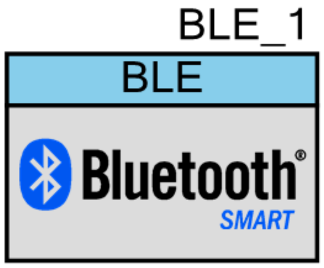
Bluetooth Low Energy (BLE)

1.0

Features



- Bluetooth v4.1 compliant protocol stack
- Generic Access Profile (GAP) Features
  - Broadcaster, Observer, Peripheral and Central roles
  - Supports role reversal between Peripheral and Central
  - User-defined advertising data
  - Bonding support for up to four devices
  - Security modes 1 and 2
- Generic Attribute Profile (GATT) Features
  - GATT Client and Server
  - 32-bit UUIDs
- Special Interest Group (SIG) adopted GATT-based Profiles (15) and Services (20), and quick prototype of new profile design through intuitive GUI Custom Profile development
- Security Manager features
  - Pairing methods: Just works, Passkey Entry, Out of Band
  - Authenticated man-in-the-middle (MITM) protection and data signing
- Logical Link Adaption Protocol (L2CAP) Connection Oriented Channel
- Link Layer (LL) Features
  - Master and Slave role
  - 128-bit AES encryption
  - Low Duty Cycle Advertising
  - LE Ping

General Description

The Bluetooth Low Energy (BLE) Component provides a comprehensive GUI-based configuration window to quickly design applications requiring BLE connectivity. The Component incorporates a Bluetooth Core Specification v4.1 compliant protocol stack and provides API functions to enable user applications to interface with the underlying hardware via the stack.

SIG adopted Profiles and Services

The BLE Component supports SIG-adopted GATT-based Profiles (15) and Services (20). Each of these can be configured for either a GATT Client or GATT Server. The Component can support several Profiles at a time by adding the required Services of a Profile to a base Profile. For example, you can select HID as a base Profile. Then to add a Find Me Profile, and add the Immediate Alert Service to the HID Profile.The Component generates all the necessary code for a particular Profile/Service operation, as configured in the GUI.

The following table lists the supported Profiles and Services.

Acronym	Profile	Version
ANP	Alert Notification Profile	1.0
ANS	Alert Notification Service	1.0
BAS	Battery Service	1.0
BLP	Blood Pressure Profile	1.0
BLS	Blood Pressure Service	1.0
CPP	Cycling Power Profile	1.0
CPS	Cycling Power Service	1.0
CSCP	Cycling Speed and Cadence Profile	1.0
CSCS	Cycling Speed and Cadence Service	1.0
CTS	Current Time Service	1.0
DIS	Device Information Service	1.1
FMP	Find Me Profile	1.0
GLP	Glucose Profile	1.0
GLS	Glucose Service	1.0
HOGP	HID over GATT Profile	1.0
HIDS	Human Interface Device Service	1.0
HTP	Health Thermometer Profile	1.0
HTS	Health Thermometer Service	1.0
HRP	Heart Rate Profile	1.0
HRS	Heart Rate Service	1.0
IAS	Immediate Alert Service	1.0

Acronym	Profile	Version
LLS	Link Loss Service	1.0
LNP	Location and Navigation Profile	1.0
LNS	Location and Navigation Service	1.0
NDCS	Next DST Change Service	1.0
PASP	Phone Alert Status Profile	1.0
PASS	Phone Alert Status Service	1.0
PXP	Proximity Profile	1.0
RSCP	Running Speed and Cadence Profile	1.0
RSCS	Running Speed and Cadence Service	1.0
RTUS	Reference Time Update Service	1.0
ScPP	Scan Parameters Profile	1.0
ScPS	Scan Parameters Service	1.0
TIP	Time Profile	1.0
TPS	Tx Power Service	1.0

Custom Profiles

You can create custom Profiles that use existing Services, and you can create custom Services with custom Characteristics and Descriptors. There are no restrictions for GAP roles for a custom Profile. Custom Services cannot be used in stand-alone mode; they need to be used in a Profile. For example, the Device Information Service is used in the Heart Rate Profile. It can be used in a custom Profile, or it can be added to any of existing Profiles.

Comprehensive APIs

The BLE Component provides application-level APIs to design solutions without requiring manual stack level configuration. The BLE Component API documentation is also provided in a separate HTML-based file that can be opened by right-clicking on the Component and selecting **Open API documentation**.

Debug Support

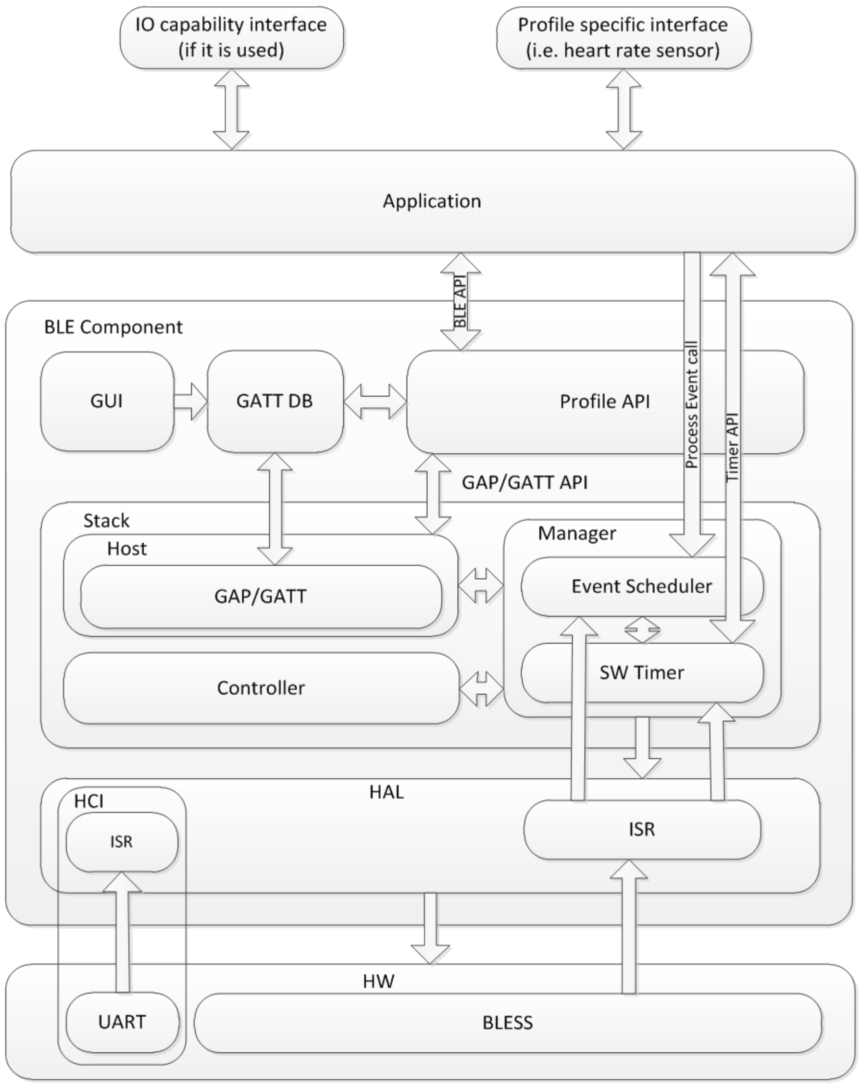
For testing and debugging, the Component can be configured to HCI mode through a Component embedded UART. For over-the-air verification, Cypress CySmart verification tool can be used for generic Bluetooth host stack emulation. To launch this tool, right click on the Component to bring up the context menu, and choose to deploy the CySmart tool.

When to use the BLE Component

BLE is used in very low power Wireless Personal Area Network (WPAN) and Internet of Things (IoT) solutions aimed for low-cost mobile battery operated devices that can quickly connect and form simple links. Target applications include HID, remote controls, sports and fitness monitors, portable medical devices and smart phone accessories, among many others that are being added to a long list of BLE supporting solutions.

BLE Component Architecture

The BLE Component consists of the BLE Stack, BLE Profile, BLE Component Hardware Abstraction Layer (HAL), and the Link Layer. The following figure shows a high-level architecture of the BLE Component, illustrating the relationship between each of the layers and the route in which the application interacts with the Component. Note that the application is informed of the BLE events through the use of callback functions. You may build your state machine using these. Refer to the Callback Functions section for more details.

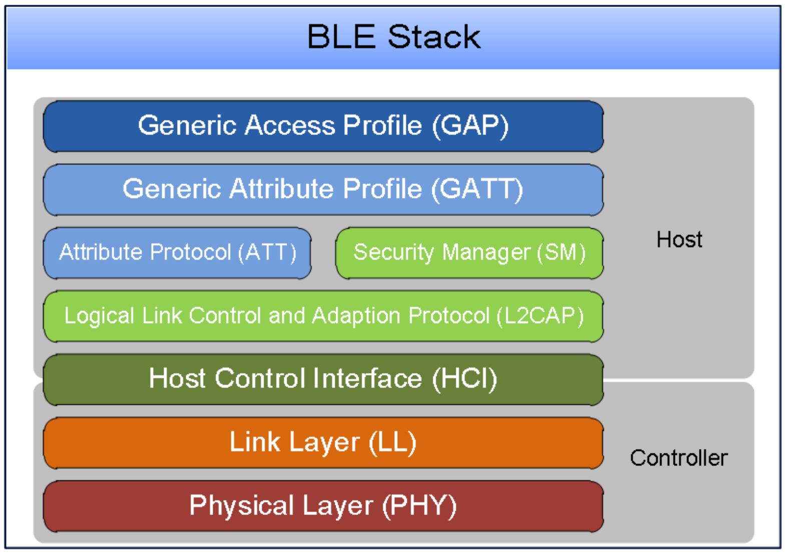


The following sub-sections give an overview of each of these layers.

BLE Stack

The BLE stack implements the core BLE functionality as defined in the Bluetooth Core Specification 4.1. The stack is included as a precompiled library and it is embedded inside the BLE Component.

The BLE stack implements all the mandatory and optional features of Low Energy Single Mode compliant to Bluetooth Core Specification 4.1. The BLE Stack implements a layered architecture of the BLE protocol stack as shown in the following figure.


BLE Stack
Generic Access Profile (GAP)
Generic Attribute Profile (GATT)
Attribute Protocol (ATT)
Security Manager (SM)
Host
Logical Link Control and Adaption Protocol (L2CAP)
Host Control Interface (HCI)
Link Layer (LL)
Controller
Physical Layer (PHY)

Generic Access Profile (GAP)

The Generic Access Profile defines the generic procedures related to discovery of Bluetooth devices and link management aspects of connecting to Bluetooth devices. In addition, this profile includes common format requirements for parameters accessible on the user interface level.

The Generic Access Profile defines the following roles when operating over the LE physical channel:

- **Broadcaster role:** A device operating in the Broadcaster role can send advertising events. It is referred to as a Broadcaster. It has a transmitter and may have a receiver.
- **Observer role:** A device operating in the Observer role is a device that receives advertising events. It is referred to as an Observer. It has a receiver and may have a transmitter.
- **Peripheral role:** A device that accepts the establishment of an LE physical link using any of the connection establishment procedures is termed to be in a "Peripheral role." A device operating in the Peripheral role will be in the "Slave role" in the Link Layer

Connection State. A device operating in the Peripheral role is referred to as a Peripheral. A Peripheral has both a transmitter and a receiver.

- **Central role:** A device that supports the Central role initiates the establishment of a physical connection. A device operating in the "Central role" will be in the "Master role" in the Link Layer Connection. A device operating in the Central role is referred to as a Central. A Central has a transmitter and a receiver.

Generic Attribute Profile (GATT)

The Generic Attribute Profile defines a generic service framework using the ATT protocol layer. This framework defines the procedures and formats of services and their Characteristics. It defines the procedures for Service, Characteristic, and Descriptor discovery, reading, writing, notifying, and indicating Characteristics, as well as configuring the broadcast of Characteristics.

GATT Roles

- GATT Client: This is the device that wants data. It initiates commands and requests towards the GATT Server. It can receive responses, indications, and notifications data sent by the GATT Server.
- GATT Server: This is the device that has the data and accepts incoming commands and requests from the GATT Client and sends responses, indications, and notifications to a GATT Client.

The BLE Stack supports both roles simultaneously for a custom profile use case.

Attribute Protocol (ATT)

The Attribute Protocol layer defines a Client/Server architecture above the BLE logical transport channel. The attribute protocol allows a device referred to as the GATT Server to expose a set of attributes and their associated values to a peer device referred to as the GATT Client. These attributes exposed by the GATT Server can be discovered, read, and written by a GATT Client, and can be indicated and notified by the GATT Server. All the transactions on attributes are atomic.

