNEL_34 = 0x0400000000, CHANNEL_35 = 0x0800000000, CHANNEL_36 = 0x1000000000, ALL = 0x1FFFFFFFFF }*Enumeration of channels Two or more channels can be selected by OR'ing the corresponding channel enumeration*

Public Member Functions

- CyChannelClassificationInfo (Channels channelClassification)
  Creates the information necessary to update the host channel classification

Properties

- Channels ChannelClassification `[get]`
  Gets the channel classification

Detailed Description

Holds the information necessary to modify the host channel classification

Member Enumeration Documentation

enum Channels : ulong `[strong]`

Enumeration of channels Two or more channels can be selected by OR'ing the corresponding channel enumeration

Enumerator

NONE No channels selected

CHANNEL_00 Channel 00

CHANNEL_01 Channel 01

CHANNEL_02 Channel 02

CHANNEL_03 Channel 03

CHANNEL_04 Channel 04

CHANNEL_05 Channel 05

CHANNEL_06 Channel 06

CHANNEL_07 Channel 07

CHANNEL_08 Channel 08

CHANNEL_09 Channel 09

CHANNEL_10 Channel 10

CHANNEL_11 Channel 11

CHANNEL_12 Channel 12

CHANNEL_13 Channel 13

CHANNEL_14 Channel 14

CHANNEL_15 Channel 15

CHANNEL_16 Channel 16

CHANNEL_17 Channel 17

CHANNEL_18 Channel 18

CHANNEL_19 Channel 19

CHANNEL_20 Channel 20

CHANNEL_21 Channel 21

CHANNEL_22 Channel 22

CHANNEL_23 Channel 23

CHANNEL_24 Channel 24

CHANNEL_25 Channel 25

CHANNEL_26 Channel 26

CHANNEL_27 Channel 27

CHANNEL_28 Channel 28

CHANNEL_29 Channel 29

CHANNEL_30 Channel 30

CHANNEL_31 Channel 31

CHANNEL_32 Channel 32

CHANNEL_33 Channel 33

CHANNEL_34 Channel 34

CHANNEL_35 Channel 35

CHANNEL_36 Channel 36

ALL All Channels

Constructor & Destructor Documentation

CyChannelClassificationInfo (Channels *channelClassification*)

Creates the information necessary to update the host channel classification

Parameters:

channelClassification	Channel classification It is mandatory that at least one channel is selected

---

Property Documentation

Channels ChannelClassification [get]

Gets the channel classification

CyCharacteristicChangedInfo

Holds the information of the characteristic value changed as a result of notification or indication

Properties

- ushort Handle `[get]`
  Gets the handle of the changed characteristic
- byte[] Value `[get]`
  Gets the changed value

Detailed Description

Holds the information of the characteristic value changed as a result of notification or indication

Property Documentation

ushort Handle `[get]`

Gets the handle of the changed characteristic

byte [] Value `[get]`

Gets the changed value

CyConnectInfo

Holds the information necessary to connect to a BLE device

Public Member Functions

- CyConnectInfo (CyBleBdAddress peerDeviceAddress, CyBleConnectionSettings settings)
  Creates an instance of the connection information class

Properties

- CyBleBdAddress PeerDeviceAddress `[get]`
  Gets the address of the peer device to be connected
- CyBleConnectionSettings Settings `[get]`
  Gets the connection settings

Detailed Description

Holds the information necessary to connect to a BLE device

Constructor & Destructor Documentation

CyConnectInfo (CyBleBdAddress *peerDeviceAddress*, CyBleConnectionSettings *settings*)

Creates an instance of the connection information class

Parameters:

peerDeviceAddress	Address of the peer device to be connected
settings	Connection settings to be used for this connection

Property Documentation

CyBleBdAddress PeerDeviceAddress [get]

Gets the address of the peer device to be connected

CyBleConnectionSettings Settings [get]

Gets the connection settings

CyConnectionParameters

Connection parameter settings

Public Member Functions

- CyConnectionParameters (ushort connectionIntervalMin, ushort connectionIntervalMax, ushort slaveLatency, ushort supervisionTimeout)
  Creates the connection parameter settings

Properties

- ushort ConnectionIntervalMinimum `[get]`
  Gets the minimum value for the connection interval
- ushort ConnectionIntervalMaximum `[get]`
  Gets the maximum value for the connection interval
- ushort SlaveLatency `[get]`
  Gets the slave latency for the connection in number of connection events
- ushort SupervisionTimeout `[get]`
  Gets the supervision timeout for the LE link

Detailed Description

Connection parameter settings

Constructor & Destructor Documentation

CyConnectionParameters (ushort *connectionIntervalMin*, ushort *connectionIntervalMax*, ushort *slaveLatency*, ushort *supervisionTimeout*)

Creates the connection parameter settings

Parameters:

connectionIntervalMin	Minimum value for the connection interval This shall be less than or equal to *connectionIntervalMax*

Range: 0x0006 to 0x0C80 Time: N * 1.25 ms; where N is the connection interval value

Parameters:

connectionIntervalMax	Maximum value for the connection interval. This shall be greater than or equal to *connectionIntervalMin*

Range: 0x0006 to 0x0C80 Time: N * 1.25 ms; where N is the connection interval value

Parameters:

slaveLatency	Slave latency for the connection in number of connection events Range: 0x0000 to 0x01F3
supervisionTimeout	Supervision timeout for the LE link

Range: 0x000A to 0x0C80 Time = N * 10 ms

Property Documentation

ushort ConnectionIntervalMinimum [get]

Gets the minimum value for the connection interval

ushort ConnectionIntervalMaximum [get]

Gets the maximum value for the connection interval

ushort SlaveLatency [get]

Gets the slave latency for the connection in number of connection events

ushort SupervisionTimeout [get]

Gets the supervision timeout for the LE link

CyConnectResult

Holds the connection result

Properties

- ICyBleDevice Device [get]
  Gets the connected device instance

Detailed Description

Holds the connection result

Property Documentation

ICyBleDevice Device [get]

Gets the connected device instance

CyConvertOctetToTimeInfo

Holds the information necessary to convert octet to time

Public Member Functions

- CyConvertOctetToTimeInfo (ushort octet)
  Creates an information necessary to convert octet to time

Properties

- CyPhyType PhysicalLayerType `[get]`
  Gets the physical layer type
- ushort Octet `[get]`
  Gets the octet value to be converted

Detailed Description

Holds the information necessary to convert octet to time

Constructor & Destructor Documentation

CyConvertOctetToTimeInfo (ushort *octet*)

Creates an information necessary to convert octet to time

Parameters:

octet	Octet value to be converted

Property Documentation

CyPhyType PhysicalLayerType `[get]`

Gets the physical layer type

ushort Octet [get]

Gets the octet value to be converted

CyConvertOctetToTimeResult

Holds the result of the octet to time conversion

Properties

- ushort Octet [get]
  Gets the octet value that was converted
- ushort Time [get]
  Gets the converted time value in µs

Detailed Description

Holds the result of the octet to time conversion

Property Documentation

ushort Octet [get]

Gets the octet value that was converted

ushort Time [get]

Gets the converted time value in µs

CyCurrentConnectionParameters

Holds the current connection parameters

Properties

- ushort ConnectionInterval [get]
  Gets the connection interval

- ushort SlaveLatency [get]
  Gets the slave latency for the connection in number of connection events
- ushort SupervisionTimeout [get]
  Gets the supervision timeout for the LE link

Detailed Description

Holds the current connection parameters

Property Documentation

ushort ConnectionInterval [get]

Gets the connection interval

ushort SlaveLatency [get]

Gets the slave latency for the connection in number of connection events

ushort SupervisionTimeout [get]

Gets the supervision timeout for the LE link

CyCurrentDataLength

Holds the current data length

Properties

- ushort ConnectionMaximumTxOctets [get]
  Gets the maximum Tx octet size for the current connection
- ushort ConnectionMaximumTxTime [get]
  Gets the maximum Tx time (in µs) for the current connection
- ushort ConnectionMaximumRxOctets [get]
  Gets the maximum Rx octet size for the current connection
- ushort ConnectionMaximumRxTime [get]
  Gets the maximum Rx time (in µs) for the current connection

---

Detailed Description

Holds the current data length

Property Documentation

ushort ConnectionMaximumTxOctets [get]

Gets the maximum Tx octet size for the current connection

ushort ConnectionMaximumTxTime [get]

Gets the maximum Tx time (in µs) for the current connection

ushort ConnectionMaximumRxOctets [get]

Gets the maximum Rx octet size for the current connection

ushort ConnectionMaximumRxTime [get]

Gets the maximum Rx time (in µs) for the current connection

CyDataLengthInfo

Holds the information necessary to change the data length for the current connection

Public Member Functions

- CyDataLengthInfo (ushort maxTxOctets, ushort maxTxTime)
  Creates the information necessary to set the preferred data length

Properties

- ushort MaxTxOctets [get]
  Gets the preferred maximum Tx octet size for the connection
- ushort MaxTxTime [get]
  Gets the preferred maximum Tx time (in µs) for the connection

---

Detailed Description

Holds the information necessary to change the data length for the current connection

Constructor & Destructor Documentation

CyDataLengthInfo (ushort *maxTxOctets*, ushort *maxTxTime*)

Creates the information necessary to set the preferred data length

Parameters:

maxTxOctets	Preferred maximum Tx octet size for the connection
maxTxTime	Preferred maximum Tx time (in µs) for the connection Use ICyBleMgr.ConvertDataLengthOctetToTime API to convert octet to time

---

Property Documentation

ushort MaxTxOctets [get]

Gets the preferred maximum Tx octet size for the connection

ushort MaxTxTime [get]

Gets the preferred maximum Tx time (in µs) for the connection

---

CyDefaultDataLengthResult

Holds the default data length of the local device (dongle)

Properties

- ushort SuggestedMaxTxOctets [get]
  Gets the suggested data octet size to be used for new connections
- ushort SuggestedMaxTxTime [get]
  Gets the suggested data transmission time (in µs) for new connections

- ushort SupportedMaxTxOctets [get]
  Gets the maximum supported data transmission octet size
- ushort SupportedMaxTxTime [get]
  Gets the maximum supported data transmission time (in µs)
- ushort SupportedMaxRxOctets [get, set]
  Gets the maximum supported data receive octet size
- ushort SupportedMaxRxTime [get]
  Gets the maximum supported data receive time (in µs)

Detailed Description

Holds the default data length of the local device (dongle)

Property Documentation

ushort SuggestedMaxTxOctets [get]

Gets the suggested data octet size to be used for new connections

ushort SuggestedMaxTxTime [get]

Gets the suggested data transmission time (in µs) for new connections

ushort SupportedMaxTxOctets [get]

Gets the maximum supported data transmission octet size

ushort SupportedMaxTxTime [get]

Gets the maximum supported data transmission time (in µs)

ushort SupportedMaxRxOctets [get], [set]

Gets the maximum supported data receive octet size

ushort SupportedMaxRxTime [get]

Gets the maximum supported data receive time (in µs)

CyDeviceAddressMgrCallback

Device address manager callback The interface defines callback methods which needs to be implemented by the API client

Public Member Functions

- virtual void OnBdAddressGenerated (CyGenerateBdAddressResult result, CyStatus status)
  Reports the generated Bluetooth device address
- virtual void OnGetBdAddress (CyBleBdAddress address, CyStatus status)
  Reports the Bluetooth device address of the local device (dongle)
- virtual void OnSetBdAddress (CyStatus status)
  Reports the status of the ICyBleDeviceAddressMgr.SetBdAddress method
- virtual void OnSetIdAddress (CyStatus status)
  Reports the status of the ICyBleDeviceAddressMgr.SetIdAddress method
- virtual void OnGetPeerResolvableAddress (CyResolvableAddressResult result, CyStatus status)
  Reports the resolvable address of the peer device
- virtual void OnGetLocalResolvableAddress (CyResolvableAddressResult result, CyStatus status)
  Reports the resolvable address of the local device

Detailed Description

Device address manager callback The interface defines callback methods which needs to be implemented by the API client

Member Function Documentation

virtual void OnBdAddressGenerated (CyGenerateBdAddressResult *result*, CyStatus *status*) `[virtual]`

Reports the generated Bluetooth device address

Parameters:

result	Contains the generated Bluetooth device address, if *status* is OK; otherwise is null

status	Status of the ICyBleDeviceAddressMgr.GenerateBdAddressOfType method execution

virtual void OnGetBdAddress (CyBleBdAddress *address*, CyStatus *status*) [virtual]

Reports the Bluetooth device address of the local device (dongle)

Parameters:

address	Contains the Bluetooth device address of the local device, if *status* is OK; otherwise is null
status	Status of the ICyBleDeviceAddressMgr.GetBdAddress method execution

virtual void OnSetBdAddress (CyStatus *status*) [virtual]

Reports the status of the ICyBleDeviceAddressMgr.SetBdAddress method

Parameters:

status	Status of the ICyBleDeviceAddressMgr.SetBdAddress method execution

virtual void OnSetIdAddress (CyStatus *status*) [virtual]

Reports the status of the ICyBleDeviceAddressMgr.SetIdAddress method

Parameters:

status	Status of the ICyBleDeviceAddressMgr.SetIdAddress method execution

virtual void OnGetPeerResolvableAddress (CyResolvableAddressResult *result*, CyStatus *status*) [virtual]

Reports the resolvable address of the peer device

Parameters:

result	Contains the resolvable private address of the peer device, if *status* is OK; otherwise is null
status	Status of the ICyBleDeviceAddressMgr.GetPeerResolvableAddress method execution

virtual void OnGetLocalResolvableAddress (CyResolvableAddressResult *result*, CyStatus *status*) [virtual]

Reports the resolvable address of the local device

Parameters:

result	Contains the resolvable private address of the local device, if *status* is OK; otherwise is null
status	Status of the ICyBleDeviceAddressMgr.GetLocalResolvableAddress method execution

---

CyDeviceListCallback

Device list callback Defines callback method from device list APIs

Public Member Functions

- virtual void OnAddDeviceToWhitelist (CyWhitelistDevice device, CyStatus status)
  Reports the device added to the whitelist
- virtual void OnRemoveDeviceFromWhitelist (CyStatus status)
  Reports the status of ICyBleDeviceList.RemoveDeviceFromWhitelist
- virtual void OnClearWhitelist (CyStatus status)
  Reports the status of ICyBleDeviceList.ClearWhitelist
- virtual void OnGetWhitelistDevices (List< CyWhitelistDevice > whitelist, CyStatus status)
  Reports the whitelist devices
- virtual void OnGetBondListDevices (List< CyBondListDevice > bondList, CyStatus status)
  Reports the bond list devices

- virtual void OnAddDeviceToResolvingList (CyResolvingListDevice device, CyStatus status)
  Reports the device added to the resolving list
- virtual void OnRemoveDeviceFromResolvingList (CyStatus status)
  Reports the status of ICyBleDeviceList.RemoveDeviceFromResolvingList
- virtual void OnClearResolvingList (CyStatus status)
  Reports the status of ICyBleDeviceList.ClearResolvingList
- virtual void OnGetResolvingListDevices (List< CyResolvingListDevice > resolvingList, CyStatus status)
  Reports the resolving list devices

Detailed Description

Device list callback Defines callback method from device list APIs

Member Function Documentation

virtual void OnAddDeviceToWhitelist (CyWhitelistDevice *device*, CyStatus *status*) [virtual]

Reports the device added to the whitelist

Parameters:

device	Contains the device added to the whitelist, if *status* is OK; Otherwise is null
status	Status of ICyBleDeviceList.AddDeviceToWhitelist

virtual void OnRemoveDeviceFromWhitelist (CyStatus *status*) [virtual]

Reports the status of ICyBleDeviceList.RemoveDeviceFromWhitelist

Parameters:

status	Status of ICyBleDeviceList.RemoveDeviceFromWhitelist

virtual void OnClearWhitelist (CyStatus *status*) [virtual]