Security Manager Protocol (SMP)

Security Manager Protocol defines the procedures and behavior to manage pairing, authentication, and encryption between the devices. These include:

- Encryption and Authentication
- Pairing and Bonding
  - Pass Key and Out of band bonding
- Key Generation for a device identity resolution, data signing and encryption

- Pairing method selection based on the IO capability of the GAP central and GAP peripheral device

Logical Link Control Adaptation Protocol (L2CAP)

L2CAP provides a connectionless data channel. LE L2CAP provides the following features:

- Channel multiplexing, which manages three fixed channels. Two channels are dedicated for higher protocol layers like ATT, SMP. One channel is used for the LE-L2CAP protocol signaling channel for its own use.
- Segmentation and reassembly of packets whose size is up to the BLE Controller managed maximum packet size.
- Connection-oriented channel over a specific application registered using the PSM (protocol service multiplexer) channel. It implements credit-based flow control between two LE L2CAP entities. This feature can be used for BLE applications that require transferring large chunks of data.

Host Controller Interface (HCI)

The HCI layer implements a command, event, and data interface to allow link layer access from upper layers such as GAP, L2CAP, and SMP.

Link Layer (LL)

The LL protocol manages the physical BLE connections between devices. It supports all LL states such as Advertising, Scanning, Initiating, and Connecting (Master and Slave). It implements all the key link control procedures such as LE Encryption, LE Connection Update, LE Channel Update, and LE Ping. The Link Layer is a hardware-firmware co-implementation, where the key time critical LL functions are implemented in the LL hardware. The LL firmware maintains and controls the key LL procedure state machines. It supports all the BLE chip specific low power modes.

The BLE Stack is a pre-compiled library in the BLE Component solution. The appropriate configuration of the BLE Stack library is linked during a build process based on application. The BLE Stack libraries are ARM Embedded Application Binary Interface (eabi) compliant and they are compiled using ARM compiler version 5.03.

The following table shows the mapping between the BLE Stack library to the user-configured Profile Role in Profile Mode or HCI Mode. Refer to the Generic Tab section for selection of stack configuration.

BLE Component Configuration	GAP Role	BLE Stack Library
BLE Profile	Central + Peripheral	CyBLEStack_BLE_SOC_CENTRAL_PERIPHERAL.a
BLE Profile	Central	CyBLEStack_BLE_SOC_CENTRAL.a

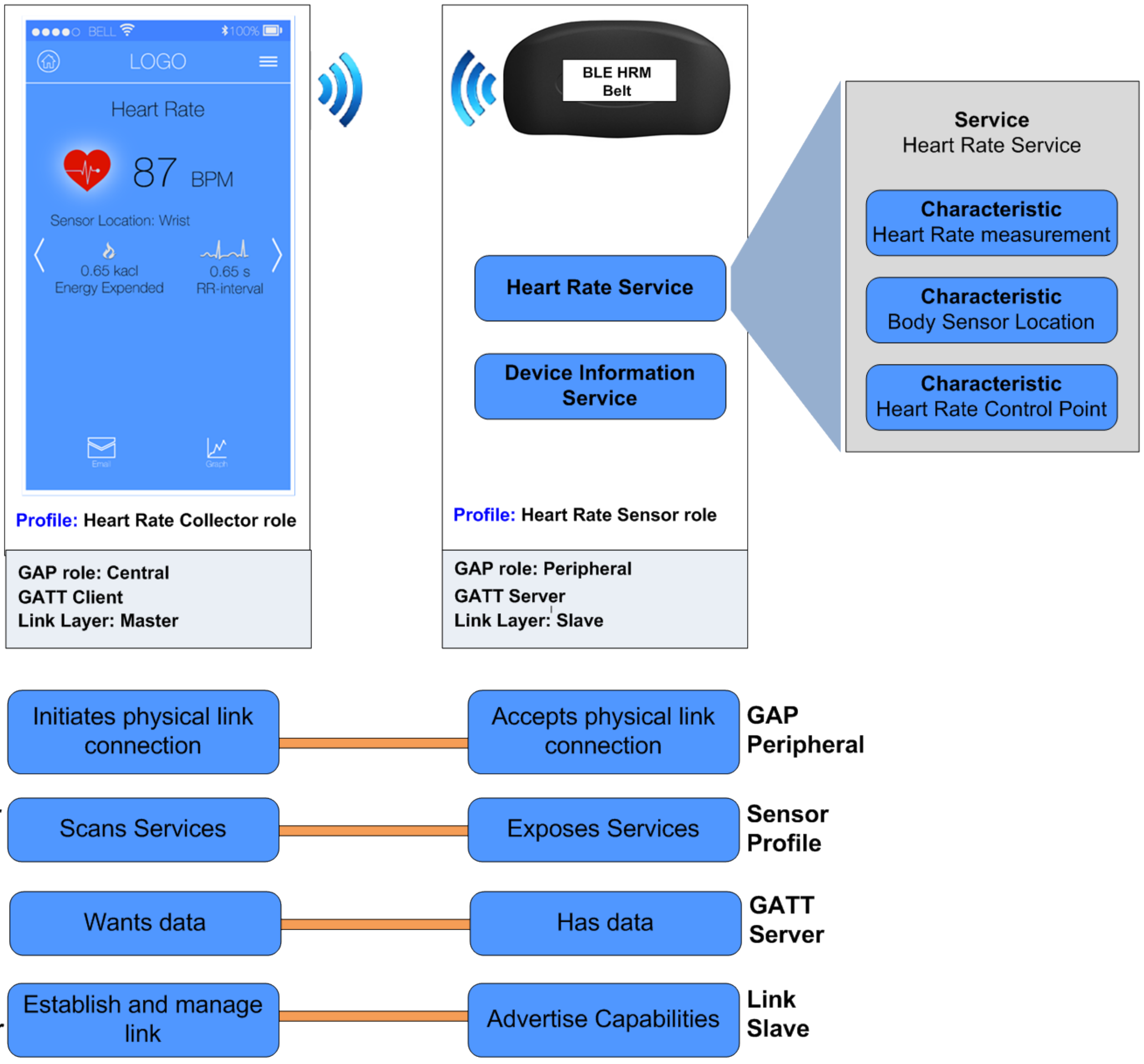
BLE Component Configuration	GAP Role	BLE Stack Library
BLE Profile	Peripheral	CyBLEStack_BLE_SOC_PERIPHERAL.a
HCI Mode	N/A	CyBLEStack_HCI_MODE_CENTRAL_PERIPHERAL.a

Profile Layer

In BLE, data is organized into concepts called Profiles, Services, and Characteristics.

- A **Profile** describes how devices connect to each other to find and use Services. It is a definition used by Bluetooth devices to describe the type of application and the general expected behavior of that device. See the Profile parameter for how to configure to the BLE Component.
- A **Service** is a collection of data entities called Characteristics. A Service is used to define a certain function in a Profile. A Service may also define its relationship to other Services. A Service is assigned a Universally Unique Identifier (UUID). This is 16 bits for SIG adopted Services and 128 bits for custom Services. See the Toolbar section for information about adding Services to a Profile.
- A **Characteristic** contains a Value and the Descriptor that describes a Characteristic Value. It is an attribute type for a specific piece of information within a Service. Like a Service, each Characteristic is designated with a UUID; 16 bits for SIG adopted Characteristics and 128 bits for custom Characteristics. See the Toolbar section for information about adding Characteristics and Descriptors.

The following diagram shows the relationship between Profiles, Services, and Characteristics in a sample BLE heart rate monitor application using a Heart Rate Profile.



The Heart Rate Profile contains a Heart Rate Service and a Device Information Service. Within the Heart Rate Service, there are three Characteristics, each containing different information. The device in the diagram is configured as a Sensor role, meaning that in the context of the Heart Rate Profile, the device is a GAP Peripheral and a GATT Server. These concepts are explained in the BLE Stack description.

The Profile layer is generated by PSoC Creator using the parameter configurations specified in the GUI. The Profile implements the Profile specific attribute database and APIs required for the application. You can choose to configure the standard SIG adopted Profile and generate a design or define a Custom Profile required by an application. The GUI also allows import/export of a Profile design in XML format for Profile design reuse.

Hardware Abstraction Layer (HAL)

The HAL implements the interface between the BLE stack and the underlying hardware. This layer is meant for the stack only and is not advisable to modify it.

Input/Output Connections

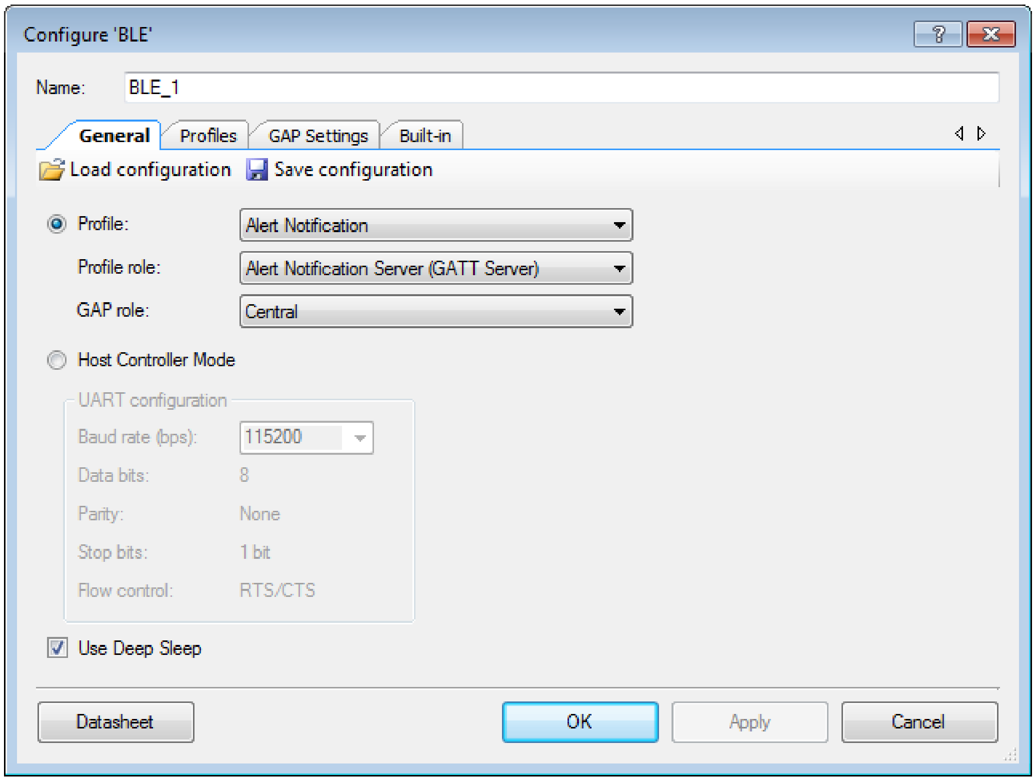
This Component does not require hardware terminals. All the hardware connections are direct and dedicated to specific pins in the underlying BLE hardware.

Component Parameters

Drag a BLE Component onto your design and double-click it to open the Configure dialog. This dialog has the following tabs with different parameters.

General Tab

The **General** tab allows general configuration of the BLE Component. It is possible to import and export the customizer configuration in xml format.



Name: BLE_1

(●) Profile: Alert Notification

Profile role: Alert Notification Server (GATT Server)

GAP role: Central

( ) Host Controller Mode

UART configuration

Baud rate (bps): 115200

Data bits: 8

Parity: None

Stop bits: 1 bit

Flow control: RTS/CTS

[✓] Use Deep Sleep

Load Configuration/Save Configuration

Use the **Load Configuration** button to load the previously saved xml Component configuration; use the Save Configuration button to save the current configuration for use in other designs.

Profile

The **Profile** mode allows you to choose the target Profile from a list of supported Profiles. See Profile, Service, and Characteristic. When a mode is chosen, the **Profile role** and **GAP role** parameters are enabled. The following Profiles are available for selection.

Alert Notification

This Profile enables a GATT Client device to receive different types of alerts and event information, as well as information on the count of new alerts and unread items, which exist in the GATT Server device.

- **Alert Notification Server** Profile role – Specified as a GATT Server. Requires the following Service: **Alert Notification Service**.
  - Central GAP role
  - Peripheral and Central GAP role
- **Alert Notification Client** Profile role – Specified as a GATT Client.
  - Peripheral GAP role
  - Peripheral and Central GAP role

Blood Pressure

This Profile enables a device to connect and interact with a Blood Pressure Sensor device for use in consumer and professional health care applications.