Reports the status of ICyBleDeviceList.ClearWhitelist

Parameters:

status	Status of ICyBleDeviceList.ClearWhitelist

virtual void OnGetWhitelistDevices (List< CyWhitelistDevice > *whitelist*, CyStatus *status*) [virtual]

Reports the whitelist devices

Parameters:

whitelist	Contains the devices in the whitelist, if *status* is OK; Otherwise is null
status	Status of ICyBleDeviceList.GetWhitelistDevices

virtual void OnGetBondListDevices (List< CyBondListDevice > *bondList*, CyStatus *status*) [virtual]

Reports the bond list devices

Parameters:

bondList	Contains the devices in the bond list, if *status* is OK; Otherwise is null
status	Status of ICyBleDeviceList.GetBondListDevices

virtual void OnAddDeviceToResolvingList (CyResolvingListDevice *device*, CyStatus *status*) [virtual]

Reports the device added to the resolving list

Parameters:

device	Resolving list device
status	Status of ICyBleDeviceList.AddDeviceToResolvingList

virtual void OnRemoveDeviceFromResolvingList (CyStatus *status*) [virtual]

Reports the status of ICyBleDeviceList.RemoveDeviceFromResolvingList

Parameters:

status	Status of ICyBleDeviceList.RemoveDeviceFromResolvingList

virtual void OnClearResolvingList (CyStatus *status*) [virtual]

Reports the status of ICyBleDeviceList.ClearResolvingList

Parameters:

status	Status of ICyBleDeviceList.ClearResolvingList

virtual void OnGetResolvingListDevices (List< CyResolvingListDevice > *resolvingList*, CyStatus *status*) [virtual]

Reports the resolving list devices

Parameters:

resolvingList	Contains the devices in the resolving list, if *status* is OK; Otherwise is null
status	Status of ICyBleDeviceList.GetResolvingListDevices

CyDiscoverAllServicesResult

Holds the result of service discovery

Properties

- List< CyGattService > Services [get]

Gets all the discovered services

Detailed Description

Holds the result of service discovery

Property Documentation

List<CyGattService> Services [get]

Gets all the discovered services

CyDiscoverCharacteristicsByUUIDInfo

Holds the information necessary to discover characteristics by UUID

Public Member Functions

- CyDiscoverCharacteristicsByUUIDInfo (CyUUID uuid, ushort startHandle, ushort endHandle)
  Creates the information necessary to discover characteristics by UUID

Properties

- CyUUID UUID [get]
- ushort StartHandle [get]
  Gets the start handle to begin the discovery
- ushort EndHandle [get]
  Gets the end handle to stop the discovery

Detailed Description

Holds the information necessary to discover characteristics by UUID

Constructor & Destructor Documentation

CyDiscoverCharacteristicsByUUIDInfo (CyUUID *uuid*, ushort *startHandle*, ushort *endHandle*)

Creates the information necessary to discover characteristics by UUID

Parameters:

uuid	UUID of the characteristic to be discovered
startHandle	Start handle
endHandle	End handle

Property Documentation

CyUUID UUID [get]

Gets the UUID of the characteristics to be discovered

ushort StartHandle [get]

Gets the start handle to begin the discovery

ushort EndHandle [get]

Gets the end handle to stop the discovery

CyDiscoverCharacteristicsCallback

Defines the callback methods for characteristic discovery

Public Member Functions

- abstract void OnCharacteristicsDiscovered (CyDiscoverCharacteristicsResult result, CyStatus status)
  Reports the discovered characteristics

Detailed Description

Defines the callback methods for characteristic discovery

Member Function Documentation

abstract void OnCharacteristicsDiscovered (CyDiscoverCharacteristicsResult *result*, CyStatus *status*) [pure virtual]

Reports the discovered characteristics

Parameters:

result	Characteristics discovered
status	Status of ICyGattClient.DiscoverCharacteristics or ICyGattClient.DiscoverCharacteristicsByUUID method execution

---

CyDiscoverCharacteristicsInfo

Holds the information necessary to discover characteristics

Public Member Functions

- CyDiscoverCharacteristicsInfo (ushort startHandle, ushort endHandle)
  Creates the information necessary to discover characteristics

Properties

- ushort StartHandle [get]
  Gets the starting handle to begin the characteristic discovery
- ushort EndHandle [get]
  Gets the end handle to stop the characteristic discovery

Detailed Description

Holds the information necessary to discover characteristics

Constructor & Destructor Documentation

CyDiscoverCharacteristicsInfo (ushort *startHandle*, ushort *endHandle*)

Creates the information necessary to discover characteristics

Parameters:

startHandle	Start handle This is typically set to a service start handle
endHandle	End handle This is typically set to a service end handle

---

Property Documentation

ushort StartHandle [get]

Gets the starting handle to begin the characteristic discovery

ushort EndHandle [get]

Gets the end handle to stop the characteristic discovery

CyDiscoverCharacteristicsResult

Holds the result of characteristic discovery

Properties

- List< CyGattCharacteristic > Characteristics [get]
  Gets the list of discovered characteristics

Detailed Description

Holds the result of characteristic discovery

Property Documentation

List<CyGattCharacteristic> Characteristics [get]

Gets the list of discovered characteristics

CyDiscoverDescriptorsCallback

Defines the callback methods for descriptor discovery

Public Member Functions

- abstract void OnDescriptorDiscovered (CyDiscoverDescriptorsResult result, CyStatus status)
  Reports the discovered descriptors

Detailed Description

Defines the callback methods for descriptor discovery

Member Function Documentation

abstract void OnDescriptorDiscovered (CyDiscoverDescriptorsResult *result*, CyStatus *status*) [pure virtual]

Reports the discovered descriptors

Parameters:

result	Descriptors discovered
status	Status of the ICyGattClient.DiscoverDescriptors method execution

---

CyDiscoverDescriptorsInfo

Holds the information necessary to discover descriptors

Public Member Functions

- CyDiscoverDescriptorsInfo (ushort startHandle, ushort endHandle)
  Creates the information necessary to discover descriptors

Properties

- ushort StartHandle `[get]`
  Gets the start handle to begin the discovery
- ushort EndHandle `[get]`
  Gets the end handle to stop the discovery

Detailed Description

Holds the information necessary to discover descriptors

Constructor & Destructor Documentation

CyDiscoverDescriptorsInfo (ushort *startHandle*, ushort *endHandle*)

Creates the information necessary to discover descriptors

Parameters:

startHandle	Start handle This is typically set to the characteristic value handle + 1
endHandle	End handle This is typically set to the end handle of a characteristic

---

Property Documentation

ushort StartHandle `[get]`

Gets the start handle to begin the discovery

ushort EndHandle `[get]`

Gets the end handle to stop the discovery

CyDiscoverDescriptorsResult

Holds the result of descriptor discovery

Properties

- List< CyGattDescriptor > Descriptors `[get]`
  Gets the list of discovered descriptors

Detailed Description

Holds the result of descriptor discovery

Property Documentation

List<CyGattDescriptor> Descriptors `[get]`

Gets the list of discovered descriptors

CyDiscoverPrimaryServiceCallback

Defines the callback methods for primary service discovery

Public Member Functions

- abstract void OnPrimaryServiceDiscovered (List< CyGattService > services, CyStatus status)
  Reports the discovered primary services

Detailed Description

Defines the callback methods for primary service discovery

Member Function Documentation

abstract void OnPrimaryServiceDiscovered (List< CyGattService > *services*, CyStatus *status*)[pure virtual]

Reports the discovered primary services

Parameters:

services	Primary services discovered
status	Status of ICyGattClient.DiscoverPrimaryServices or ICyGattClient.DiscoverPrimaryServicesByUUID method execution

---

CyDiscoverPrimaryServicesByUUIDInfo

Holds the information necessary to discover primary services by UUID

Public Member Functions

- CyDiscoverPrimaryServicesByUUIDInfo (CyUUID uuid)
  Creates the information necessary to discover primary services by UUID

Properties

- CyUUID UUID [get]
  Gets the UUID of the primary service to be discovered

Detailed Description

Holds the information necessary to discover primary services by UUID

Constructor & Destructor Documentation

CyDiscoverPrimaryServicesByUUIDInfo (CyUUID *uuid*)

Creates the information necessary to discover primary services by UUID

Parameters:

uuid	UUID of the primary service to be discovered

Property Documentation

CyUUID UUID [get]

Gets the UUID of the primary service to be discovered

CyDiscoverPrimaryServicesResult

Holds the result of primary services discovery

Properties

- List< CyGattService > Services [get]
  Gets the list of primary services discovered

Detailed Description

Holds the result of primary services discovery

Property Documentation

List<CyGattService> Services [get]

Gets the list of primary services discovered

CyDongleInfo

Holds the information necessary to get a dongle communicator

Public Types

- enum CySmartDongleType { CY5670, CY5672, CY5677 }*Enumeration of supported dongle types*

Public Member Functions

- CyDongleInfo (string comPortName)
  Creates an instance of the DongleInfo class
- CyDongleInfo (string comPortName, CySmartDongleType type)
  Creates an instance of the DongleInfo class

Properties

- string COMPortName `[get]`
  Gets the COM port name of the CySmart dongle
- CySmartDongleType DongleType `[get]`
  Gets the dongle type

Detailed Description

Holds the information necessary to get a dongle communicator

Member Enumeration Documentation

enum CySmartDongleType `[strong]`

Enumeration of supported dongle types

Enumerator

CY5670 CySmart BLE dongle (supports BLE v4.1)

CY5672 CySmart HID dongle (supports BLE v4.1)

CY5677 CySmart BLE dongle (supports BLE v4.2)

Constructor & Destructor Documentation

CyDongleInfo (string *comPortName*)

Creates an instance of the *DongleInfo* class

Parameters:

comPortName	COM port name of the CySmart dongle

CyDongleInfo (string *comPortName*, CySmartDongleType *type*)

Creates an instance of the *DongleInfo* class

Parameters:

comPortName	COM port name of the CySmart dongle
type	CySmart BLE dongle type

Property Documentation

string COMPortName [get]

Gets the COM port name of the CySmart dongle

CySmartDongleType DongleType [get]

Gets the dongle type

CyEstablishL2CapChannelInfo

Holds the information necessary to create a new L2CAP channel

Public Member Functions

- CyEstablishL2CapChannelInfo (ushort remotePSM, ushort localPSM, ushort mtu, ushort mps, ushort initialCredits)
  Creates the information necessary to establish an L2CAP channel

Properties

- ushort RemotePSM [get]
  Gets the remote PSM
- ushort LocalPSM [get]

Gets the local PSM

- ushort MTU [get]
  Gets the local MTU
- ushort MPS [get]
  Gets the local MPS
- ushort InitialCredits [get]
  Gets the initial credits

Detailed Description

Holds the information necessary to create a new L2CAP channel

Constructor & Destructor Documentation

CyEstablishL2CapChannelInfo (ushort *remotePSM*, ushort *localPSM*, ushort *mtu*, ushort *mps*, ushort *initialCredits*)

Creates the information necessary to establish an L2CAP channel

Parameters:

remotePSM	Remote PSM
localPSM	Local PSM
mtu	Local MTU
mps	Local MPS
initialCredits	Initial credits

Property Documentation

ushort RemotePSM [get]

Gets the remote PSM

ushort LocalPSM [get]

Gets the local PSM

ushort MTU [get]

Gets the local MTU

ushort MPS [get]

Gets the local MPS

ushort InitialCredits [get]

Gets the initial credits

CyFindIncludedServicesCallback

Defines the callback methods for find included services API

Public Member Functions

- abstract void OnFindIncludedServices (CyFindIncludedServicesResult result, CyStatus status)
  Reports the discovered included services

Detailed Description

Defines the callback methods for find included services API

Member Function Documentation

abstract void OnFindIncludedServices (CyFindIncludedServicesResult *result*, CyStatus *status*) [pure virtual]

Reports the discovered included services

Parameters:

result	Discovered included services
status	Status of the ICyGattClient.FindIncludedServices method execution

CyFindIncludedServicesInfo

Holds the information necessary to find an included service

Public Member Functions

- CyFindIncludedServicesInfo (ushort startHandle, ushort endHandle)
  Creates the information necessary to find included services

Properties

- ushort StartHandle `[get]`
  Gets the handle from which the discovery needs to be started
- ushort EndHandle `[get]`
  Gets the attribute handle till which the discovery needs to be executed

Detailed Description

Holds the information necessary to find an included service

Constructor & Destructor Documentation

CyFindIncludedServicesInfo (ushort *startHandle*, ushort *endHandle*)

Creates the information necessary to find included services

Parameters:

startHandle	Start handle This is typically the start handle of a primary service
endHandle	End handle This is typically the end handle of a primary service

Property Documentation

ushort StartHandle [get]

Gets the handle from which the discovery needs to be started

ushort EndHandle [get]

Gets the attribute handle till which the discovery needs to be executed

---

CyFindIncludedServicesResult

Holds the result of included service discovery

Properties

- List< CyGattIncludedService > IncludedServices [get]
  Gets the included service discovered

Detailed Description

Holds the result of included service discovery

Property Documentation

List<CyGattIncludedService> IncludedServices [get]

Gets the included service discovered

CyGattAttribute

Represents an attribute in the GATT server

Protected Member Functions

- CyGattAttribute (CyUUID uuid, ushort handle, byte[] value)
  Creates an instance of the attribute

Properties

- CyUUID UUID `[get]`
  Gets the UUID of the attribute
- ushort Handle `[get]`
  Gets the attribute handle
- byte[] Value `[get]`
  Gets the attribute value

Detailed Description

Represents an attribute in the GATT server

Constructor & Destructor Documentation

CyGattAttribute (CyUUID *uuid*, ushort *handle*, byte[] *value*) `[protected]`

Creates an instance of the attribute

Parameters:

uuid	Attribute UUID
handle	Attribute handle
value	Attribute value

Property Documentation

CyUUID UUID `[get]`

Gets the UUID of the attribute

ushort Handle `[get]`

Gets the attribute handle

byte [] Value [get]

Gets the attribute value

CyGattCharacteristic

Represents a GATT characteristic

Properties

- CyUUID UUID [get]
  Gets the characteristic UUID
- ushort DeclarationHandle [get]
  Gets the handle at which the characteristic is declared
- ushort Handle [get]
  Gets the handle to characteristic value
- CyGattCharacteristicProperty Properties [get]
  Gets the characteristic properties
- CyGattService Service [get, set]
  Gets the service which includes this characteristic
- List< CyGattDescriptor > Descriptors [get]
  Gets the list of descriptors associated with this characteristic

---

Detailed Description

Represents a GATT characteristic

Property Documentation

CyUUID UUID [get]

Gets the characteristic UUID

ushort DeclarationHandle [get]

Gets the handle at which the characteristic is declared

ushort Handle [get]

Gets the handle to characteristic value

CyGattCharacteristicProperty Properties [get]

Gets the characteristic properties

CyGattService Service [get], [set]

Gets the service which includes this characteristic

This will be null, if the characteristic was discovered via ICyGattClient.DiscoverCharacteristics or ICyGattClient.DiscoverCharacteristicsByUUID

List<CyGattDescriptor> Descriptors [get]

Gets the list of descriptors associated with this characteristic

This will be null, if the characteristic was discovered via ICyGattClient.DiscoverCharacteristics or ICyGattClient.DiscoverCharacteristicsByUUID

CyGattClientCallback

Public Member Functions

- virtual void OnGattMtuExchanged (CyGattExchangeMtuResult result, CyStatus status)
  Reports the negotiated GATT MTU size
- virtual void OnServiceDiscovered (CyDiscoverAllServicesResult result, CyStatus status)
  Reports all the discovered services
- virtual void OnCharacteristicChanged (CyCharacteristicChangedInfo info)
  Reports the received characteristic value notifications and indications
- virtual void OnCharacteristicRead (CyGattReadResult result, CyStatus status)
  Reports the characteristic value that was read
- virtual void OnCharacteristicReadByUUID (CyReadCharacteristicByUUIDResult result, CyStatus status)
  Reports the value of characteristics with a specific UUID
- virtual void OnReadMultipleCharacteristics (CyReadMultipleCharacteristicResult result, CyStatus status)
  Reports the value read by ICyGattClient.ReadMultipleCharacteristic method

- virtual void OnCharacteristicWrite (CyGattWriteResult result, CyStatus status)
  Reports the status of a characteristic value write
- virtual void OnReliableWriteCompleted (CyStatus status)
  Reports the result of a reliable write
- virtual void OnWriteBufferFull (CyWriteBufferFullResponse response)
  This callback is invoked when the peer device reports that the write buffer is full
- virtual void OnDescriptorRead (CyGattReadResult result, CyStatus status)
  Reports the descriptor value that was read
- virtual void OnDescriptorWrite (CyGattWriteResult result, CyStatus status)
  Reports the status of descriptor value write
- virtual void OnGattStop (CyStatus status)
  Reports the status of ICyGattClient.GattStop

Member Function Documentation

virtual void OnGattMtuExchanged (CyGattExchangeMtuResult *result*, CyStatus *status*) [virtual]

Reports the negotiated GATT MTU size

Parameters:

result	Contains the negotiated GATT MTU size, if *status* is OK; Otherwise is null
status	Status of ICyGattClient.ExchangeMtu method execution

virtual void OnServiceDiscovered (CyDiscoverAllServicesResult *result*, CyStatus *status*) [virtual]

Reports all the discovered services

Parameters:

result	Contains the discovered services, if *status* is OK; Otherwise is null
status	Status of ICyGattClient.DiscoverAllServices method execution

virtual void OnCharacteristicChanged (CyCharacteristicChangedInfo *info*) [virtual]

Reports the received characteristic value notifications and indications

Parameters:

info	Characteristic that was changed

virtual void OnCharacteristicRead (CyGattReadResult *result*, CyStatus *status*) [virtual]

Reports the characteristic value that was read

Parameters:

result	Contains the characteristic value, if *status* is OK; Otherwise is null
status	Status of ICyGattClient.ReadCharacteristic or ICyGattClient.ReadLongCharacteristic methods

virtual void OnCharacteristicReadByUUID (CyReadCharacteristicByUUIDResult *result*, CyStatus *status*) [virtual]

Reports the value of characteristics with a specific UUID

Parameters:

result	Contains the characteristic value, if *status* is OK; Otherwise is null
status	Status of ICyGattClient.ReadCharacteristicByUUID method

virtual void OnReadMultipleCharacteristics (CyReadMultipleCharacteristicResult *result*, CyStatus *status*) [virtual]

Reports the value read by ICyGattClient.ReadMultipleCharacteristic method

Parameters:

result	Contains the value read, if *status* is OK; Otherwise is null
status	Status of ICyGattClient.ReadMultipleCharacteristic method

virtual void OnCharacteristicWrite (CyGattWriteResult *result*, CyStatus *status*) [virtual]

Reports the status of a characteristic value write

Parameters:

result	Contains the characteristic that was written
status	Status of ICyGattClient.WriteCharacteristicWithoutResponse, ICyGattClient.WriteCharacteristic, ICyGattClient.WriteLongCharacteristic or ICyGattClient.SignedCharacteristicWriteWithoutResponse

virtual void OnReliableWriteCompleted (CyStatus *status*) [virtual]

Reports the result of a reliable write

Parameters:

status	Status of ICyGattClient.ReliableWrite method

virtual void OnWriteBufferFull (CyWriteBufferFullResponse *response*) [virtual]

This callback is invoked when the peer device reports that the write buffer is full

Parameters:

response	Holds the information about the attribute reported by the peer device and the response to be sent

Use ICyGattClient.SendReliableWriteBufferFullResponse to send the response

virtual void OnDescriptorRead (CyGattReadResult *result*, CyStatus *status*) [virtual]

Reports the descriptor value that was read

Parameters:

result	Contains the descriptor value
status	Status of ICyGattClient.ReadDescriptor or ICyGattClient.ReadLongDescriptor methods

virtual void OnDescriptorWrite (CyGattWriteResult *result*, CyStatus *status*) [virtual]

Reports the status of descriptor value write

Parameters:

result	Contains the descriptor that was written
status	Status of ICyGattClient.WriteDescriptor or ICyGattClient.WriteLongDescriptor methods

virtual void OnGattStop (CyStatus *status*) [virtual]

Reports the status of ICyGattClient.GattStop

Parameters:

status	Status of ICyGattClient.GattStop method execution

CyGattDescriptor

Represents a characteristic descriptor

Properties

- CyUUID UUID [get]
  Gets the characteristic descriptor UUID
- ushort Handle [get]
  Gets the handle to the characteristic descriptor
- CyGattCharacteristic Characteristic [get, set]
  Gets the characteristic which includes this descriptor

Detailed Description

Represents a characteristic descriptor

Property Documentation

CyUUID UUID [get]

Gets the characteristic descriptor UUID

ushort Handle [get]

Gets the handle to the___14 characteristic descriptor

CyGattCharacteristic Characteristic [get], [set]

Gets the characteristic which includes this descriptor
This will be null, if the descriptor was discovered via ICyGattClient.DiscoverDescriptors

---

CyGattExchangeMtuInfo

Holds the information necessary to exchange GATT MTU size

Public Member Functions

- CyGattExchangeMtuInfo (ushort gattMtu)
  Creates the information necessary to exchange GATT MTU

Properties

- ushort GattMtu `[get]`
  Gets the GATT MTU size to be used for negotiation

Detailed Description

Holds the information necessary to exchange GATT MTU size

Constructor & Destructor Documentation

CyGattExchangeMtuInfo (ushort *gattMtu*)

Creates the information necessary to exchange GATT MTU

Parameters:

gattMtu	GATT MTU size to be negotiated

Property Documentation

ushort GattMtu `[get]`

Gets the GATT MTU size to be used for negotiation

CyGattExchangeMtuResult

Holds the result of GATT MTU exchange

Properties

- ushort GattMtuRequested `[get]`
  Gets the GATT MTU size requested
- ushort NegotiatedGattMtu `[get]`
  Gets the negotiated GATT MTU size

Detailed Description

Holds the result of GATT MTU exchange

Property Documentation

ushort GattMtuRequested [get]

Gets the GATT MTU size requested

ushort NegotiatedGattMtu [get]

Gets the negotiated GATT MTU size

CyGattIncludedService

Represents an included service

Properties

- ushort IncludedHandle [get]
  Gets the handle at which the service is included
- CyGattService IncludedService [get]
  Gets the service which is being included
- CyGattService Service [get, set]
  Gets the service which includes the IncludedService

Detailed Description

Represents an included service

Property Documentation

ushort IncludedHandle [get]

Gets the handle at which the service is included

CyGattService IncludedService [get]

Gets the service which is being included

CyGattService Service [get], [set]

Gets the service which includes the *IncludedService*
This will be null, if the included service was discovered via ICyGattClient.FindIncludedServices

---

CyGattReadInfo

Holds the information necessary to read an attribute value

Public Member Functions

- CyGattReadInfo (CyGattCharacteristic characteristic)
  Creates the information necessary to read a characteristic value
- CyGattReadInfo (CyGattCharacteristic characteristic, ushort offset)
  Creates the information necessary to read a long characteristic value
- CyGattReadInfo (CyGattDescriptor descriptor)
  Creates the information necessary to read a descriptor value
- CyGattReadInfo (CyGattDescriptor descriptor, ushort offset)
  Creates the information necessary to read a long descriptor value
- CyGattReadInfo (ushort handle)
  Creates the information necessary to read an attribute
- CyGattReadInfo (ushort handle, ushort offset)
  Creates the information necessary to read a long attribute value

Properties

- ushort Handle [get]
  Gets the attribute handle
- ushort Offset [get]
  Offset within the attribute value, to begin reading the value This is ignored if the operation is not a long operation

---

Detailed Description

Holds the information necessary to read an attribute value

Constructor & Destructor Documentation

CyGattReadInfo (CyGattCharacteristic *characteristic*)

Creates the information necessary to read a characteristic value

Parameters:

characteristic	Characteristic to be read

CyGattReadInfo (CyGattCharacteristic *characteristic*, ushort *offset*)

Creates the information necessary to read a long characteristic value

Parameters:

characteristic	Characteristic to be read
offset	Offset within the characteristic value, to begin reading the value

CyGattReadInfo (CyGattDescriptor *descriptor*)

Creates the information necessary to read a descriptor value

Parameters:

descriptor	Descriptor to be read

CyGattReadInfo (CyGattDescriptor *descriptor*, ushort *offset*)

Creates the information necessary to read a long descriptor value

Parameters:

descriptor	Descriptor to be read
offset	Offset within the descriptor to begin the read

CyGattReadInfo (ushort *handle*)

Creates the information necessary to read an attribute

Parameters:

handle	Attribute handle

CyGattReadInfo (ushort *handle*, ushort *offset*)

Creates the information necessary to read a long attribute value

Parameters:

handle	Attribute handle
offset	Offset within the attribute value, to begin reading the value

---

Property Documentation

ushort Handle [get]

Gets the attribute handle

ushort Offset [get]

Offset within the attribute value, to begin reading the value This is ignored if the operation is not a long operation

---

CyGattReadResult

Holds the result of an attribute read request

Properties

- ushort Handle [get]
  Handle to the attribute that was read

- byte[] Value `[get]`
  The value that was read from the peer device
- ushort Offset `[get]`
  The offset from which the value was read.

Detailed Description

Holds the result of an attribute read request

Property Documentation

ushort Handle `[get]`

Handle to the attribute that was read

byte [] Value `[get]`

The value that was read from the peer device

ushort Offset `[get]`

The offset from which the value was read.

CyGattService

Represents a GATT service

Properties

- bool IsPrimaryService `[get]`
  Gets whether the service is a primary service or not
- CyUUID UUID `[get, set]`
  Gets the service UUID
- ushort StartHandle `[get]`
  Gets the start handle of the service
- ushort EndHandle `[get]`
  Gets the end handle of the service
- List< CyGattCharacteristic > Characteristics `[get]`

Gets the list of characteristics associated with the service

- List< CyGattIncludedService > IncludedServices `[get]`
  Gets the list of included services

Detailed Description

Represents a GATT service

Property Documentation

bool IsPrimaryService `[get]`

Gets whether the service is a primary service or not

CyUUID UUID `[get], [set]`

Gets the service UUID

ushort StartHandle `[get]`

Gets the start handle of the service

ushort EndHandle `[get]`

Gets the end handle of the service

List<CyGattCharacteristic> Characteristics `[get]`

Gets the list of characteristics associated with the service

This will be null, if the service was discovered via ICyGattClient.DiscoverPrimaryServices or ICyGattClient.DiscoverPrimartServicesByUUID

List<CyGattIncludedService> IncludedServices `[get]`

Gets the list of included services

This will be null, if the service was discovered via ICyGattClient.DiscoverPrimaryServices or ICyGattClient.DiscoverPrimartServicesByUUID

CyGattWriteInfo

Holds the information necessary to write a attribute value

Public Member Functions

- CyGattWriteInfo (CyGattCharacteristic characteristic, params byte[] value)
  Creates the information necessary to write a characteristic value
- CyGattWriteInfo (CyGattCharacteristic characteristic, ushort offset, params byte[] value)
  Creates the information necessary to write a characteristic value
- CyGattWriteInfo (CyGattDescriptor descriptor, params byte[] value)
  Creates the information necessary to write a descriptor value
- CyGattWriteInfo (CyGattDescriptor descriptor, ushort offset, params byte[] value)
  Creates the information necessary to write a descriptor value
- CyGattWriteInfo (ushort handle, params byte[] value)
  Creates the information necessary to write a attribute value
- CyGattWriteInfo (ushort handle, ushort offset, params byte[] value)
  Creates the information necessary to write a attribute value

Properties

- ushort Handle `[get]`
  Gets the handle of the attribute to be written
- ushort Offset `[get]`
  Gets the offset within the attribute value to be written This is ignored, if the operation is not a long operation
- byte[] Value `[get]`
  Gets the value to be written

Detailed Description

Holds the information necessary to write a attribute value

Constructor & Destructor Documentation

CyGattWriteInfo (CyGattCharacteristic *characteristic*, params byte[] *value*)

Creates the information necessary to write a characteristic value

Parameters:

characteristic	Characteristic to be written
value	Value to be written

CyGattWriteInfo (CyGattCharacteristic *characteristic*, ushort *offset*, params byte[] *value*)

Creates the information necessary to write a characteristic value

Parameters:

characteristic	Characteristic to be written
offset	Offset within the characteristic value to be written
value	Value to be written

CyGattWriteInfo (CyGattDescriptor *descriptor*, params byte[] *value*)

Creates the information necessary to write a descriptor value

Parameters:

descriptor	Descriptor to be written
value	Value to be written

CyGattWriteInfo (CyGattDescriptor *descriptor*, ushort *offset*, params byte[] *value*)

Creates the information necessary to write a descriptor value

Parameters:

descriptor	Descriptor to be written
offset	Offset within the descriptor, to begin the value write
value	Value to be written

CyGattWriteInfo (ushort *handle*, params byte[] *value*)

Creates the information necessary to write a attribute value

Parameters:

handle	Attribute handle to be written
value	Value to be written

CyGattWriteInfo (ushort *handle*, ushort *offset*, params byte[] *value*)

Creates the information necessary to write a attribute value

Parameters:

handle	Attribute handle to be written
offset	Offset within the attribute value to be written
value	Value to be written

Property Documentation

ushort Handle [get]

Gets the handle of the attribute to be written

ushort Offset [get]

Gets the offset within the attribute value to be written This is ignored, if the operation is not a long operation

byte [] Value [get]

Gets the value to be written

CyGattWriteResult

Holds the information about the completion of an attribute write

Properties

- ushort Handle `[get]`
  Gets the handle of the attribute that was written

Detailed Description

Holds the information about the completion of an attribute write

Property Documentation

ushort Handle `[get]`

Gets the handle of the attribute that was written

CyGenerateBdAddressInfo

Holds the information required to generate Bluetooth device address

Public Member Functions

- CyGenerateBdAddressInfo (CyExpandedBdAddrType type)
  Creates the information required to generate Bluetooth device address Do not use this constructor to generate random resolvable address
- CyGenerateBdAddressInfo (byte[] irk)
  Creates the information required to generate random resolvable address

Properties

- CyExpandedBdAddrType AddressType `[get]`
  Gets the type of address to be generated
- byte[] IRK `[get]`
  Gets the 16-byte IRK to be used to generate random resolvable address

---

Detailed Description

Holds the information required to generate Bluetooth device address

Constructor & Destructor Documentation

CyGenerateBdAddressInfo (CyExpandedBdAddrType *type*)

Creates the information required to generate Bluetooth device address Do not use this constructor to generate random resolvable address

Parameters:

type	Type of address to be generated

CyGenerateBdAddressInfo (byte[] *irk*)

Creates the information required to generate random resolvable address

Parameters:

irk	IRK to be used to generate the random resolvable address

Property Documentation

CyExpandedBdAddrType AddressType [get]

Gets the type of address to be generated

byte [] IRK [get]

Gets the 16-byte IRK to be used to generate random resolvable address

---

CyGenerateBdAddressResult

Holds the generated Bluetooth device address

Properties

- CyBleBdAddress GeneratedAddress [get]
  Gets the generated Bluetooth device address

- CyExpandedBdAddrType AddressType [get]
  Gets the address type of the generated Bluetooth device address

---

Detailed Description

Holds the generated Bluetooth device address

Property Documentation

CyBleBdAddress GeneratedAddress [get]