- **Blood Pressure Sensor** Profile role – Specified as a GATT Server. Requires the following Services: **Blood Pressure Service**, **Device Information Service**.
  - Peripheral GAP role
- **Blood Pressure Collector** Profile role – Specified as a GATT Client. Requires support of the following Services: **Blood Pressure Service**. Support of **Device Information Service** is optional.
  - Central GAP role

Cycling Power

This Profile enables a Collector device to connect and interact with a Cycling Power Sensor for use in sports and fitness applications.

- **Cycling Power Sensor** Profile role – Specified as a GATT Server. Requires the following Service: **Cycling Power Service**. Optionally may include **Device Information Service** and **Battery Service**.
    - Peripheral GAP role
- **Cycling Power Broadcaster** Profile role. Requires the following Service: **Cycling Power Service**.
    - Broadcaster GAP role
- **Cycling Power Observer** Profile role. Can only talk to a device with the **Cycling Power Broadcaster** role. Requires support of the following Service: **Cycling Power Service**.
    - Observer GAP role
- **Collector** Profile role – Specified as a GATT Client. Requires support of the following Service: **Cycling Power Service**. Support of **Device Information Service** and **Battery Service** is optional.
    - Central GAP role

Cycling Speed and Cadence

This Profile enables a Collector device to connect and interact with a Cycling Speed and Cadence Sensor for use in sports and fitness applications.

- **Cycling Speed and Cadence Sensor** Profile role – Specified as a GATT Server. Requires the following Service: **Cycling Speed and Cadence Service**. Optionally may include **Device Information Service**.
    - Peripheral GAP role
- **Collector** Profile role – Specified as a GATT Client. Requires support of the following Service: **Cycling Speed and Cadence Service**. Support of **Device Information Service** is optional.
    - Central GAP role

Find Me

The Find Me Profile defines the behavior when a button is pressed on one device to cause an alerting signal on a peer device.

- **Find Me Target** Profile role – Specified as a GATT Server. Requires the following Service: **Immediate Alert Service**.

  - Peripheral GAP role
  - Central GAP role
  - Peripheral and Central GAP roles

- **Find Me Locator** Profile role – Specified as a GATT Client. Requires support of the following Service: **Immediate Alert Service**.
  - Peripheral GAP role
  - Central GAP role
  - Peripheral and Central GAP roles

Glucose

This Profile enables a device to connect and interact with a Glucose Sensor for use in consumer healthcare applications.

- **Glucose Sensor** Profile role – Specified as a GATT Server. Requires the following Services: **Glucose Service**, **Device Information Service**.
  - Peripheral GAP role
- **Collector** Profile role – Specified as a GATT Client. Requires support of the following Service: **Glucose Service**. Support of **Device Information Service** is optional.
  - Central GAP role

Health Thermometer

This Profile enables a Collector device to connect and interact with a Thermometer sensor for use in healthcare applications.

- **Thermometer** Profile role – Specified as a GATT Server. Requires the following Services: **Health Thermometer Service**, **Device Information Service**.
  - Peripheral GAP role
- **Collector** Profile role – Specified as a GATT Client. Requires support of the following Service: **Health Thermometer Service**. Support of **Device Information Service** is optional.
  - Central GAP role

Heart Rate

This Profile enables a Collector device to connect and interact with a Heart Rate Sensor for use in fitness applications.

- **Heart Rate Sensor** Profile role – Specified as a GATT Server. Requires the following Services: **Heart Rate Service**, **Device Information Service**.
  - Peripheral GAP role
- **Collector** Profile role – Specified as a GATT Client. Requires support of the following Service: **Heart Rate Service**. Support of **Device Information Service** is optional.
  - Central GAP role

HID over GATT

This Profile defines how a device with BLE wireless communications can support HID Services over the BLE protocol stack using the Generic Attribute Profile.

- **HID Device** Profile role – Specified as a GATT Server. Requires the following Services: **HID Service**, **Battery Service**, and **Device Information Service**. Optionally may include **Scan Parameters Service** as part of the **Scan Server** role of the **Scan Parameters** Profile. **HID Device** supports multiple instances of **HID Service** and **Battery Service** and may include any other optional Services.
  - Peripheral GAP role
- **Boot Host** Profile role – Specified as a GATT Client. Requires support of the following Service: **HID Service**. Support of **Battery Service** and **Device Information Service** is optional.
  - Central GAP role
- **Report Host** Profile role – Specified as a GATT Client. Requires support of the following Services: **HID Service**, **Battery Service**, **Device Information Service**. Support of **Scan Client** role of the **Scan Parameters** is optional.
  - Central GAP role
- **Report and Boot Host** Profile role – Specified as a GATT Client. Requires support of the following Services: **HID Service**, **Battery Service**, **Device Information Service**. Support of **Scan Client** role of the **Scan Parameters** is optional.
  - Central GAP role

Location and Navigation

This Profile enables a Collector device to connect and interact with a Location and Navigation Sensor for use in outdoor activity applications.

- **Location and Navigation Sensor** Profile role – Specified as a GATT Server. Requires the following Service: **Location and Navigation Service**. Optionally may include **Device Information Service** and **Battery Service**.
  - Peripheral GAP role

- **Collector** Profile role – Specified as a GATT Client. Requires support of the following Services: **Location and Navigation Service**. Support of **Device Information Service** and **Battery Service** is optional.
    - Central GAP role

Phone Alert Status

This Profile enables a device to alert its user about the alert status of a phone connected to the device.

- **Phone Alert Server** Profile role – Specified as a GATT Server. Requires the following Services: **Phone Alert Status Service**.
    - Central GAP role
    - Peripheral and Central GAP role
- **Phone Alert Client** Profile role – Specified as a GATT Client. Requires support of the following Service: **Phone Alert Service**.
    - Peripheral GAP role
    - Peripheral and Central GAP role

Proximity

The Proximity Profile enables proximity monitoring between two devices.

- **Proximity Reporter** Profile role – Specified as a GATT Server. Requires the following Service: **Link Loss Service**. Optionally may include **Immediate Alert Service** and **Tx Power Service** if both are used. Using only one of the optional Services is not allowed.
    - Peripheral GAP role
    - Central GAP role
- **Proximity Monitor** Profile role – Specified as a GATT Client. Requires support of the following Services: **Link Loss Service**. Support of **Immediate Alert Service** and **Tx Power Service** is optional. Same restrictions apply as to **Proximity Reporter**.
    - Central GAP role
    - Peripheral GAP role
    - Peripheral and Central GAP role

Running Speed and Cadence

This Profile enables a Collector device to connect and interact with a Running Speed and Cadence Sensor for use in sports and fitness applications.

- **Running Speed and Cadence Sensor** Profile role – Specified as a GATT Server. Requires the following Service: **Running Speed and Cadence Service**. Optionally may include **Device Information Service**.
    - Peripheral GAP role
- **Collector** Profile role – Specified as a GATT Client. Requires support of the following Services: **Running Speed and Cadence Service**. Support of **Device Information Service** is optional.
    - Central GAP role

Scan Parameters

This Profile defines how a Scan Client device with BLE wireless communications can write its scanning behavior to a Scan Server, and how a Scan Server can request updates of the Scan Client scanning behavior.

- **Scan Server** Profile role – Specified as a GATT Server. Requires the following Service: **Scan Parameters Service**.
    - Peripheral GAP role
- **Scan Client** Profile role – Specified as a GATT Client. Required support of the following Service: **Scan Parameters Service**.
    - Central GAP role

Time

The Time Profile enables the device to get the date, time, time zone, and DST information and control the functions related to time.

- **Time Server** Profile role – Specified as a GATT Server. Requires the following Service: **Current Time Service**. Optionally may include **Next DST Change Service** and **Reference Time Update Service**.
    - Central GAP role
    - Peripheral and Central GAP role
- **Time Client** Profile role – Specified as a GATT Client. Requires support of the following Service: **Current Time Service**. Support of **Next DST Change Service** and **Reference Time Update Service** is optional.
    - Peripheral GAP role

- Peripheral and Central GAP role

Custom

Used to create a custom Profile. This Profile mode allows you to add in a **Custom Service** and gives control over the Service types.

- **Server (GATT Server)** Profile role
  - Peripheral GAP role
  - Central GAP role
  - Peripheral and Central GAP roles
  - Broadcaster GAP role
  - Observer GAP role
- **Client (GATT Client)** Profile role
  - Peripheral GAP role
  - Central GAP role
  - Peripheral and Central GAP roles
  - Broadcaster GAP role
  - Observer GAP role
- **Client and Server (GATT Client and Server)** Profile role
  - Peripheral GAP role
  - Central GAP role
  - Peripheral and Central GAP roles
  - Broadcaster GAP role
  - Observer GAP role

Profile Role

The **Profile role** parameter configuration depends on the chosen **Profile**, and the **Profile role** selection affects the **GAP role** parameter. These parameters affect the options available on the **Profiles** tab.

- **GATT Server –** Defines the role of the device that contains a specific data in a structured form. The device in this role is usually a sensor that gets the data. The data is structured in the GATT database. BLE Profiles can introduce their own names to identify GATT Server device (e.g. Find Me Profile uses “Find Me Target”). GATT Server devices usually utilize the GAP Peripheral role.

- **GATT Client –** Defines the role of the device that generates requests to the GATT Server device to fetch data. BLE Profiles can introduce their own names to identify GATT Client device (e.g. Find Me Profile uses “Find Me Locator”). GATT Client devices usually utilize the GAP Peripheral role.
- **Client and Server –** Defines the role of the device that concurrently can perform functionality of a GATT Client and Server Profile role. A device in this role should be configured for Peripheral and Central GAP role. For example, a peripheral device can act as a GATT Client and start discovering the iOS device's (acting as GATT Server) Services (battery, time and Apple notification central Service).

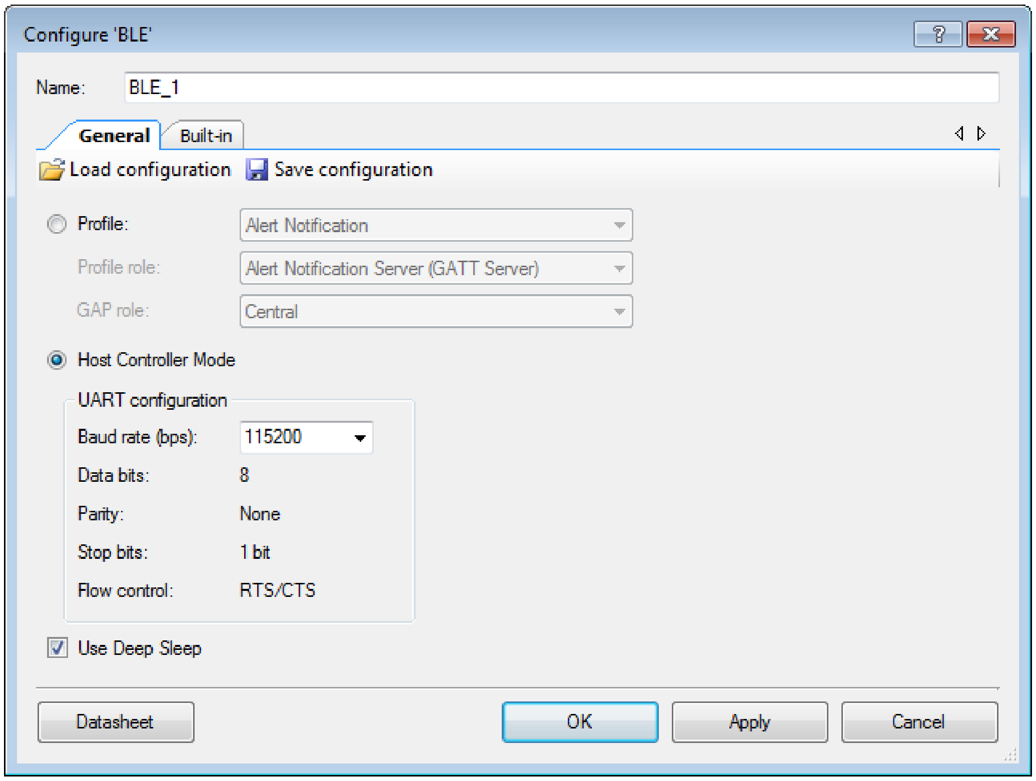
Gap Role

The **GAP role** parameter can take the following values:

- **Peripheral** – Defines a device that advertises using connectable advertising packets and so becomes a slave once connected. Peripheral devices need a Central device, as the Central device initiates connections. Through the advertisement data, a Peripheral device can broadcast the general information about a device.
- **Central** – Defines a device that initiates connections to peripherals and will therefore become a master when connected. Peripheral devices need a Central device, as the Central device initiates connections.
- **Broadcaster** – Similar to the Peripheral role, the device sends advertising data. However Broadcaster does not support connections and can only send data but not receive them.
- **Observer** – When in this role, the device scans for Broadcasters and reports the received information to an application. The Observer role does not allow transmissions.
- **Peripheral and Central** – In this role, the application can perform role reversal between Peripheral and Central roles at run time. For example, Bluetooth Smart watch (Peripheral) can connect to a smartphone (Central device). The same sports watch can then switch to the Central device mode to obtain data from other Peripheral devices such as a heart rate monitor and a blood pressure sensor.