Gets the generated Bluetooth device address

CyExpandedBdAddrType AddressType [get]

Gets the address type of the generated Bluetooth device address

---

CyGenerateSecureConnectionOobDataInfo

Holds the information necessary to generate OOOB data for secure connection pairing

Public Member Functions

- CyGenerateSecureConnectionOobDataInfo (byte[] rand)
  Create information necessary to generate secure connection OOB data

Properties

- byte[] Rand [get]
  16-byte random number to be used for OOB data generation

Detailed Description

Holds the information necessary to generate OOOB data for secure connection pairing

Constructor & Destructor Documentation

CyGenerateSecureConnectionOobDataInfo (byte[] *rand*)

Create information necessary to generate secure connection OOB data

Parameters:

rand	16-byte random number to be used for OOB data generation

---

Property Documentation

byte [] Rand `[get]`

16-byte random number to be used for OOB data generation

CyGetPeerDeviceAuthenticationKeyInfo

Holds the information necessary to get the authentication keys of a peer device

Public Member Functions

- CyGetPeerDeviceAuthenticationKeyInfo (byte deviceHandle)
  Creates the information necessary to get the authentication keys

Properties

- byte DeviceHandle `[get]`
  Gets the device handle of the peer device

Detailed Description

Holds the information necessary to get the authentication keys of a peer device

Constructor & Destructor Documentation

CyGetPeerDeviceAuthenticationKeyInfo (byte *deviceHandle*)

Creates the information necessary to get the authentication keys

Parameters:

deviceHandle	Device handle to the peer device

Property Documentation

byte DeviceHandle [get]

Gets the device handle of the peer device

CyGetPeerDeviceAuthenticationKeyResult

Holds the information necessary to get the authentication keys of a peer device

Properties

- byte DeviceHandle [get]
  Gets the device handle of the peer device
- CyAuthenticationKeys AuthenticationKeys [get]
  Gets the peer device authentication keys

Detailed Description

Holds the information necessary to get the authentication keys of a peer device

Property Documentation

byte DeviceHandle [get]

Gets the device handle of the peer device

CyAuthenticationKeys AuthenticationKeys [get]

Gets the peer device authentication keys

CyL2CapConnectionResponseInfo

Holds the L2CAP connection request details and the response to be sent

Properties

- ushort LocalChannelID `[get]`
  Gets the local channel ID
- ushort LocalPSM `[get]`
  Gets the local PSM
- ushort RemoteMTU `[get]`
  Gets the remote MTU
- ushort RemoteMPS `[get]`
  Gets the remote MPS
- ushort InitialCreditsForLocalDevice `[get]`
  Gets the initial credits sent by the remote device
- ushort LocalMTU `[get, set]`
  Gets/Sets the local MTU
- ushort LocalMPS `[get, set]`
  Gets/Sets the local MPS
- ushort InitialCreditsToRemoteDevice `[get, set]`
  Gets/Sets the initial credits
- CyL2CapConnectionResponseCode Response `[get, set]`
  Gets/Sets the response to the request

Detailed Description

Holds the L2CAP connection request details and the response to be sent

Property Documentation

ushort LocalChannelID `[get]`

Gets the local channel ID

ushort LocalPSM `[get]`

Gets the local PSM

ushort RemoteMTU `[get]`

Gets the remote MTU

ushort RemoteMPS [get]

Gets the remote MPS

ushort InitialCreditsForLocalDevice [get]

Gets the initial credits sent by the remote device

ushort LocalMTU [get], [set]

Gets/Sets the local MTU

ushort LocalMPS [get], [set]

Gets/Sets the local MPS

ushort InitialCreditsToRemoteDevice [get], [set]

Gets/Sets the initial credits

CyL2CapConnectionResponseCode Response [get], [set]

Gets/Sets the response to the request

CyL2CapDataReceivedInfo

Holds the information about the data received over an L2CAP channel

Properties

- ushort LocalChannelID [get]
  Gets the channel ID of L2CAP channel on which the data was received
- CyL2CapResultCode Status [get]
  Gets the status

- byte[] Data `[get]`
  Gets the received data

Detailed Description

Holds the information about the data received over an L2CAP channel

Property Documentation

ushort LocalChannelID `[get]`

Gets the channel ID of L2CAP channel on which the data was received

CyL2CapResultCode Status `[get]`

Gets the status

byte [] Data `[get]`

Gets the received data

CyL2CapDisconnectConfirmation

Holds the result of an L2CAP channel disconnect request

Properties

- ushort LocalChannelID `[get]`
  Gets the channel ID of L2CAP channel that was disconnected
- CyL2CapResultCode ResultCode `[get]`
  Gets result of the disconnect

Detailed Description

Holds the result of an L2CAP channel disconnect request

Property Documentation

ushort LocalChannelID [get]

Gets the channel ID of L2CAP channel that was disconnected

CyL2CapResultCode ResultCode [get]

Gets result of the disconnect

CyL2CapDisconnectIndicationInfo

Holds the information about the L2CAP channel disconnected by the peer device or by the local device stack

Properties

- ushort LocalChannelID [get]
  Gets the channel ID of L2CAP channel that was disconnected
- bool IsDisconnectedByPeerDevice [get]
  Gets whether the channel was disconnected by the peer device
- CyL2CapResultCode Reason [get]
  Gets the reason for disconnect This will always be CyL2CapResultCode.SUCCESS, if the disconnect was triggered by the peer device

---

Detailed Description

Holds the information about the L2CAP channel disconnected by the peer device or by the local device stack

Property Documentation

ushort LocalChannelID [get]

Gets the channel ID of L2CAP channel that was disconnected

bool IsDisconnectedByPeerDevice [get]

Gets whether the channel was disconnected by the peer device

CyL2CapResultCode Reason [get]

Gets the reason for disconnect This will always be CyL2CapResultCode.SUCCESS, if the disconnect was triggered by the peer device

---

CyL2CapMgrCallback

L2CAP manager callback Defines callback method for L2CAP manager APIs

Public Member Functions

- virtual void OnChannelConnectionIndication (CyL2CapConnectionResponseInfo response)
  Reports an L2CAP channel connection request received from a peer device
- virtual void OnChannelDisconnectIndication (CyL2CapDisconnectIndicationInfo info)
  Reports an L2CAP channel disconnect initiated by the local device stack or by the peer device
- virtual void OnDisconnectChannel (CyL2CapDisconnectConfirmation result, CyStatus status)
  Reports the status of ICyL2CapMgr.DisconnectChannel
- virtual void OnChannelEstablished (ICyL2CapChannel channel, CyStatus status)
  Reports the established L2CAP channel
- virtual void OnDataReceived (CyL2CapDataReceivedInfo info)
  Reports the data received over an L2CAP channel
- virtual void OnReceiveCreditLowIndication (CyL2CapReceiveCreditLowInfo info)
  Reports that the receive credit of an L2CAP channel is low
- virtual void OnTransmitCreditIndication (CyL2CapTransmitCreditInfo info)
  Reports the transmit flow credit received from a peer device, for an L2CAP channel
- virtual void OnSendData (ushort channelID, CyStatus status)
  Reports the status of ICyL2CapMgr.SendData
- virtual void OnSendCredits (ushort channelID, CyStatus status)
  Reports the status of ICyL2CapMgr.SendCredits

Detailed Description

L2CAP manager callback Defines callback method for L2CAP manager APIs

Member Function Documentation

virtual void OnChannelConnectionIndication (CyL2CapConnectionResponseInfo *response*) [virtual]

Reports an L2CAP channel connection request received from a peer device

Parameters:

response	Contains the details of the channel request and the response instance

Use ICyL2CapMgr.RespondToChannelRequest method to respond to the request

virtual void OnChannelDisconnectIndication (CyL2CapDisconnectIndicationInfo *info*) [virtual]

Reports an L2CAP channel disconnect initiated by the local device stack or by the peer device

Parameters:

info	Information about the L2CAP channel that was disconnected

virtual void OnDisconnectChannel (CyL2CapDisconnectConfirmation *result*, CyStatus *status*) [virtual]

Reports the status of ICyL2CapMgr.DisconnectChannel

Parameters:

result	Disconnect confirmation
status	Status of ICyL2CapMgr.DisconnectChannel

virtual void OnChannelEstablished (ICyL2CapChannel *channel*, CyStatus *status*) [virtual]

Reports the established L2CAP channel

Parameters:

channel	Contains the newly created L2CAP channel, if status is OK; Otherwise is null

status	Status of ICyL2CapMgr.EstablishChannel. Ignore, if the channel establishment was triggered by a peer device

virtual void OnDataReceived (CyL2CapDataReceivedInfo *info*) [virtual]

Reports the data received over an L2CAP channel

Parameters:

info	Information about the received data

virtual void OnReceiveCreditLowIndication (CyL2CapReceiveCreditLowInfo *info*) [virtual]

Reports that the receive credit of an L2CAP channel is low

Parameters:

info	Information about the receive credit low indication

virtual void OnTransmitCreditIndication (CyL2CapTransmitCreditInfo *info*) [virtual]

Reports the transmit flow credit received from a peer device, for an L2CAP channel

Parameters:

info	Information about the transmit flow credit received

virtual void OnSendData (ushort *channelID*, CyStatus *status*) [virtual]

Reports the status of ICyL2CapMgr.SendData

Parameters:

channelID	L2CAP channel ID on which the data was sent
status	Status of ICyL2CapMgr.SendData

virtual void OnSendCredits (ushort *channelID*, CyStatus *status*) [virtual]

Reports the status of ICyL2CapMgr.SendCredits

Parameters:

channelID	L2CAP channel ID on which the credits were sent
status	Status of ICyL2CapMgr.SendCredits

CyL2CapReceiveCreditLowInfo

Holds the information about the receive credit low indication

Properties

- ushort LocalChannelID [get]
  Gets the channel ID of L2CAP channel on which the credit limit has reached the low watermark
- ushort Credits [get]
  Gets the current receive credits

Detailed Description

Holds the information about the receive credit low indication

Property Documentation

ushort LocalChannelID [get]

Gets the channel ID of L2CAP channel on which the credit limit has reached the low watermark

ushort Credits [get]

Gets the current receive credits

CyL2CapSendCreditsInfo

Holds the information necessary to send L2CAP flow control credits

Public Member Functions

- CyL2CapSendCreditsInfo (ICyL2CapChannel channel, ushort credits)
  Creates the information necessary to send flow control credits

Properties

- ICyL2CapChannel Channel `[get]`
  Gets the L2CAP channel for which flow control credits need to be sent
- ushort Credits `[get]`
  Gets the flow control credit to be sent

Detailed Description

Holds the information necessary to send L2CAP flow control credits

Constructor & Destructor Documentation

CyL2CapSendCreditsInfo (ICyL2CapChannel *channel*, ushort *credits*)

Creates the information necessary to send flow control credits

Parameters:

channel	L2CAP channel for which the credits need to be sent
credits	Flow control credits

Property Documentation

ICyL2CapChannel Channel `[get]`

Gets the L2CAP channel for which flow control credits need to be sent

ushort Credits [get]

Gets the flow control credit to be sent

CyL2CapSendDataInfo

Holds the information necessary to send data over an L2CAP channel

Public Member Functions

- CyL2CapSendDataInfo (ICyL2CapChannel channel, params byte[] data)
  Creates the information necessary to send data

Properties

- ICyL2CapChannel Channel [get]
  Gets the L2CAP channel over which the data needs to be sent
- byte[] Data [get]
  Gets the data to be sent

Detailed Description

Holds the information necessary to send data over an L2CAP channel

Constructor & Destructor Documentation

CyL2CapSendDataInfo (ICyL2CapChannel *channel*, params byte[] *data*)

Creates the information necessary to send data

Parameters:

channel	L2CAP channel over which the data needs to be sent
data	Data to be sent

Property Documentation

ICyL2CapChannel Channel [get]

Gets the L2CAP channel over which the data needs to be sent

byte [] Data [get]

Gets the data to be sent

CyL2CapTransmitCreditInfo

Holds the transmit flow credits received from a peer device

Properties

- ushort LocalChannelID [get]
  Gets the channel ID of L2CAP channel for which transmit credits were received
- ushort Credits [get]
  Gets the current available transmit credits
- CyL2CapResultCode Status [get]
  Gets the status

Detailed Description

Holds the transmit flow credits received from a peer device

Property Documentation

ushort LocalChannelID [get]

Gets the channel ID of L2CAP channel for which transmit credits were received

ushort Credits [get]

Gets the current available transmit credits

CyL2CapResultCode Status [get]

Gets the status

CyNumericComparisonResponse

Holds the information necessary to respond to a secure connection numeric comparison request

Properties

- uint Passkey [get]
  Gets the 6 digit numeric value
- CyPairingResponseCode Response [get, set]
  Gets/Sets the response

Detailed Description

Holds the information necessary to respond to a secure connection numeric comparison request

Property Documentation

uint Passkey [get]

Gets the 6 digit numeric value

CyPairingResponseCode Response [get], [set]

Gets/Sets the response

CyOobData

Holds the OOB data to be set

Public Member Functions

- CyOobData (byte[] oobKey)
  Set the OOB key

- CyOobData (byte[] oobKey, byte[] oobData)
  Set OOB key and data

Properties

- byte[] OobKey `[get]`
  Gets the 16-byte OOB key
- byte[] OobData `[get]`
  Gets the 16-byte OOB data

Detailed Description

Holds the OOB data to be set

Constructor & Destructor Documentation

CyOobData (byte[] *oobKey*)

Set the OOB key

Parameters:

oobKey	16-byte OOB key to be used

CyOobData (byte[] *oobKey*, byte[] *oobData*)

Set OOB key and data

Parameters:

oobKey	16-byte OOB key to be used
oobData	16-byte OOB data to be used

Property Documentation

byte [] OobKey `[get]`

Gets the 16-byte OOB key

byte [] OobData [get]

Gets the 16-byte OOB data

CyPairSettings

Holds the settings to be used for pairing

Public Member Functions

- CyPairSettings (CySecurityLevel securityLevel, bool bonding, byte encryptionKeySize, CyPairingProperties properties)
  Creates the settings to be used for pairing

Properties

- CySecurityLevel SecurityLevel [get]
  Gets the security level to be used for pairing
- bool Bonding [get]
  Gets whether bonding is enabled or disabled
- byte EncryptionKeySize [get]
  Gets the encryption size
- CyPairingProperties PairingProperties [get]
  Gets the pairing properties

Detailed Description

Holds the settings to be used for pairing

Constructor & Destructor Documentation

CyPairSettings (CySecurityLevel *securityLevel*, bool *bonding*, byte *encryptionKeySize*, CyPairingProperties *properties*)

Creates the settings to be used for pairing

Parameters:

securityLevel	Security level

bonding	Enable/Disable bonding
encryptionKeySize	Encryption key size Range: 7 to 16

Property Documentation

CySecurityLevel SecurityLevel [get]

Gets the security level to be used for pairing

bool Bonding [get]

Gets whether bonding is enabled or disabled

byte EncryptionKeySize [get]

Gets the encryption size

CyPairingProperties PairingProperties [get]

Gets the pairing properties

CyPasskeyDisplayInfo

Holds the passkey information to be displayed

Properties

- uint Passkey [get]
  Gets the 6 digit passkey to be displayed

Detailed Description

Holds the passkey information to be displayed

Property Documentation

uint Passkey [get]

Gets the 6 digit passkey to be displayed

CyPasskeyEntryResponse

Holds the information necessary to respond to a passkey entry request

Properties

- uint Passkey [get, set]
  Gets/Sets the 6 digit passkey
- bool IsKeyPressNotificationRequired [get]
  Gets whether keypress notification is required or not. This is applicable only for secure connection
- CyPairingResponseCode Response [get, set]
  Gets/Sets the response