Host Controller Mode

Choosing this configuration places the Component in HCI mode, which enables use of the device as a BLE controller. It also allows communication with a host stack using a Component embedded UART. When choosing this mode, the **Profile** mode options, **Profiles** tab, and **GAP Settings** tab become unavailable.



Name: BLE_1

Profile: Alert Notification

Profile role: Alert Notification Server (GATT Server)

GAP role: Central

(●) Host Controller Mode

UART configuration

Baud rate (bps): 115200

Data bits: 8

Parity: None

Stop bits: 1 bit

Flow control: RTS/CTS

[✓] Use Deep Sleep

It also reveals the UART configuration information.

- UART Configuration – The UART is a full-duplex 8 data bit, 1 stop bit, no parity with Flow control interface. These settings are fixed.
- Baud rate (bps) – Configures the UART baud rate.

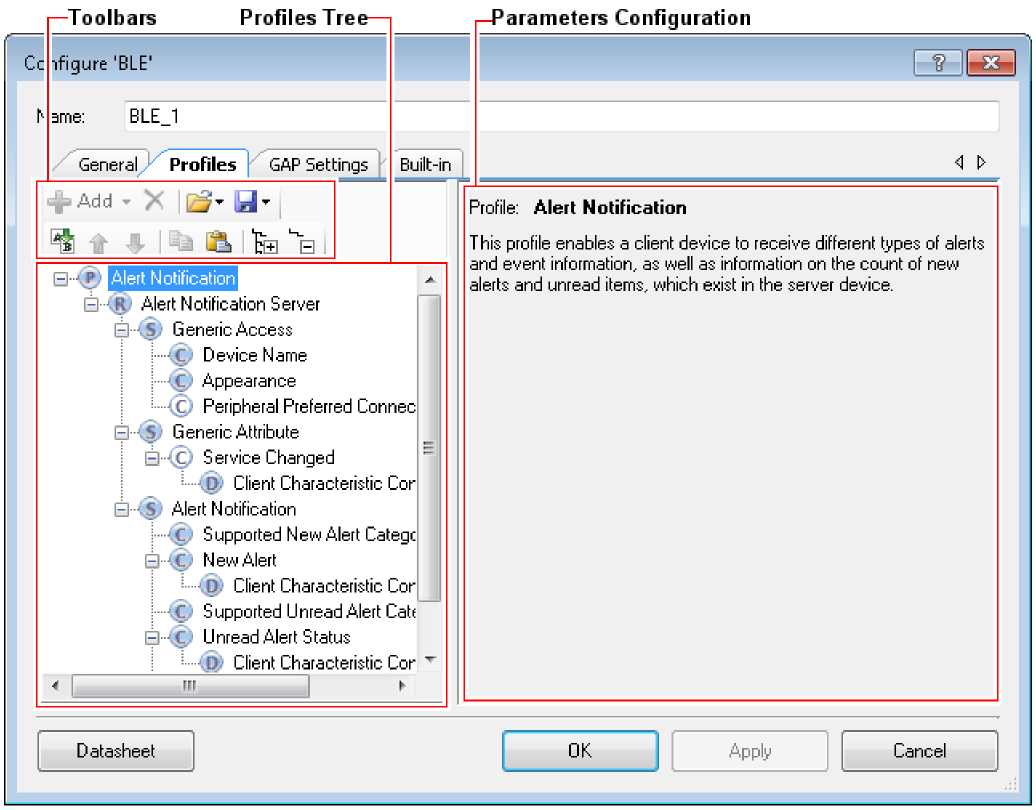
Use Deep Sleep

This parameter identifies if Deep Sleep Mode support is required for the BLE Component. Default: true.

When this parameter is set, WCO must be selected as the LFCLK source in the Design-Wide Resources Clock Editor. This configuration is a requirement if you intend to use the Component in Deep Sleep Mode.

Profiles Tab

The **Profiles** tab is used to configure Profile-specific parameters. It is directly affected by the choice of **Profile** settings set in the **General** tab. The **Profiles** tab has 3 areas: toolbars, a Profiles tree, and a parameters configuration section.



Profile: **Alert Notification**

This profile enables a client device to receive different types of alerts and event information, as well as information on the count of new alerts and unread items, which exist in the server device.

Toolbars

The toolbars contain navigation options and a means to add or delete Services, Characteristics, and Descriptors.

- **Add Service –** This option is available when the **Profile Role** is highlighted in the Profile tree. It allows loading of Services in the selected **Profile Role**. In GATT server configuration, this option adds the selected service data to the server GATT database and enables service specific APIs. In GATT client configuration, the data structures for auto discovery of this service is created by the Component. If services that are not populated in the GUI are discovered during auto discovery, the Component ignores those service and the application is responsible for discovering the details of such services. Refer to the Profile section for the available Services.

- **Add Characteristic –** This option is available when a Service is highlighted in the Profile tree. The Characteristic options are unique to each Service and are all loaded automatically when a Service is added to the design. The **Add Characteristic** button can be used to manually add new Characteristics to the Service. All Characteristics for the above mentioned Services plus Custom Characteristic are available for selection.

- **Add Descriptor –** This option is available when a Characteristic is highlighted in the Profile tree. Similar to the Characteristic options, Descriptor options are unique to a Characteristic and are all automatically loaded when a Characteristic is added to the design. For more information about BLE Characteristic Descriptors, refer to developer.bluetooth.org. (**Note** You should be a member of Bluetooth SIG to have full access to this site.)
- **Delete –** Deletes the selected Service, Characteristic, or Descriptor.
- **Load/Save –** Imports/Exports Profiles, Services, Characteristics, and Descriptors as shown in the tree. This functionality is independent of the **Load Configuration/Save Configuration** buttons on the **General** tab. That is, this allows you to customize this tree independent of the general settings. Each exported file type will have its own extension.
- **Rename –** Renames the selected item in the Profiles tree.
- **Move Up/Down –** Moves the selected item up or down in the Profiles tree.
- **Copy/Paste –** Copies/pastes items in the Profiles tree.
- **Expand All –** Expands all items in the Profiles tree.
- **Collapse all Services –** Collapses all Services in the Profiles tree.

Profiles Tree

The Profiles tree is used to view Services, Characteristics, and Descriptors in the selected Profile. By navigating through the tree, you can quickly add, delete, or modify Services, Characteristics, and Descriptors using the toolbar buttons or the context menu. You can configure the parameters by clicking an item on the tree. These parameters will show in the Parameters Configuration section.

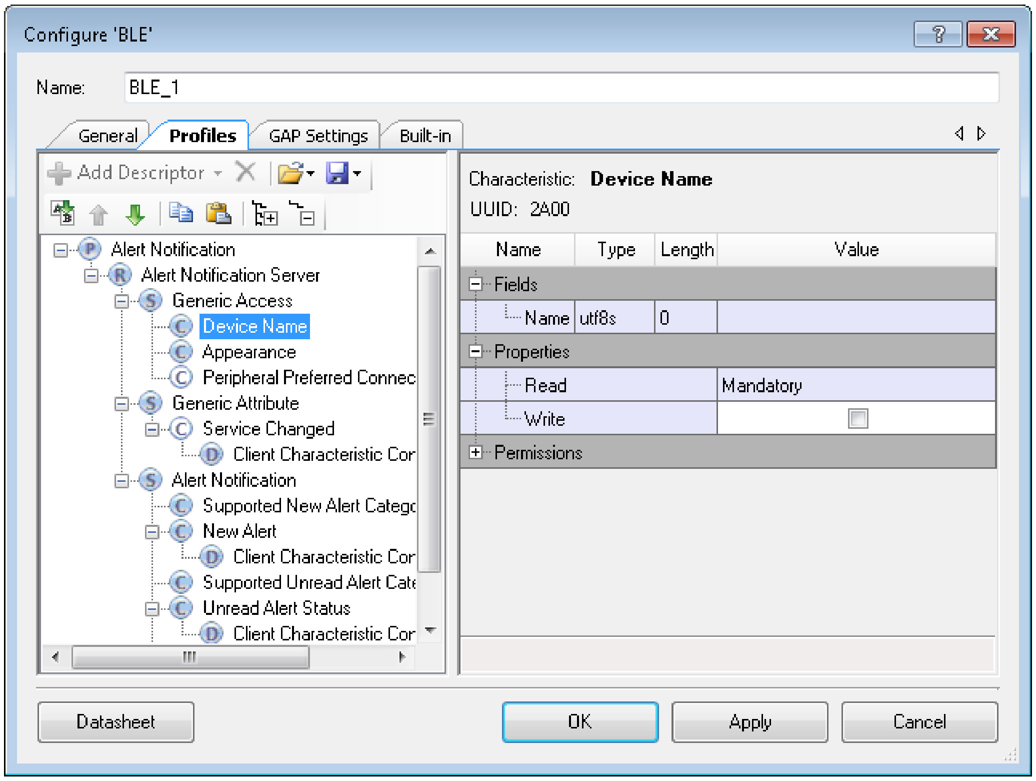
Parameters Configuration

The Parameters Configuration section allows you to configure a Service or Characteristic by selecting the type of Service or Characteristic in the tree.

Notes

- All Profiles must have a **Generic Access Service** and a **Generic Attribute Service**.
- The Service Characteristics are configurable only when the device is a GATT Server.
- The security settings located in the **GAP Settings** tab are applied globally. In addition to this, you may manually configure the security of each Characteristic/Descriptor.

Generic Access Service



Characteristic: **Device Name**
UUID: 2A00

Name	Type	Length	Value
Fields			
Name	utf8s	0	
Properties			
Read			Mandatory
Write			☐
Permissions			

This Service is used to define the basic Bluetooth connection and discovery parameters. Click on the Characteristic under the **Generic Access Service** to view that particular Characteristic settings. You perform the actual Characteristics configuration in the **General** options located in the **GAP Settings** tab.

- **Device Name**: This is the name of your device. It has a read (without authentication/authorization) property associated with it by default. This parameter can be up to 248 bytes. The value comes from the **Device Name** field on the GAP Settings tab, under General.

- **Appearance**: The device's logo or appearance, which is a SIG defined 2-byte value. It has a read (without authentication/authorization) property associated with it by default. The value comes from the **Appearance** field on the GAP Settings tab, under General.

- **Peripheral Preferred Connection**: A device in the peripheral role can convey its preferred connection parameter to the peer device. This parameter is 8 bytes in total and is composed of the following sub-parameters.

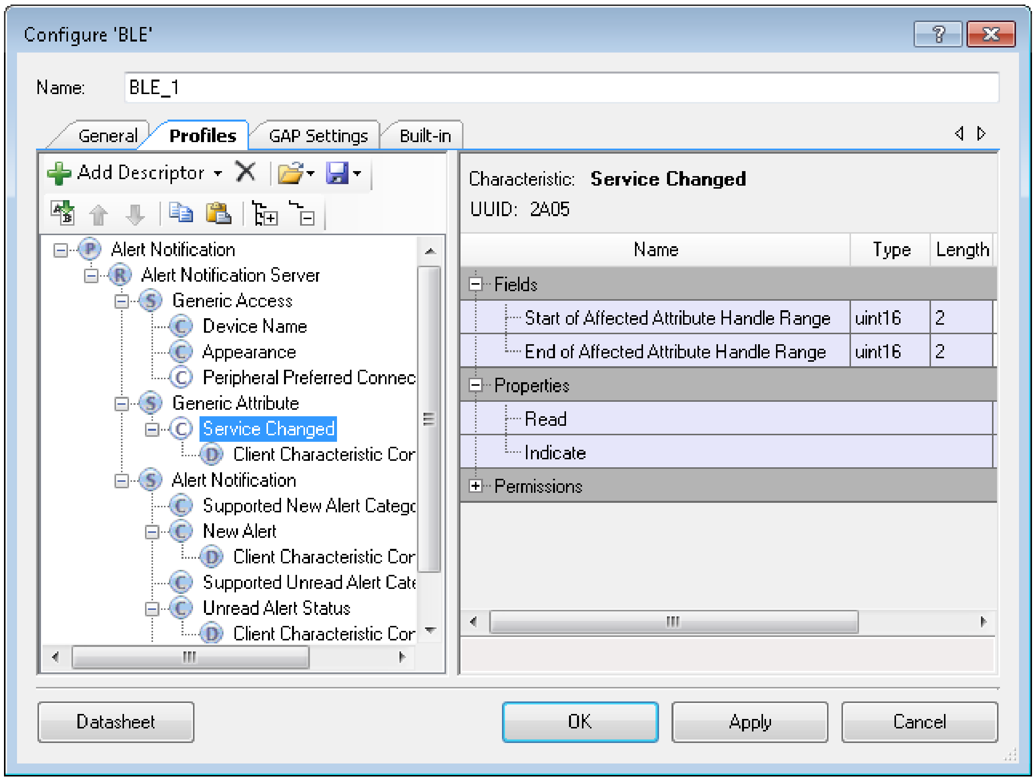
  Note This parameter is read-only and is derived from the **Advertisement settings Connection Parameters**. It will only be available when the device supports a Peripheral role. Refer to the Connection Parameters section for more information.