Detailed Description

Holds the information necessary to respond to a passkey entry request

Property Documentation

uint Passkey [get], [set]

Gets/Sets the 6 digit passkey

bool IsKeyPressNotificationRequired [get]

Gets whether keypress notification is required or not. This is applicable only for secure connection
If true, use ICyBleDevice.SendKeyPressNotification to send the keypress notification

CyPairingResponseCode Response [get], [set]

Gets/Sets the response

CyReadCharacteristicByUUIDInfo

Holds the information necessary to discover characteristics by UUID

Public Member Functions

- CyReadCharacteristicByUUIDInfo (CyUUID uuid, ushort startHandle, ushort endHandle)
  Creates the information necessary to read characteristics by UUID

Properties

- CyUUID UUID `[get]`
  UUID of the characteristic to be read
- ushort StartHandle `[get]`
  Handle from which the search for the characteristic should begin
- ushort EndHandle `[get]`
  Handle at which the search for the characteristic should end

---

Detailed Description

Holds the information necessary to discover characteristics by UUID

Constructor & Destructor Documentation

CyReadCharacteristicByUUIDInfo (CyUUID *uuid*, ushort *startHandle*, ushort *endHandle*)

Creates the information necessary to read characteristics by UUID

Parameters:

uuid	UUID of the characteristics to be read
startHandle	Start handle from which the search for the characteristic should begin
endHandle	End handle at which the search for the characteristic should end

---

Property Documentation

CyUUID UUID [get]

UUID of the characteristic to be read

ushort StartHandle [get]

Handle from which the search for the characteristc should begin

ushort EndHandle [get]

Handle at which the search for the characteristic should end

CyReadCharacteristicByUUIDResult

Holds the result of read characteristic by UUID

Properties

- CyGattReadResult[] Records [get]
  Gets all the characteristics that were read

Detailed Description

Holds the result of read characteristic by UUID

Property Documentation

CyGattReadResult [] Records [get]

Gets all the characteristics that were read

CyReadMultipleCharacteristicInfo

Holds the information necessary to read multiple characteristic

Public Member Functions

- CyReadMultipleCharacteristicInfo (params CyGattCharacteristic[] characteristics)
  Creates the information necessary to read multiple characteristics
- CyReadMultipleCharacteristicInfo (params ushort[] handles)
  Creates the information necessary to read multiple characteristics

Properties

- ushort[] Handles `[get]`
  Get the characteristic value handles to be read

Detailed Description

Holds the information necessary to read multiple characteristic

Constructor & Destructor Documentation

CyReadMultipleCharacteristicInfo (params CyGattCharacteristic[] *characteristics*)

Creates the information necessary to read multiple characteristics

Parameters:

characteristics	Array of characteristics to be read

CyReadMultipleCharacteristicInfo (params ushort[] *handles*)

Creates the information necessary to read multiple characteristics

Parameters:

handles	Array of characteristic value handles to be read

Property Documentation

ushort [] Handles `[get]`

Get the characteristic value handles to be read

CyReadMultipleCharacteristicResult

Holds the result of multiple characteristic request

Properties

- byte[] Value `[get]`
  Gets the value returned by the peer device

Detailed Description

Holds the result of multiple characteristic request

Property Documentation

byte [] Value `[get]`

Gets the value returned by the peer device

CyRegisteredPsm

Represents a registered L2CAP PSM

Properties

- ushort PSM `[get]`
  Gets the registered PSM
- ushort ReceiveLowWatermark `[get]`
  Gets the receive low watermark value for the PSM

Detailed Description

Represents a registered L2CAP PSM

Property Documentation

ushort PSM `[get]`

Gets the registered PSM

ushort ReceiveLowWatermark [get]

Gets the receive low watermark value for the PSM

CyRegisterPsmInfo

Holds the information necessary to register a PSM

Public Member Functions

- CyRegisterPsmInfo (ushort psm, ushort receiveLowWatermark)
  Creates the info class object needed to register a PSM

Properties

- ushort PSM [get]
  Gets the PSM to be registered
- ushort ReceiveLowWatermark [get]
  Gets the receive low watermark value for the PSM

Detailed Description

Holds the information necessary to register a PSM

Constructor & Destructor Documentation

CyRegisterPsmInfo (ushort *psm*, ushort *receiveLowWatermark*)

Creates the info class object needed to register a PSM

Parameters:

psm	PSM to be registered
receiveLowWatermark	Low credit watermark to be set for the PSM

Property Documentation

ushort PSM [get]

Gets the PSM to be registered

ushort ReceiveLowWatermark [get]

Gets the receive low watermark value for the PSM

CyReliableWriteInfo

Holds the information necessary to perform a reliable write

Public Member Functions

- CyReliableWriteInfo (params CyGattWriteInfo[] records)
  Creates the information necessary to perform a reliable write

Properties

- CyGattWriteInfo[] Records [get]
  Gets all the characteristics to be written

Detailed Description

Holds the information necessary to perform a reliable write

Constructor & Destructor Documentation

CyReliableWriteInfo (params CyGattWriteInfo[] *records*)

Creates the information necessary to perform a reliable write

Parameters:

records	Characteristics to be written

Property Documentation

CyGattWriteInfo [] Records [get]

Gets all the characteristics to be written

CyResolvableAddressResult

Holds the local or peer resolvable private address

Properties

- CyBleBdAddress PeerIdAddress [get]
  Gets the identity address of the peer device
- CyBleBdAddress ResolvableAddress [get]
  Gets the resolvable private address

Detailed Description

Holds the local or peer resolvable private address

Property Documentation

CyBleBdAddress PeerIdAddress [get]

Gets the identity address of the peer device

CyBleBdAddress ResolvableAddress [get]

Gets the resolvable private address

CyResolvableAddressTimeoutInfo

Holds the information necessary to set the resolvable address generation timeout

Public Member Functions

- CyResolvableAddressTimeoutInfo (ushort timeoutInSeconds)
  Creates the information to set the address generation timeout

Properties

- ushort TimeoutInSeconds `[get]`
  Gets the resolvable address generation timeout in seconds

Detailed Description

Holds the information necessary to set the resolvable address generation timeout

Constructor & Destructor Documentation

CyResolvableAddressTimeoutInfo (ushort *timeoutInSeconds*)

Creates the information to set the address generation timeout

Parameters:

timeoutInSeconds	Timeout in seconds. Range - 0x0001 to 0xA1B8

Property Documentation

ushort TimeoutInSeconds `[get]`

Gets the resolvable address generation timeout in seconds

CyResolvePeerDeviceInfo

Holds the information necessary to resolve a peer device

Public Member Functions

- CyResolvePeerDeviceInfo (CyBleBdAddress peerDeviceAddress)
  Create the information necessary to resolve a peer device address

Properties

- CyBleBdAddress PeerDeviceAddress [get]
  Gets the peer device address that needs to be resolved

Detailed Description

Holds the information necessary to resolve a peer device

Constructor & Destructor Documentation

CyResolvePeerDeviceInfo (CyBleBdAddress *peerDeviceAddress*)

Create the information necessary to resolve a peer device address

Parameters:

peerDeviceAddress	Address of the peer device This is typically the peer device address in the received advertisement

---

Property Documentation

CyBleBdAddress PeerDeviceAddress [get]

Gets the peer device address that needs to be resolved

CyResolvingListDevice

Represents a device in the resolving list

Properties

- CyBleBdAddress PeerIdAddress [get]
  Gets the peer device ID address
- byte[] LocalIRK [get]
  Gets the 16-byte local IRK
- byte[] PeerIRK [get]
  Gets the 16-byte peer IRK

Detailed Description

Represents a device in the resolving list

Property Documentation

CyBleBdAddress PeerIdAddress [get]

Gets the peer device ID address

byte [] LocalIRK [get]

Gets the 16-byte local IRK

byte [] PeerIRK [get]

Gets the 16-byte peer IRK

CyScanCallback

Defines scan callback methods

Public Member Functions

- abstract void OnScanResult (CyScanResult result)
  This callback reports the discovered BLE devices
- virtual void OnScanStatusChanged (CyScanStatus scanStatus)
  This callback reports the scan status
- virtual void OnStartScanError (CyStatus status)
  Reports the ICyBleMgr.StartScan error. This callback will not be called when there is no error
- virtual void OnStopScanError (CyStatus status)
  Reports the ICyBleMgr.StopScan error. This callback will not be called when there is no error

---

Detailed Description

Defines scan callback methods

Member Function Documentation

abstract void OnScanResult (CyScanResult *result*) [pure virtual]

This callback reports the discovered BLE devices

Parameters:

result	Scan result

This callback may be called multiple times while the scan is active

virtual void OnScanStatusChanged (CyScanStatus *scanStatus*) [virtual]

This callback reports the scan status

Parameters:

scanStatus	True if the scan is in-progress; otherwise, False

virtual void OnStartScanError (CyStatus *status*) [virtual]

Reports the ICyBleMgr.StartScan error. This callback will not be called when there is no error

Parameters:

status	Status of the ICyBleMgr.StartScan method execution

virtual void OnStopScanError (CyStatus *status*) [virtual]

Reports the ICyBleMgr.StopScan error. This callback will not be called when there is no error

Parameters:

status	Status of the ICyBleMgr.StopScan method execution

CyScanRecord

Represents an advertisement or scan response of a BLE device

Properties

- CyBleAdvEventType AdvertisementType `[get]`
  Gets the advertisement type
- bool IsDirectedRpaAdvertisement `[get]`
  Gets whether the advertisement is a directed advertisement with RPA used for InitA This is applicable only when Privacy 1.2 is used
- CyBleBdAddress LocalDeviceAddress `[get]`
  Gets the local Bluetooth device address This is valid only if IsDirectedRpaAdvertisement is True
- CyBleBdAddress PeerDeviceAddress `[get]`
  Gets the address of the advertising BLE device
- sbyte RSSI `[get]`
  Gets the RSSI
- CyAdvertisementData AdvertisementData `[get]`
  Gets the advertisement or scan response data

Detailed Description

Represents an advertisement or scan response of a BLE device

Property Documentation

CyBleAdvEventType AdvertisementType `[get]`

Gets the advertisement type

bool IsDirectedRpaAdvertisement `[get]`

Gets whether the advertisement is a directed advertisement with RPA used for InitA This is applicable only when Privacy 1.2 is used

CyBleBdAddress LocalDeviceAddress `[get]`

Gets the local Bluetooth device address This is valid only if IsDirectedRpaAdvertisement is True

CyBleBdAddress PeerDeviceAddress [get]

Gets the address of the advertising BLE device

sbyte RSSI [get]

Gets the RSSI

CyAdvertisementData AdvertisementData [get]

Gets the advertisement or scan response data

CyScanResult

Holds the scan results

Properties

- List< CyScanRecord > ScanRecords [get]
  Gets the scan records

Detailed Description

Holds the scan results

Property Documentation

List<CyScanRecord> ScanRecords [get]

Gets the scan records

CySecureConnectionOobDataResult

Holds the generated secure connection OOB data

Public Member Functions

- CySecureConnectionOobDataResult (byte[] oobKey, byte[] oobData)
  Creates an instance to hold the generated OOB key and data for secure connection

Properties

- byte[] OobKey `[get]`
  Gets the 16-byte OOB key
- byte[] OobData `[get]`
  Gets the generated 16-byte OOB data

Detailed Description

Holds the generated secure connection OOB data

Constructor & Destructor Documentation

CySecureConnectionOobDataResult (byte[] *oobKey*, byte[] *oobData*)

Creates an instance to hold the generated OOB key and data for secure connection

Parameters:

oobKey	16-byte OOB key
oobData	16-byte OOB data

Property Documentation

byte [] OobKey `[get]`

Gets the 16-byte OOB key

byte [] OobData `[get]`

Gets the generated 16-byte OOB data

CySecurityMgrCallback

Security manager callback Defines callback method for security manager APIs

Public Member Functions

- virtual void OnGetAuthenticationKeys (CyAuthenticationKeys result, CyStatus status)
  Reports the local authentication keys
- virtual void OnGenerateAuthenticationKeys (CyAuthenticationKeys result, CyStatus status)
  Reports the generated authentication keys
- virtual void OnSetAuthenticationKeys (CyStatus status)
  Reports status of the ICyBleSecurityMgr.SetAuthenticationKeys method
- virtual void OnGetPeerDeviceAuthenticationKeys (CyGetPeerDeviceAuthenticationKeyResult result, CyStatus status)
  Reports the peer device authentication keys
- virtual void OnResolvePeerDeviceAddress (CyResolveAddressResult result, CyStatus status)
  Reports the status of ICyBleSecurityMgr.ResolvePeerDeviceAddress method
- virtual void OnGenerateSecureConnectionOobData (CySecureConnectionOobDataResult result, CyStatus status)
  Reports the generated secure connection OOB data
- virtual void OnGenerateLocalECDHKey (CyStatus status)
  Reports the status of ICyBleSecurityMgr.GenerateLocalECDHKey method

---

Detailed Description

Security manager callback Defines callback method for security manager APIs

Member Function Documentation

virtual void OnGetAuthenticationKeys (CyAuthenticationKeys *result*, CyStatus *status*) [virtual]

Reports the local authentication keys

Parameters:

result	Contains the current local authentication keys, if *status* is OK; Otherwise is null
status	Status of the ICyBleSecurityMgr.GetAuthenticationKeys method execution

virtual void OnGenerateAuthenticationKeys (CyAuthenticationKeys *result*, CyStatus *status*) [virtual]

Reports the generated authentication keys

Parameters:

result	Contains the generated authentication keys, if *status* is OK; Otherwise is null
status	Status of the ICyBleSecurityMgr.GenerateAuthenticationKeys method execution

virtual void OnSetAuthenticationKeys (CyStatus *status*) [virtual]

Reports status of the ICyBleSecurityMgr.SetAuthenticationKeys method

Parameters:

status	Status of the ICyBleSecurityMgr.SetAuthenticationKeys method execution

virtual void OnGetPeerDeviceAuthenticationKeys (CyGetPeerDeviceAuthenticationKeyResult *result*, CyStatus *status*) [virtual]

Reports the peer device authentication keys

Parameters:

result	Contains the peer device authentication keys
status	Status of ICyBleSecurityMgr.GetPeerDeviceAuthenticationKeys

virtual void OnResolvePeerDeviceAddress (CyResolveAddressResult *result*, CyStatus *status*) [virtual]

Reports the status of ICyBleSecurityMgr.ResolvePeerDeviceAddress method

Parameters:

result	Contains the result of the peer device address resolution, if *status* is OK; Otherwise is null
status	Status of ICyBleSecurityMgr.ResolvePeerDeviceAddress method execution

virtual void OnGenerateSecureConnectionOobData (CySecureConnectionOobDataResult *result*, CyStatus *status*) [virtual]

Reports the generated secure connection OOB data

Parameters:

result	Contains the generated OOB data, if *status* is OK; Otherwise is null
status	Status of ICyBleSecurityMgr.GenerateSecureConnectionOobData method execution

virtual void OnGenerateLocalECDHKey (CyStatus *status*) [virtual]

Reports the status of ICyBleSecurityMgr.GenerateLocalECDHKey method

Parameters:

status	Status of ICyBleSecurityMgr.GenerateLocalECDHKey method execution

---

CySetSuggestedDataLengthInfo

Holds the information necessary to set the suggested data length

Public Member Functions

- CySetSuggestedDataLengthInfo (ushort suggestedMaxTxOctets, ushort suggestedMaxTxTime)
  Creates the information necessary to set the suggested data length

Properties

- ushort SuggestedMaxTxOctets `[get]`
  Gets the suggested max. Tx octets
- ushort SuggestedMaxTxTime `[get]`
  Gets the suggested max. Tx time in µs

Detailed Description

Holds the information necessary to set the suggested data length

Constructor & Destructor Documentation

CySetSuggestedDataLengthInfo (ushort *suggestedMaxTxOctets*, ushort *suggestedMaxTxTime*)