  - **Minimum Connection Interval:** This is a 2-byte parameter that denotes the minimum permissible connection time.
  - **Maximum Connection Interval**: This is a 2-byte parameter that denotes the maximum permissible connection time.
  - **Slave Latency**: This is a 2-byte value and defines the latency between consecutive connection events.
  - **Connection Supervision Timeout Multiplier**: This is a 2-byte value that denotes the LE link supervision timeout interval. It defines the timeout duration for which an LE link needs to be sustained in case of no response from the peer device over the LE link.

  Note The above parameters are used for connection parameters update procedure over L2CAP if a GAP central device does not use the peripheral preferred connection parameters. For example, iOS7 ignores peripheral preferred connection parameter Characteristics and establishes a connection with a default 30 ms connection interval. The peripheral device should request a connection parameter update by sending an L2CAP connection parameter update request at an appropriate time.

  A typical peripheral implementation should initiate L2CAP connection parameter update procedure once any Characteristic is configured for periodic notification or indication.

Generic Attribute Service



Name: BLE_1

Alert Notification
Alert Notification Server
Generic Access
Device Name
Appearance
Peripheral Preferred Connec
Generic Attribute
Service Changed
Client Characteristic Cor
Alert Notification
Supported New Alert Catego
New Alert
Client Characteristic Cor
Supported Unread Alert Cate
Unread Alert Status
Client Characteristic Cor

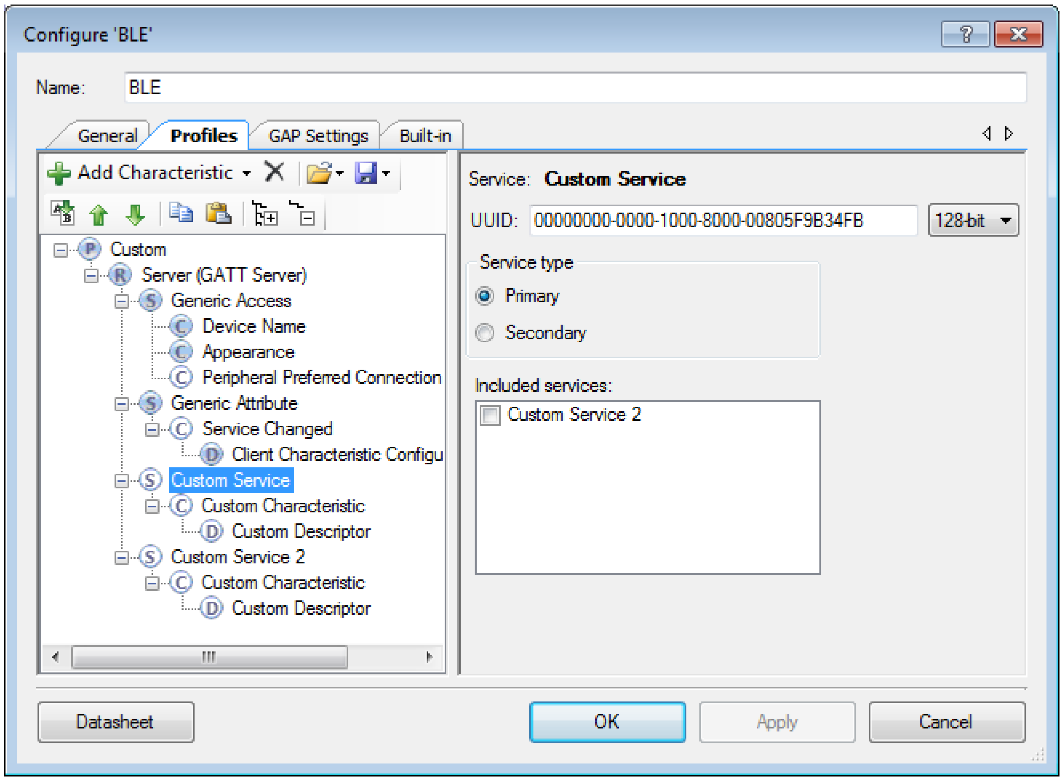
Characteristic: **Service Changed**
UUID: 2A05

Name	Type	Length
Fields		
Start of Affected Attribute Handle Range	uint16	2
End of Affected Attribute Handle Range	uint16	2
Properties		
Read		
Indicate		
Permissions		

Click on the Characteristic under the Generic Attribute Service to configure that particular Characteristic.

- **Service Changed** - This Characteristic is used to indicate to the connected devices that a Service has changed (i.e., added, removed, or modified). It is used to indicate to GATT Clients that have a trusted relationship (i.e., bond) with the GATT Server when GATT based Services have changed when they re-connect to the GATT Server. It is mandatory for the device in the GATT Client role. For the device in the GATT Server role, the Characteristic is mandatory if the GATT Server changes the supported Services in the device.

Custom Service Configuration



Name: BLE

Service: **Custom Service**

UUID: 00000000-0000-1000-8000-00805F9B34FB 128-bit

Service type

[✓ Primary]

Secondary

Included services:

☐ Custom Service 2

UUID

A universally unique identifier of the service. This field is editable for Custom Services.

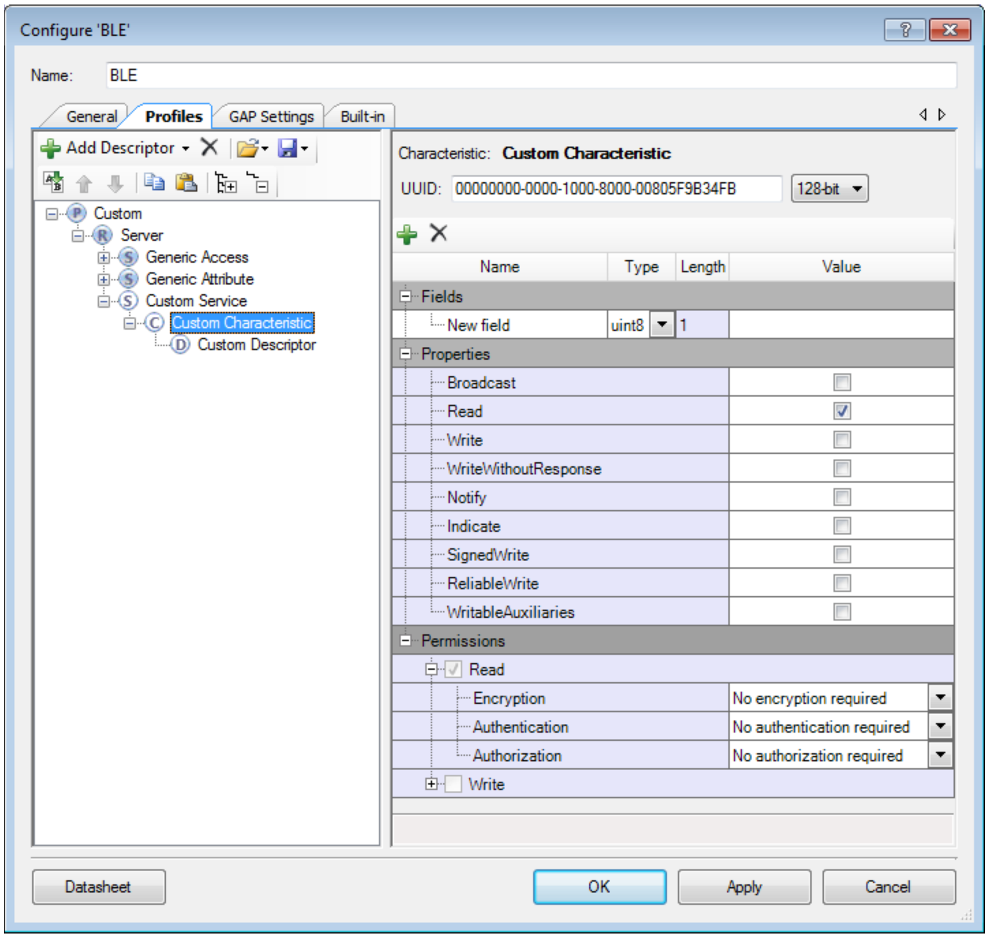
Service type

- **Primary** – Represents the primary functionality of the device.
- **Secondary** – Represents an additional functionality of the device. The secondary service must be included in another service.

Included services

- The list of the Services that can be included in the selected Service. Each Service may have one or more included Services. The included Services provide the additional functionality for the Service.

Custom Characteristic Configuration



Name: BLE

- Custom
  - Server
    - Generic Access
    - Generic Attribute
    - Custom Service
      - Custom Characteristic
        - Custom Descriptor

Characteristic: **Custom Characteristic**

UUID: 00000000-0000-1000-8000-00805F9B34FB 128-bit

Name	Type	Length	Value
Fields			
New field	uint8	1	
Properties			
Broadcast			☐
Read			☑
Write			☐
WriteWithoutResponse			☐
Notify			☐
Indicate			☐
SignedWrite			☐
ReliableWrite			☐
WritableAuxiliaries			☐
Permissions			
☑ Read			
Encryption			No encryption required
Authentication			No authentication required
Authorization			No authorization required
☐ Write			

UUID

A universally unique identifier of the Characteristic. This field is editable for Custom Characteristics.

Fields

Fields represent a Characteristic value. The default value for each field can be set in the **Value** column. In case of the Custom Characteristic, the fields are customizable.

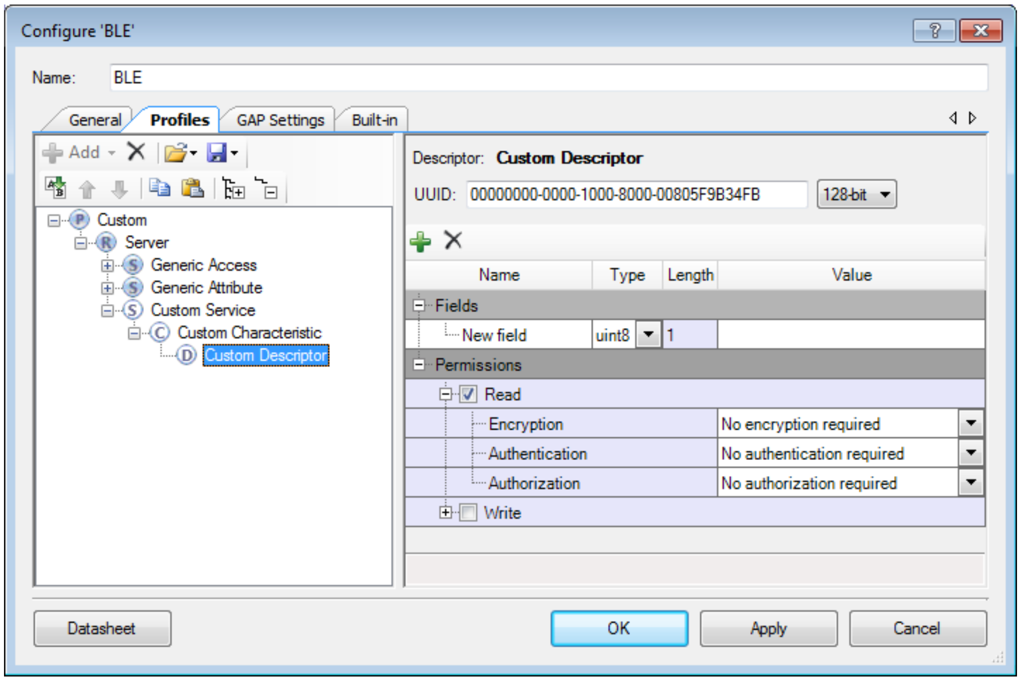
Properties

The Characteristic properties define how the Characteristic value can be used. Some properties (Broadcast, Notify, Indicate, Reliable Write, Writable Auxilires) require the presence of a corresponding Characteristic Descriptor.