Creates the information necessary to set the suggested data length

Parameters:

suggestedMaxTxOctets	Suggested maximum Tx octet
suggestedMaxTxTime	Suggested maximum Tx time in µs Use ICyBleMgr.ConvertDataLengthOctetToTime API to get the time from the octet value

---

Property Documentation

ushort SuggestedMaxTxOctets `[get]`

Gets the suggested max. Tx octets

ushort SuggestedMaxTxTime `[get]`

Gets the suggested max. Tx time in µs

CySmartDongleMgr

Provides APIs to manage the CySmart dongle

Public Member Functions

- CyApiErr TryGetCySmartDongleCommunicator (CyDongleInfo info, out ICySmartDongleCommunicator communicator)
  Get the CySmart dongle communicator
- CyApiErr CloseCommunicator (ICySmartDongleCommunicator communicator)
  Close a communicator

Static Public Member Functions

- static CySmartDongleMgr GetInstance ()
  Gets the CySmart dongle manager instance

Detailed Description

Provides APIs to manage the CySmart dongle

Member Function Documentation

CyApiErr TryGetCySmartDongleCommunicator (CyDongleInfo *info*, out ICySmartDongleCommunicator *communicator*)

Get the CySmart dongle communicator

Parameters:

info	Information necessary to get the communicator
communicator	When this method returns, contains the communicator for the specified CySmart dongle, if the CySmart dongle is valid; otherwise, the communicator is null This parameter is passed uninitialized

Returns:

CyApiErr.OK if the communicator was successfully created; otherwise, contains the error

CyApiErr CloseCommunicator (ICySmartDongleCommunicator *communicator*)

Close a communicator

Parameters:

communicator	Communicator to be closed

Returns:

CyApiErr.OK if the communicator was successfully closed; otherwise, contains the error

Once closed, the communicator instances cannot be used again. Get a new communicator instance by calling the TryGetCySmartDongleCommunicator method

static CySmartDongleMgr GetInstance () `[static]`

Gets the CySmart dongle manager instance

Returns:

CyTxPowerInfo

Holds channel power level information

Public Member Functions

- CyTxPowerInfo (CyChannelGroup channelGroup, CyPowerLevel powerLevel)
  Creates an instance of the channel Tx power level

Properties

- CyChannelGroup ChannelGroup `[get]`
  Gets the channel group
- CyPowerLevel PowerLevel `[get]`
  Gets the power level

Detailed Description

Holds channel power level information

Constructor & Destructor Documentation

CyTxPowerInfo (CyChannelGroup *channelGroup*, CyPowerLevel *powerLevel*)

Creates an instance of the channel Tx power level

Parameters:

channelGroup	Channel group
powerLevel	Channel power level

Property Documentation

CyChannelGroup ChannelGroup [get]

Gets the channel group

CyPowerLevel PowerLevel [get]

Gets the power level

CyUUID

Represents an UUID

Public Member Functions

- CyUUID (ushort uuid16)
  Create a 16-bit UUID
- CyUUID (params byte[] uuid128)
  Create a 128-bit UUID

Public Attributes

- const int **UUID_16_LENGTH_IN_BYTES** = 2
- const int **UUID_128_LENGTH_IN_BYTES** = 16
- const int **UUID16_START_INDEX_IN_UUID128** = 12

Properties

- ushort UUID16 `[get]`
  Gets the 16-bit UUID
- bool IsUUID16Valid `[get]`
  Gets whether the 16-bit UUID representation is valid or not
- byte[] UUID128 `[get]`
  Gets the 128-bit UUID byte array in little endian

Detailed Description

Represents an UUID

Constructor & Destructor Documentation

CyUUID (ushort *uuid16*)

Create a 16-bit UUID

Parameters:

uuid16	16-bit UUID value

CyUUID (params byte[] *uuid128*)

Create a 128-bit UUID

Parameters:

uuid128	128-bit UUID bytes in little endian format

Property Documentation

ushort UUID16 `[get]`

Gets the 16-bit UUID

bool IsUUID16Valid [get]

Gets whether the 16-bit UUID representation is valid or not

byte [] UUID128 [get]

Gets the 128-bit UUID byte array in little endian

CyWhitelistDevice

Represents a device in the whitelist

Properties

- CyBleBdAddress DeviceAddress [get]
  Gets the device address

Detailed Description

Represents a device in the whitelist

Property Documentation

CyBleBdAddress DeviceAddress [get]

Gets the device address

CyWriteBufferFullResponse

Holds the response to be sent when the peer device buffer is full before the complete write request could be queued

Public Types

- enum ResponseCode { ABORT, EXECUTE }*Response to be sent*

Properties

- ushort Handle `[get]`
  Gets the attribute handle that was not queued
- ResponseCode Response `[get, set]`
  Gets/Sets the response to be sent to the peer device

Detailed Description

Holds the response to be sent when the peer device buffer is full before the complete write request could be queued

Member Enumeration Documentation

enum ResponseCode `[strong]`

Response to be sent

Enumerator

ABORT Abort all queued requests

EXECUTE Execute all queued requests

Property Documentation

ushort Handle `[get]`

Gets the attribute handle that was not queued

ResponseCode Response `[get], [set]`

Gets/Sets the response to be sent to the peer device

ICyBleDevice

Represents a remote device which is in connected state

Public Member Functions

- CyApiErr RegisterCallback (CyBleDeviceCallback cb)
  Register device callback
- CyApiErr SetOob (bool enable, CyOobData data)
  Set OOB status and data
- CyApiErr Pair (CyPairSettings settings)
  Initiate pairing with the remote device
- CyApiErr SendKeyPressNotification (CyKeyPressNotification notification)
  Send the key press notification This needs to be called only for secure connection pairing with keypress enabled
- CyApiErr SendPasskeyResponse (CyPasskeyEntryResponse response)
  Send passkey response for a passkey entry request
- CyApiErr SendNumericComparisonResponse (CyNumericComparisonResponse response)
  Send numeric comparison response for numeric comparison request
- CyApiErr SetDataLength (CyDataLengthInfo info)
  Set the data packet length for the current connection This data is then negotiated with the remote device
- CyApiErr UpdateConnectionParameter (CyConnectionParameters settings)
  Update connection parameters of the remote device
- CyApiErr SendConnectionParametersResponse (CyConnectionParametersResponse response)
  Send response to connection parameter update request received from peer device

Properties

- CyBleBdAddress Address `[get]`
  Gets the address of the remote device
- ushort Handle `[get]`
  Gets the connection handle
- ICyGattClient GattClient `[get]`
  Gets the GATT client
- ICyL2CapMgr L2CapMgr `[get]`
  Gets the L2CAP manager

Detailed Description

Represents a remote device which is in connected state

Member Function Documentation

CyApiErr RegisterCallback (CyBleDeviceCallback *cb*)

Register device callback

Parameters:

cb	Device callback

Returns:

CyApiErr.OK if the callback was successfully registered; otherwise, contains the error

CyApiErr SetOob (bool *enable*, CyOobData *data*)

Set OOB status and data

Parameters:

enable	Enable or disable OOB
data	OOB data received from the peer device

Returns:

CyApiErr.OK if the method parameters are correct; otherwise, contains the error

Status of this operation is reported via the CyBleDeviceCallback.OnSetOob callback

CyApiErr Pair (CyPairSettings *settings*)

Initiate pairing with the remote device

Parameters:

settings	Settings to be used for pairing

Returns:

CyApiErr.OK if the method parameters are correct; otherwise, contains the error

Based on the security settings and device IO capabilities, one of the callback may be invoked to complete the pairing procedure

1. CyBleDeviceCallback.OnPasskeyEntryRequest
2. CyBleDeviceCallback.OnPasskeyDisplayRequest
3. CyBleDeviceCallback.OnNumericComparisonRequest

Status of the overall pairing procedure is report via the CyBleDeviceCallback.OnPairingCompleted callback method

CyApiErr SendKeyPressNotification (CyKeyPressNotification *notification*)

Send the key press notification This needs to be called only for secure connection pairing with keypress enabled

Parameters:

notification	Keypress notification

Returns:

CyApiErr.OK if the method parameters are correct; otherwise, contains the error
Status of this operation is reported via the CyBleDeviceCallback.OnSendKeyPressNotification callback

CyApiErr SendPasskeyResponse (CyPasskeyEntryResponse *response*)

Send passkey response for a passkey entry request

Parameters:

response	Passkey entry response

Returns:

CyApiErr.OK if the method parameters are correct; otherwise, contains the error
Response instance is received from CyBleDeviceCallback.OnPasskeyEntryRequest callback

CyApiErr SendNumericComparisonResponse (CyNumericComparisonResponse *response*)

Send numeric comparison response for numeric comparison request

Parameters:

response	Numeric comparison response

Returns:

CyApiErr.OK if the method parameters are correct; otherwise, contains the error
Response instance is received from CyBleDeviceCallback.OnNumericComparisonRequest callback

CyApiErr SetDataLength (CyDataLengthInfo *info*)

Set the data packet length for the current connection This data is then negotiated with the remote device

Parameters:

info	Information necessary to set the data length for this connection

Returns:

CyApiErr.OK if the method parameters are correct; otherwise, contains the error

The status of the method is reported via the CyBleDeviceCallback.OnSetDataLength. The change in data length will be reported via the CyBleDeviceCallback.OnDataLengthChanged callback

CyApiErr UpdateConnectionParameter (CyConnectionParameters *settings*)

Update connection parameters of the remote device

Parameters:

settings	New connection parameter settings to be used

Returns:

CyApiErr.OK if the method parameters are correct; otherwise, contains the error

The status is reported via the CyBleDeviceCallback.OnUpdateConnectionParameter callback method If the connection parameter changed, then the change is reported via the CyBleDeviceCallback.OnConnectionParameterChanged callback method

CyApiErr SendConnectionParametersResponse (CyConnectionParametersResponse *response*)

Send response to connection parameter update request received from peer device

Parameters:

response	Response to be sent

Returns:

CyApiErr.OK if the method parameters are correct; otherwise, contains the error

---

Property Documentation

CyBleBdAddress Address `[get]`

Gets the address of the remote device

ushort Handle [get]

Gets the connection handle

ICyGattClient GattClient [get]

Gets the GATT client

ICyL2CapMgr L2CapMgr [get]

Gets the L2CAP manager

ICyBleDeviceAddressMgr

Device address manager Provides API to generate and set device address

Public Member Functions

- CyApiErr RegisterDeviceAddressMgrCallback (CyDeviceAddressMgrCallback cb)
  Register device address manager callback
- CyApiErr GetBdAddress (CyBleBdAddressType addressType)
  Get the current Bluetooth device address of the local device (dongle)
- CyApiErr GenerateBdAddressOfType (CyGenerateBdAddressInfo info)
  Generate Bluetooth device address
- CyApiErr SetBdAddress (CyBleBdAddress address)
  Set the Bluetooth device address of the local device (dongle)
- CyApiErr SetIdAddress (CyBleBdAddress address)
  Set the identity address of the local device (dongle)
- CyApiErr GetPeerResolvableAddress (CyBleBdAddress peerIdAddr)
  Get the resolvable private address of a peer device
- CyApiErr GetLocalResolvableAddress (CyBleBdAddress peerIdAddr)
  Get the resolvable private address of the local device for a peer device

---

Detailed Description

Device address manager Provides API to generate and set device address

Member Function Documentation

CyApiErr RegisterDeviceAddressMgrCallback (CyDeviceAddressMgrCallback *cb*)

Register device address manager callback

Parameters:

cb	Device address manager callback

Returns:

CyApiErr.OK if the callback was successfully registered; otherwise, contains the error
The status of device address manager APIs are reported via the CyDeviceAddressMgrCallback callback

CyApiErr GetBdAddress (CyBleBdAddressType *addressType*)

Get the current Bluetooth device address of the local device (dongle)

Parameters:

addressType	Type of address - public or random, to be returned

Returns:

CyApiErr.OK if the callback was successfully registered; otherwise, contains the error
The Bluetooth device address is reported via the registered CyDeviceAddressMgrCallback callback

CyApiErr GenerateBdAddressOfType (CyGenerateBdAddressInfo *info*)

Generate Bluetooth device address

Parameters:

info	Information required to generate Bluetooth device address

Returns:

CyApiErr.OK if the method parameters are correct; otherwise, contains the error
The generated Bluetooth device address is reported via the registered CyDeviceAddressMgrCallback callback

CyApiErr SetBdAddress (CyBleBdAddress *address*)

Set the Bluetooth device address of the local device (dongle)

Parameters:

address	Bluetooth device address

Returns:

CyApiErr.OK if the method parameters are correct; otherwise, contains the error
The status of this operation is reported via the registered CyDeviceAddressMgrCallback callback

CyApiErr SetIdAddress (CyBleBdAddress *address*)

Set the identity address of the local device (dongle)

Parameters:

address	Identity address

Returns:

CyApiErr.OK if the method parameters are correct; otherwise, contains the error
The ID address should be one of the bluetooth device address, set using SetBdAddress; otherwise, could result in unexpected behavior.
The status of this operation is reported via the registered CyDeviceAddressMgrCallback callback

CyApiErr GetPeerResolvableAddress (CyBleBdAddress *peerIdAddr*)

Get the resolvable private address of a peer device

Parameters:

peerIdAddr	Identity address of the peer device

Returns:

CyApiErr.OK if the method parameters are correct; otherwise, contains the error
Notes: (1) The resolvable private address is reported via the registered CyDeviceAddressMgrCallback callback
(2) This feature is available from BLE version 4.2

CyApiErr GetLocalResolvableAddress (CyBleBdAddress *peerIdAddr*)

Get the resolvable private address of the local device for a peer device

Parameters:

peerIdAddr	Identity address of the peer device

Returns:

CyApiErr.OK if the method parameters are correct; otherwise, contains the error

Notes: (1) The resolvable private address is reported via the registered CyDeviceAddressMgrCallback callback (2) This feature is available from BLE version 4.2

ICyBleDeviceList

Represents the whitelist, bond list and the resolving list of the local device (dongle)

Public Member Functions

- CyApiErr RegisterDeviceListCallback (CyDeviceListCallback cb)
  Register callback for all device list APIs
- CyApiErr AddDeviceToWhitelist (CyBleBdAddress address)
  Add a device to the whitelist
- CyApiErr RemoveDeviceFromWhitelist (CyWhitelistDevice device)
  Remove a device from the whitelist
- CyApiErr ClearWhitelist ()
  Remove all devices in the whitelist
- CyApiErr GetWhitelistDevices ()
  Get all devices in the whitelist
- CyApiErr GetBondListDevices ()
  Get devices in the bond list
- CyApiErr AddDeviceToResolvingList (CyAddToResolvingListInfo info)
  Add a device to the resolving list
- CyApiErr RemoveDeviceFromResolvingList (CyResolvingListDevice device)
  Remove a device from the resolving list
- CyApiErr ClearResolvingList ()
  Remove all devices from the resolving list
- CyApiErr GetResolvingListDevices ()
  Get devices in the resolving list

Detailed Description

Represents the whitelist, bond list and the resolving list of the local device (dongle)

Member Function Documentation

CyApiErr RegisterDeviceListCallback (CyDeviceListCallback *cb*)

Register callback for all device list APIs

Parameters:

cb	Device list callback

Returns:

CyApiErr.OK if the callback was successfully registered; otherwise, contains the error

CyApiErr AddDeviceToWhitelist (CyBleBdAddress *address*)

Add a device to the whitelist

Parameters:

address	Address of the device to be added to the whitelist

Returns:

CyApiErr.OK if the method parameters are correct; otherwise, contains the error

Result is reported via the CyDeviceListCallback.OnAddDeviceToWhitelist callback method

CyApiErr RemoveDeviceFromWhitelist (CyWhitelistDevice *device*)

Remove a device from the whitelist

Parameters:

device	Device to be removed from whitelist

Returns:

CyApiErr.OK if the method parameters are correct; otherwise, contains the error

Status is reported via the CyDeviceListCallback.OnRemoveDeviceFromWhitelist callback method

CyApiErr ClearWhitelist ()

Remove all devices in the whitelist

Returns:

CyApiErr.OK if the method parameters are correct; otherwise, contains the error
Status is reported via the CyDeviceListCallback.OnClearWhitelist callback method

CyApiErr GetWhitelistDevices ()

Get all devices in the whitelist

Returns:

CyApiErr.OK if the method parameters are correct; otherwise, contains the error
The whitelist devices are reported via the CyDeviceListCallback.OnGetWhitelistDevices callback method

CyApiErr GetBondListDevices ()

Get devices in the bond list

Returns:

CyApiErr.OK if the method parameters are correct; otherwise, contains the error
The bond list devices are reported via the CyDeviceListCallback.OnGetBondListDevices callback method

CyApiErr AddDeviceToResolvingList (CyAddToResolvingListInfo *info*)

Add a device to the resolving list

Parameters:

info	Information necessary to add a device to the resolving list

Returns:

CyApiErr.OK if the method parameters are correct; otherwise, contains the error
Status is reported via the CyDeviceListCallback.OnAddDeviceToResolvingList callback method

CyApiErr RemoveDeviceFromResolvingList (CyResolvingListDevice *device*)

Remove a device from the resolving list

Parameters:

device	Device to be removed from the resolving list

Returns:

CyApiErr.OK if the method parameters are correct; otherwise, contains the error
Status is reported via the CyDeviceListCallback.OnRemoveDeviceFromResolvingList callback method

CyApiErr ClearResolvingList ()

Remove all devices from the resolving list

Returns:

CyApiErr.OK if the method parameters are correct; otherwise, contains the error
Status is reported via the CyDeviceListCallback.OnClearResolvingList callback method

CyApiErr GetResolvingListDevices ()

Get devices in the resolving list

Returns:

CyApiErr.OK if the method parameters are correct; otherwise, contains the error
The resolving list devices are reported via the CyDeviceListCallback.OnGetResolvingListDevices callback method

---

ICyBleMgr

BLE Manager Provides APIs to scan, connect and to perform other GAP central operations

Public Member Functions

- CyApiErr RegisterBleMgrCallback (CyBleMgrCallback cb)
  Register the callback for BLE manager APIs
- CyApiErr StartScan (CyBleScanSettings settings, CyScanCallback cb)
  Start scan The dongle starts scanning for nearby BLE devices. The discovered devices are reported in the CyScanCallback callback

- CyApiErr StopScan ()
  Stop an ongoing scan
- CyApiErr Connect (CyConnectInfo info)
  Connect to a BLE device Establishes connection with a BLE device
- CyApiErr CancelConnection (CyBleBdAddress deviceAddress)
  Cancels an ongoing connect request with the given device
- CyApiErr Disconnect (ICyBleDevice device)
  Disconnect from a BLE device
- CyApiErr GetDeviceIoCapabilities ()
  Get the current IO capabilities of the local device (dongle)
- CyApiErr SetDeviceIoCapabilities (CyBleDeviceIoCapabilities ioCapability)
  Set the IO capabilities of the local device (dongle)
- CyApiErr RegisterPsm (CyRegisterPsmInfo info)
  Register a L2CAP PSM
- CyApiErr UnregisterPsm (CyRegisteredPsm psm)
  Unregister a previously registered L2CAP PSM
- CyApiErr GetRSSI ()
  Get RSSI of the last received packet
- CyApiErr SetTxPower (CyTxPowerInfo info)
  Set channel transmission power
- CyApiErr GetTxPower (CyChannelGroup channel)
  Get the current channel transmission power
- CyApiErr SetHostChannelClassification (CyChannelClassificationInfo info)
  Set host channel classification
- CyApiErr GetDefaultDataLength ()
  Get default data packet length
- CyApiErr SetSuggestedDataLength (CySetSuggestedDataLengthInfo info)
  Set the default data packet length
- CyApiErr ConvertDataLengthOctetToTime (CyConvertOctetToTimeInfo info)
  Utility method to convert data length octet to time
- CyApiErr SetResolvableAddressTimeout (CyResolvableAddressTimeoutInfo info)
  Set the length of time the controller uses a resolvable private address before a new address is generated
- CyApiErr SetAddressResolutionControl (CyAddressResolutionControlInfo info)
  Enable or disable resolvable address resolution in controller

Properties

- ICyBleDeviceList DeviceList `[get]`
  Gets the device list Device list includes whitelist, bond list and the resolving list
- ICyBleSecurityMgr SecurityMgr `[get]`
  Gets the security manager
- ICyBleDeviceAddressMgr DeviceAddressMgr `[get]`
  Gets the device address manager

Detailed Description

BLE Manager Provides APIs to scan, connect and to perform other GAP central operations

Member Function Documentation

CyApiErr RegisterBleMgrCallback (CyBleMgrCallback *cb*)

Register the callback for BLE manager APIs

Parameters:

cb	BLE manager callback

Returns:

CyApiErr.OK if the callback was successfully registered; otherwise, contains the error

CyApiErr StartScan (CyBleScanSettings *settings*, CyScanCallback *cb*)

Start scan The dongle starts scanning for nearby BLE devices. The discovered devices are reported in the CyScanCallback callback

Parameters:

settings	Scan settings
cb	Scan callback Scan results are reported via this callback

Returns:

CyApiErr.OK if the method parameters are correct; otherwise, contains the error

Scan status is reported via the CyScanCallback callback

CyApiErr StopScan ()

Stop an ongoing scan

Returns:

CyApiErr.OK if the method parameters are correct; otherwise, contains the error

Scan status is reported via the CyScanCallback callback registered in ICyBleMgr.StartScan

CyApiErr Connect (CyConnectInfo *info*)

Connect to a BLE device Establishes connection with a BLE device

Parameters:

info	Information necessary to establish a connection

Returns:

CyApiErr.OK if the method parameters are correct; otherwise, contains the error
The status of the operation is reported via the registered CyBleMgrCallback callback

CyApiErr CancelConnection (CyBleBdAddress *deviceAddress*)

Cancels an ongoing connect request with the given device

Parameters:

deviceAddress	Cancel connection with device

Returns:

CyApiErr.OK if the method parameters are correct; otherwise, contains the error
The status of the operation is reported via the registered CyBleMgrCallback callback

CyApiErr Disconnect (ICyBleDevice *device*)

Disconnect from a BLE device

Parameters:

device	Device to be disconnected

Returns:

CyApiErr.OK if the method parameters are correct; otherwise, contains the error
The status of the operation is reported via the registered CyBleMgrCallback callback

CyApiErr GetDeviceIoCapabilities ()

Get the current IO capabilities of the local device (dongle)

Returns:

CyApiErr.OK if the method parameters are correct; otherwise, contains the error

CyApiErr SetDeviceIoCapabilities (CyBleDeviceIoCapabilities *ioCapability*)

Set the IO capabilities of the local device (dongle)

Parameters:

ioCapability	IO capabilities to be set

Returns:

CyApiErr.OK if the method parameters are correct; otherwise, contains the error

CyApiErr RegisterPsm (CyRegisterPsmInfo *info*)

Register a L2CAP PSM

Parameters:

info	Information necessary to register a PSM

Returns:

CyApiErr.OK if the method parameters are correct; otherwise, contains the error

Registered PSM is used to create a L2CAP channel. ICyBleL2CapMgr provides API to create and manage L2CAP channel communication

CyApiErr UnregisterPsm (CyRegisteredPsm *psm*)

Unregister a previously registered L2CAP PSM

Parameters:

psm	PSM to be unregistered

Returns:

CyApiErr.OK if the method parameters are correct; otherwise, contains the error

CyApiErr GetRSSI ()

Get RSSI of the last received packet

Returns:

CyApiErr.OK if the method parameters are correct; otherwise, contains the error
The RSSI value is reported via the registered CyBleMgrCallback callback

CyApiErr SetTxPower (CyTxPowerInfo *info*)

Set channel transmission power

Parameters:

info	Information necessary to set the channel power

Returns:

CyApiErr.OK if the method parameters are correct; otherwise, contains the error

CyApiErr GetTxPower (CyChannelGroup *channel*)

Get the current channel transmission power

Parameters:

channel	Channel group

Returns:

CyApiErr.OK if the method parameters are correct; otherwise, contains the error

CyApiErr SetHostChannelClassification (CyChannelClassificationInfo *info*)

Set host channel classification

Parameters:

info	Information necessary to set the host channel classification

Returns:

CyApiErr.OK if the method parameters are correct; otherwise, contains the error

CyApiErr GetDefaultDataLength ()

Get default data packet length

Returns:

CyApiErr.OK if the method parameters are correct; otherwise, contains the error
This feature is available from BLE version 4.2

CyApiErr SetSuggestedDataLength (CySetSuggestedDataLengthInfo *info*)

Set the default data packet length

Parameters:

info	Information necessary to set the default data packet length

Returns:

CyApiErr.OK if the method parameters are correct; otherwise, contains the error
This feature is available from BLE version 4.2

CyApiErr ConvertDataLengthOctetToTime (CyConvertOctetToTimeInfo *info*)

Utility method to convert data length octet to time

Parameters:

info	Information required to convert

Returns:

CyApiErr.OK if the method parameters are correct; otherwise, contains the error
The converted time value is returned via the registered CyBleMgrCallback.OnConvertDataLengthOctetToTime callback Use the converted time value when setting the suggested data length using SetSuggestedDataLength API

CyApiErr SetResolvableAddressTimeout (CyResolvableAddressTimeoutInfo *info*)

Set the length of time the controller uses a resolvable private address before a new address is generated

Parameters:

info	Information necessary to set the resolvable address generation timeout

Returns:

CyApiErr.OK if the method parameters are correct; otherwise, contains the error
The status is reported via the CyBleMgrCallback.OnSetResolvableAddressTimeout callback method This feature is available from BLE version 4.2

CyApiErr SetAddressResolutionControl (CyAddressResolutionControlInfo *info*)

Enable or disable resolvable address resolution in controller

Parameters:

info	Information necessary to enable/disable resolvable address resolution

Returns:

CyApiErr.OK if the method parameters are correct; otherwise, contains the error
The status is reported via the CyBleMgrCallback.OnSetAddressResolutionControl callback method This feature is available from BLE version 4.2

Property Documentation

ICyBleDeviceList DeviceList [get]

Gets the device list Device list includes whitelist, bond list and the resolving list

ICyBleSecurityMgr SecurityMgr [get]

Gets the security manager

ICyBleDeviceAddressMgr DeviceAddressMgr [get]

Gets the device address manager

ICyBleSecurityMgr

Security Manager Provide APIs to manage authentication keys and OOB data to be used when pairing with a device

Public Member Functions

- CyApiErr RegisterSecurityMgrCallback (CySecurityMgrCallback cb)
  Register the security manager callback
- CyApiErr GetAuthenticationKeys ()
  Get the authentication keys of the local device (dongle)
- CyApiErr GenerateAuthenticationKeys (CyAuthenticationKeyFlags distributeKeys)
  Generate authentication keys Generates new set of LTK, IRK and CSRK
- CyApiErr SetAuthenticationKeys (CyAuthenticationKeys keys)
  Set authentication keys
- CyApiErr GetPeerDeviceAuthenticationKeys (CyGetPeerDeviceAuthenticationKeyInfo info)
  Get peer device authentication keys
- CyApiErr ResolvePeerDeviceAddress (CyResolvePeerDeviceInfo info)
  Resolve peer device address The peer device address needs to be resolved to ensure that the previously bonded data is used when connecting with a device
- CyApiErr GenerateLocalECDHKey ()
  Generates local ECDH key
- CyApiErr GenerateSecureConnectionOobData (CyGenerateSecureConnectionOobDataInfo info)
  Generate OOB data for secure connection

Detailed Description

Security Manager Provide APIs to manage authentication keys and OOB data to be used when pairing with a device

Member Function Documentation

CyApiErr RegisterSecurityMgrCallback (CySecurityMgrCallback *cb*)

Register the security manager callback

Parameters:

cb	Security manager callback

Returns:

CyApiErr.OK if the callback was successfully registered; otherwise, contains the error

CyApiErr GetAuthenticationKeys ()

Get the authentication keys of the local device (dongle)

Returns:

CyApiErr.OK
The local authentication keys are reported via the CySecurityMgrCallback.OnGetAuthenticationKeys callback

CyApiErr GenerateAuthenticationKeys (CyAuthenticationKeyFlags *distributeKeys*)

Generate authentication keys Generates new set of LTK, IRK and CSRK

Parameters:

distributeKeys	Keys to be distributed and requested

Returns:

CyApiErr.OK if the method parameters are correct; otherwise, contains the error
The generated authentication keys are reported via the CySecurityMgrCallback.OnGenerateAuthenticationKeys callback

CyApiErr SetAuthenticationKeys (CyAuthenticationKeys *keys*)

Set authentication keys

Parameters:

keys	Authentication keys

Returns:

CyApiErr.OK if the method parameters are correct; otherwise, contains the error
The status of this operation is reported via the CySecurityMgrCallback.OnSetAuthenticationKeys callback

CyApiErr GetPeerDeviceAuthenticationKeys (CyGetPeerDeviceAuthenticationKeyInfo *info*)

Get peer device authentication keys

Parameters:

info	Information necessary to get the peer device authentication keys

Returns:

CyApiErr.OK if the method parameters are correct; otherwise, contains the error

Peer device authentication keys are reported via the CySecurityMgrCallback.OnGetPeerDeviceAuthenticationKeys callback

CyApiErr ResolvePeerDeviceAddress (CyResolvePeerDeviceInfo *info*)

Resolve peer device address The peer device address needs to be resolved to ensure that the previously bonded data is used when connecting with a device

Parameters:

info	Information necessary to resolve a peer device address

Returns:

CyApiErr.OK if the method parameters are correct; otherwise, contains the error

Notes: (1) This method must be called before connecting to a device (2) This method can be ignored, if address resolution is enabled in the controller (Privacy 1.2). Refer to ICyBleMgr.SetAddressResolutionControl (3) The status of this operation is reported via the CySecurityMgrCallback.OnResolvePeerDeviceAddress callback

CyApiErr GenerateLocalECDHKey ()

Generates local ECDH key

Returns:

CyApiErr.OK

Notes: (1) The status of this operation is reported via the CySecurityMgrCallback.OnGenerateLocalECDHKey callback

(2) This feature is available from BLE version 4.2

CyApiErr GenerateSecureConnectionOobData (CyGenerateSecureConnectionOobDataInfo *info*)

Generate OOB data for secure connection

Parameters:

info	Information necessary to generate OOB data

Returns:

CyApiErr.OK if the method parameters are correct; otherwise, contains the error

Note: (1) The generated OOB data is reported via the registered CySecurityMgrCallback.OnGenerateSecureConnectionOobData callback

(2) The local ECDH key is used for secure connection OOB data generation. The ECDH key can be changed / generated by calling the GenerateLocalECDHKey method

(3) This feature is available from BLE version 4.2

ICyGattClient

This interface defines GATT client APIs to discover, read/write/notify GATT server