Permissions

Characteristic permissions define how the Characteristic Value attribute can be accessed and the security level required for this access. Access permissions are set based on the Characteristic properties. Security permissions are automatically updated for all Characteristics when the **Security Mode** or **Security Level** parameters are changed on the GAP tab.

Custom Descriptor Configuration



Descriptor: **Custom Descriptor**

UUID: 00000000-0000-1000-8000-00805F9B34FB 128-bit

Name	Type	Length	Value
Fields			
New field	uint8	1	
Permissions			
☑ Read			
Encryption			No encryption required
Authentication			No authentication required
Authorization			No authorization required
☐ Write			

UUID

A universally unique identifier of the Descriptor. This field is editable for Custom Descriptors.

Fields

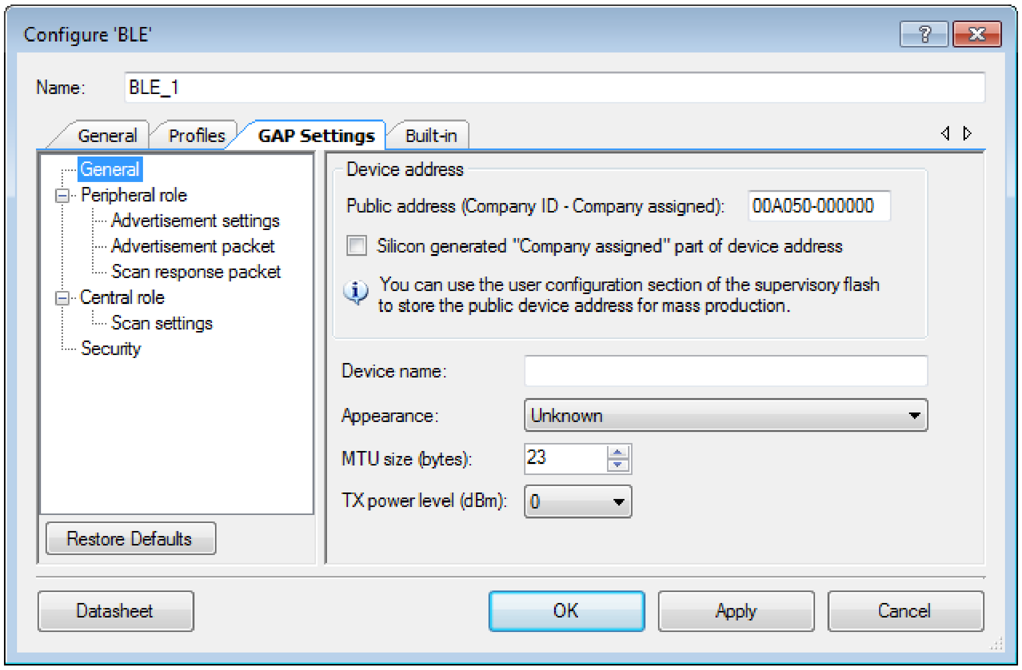
Fields represent a Descriptor value. The default value for each field can be set in the **Value** column. In case of the Custom Descriptor, the fields are customizable.

Permissions

Descriptor permissions define how the Descriptor attribute can be accessed and the security level required for this access.

GAP Settings Tab

The GAP parameters define the general connection settings required when connecting Bluetooth devices. It contains various sections of parameters based on the item you select in the tree.



Name: BLE_1

Device address

Public address (Company ID - Company assigned): 00A050-000000

☐ Silicon generated "Company assigned" part of device address

You can use the user configuration section of the supervisory flash to store the public device address for mass production.

Device name:

Appearance: Unknown

MTU size (bytes): 23

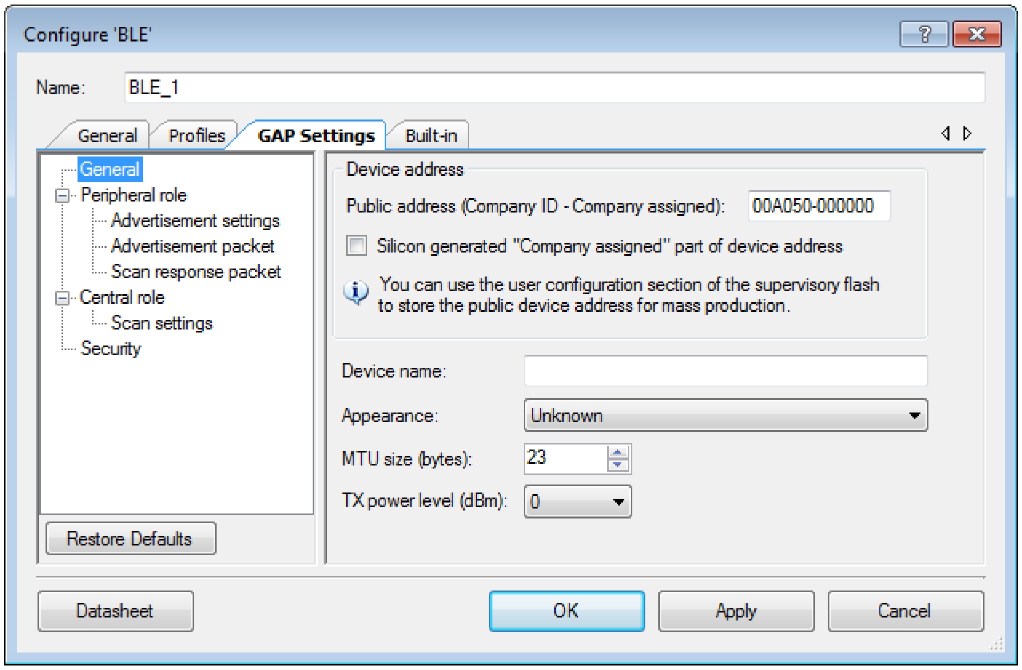
TX power level (dBm): 0

The **GAP Settings** tab displays the settings possible based on the GAP role selected in the **General** tab. This tab allows the default settings to be restored by using the **Restore Defaults** button.

The following sections show the different categories of parameters based on what item you select in the tree.

GAP Settings Tab – General

This section contains general GAP parameters:



Name: BLE_1

Device address

Public address (Company ID - Company assigned): 00A050-000000

☐ Silicon generated "Company assigned" part of device address

You can use the user configuration section of the supervisory flash to store the public device address for mass production.

Device name:

Appearance: Unknown

MTU size (bytes): 23

TX power level (dBm): 0

Public device address (Company ID – Company assigned)

This is a unique 48-bit Bluetooth public address that is used to identify the device. It is divided into the following two parts:

- **"Company ID"** part is contained in the 24 most significant bits. It is a 24-bit Organization Unique Identifier (OUI) address assigned by IEEE.
- **"Company assigned"** part is contained in the 24 least significant bits.

The address configured here is static and is designed to be used for development purposes only. This address is programmed into the device via the SWD interface. Normally this address must be programmed only once during mass production, and then never changed in-field. However, user flash can be reprogrammed in-field many times. During prototyping (FW design), this address can be programmed using MiniProg3. For that you can use the application installed in the "./Example/Misc/PSoC4-BLE-SFLASH-Update" folder of PSoC Programmer. This application is provided in source code, and can be used as a reference example for implementation in production programmers.

Silicon generated "Company assigned" part of device address

When checked, the "Company assigned" part of the device address is generated using the wafer ID and X-Y die location on the wafer.

Device Name

The device name to be displayed on the peer side. It has a read (without authentication/authorization) property associated with it by default. This parameter can be up to 248 bytes.

Note This parameter configures the **GAP Service Device name** Characteristic located in the **Profile Tree**. It is available for modification only when the device is a GATT Server.

Appearance

The device's logo or appearance, which is a SIG defined 2-byte value. It has a read (without authentication/authorization) property associated with it by default.

Note This parameter configures the **GAP Service Appearance** Characteristic located in the **Profile Tree**, It is available for modification only when the device is a GATT Server.

MTU Size

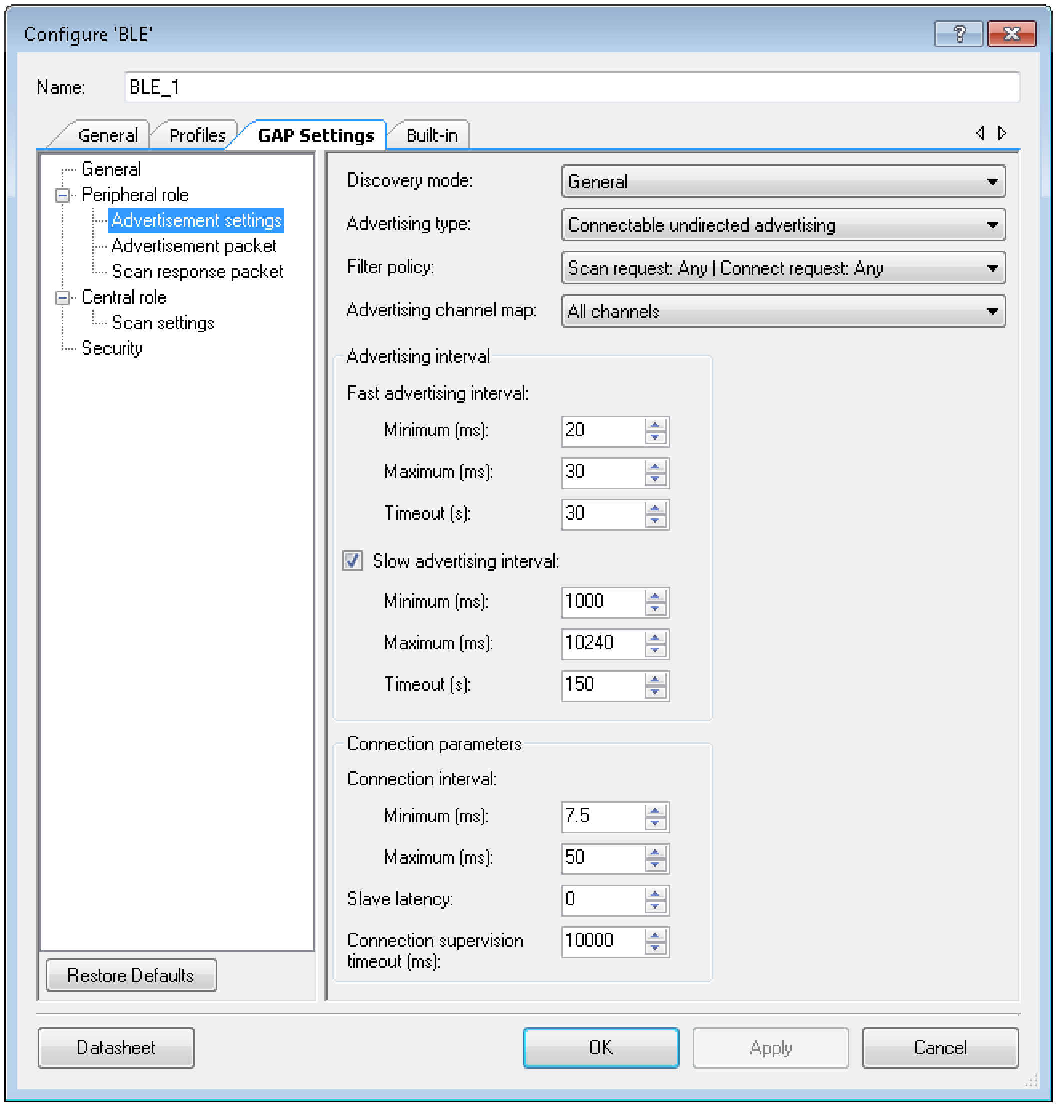
Maximum Transmission Unit size (Bytes) of an attribute to be used in the design. Valid range is from 23 to 512 bytes. This value is used to respond to an Exchange MTU request from the GATT Client.

TX Power level

The initial transmitter power level (dBm) upon startup. It is applicable for advertisement and connection channels. Default: 0 dBm. Possible values: -18 dBm, -12 dBm, -6 dBm, -3 dBm, -2 dBm, -1 dBm, 0 dBm, 3 dBm.

GAP Settings Tab – Advertisement Settings

These parameters are available when the device is configured as "Peripheral," "Broadcaster," or "Peripheral and Central" **GAP role**.