Public Member Functions

- CyApiErr RegisterCallback (CyGattClientCallback cb)
  Register GATT client callback
- CyApiErr ExchangeMtu (CyGattExchangeMtuInfo info)
  Exchange GATT MTU size with the peer device
- CyApiErr DiscoverAllServices ()
  Discover all services, characteristics and descriptors defined in the peer device GATT server
- CyApiErr DiscoverPrimaryServices (CyDiscoverPrimaryServiceCallback cb)
  Discover only primary services defined in the peer device GATT server
- CyApiErr DiscoverPrimaryServicesByUUID (CyDiscoverPrimaryServicesByUUIDInfo info, CyDiscoverPrimaryServiceCallback cb)
  Discover primary services by UUID
- CyApiErr FindIncludedServices (CyFindIncludedServicesInfo info, CyFindIncludedServicesCallback cb)
  Find included services
- CyApiErr DiscoverCharacteristics (CyDiscoverCharacteristicsInfo info, CyDiscoverCharacteristicsCallback cb)
  Discover only characteristics
- CyApiErr DiscoverCharacteristicsByUUID (CyDiscoverCharacteristicsByUUIDInfo info, CyDiscoverCharacteristicsCallback cb)
  Discover characteristics by UUID
- CyApiErr DiscoverDescriptors (CyDiscoverDescriptorsInfo info, CyDiscoverDescriptorsCallback cb)
  Discover descriptors
- CyApiErr ReadCharacteristic (CyGattReadInfo info)
  Read a characteristic value
- CyApiErr ReadCharacteristicByUUID (CyReadCharacteristicByUUIDInfo info)

Read a characteristic value by UUID

- CyApiErr ReadLongCharacteristic (CyGattReadInfo info)
  Read a long characteristic value
- CyApiErr ReadMultipleCharacteristic (CyReadMultipleCharacteristicInfo info)
  Read multiple characteristics
- CyApiErr WriteCharacteristicWithoutResponse (CyGattWriteInfo info)
  Write a characteristic value without response
- CyApiErr WriteCharacteristic (CyGattWriteInfo info)
  Write a characteristic value
- CyApiErr WriteLongCharacteristic (CyGattWriteInfo info)
  Write a long characteristic value
- CyApiErr ReliableWrite (CyReliableWriteInfo info)
  Reliable characteristic write
- CyApiErr SendWriteBufferFullResponse (CyWriteBufferFullResponse response)
  Sends the response to write buffer full status reported by the peer device
- CyApiErr SignedCharacteristicWriteWithoutResponse (CyGattWriteInfo info)
  Signed write characteristic value without response
- CyApiErr ReadDescriptor (CyGattReadInfo info)
  Read a descriptor value
- CyApiErr ReadLongDescriptor (CyGattReadInfo info)
  Read a long descriptor value
- CyApiErr WriteDescriptor (CyGattWriteInfo info)
  Write a descriptor value
- CyApiErr WriteLongDescriptor (CyGattWriteInfo info)
  Write a long descriptor value
- CyApiErr GattStop ()
  Stop an on-going long GATT operation

Detailed Description

This interface defines GATT client APIs to discover, read/write/notify GATT server

Member Function Documentation

CyApiErr RegisterCallback (CyGattClientCallback *cb*)

Register GATT client callback

Parameters:

cb	GATT client callback

Returns:

CyApiErr.OK if the callback was successfully registered; otherwise, contains the error

CyApiErr ExchangeMtu (CyGattExchangeMtuInfo *info*)

Exchange GATT MTU size with the peer device

Parameters:

info	Information necessary to exchange GATT MTU

Returns:

CyApiErr.OK if the method parameters are correct; otherwise, contains the error

The negotiated GATT MTU size is reported via the CyGattClientCallback.OnGattMtuExchanged callback

CyApiErr DiscoverAllServices ()

Discover all services, characteristics and descriptors defined in the peer device GATT server

Returns:

CyApiErr.OK if the method parameters are correct; otherwise, contains the error

The discovered services are reported via the ICyGattClientCallback.OnServiceDiscovered

CyApiErr DiscoverPrimaryServices (CyDiscoverPrimaryServiceCallback *cb*)

Discover only primary services defined in the peer device GATT server

Parameters:

cb	Callback to report the discovered primary services

Returns:

CyApiErr.OK if the method parameters are correct; otherwise, contains the error

CyApiErr DiscoverPrimaryServicesByUUID (CyDiscoverPrimaryServicesByUUIDInfo *info*, CyDiscoverPrimaryServiceCallback *cb*)

Discover primary services by UUID

Parameters:

info	Information necessary to discover primary services of a specific UUID
cb	Callback to report the discovered primary services

Returns:

CyApiErr.OK if the method parameters are correct; otherwise, contains the error

CyApiErr FindIncludedServices (CyFindIncludedServicesInfo *info*, CyFindIncludedServicesCallback *cb*)

Find included services

Parameters:

info	Information necessary to discover included services
cb	Callback to report the discovered included services

Returns:

CyApiErr.OK if the method parameters are correct; otherwise, contains the error

CyApiErr DiscoverCharacteristics (CyDiscoverCharacteristicsInfo *info*, CyDiscoverCharacteristicsCallback *cb*)

Discover only characteristics

Parameters:

info	Information necessary to discover characteristics
cb	Callback to report the discovered characteristics

Returns:

CyApiErr.OK if the method parameters are correct; otherwise, contains the error

CyApiErr DiscoverCharacteristicsByUUID (CyDiscoverCharacteristicsByUUIDInfo *info*, CyDiscoverCharacteristicsCallback *cb*)

Discover characteristics by UUID

Parameters:

info	Information necessary to discover characteristic of a specific UUID
cb	Callback to report the discovered characteristics

Returns:

CyApiErr.OK if the method parameters are correct; otherwise, contains the error

CyApiErr DiscoverDescriptors (CyDiscoverDescriptorsInfo *info*, CyDiscoverDescriptorsCallback *cb*)

Discover descriptors

Parameters:

info	Information necessary to discover descriptors
cb	Callback to report the discovered descriptors

Returns:

CyApiErr.OK if the method parameters are correct; otherwise, contains the error

CyApiErr ReadCharacteristic (CyGattReadInfo *info*)

Read a characteristic value

Parameters:

info	Information necessary to read a characteristic value

Returns:

CyApiErr.OK if the method parameters are correct; otherwise, contains the error

The read value is reported via the CyGattClientCallback.OnCharacteristicRead callback method

CyApiErr ReadCharacteristicByUUID (CyReadCharacteristicByUUIDInfo *info*)

Read a characteristic value by UUID

Parameters:

info	Information necessary to read value of characteristics with a specific UUID

Returns:

CyApiErr.OK if the method parameters are correct; otherwise, contains the error

The read value is reported via the CyGattClientCallback.OnCharacteristicReadByUUID callback method

CyApiErr ReadLongCharacteristic (CyGattReadInfo *info*)

Read a long characteristic value

Parameters:

info	Information necessary to read a long characteristic value

Returns:

CyApiErr.OK if the method parameters are correct; otherwise, contains the error

The read value is reported via the CyGattClientCallback.OnCharacteristicRead callback method

CyApiErr ReadMultipleCharacteristic (CyReadMultipleCharacteristicInfo *info*)

Read multiple characteristics

Parameters:

info	Information necessary to read multiple characteristics

Returns:

CyApiErr.OK if the method parameters are correct; otherwise, contains the error

The read value is reported via the CyGattClientCallback.OnReadMultipleCharacteristics callback method

CyApiErr WriteCharacteristicWithoutResponse (CyGattWriteInfo *info*)

Write a characteristic value without response

Parameters:

info	Information necessary to write a characteristic without response

Returns:

CyApiErr.OK if the method parameters are correct; otherwise, contains the error
The status is reported via the CyGattClientCallback.OnCharacteristicWrite callback method

CyApiErr WriteCharacteristic (CyGattWriteInfo *info*)

Write a characteristic value

Parameters:

info	Information necessary to write a characteristic value

Returns:

CyApiErr.OK if the method parameters are correct; otherwise, contains the error
The status is reported via the CyGattClientCallback.OnCharacteristicWrite callback method

CyApiErr WriteLongCharacteristic (CyGattWriteInfo *info*)

Write a long characteristic value

Parameters:

info	Information necessary to write a long characteristic value

Returns:

CyApiErr.OK if the method parameters are correct; otherwise, contains the error
The status is reported via the CyGattClientCallback.OnCharacteristicWrite callback method. If the peer device reports buffer full error, CyGattClientCallback.OnWriteBufferFull callback method will be invoked

CyApiErr ReliableWrite (CyReliableWriteInfo *info*)

Reliable characteristic write

Parameters:

info	Information necessary to initiate a reliable characteristic write

Returns:

CyApiErr.OK if the method parameters are correct; otherwise, contains the error

The status is reported via the CyGattClientCallback.OnReliableWriteCompleted callback method If the peer device reports buffer full error, CyGattClientCallback.OnWriteBufferFull callback method will be invoked

CyApiErr SendWriteBufferFullResponse (CyWriteBufferFullResponse *response*)

Sends the response to write buffer full status reported by the peer device

Parameters:

response	Response to be sent

Returns:

CyApiErr.OK if the method parameters are correct; otherwise, contains the error

CyApiErr SignedCharacteristicWriteWithoutResponse (CyGattWriteInfo *info*)

Signed write characteristic value without response

Parameters:

info	Information necessary to signed write characteristic value without response

Returns:

CyApiErr.OK if the method parameters are correct; otherwise, contains the error

The status is reported via the CyGattClientCallback.OnCharacteristicWrite callback method

CyApiErr ReadDescriptor (CyGattReadInfo *info*)

Read a descriptor value

Parameters:

info	Information necessary to read a descriptor value

Returns:

CyApiErr.OK if the method parameters are correct; otherwise, contains the error
The read value is reported via the CyGattClientCallback.OnDescriptorRead callback method

CyApiErr ReadLongDescriptor (CyGattReadInfo *info*)

Read a long descriptor value

Parameters:

info	Information necessary to read a long descriptor value

Returns:

CyApiErr.OK if the method parameters are correct; otherwise, contains the error
The read value is reported via the CyGattClientCallback.OnDescriptorRead callback method

CyApiErr WriteDescriptor (CyGattWriteInfo *info*)

Write a descriptor value

Parameters:

info	Information necessary to write a descriptor value

Returns:

CyApiErr.OK if the method parameters are correct; otherwise, contains the error
The status is reported via the CyGattClientCallback.OnDescriptorWrite callback method

CyApiErr WriteLongDescriptor (CyGattWriteInfo *info*)

Write a long descriptor value

Parameters:

info	Information necessary to write a long descriptor value

Returns:

CyApiErr.OK if the method parameters are correct; otherwise, contains the error
The status is reported via the CyGattClientCallback.OnDescriptorWrite callback method. If the peer device reports buffer full error, CyGattClientCallback.OnWriteBufferFull callback method will be invoked

CyApiErr GattStop ()

Stop an on-going long GATT operation

Returns:

CyApiErr.OK if the method parameters are correct; otherwise, contains the error
The status is reported via the CyGattClientCallback.OnGattStop callback method

---

ICyL2CapChannel

Represents an L2CAP channel

Properties

- ushort ChannelID `[get]`
  Gets the channel ID
- ushort LocalPSM `[get]`
  Gets the local PSM
- ushort MTU `[get]`
  Gets the channel local MTU
- ushort MPS `[get]`
  Gets the channel local MPS
- ushort InitialCredit `[get]`
  Gets the initial credit of the channel
- ushort RemoteMTU `[get]`
  Gets the remote MTU
- ushort RemoteMPS `[get]`
  Gets the remote MPS
- ushort RemoteInitialCredit `[get]`
  Gets the initial credit sent by the remote device

Detailed Description

Represents an L2CAP channel

Property Documentation

ushort ChannelID [get]

Gets the channel ID

ushort LocalPSM [get]

Gets the local PSM

ushort MTU [get]

Gets the channel local MTU

ushort MPS [get]

Gets the channel local MPS

ushort InitialCredit [get]

Gets the initial credit of the channel

ushort RemoteMTU [get]

Gets the remote MTU

ushort RemoteMPS [get]

Gets the remote MPS

ushort RemoteInitialCredit [get]

Gets the initial credit sent by the remote device

ICyL2CapMgr

L2CAP Manager. Provides APIs to manage L2CAP channel creation and removal. Also, provides APIs to send data and credits.

Public Member Functions

- CyApiErr RegisterL2CapCallback (CyL2CapMgrCallback cb)
  Register L2CAP manager callback
- CyApiErr EstablishChannel (CyEstablishL2CapChannelInfo info)
  Establish an L2CAP channel with the peer device
- CyApiErr DisconnectChannel (ICyL2CapChannel channel)
  Disconnect an existing L2CAP channel
- CyApiErr RespondToChannelRequest (CyL2CapConnectionResponseInfo response)
  Send a response to an L2CAP channel establishment request from peer device
- CyApiErr SendData (CyL2CapSendDataInfo info)
  Send data over an L2CAP channel
- CyApiErr SendCredits (CyL2CapSendCreditsInfo info)
  Send credits to a peer device

Detailed Description

L2CAP Manager. Provides APIs to manage L2CAP channel creation and removal. Also, provides APIs to send data and credits.

Member Function Documentation

CyApiErr RegisterL2CapCallback (CyL2CapMgrCallback *cb*)

Register L2CAP manager callback

Parameters:

cb	L2CAP callback

Returns:

CyApiErr.OK if the callback was successfully registered; otherwise, contains the error

CyApiErr EstablishChannel (CyEstablishL2CapChannelInfo *info*)

Establish an L2CAP channel with the peer device

Parameters:

info	Information necessary to create a new channel

Returns:

CyApiErr.OK if the method parameters are correct; otherwise, contains the error
The L2CAP channel is reported via the CyL2CapMgrCallback.OnChannelEstablished callback method

CyApiErr DisconnectChannel (ICyL2CapChannel *channel*)

Disconnect an existing L2CAP channel

Parameters:

channel	L2CAP channel to be disconnected

Returns:

CyApiErr.OK if the method parameters are correct; otherwise, contains the error
Status is reported via the CyL2CapMgrCallback.OnDisconnectChannel callback method

CyApiErr RespondToChannelRequest (CyL2CapConnectionResponseInfo *response*)

Send a response to an L2CAP channel establishment request from peer device

Parameters:

response	Response to the request

Returns:

CyApiErr.OK if the method parameters are correct; otherwise, contains the error
Response instance is received from the CyL2CapMgrCallback.OnChannelConnectionIndication callback method

CyApiErr SendData (CyL2CapSendDataInfo *info*)

Send data over an L2CAP channel

Parameters:

info	Information necessary to send data

Returns:

CyApiErr.OK if the method parameters are correct; otherwise, contains the error
Status is reported via the CyL2CapMgrCallback.OnSendData callback method

CyApiErr SendCredits (CyL2CapSendCreditsInfo *info*)

Send credits to a peer device

Parameters:

info	Information necessary to send credits

Returns:

CyApiErr.OK if the method parameters are correct; otherwise, contains the error
Status is reported via the CyL2CapMgrCallback.OnSendCredits callback method

---

ICySmartDongleCommunicator

Provides APIs to communicate with the CySmart dongle
Inherits IDisposable.

Properties

- CyDongleID DeviceID `[get]`
  Gets the device ID of the dongle
- Version BleStackVersion `[get]`
  Gets the BLE stack version of the dongle
- Version FirmwareVersion `[get]`
  Gets the dongle firmware version
- Version HardwareVersion `[get]`
  Gets the dongle hardware version
- ICyBleMgr BleMgr `[get]`
  Gets the BLE manager

Detailed Description

Provides APIs to communicate with the CySmart dongle

Property Documentation

CyDongleID DeviceID [get]

Gets the device ID of the dongle

Version BleStackVersion [get]

Gets the BLE stack version of the dongle

Version FirmwareVersion [get]

Gets the dongle firmware version

Version HardwareVersion [get]

Gets the dongle hardware version

ICyBleMgr BleMgr [get]

Gets the BLE manager

Revision History

Document Revision History

Document Title: CySmart™ API Reference Guide Document Number: 002-11435 Rev *A			
Revision	**Issue Date**	**Origin of Change**	**Description of Change**
**	03/04/2016	BAAM	Initial version of CySmart API reference guide
*A	03/22/2017	BAAM	Updated Cypress logo and copyright notice