Name: BLE_1

Discovery mode: General

Advertising type: Connectable undirected advertising

Filter policy: Scan request: Any | Connect request: Any

Advertising channel map: All channels

Advertising interval

Fast advertising interval:

Minimum (ms): 20

Maximum (ms): 30

Timeout (s): 30

[✓] Slow advertising interval:

Minimum (ms): 1000

Maximum (ms): 10240

Timeout (s): 150

Connection parameters

Connection interval:

Minimum (ms): 7.5

Maximum (ms): 50

Slave latency: 0

Connection supervision timeout (ms): 10000

Discovery mode

- **Non-discoverable** – In this mode, the device can't be discovered by a Central device.
- **Limited Discoverable** – This mode is used by devices that need to be discoverable only for a limited period of time, during temporary conditions, or for a specific event. The device which is advertising in Limited Discoverable mode are available for a connection to Central device which performs Limited Discovery procedure. The timeout duration is defined by the applicable advertising timeout parameter.

- **General Discoverable Mode** – In this mode, the device should be used by devices that need to be discoverable continuously or for no specific condition. The device which is advertising in General Discoverable mode are available for a connection to Central device which performs General Discovery procedure.The timeout duration is defined by the applicable advertising timeout parameter.

Advertising type

This parameter defines the advertising type to be used by the LL for an appropriate **Discovery mode**.

- **Connectable undirected advertising** – This option is used for general advertising of the advertising and scan response data. It allows any other device to connect to this device.
- **Scannable undirected advertising** – This option is used to broadcast advertising data and scan response data to active scanners.
- **Non-connectable undirected advertising** – This option is used to just broadcast advertising data.

Filter policy

This parameter defines how the scan and connection requests are filtered.

- **Scan request: Any | Connect request: Any** – Process scan and connect requests from all devices.
- **Scan request: White List | Connect request: Any** – Process scan requests only from devices in the White List and connect requests from all devices.
- **Scan request: Any | Connect request: White List** – Process scan requests from all devices and connect requests only from devices in the White List.
- **Scan request: White List | Connect request: White List** – Process scan and connect requests only from devices in the White List.

Advertising channel map

This parameter is used to enable a specific advertisement channel.

- **Channel 37** – enables advertisement channel #37
- **Channel 38** – enables advertisement channel #38
- **Channel 39** – enables advertisement channel #39
- **Channels 37 and 38** – enables advertisement channels #37 and #38

- **Channel 37 and 39** – enables advertisement channels #37 and #39
- **Channels 38 and 39** – enables advertisement channels #38 and #39
- **All channels** – enables all three advertisement channels

Advertising Interval

This parameter defines the interval between two advertising events. Set the permissible minimum and maximum values of two Advertisement interval types: **Fast advertising interval** and **Slow advertising interval**. Typically after the device initialization, a peripheral device uses the Fast advertising interval. After the **Fast advertising interval timeout** value expires, and if a connection with a Central device is not established, then the Profile switches to Slow advertising interval to save the battery life. After the **Slow advertising interval timeout** value expires, 'CYBLE_EVT_GAPP_ADVERTISEMENT_START_STOP' event is generated.

Note: The Advertising interval needs to be aligned with the selected Profile specification.

- **Fast advertising interval** – This advertisement interval results in faster LE Connection. The BLE Component uses this interval value when the connection time is between the specified minimum and maximum values of the interval.
    - Minimum: The minimum interval for advertising the data and establishing the LE Connection. The parameter is configured to increment in multiples of 0.625 ms. Valid range is from 20 ms to 10240 ms.
    - Maximum: The maximum interval for advertising the data and establishing the LE Connection. The parameter is configured to increment in multiples of 0.625 ms. Valid range is from 20 ms to 10240 ms.
    - Timeout: The timeout value of advertising with fast advertising interval parameters.
- **Slow advertising interval** – Defines the advertising interval for slow advertising. This is an optional parameter which, if enabled, allows to implement advertising with a lower duty cycle to save battery life. The Slow advertising interval parameters are applied to the device after the internal fast advertising interval timeout occurs.. The minimum and maximum values defined using this parameter allow the BLE Stack to expect the advertising to happen within these intervals.
    - Minimum: The minimum interval for advertising the data and establishing the LE Connection. The parameter is configured to increment in multiples of 0.625 ms. Valid range is from 1000 ms to 10240 ms.
    - Maximum: The maximum interval for advertising the data and establishing the LE Connection. The parameter is configured to increment in multiples of 0.625 ms. Valid range is from 1000 ms to 10240 ms.
    - Timeout: The timeout value of advertising with slow advertising interval parameters.

Connection Parameters

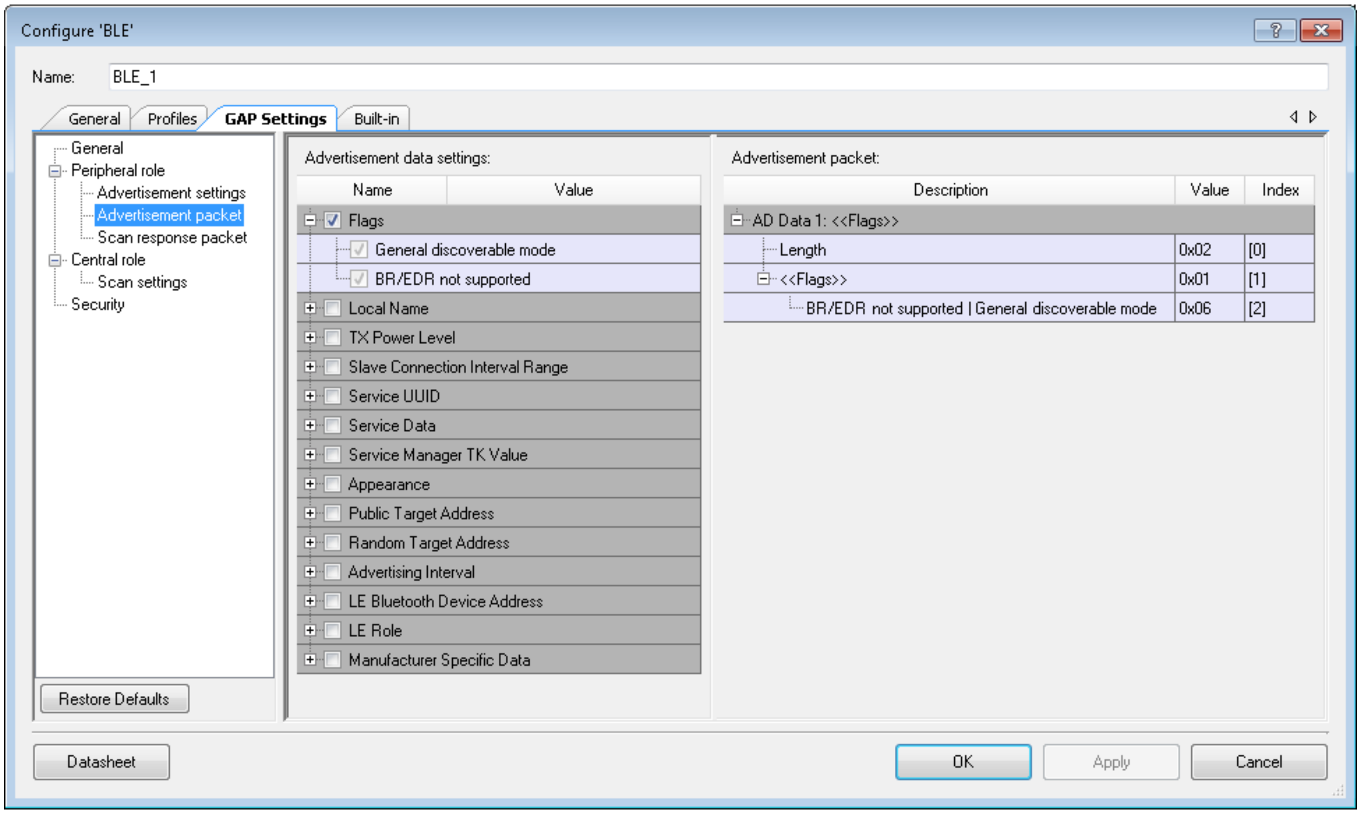
These parameters define the connection event timing for a Central device communicating with the Peripheral device. Consecutive connection events are separated by the defined Connection interval.

Note The scaled values of these parameters used internally by the BLE stack are also shown in the **Peripheral Preferred Connection Parameters**. These are the actual values sent over the air.

- **Connection interval –** The Central device connecting to a Peripheral device needs to define the time interval for a connection to happen.
  - Minimum (ms): This parameter is the minimum permissible connection time value to be used during a connection event. It is configured in steps of 1.25 ms. The range is from 7.5 ms to 4000 ms.
  - Maximum (ms): This parameter is the maximum permissible connection time value to be used during a connection event. It is configured in steps of 1.25 ms. The range is from 7.5 ms to 4000 ms.
- **Slave Latency –** Defines the latency of the slave in responding to a connection event in consecutive connection events. This is expressed in terms of multiples of connection intervals, where only one connection event is allowed per interval. The range is from 0 to 499 events.
- **Connection Supervision Timeout –** This parameter defines the LE link supervision timeout interval. It defines the timeout duration for which an LE link needs to be sustained in case of no response from peer device over the LE link. The time interval is configured in multiples of 10 ms. The range is from 100 ms to 32000 ms.

GAP Settings Tab – Advertisement packet

This section displays when the device is configured to contain "Peripheral," "Broadcaster," or "Peripheral and Central" **GAP role**. It is used to configure the **Advertisement data** to be used in device advertisements.



Advertisement data settings:

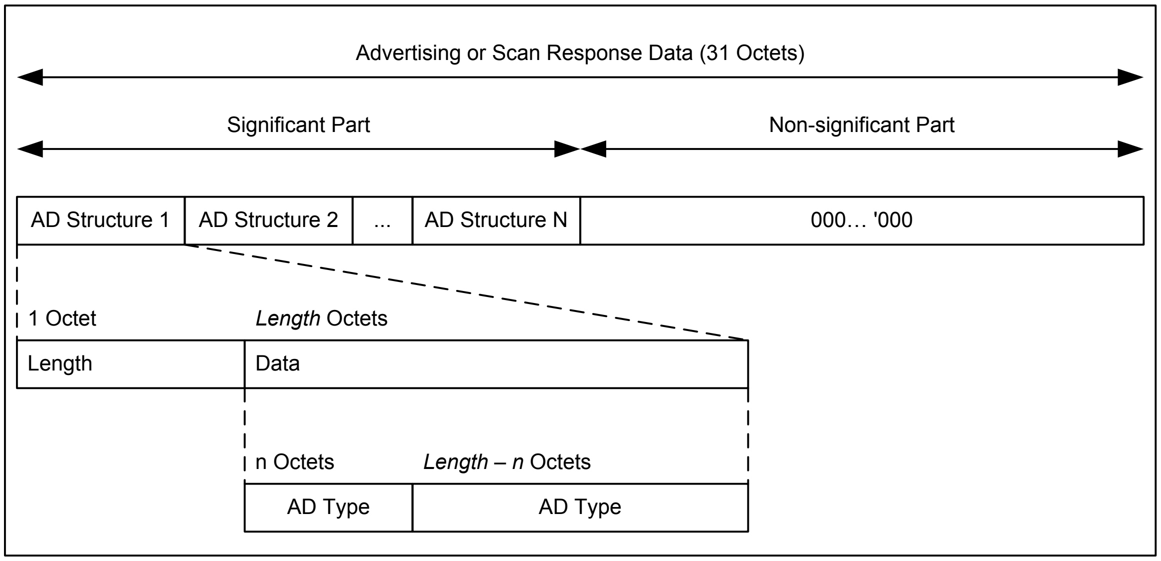
Name	Value
✓ Flags	
✓ General discoverable mode	
✓ BR/EDR not supported	
Local Name	
TX Power Level	
Slave Connection Interval Range	
Service UUID	
Service Data	
Service Manager TK Value	
Appearance	
Public Target Address	
Random Target Address	
Advertising Interval	
LE Bluetooth Device Address	
LE Role	
Manufacturer Specific Data	

Advertisement packet:

Description	Value	Index
AD Data 1: <<Flags>>		
Length	0x02	[0]
<<Flags>>	0x01	[1]
BR/EDR not supported | General discoverable mode	0x06	[2]

Advertisement / Scan response data settings

Advertisement (AD) or **Scan response data** packet is a 31 byte payload used to declare the device's BLE capability and its connection parameters. The structure of this data is shown below as specified in the Bluetooth specification.



The data packet can contain a number of AD structures. Each of these structures is composed of the following parameters.

- **AD Length**: Size of the **AD Type** and **AD Data** in bytes.
- **AD Type**: The type of advertisement within the AD structure.
- **AD Data**: Data associated with the **AD Type**.

The total length of a complete Advertising packet cannot exceed 31 bytes.

An example structure for **Advertisement data** or **Scan response data** is as follows.

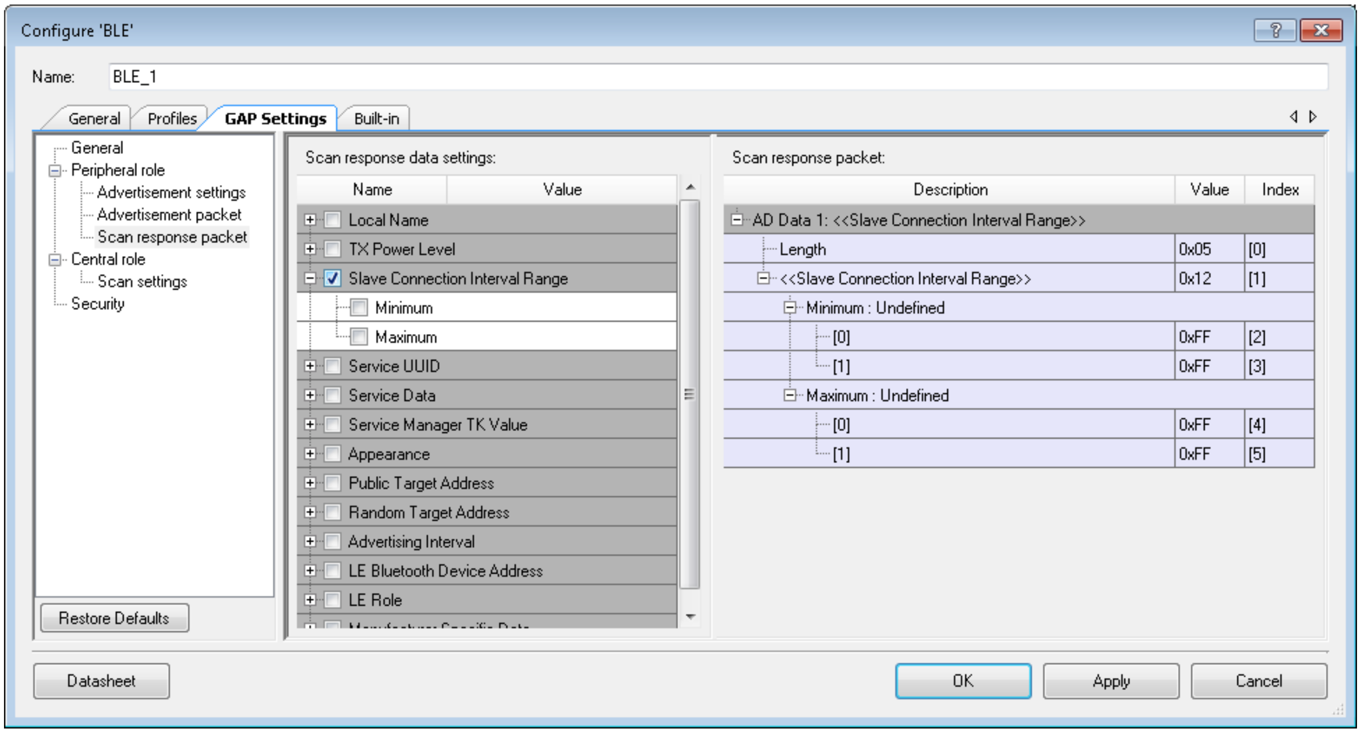
- AD Structure Element Definition:
  - **AD Length**: Size of **AD Type** and associated **AD Data** = 5 bytes
  - **AD Type** (1 byte): 0x03 (Service UUID)
  - **AD Data** (4 bytes): 0x180D, 0x180A (Heart Rate Service, Device Information Service)

The following table shows the **AD Types**.

AD Type	Description
Flags	Flags to broadcast underlying BLE transport capability such as Discoverable mode, LE only, etc.
Local Name	Device Name (complete of shortened). The device name value comes from the **Device name** field on the **GAP Settings** tab, under **General**.
Tx Power Level	Transmit Power Level. Taken from the **TX power level** field on the **GAP Settings** tab, under **General.**
Slave Connection Interval Range	Preferred connection interval range for the device.
Service UUID	List of Service UUIDs to be broadcasted that the device has implemented. There are different AD Type values to advertise 16-bit, 32-bit and 128-bit Service UUIDs. 16-bit and 32-bit Service UUIDs are used if they are assigned by the Bluetooth SIG.
Service Solicitation	List of Service UUIDs from the central device that the peripheral device would like to use. There are different AD Type values to advertise 16-bit, 32-bit and 128-bit Service UUIDs.
Service Data	2/4/16-byte Service UUID, followed by additional Service data.
Security Manager TK value	Temporal key to be used at the time of pairing.
Appearance	The external appearance of the device. The value comes from the **Appearance** field on the **GAP Settings** tab, under **General**.
Public Target Address	The public device address of intended recipients.
Random Target Address	The random device address of intended recipients.
Advertising Interval	The Advertising interval value that is calculated as an average of Fast advertising interval minimum and maximum values configured on the **GAP Settings** tab, under **Advertisement Settings**.
LE Bluetooth Device Address	The device address of the local device. The value comes from the **Public device address** field on the **GAP Settings** tab, under **General**.
LE Role	Supported LE roles
Manufacturer Specific Data	2 bytes company identifier followed by manufacturer specific data.

GAP Settings Tab – Scan response packet

This section displays when the device is configured to contain a "Peripheral," "Broadcaster," or "Peripheral and Central" **GAP role**. It is used to configure the Scan response data packet to be used in response to device scanning performed by a GATT Client device.



Name: BLE_1

Scan response data settings:

Name	Value
Local Name	
TX Power Level	
✓ Slave Connection Interval Range	
Minimum	
Maximum	
Service UUID	
Service Data	
Service Manager TK Value	
Appearance	
Public Target Address	
Random Target Address	
Advertising Interval	
LE Bluetooth Device Address	
LE Role	

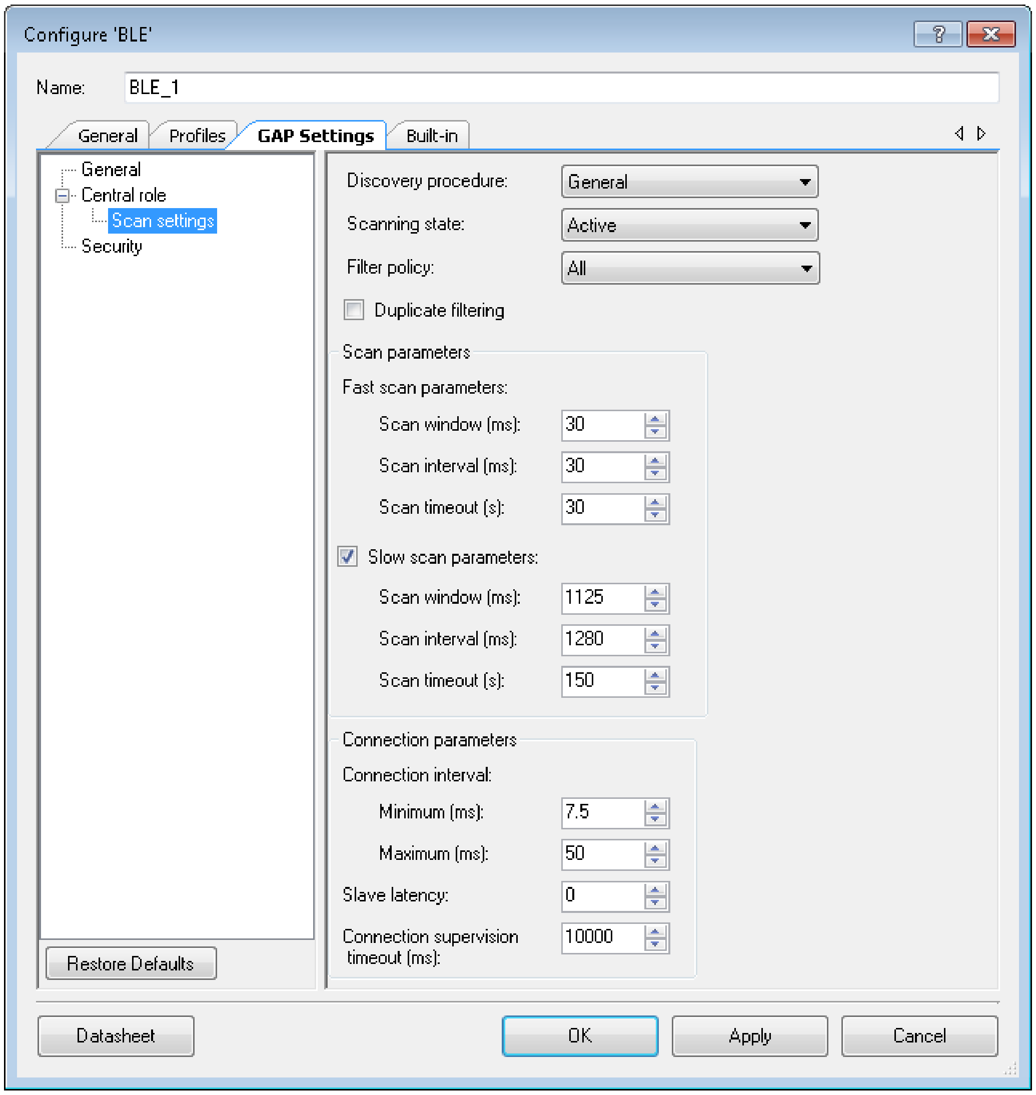
Scan response packet:

Description	Value	Index
AD Data 1: <<Slave Connection Interval Range>>		
Length	0x05	[0]
<<Slave Connection Interval Range>>	0x12	[1]
Minimum : Undefined		
[0]	0xFF	[2]
[1]	0xFF	[3]
Maximum : Undefined		
[0]	0xFF	[4]
[1]	0xFF	[5]

The packet structure of a Scan response packet is the same as an Advertisement packet. See Advertisement / Scan response data settings for information on configuring the Scan response packet.

GAP Settings Tab – Scan settings

These parameters are available when the device is configured as a "Central," "Observer," or "Peripheral and Central" **GAP role**. Typically during a device discovery, the GATT Client device initiates the scan procedure. It uses **Fast scan parameters** for a period of time, approximately 30 to 60 seconds, and then it reduces the scan frequency using the **Slow scan parameters**.



Name: BLE_1

Discovery procedure: General
Scanning state: Active
Filter policy: All
☐ Duplicate filtering

Scan parameters
Fast scan parameters:
Scan window (ms): 30
Scan interval (ms): 30
Scan timeout (s): 30
☑ Slow scan parameters:
Scan window (ms): 1125
Scan interval (ms): 1280
Scan timeout (s): 150

Connection parameters
Connection interval:
Minimum (ms): 7.5
Maximum (ms): 50
Slave latency: 0
Connection supervision timeout (ms): 10000

Note The scan interval needs to be aligned with the user-selected Profile specification.

Discovery procedure

- **Limited –** A device performing this procedure shall discover the device doing limited discovery mode advertising only.
- **General –** A device performing this procedure shall discover the devices doing general and limited discovery advertising.

Scanning state

- **Passive –** In this state a device can only listen to advertisement packets.
- **Active –** In this state a device may ask an advertiser for additional information.

Filter policy

This parameter defines how the advertisement packets are filtered.

- **All** – Process all advertisement packets.
- **White List Only** – Process advertisement packets only from devices in the White List.

Duplicate filtering

When enabled, this activates filtering of duplicated advertisement data. If disabled, the BLE stack will not perform filtering of advertisement data.

Scan parameters

These parameters define the scanning time and interval between scanning events. Two different sets of Scan parameters are used: **Fast scan parameters** and **Slow scan parameters**. Typically after the device initialization, a central device uses the Fast scan parameters. After the **Fast scan timeout** value expires, and if a connection with a Peripheral device is not established, then the Profile switches to Slow scan parameters to save the battery life. After the **Slow scan timeout** value expires, CYBLE_EVT_GAPC_SCAN_START_STOP event is generated. See API documentation.

- **Fast scan parameters** – This connection type results in a faster connection between the GATT Client and Server devices than it is possible using a normal connection.
  - **Scan Window**: This parameter defines the scan window when operating in **Fast connection**. The parameter is configured to increment in multiples of 0.625 ms. Valid range is from 2.5 ms to 10240 ms. **Scan Window** must be less than the **Scan Interval**. Default: 30 ms.
  - **Scan Interval**: This parameter defines the scan interval when operating in **Fast connection**. The parameter is configured to increment in multiples of 0.625 ms. Valid range is from 2.5 ms to 10240 ms. Default: 30 ms.
  - **Scan Timeout**: The timeout value of scanning with fast scan parameters. Default: 30 s.

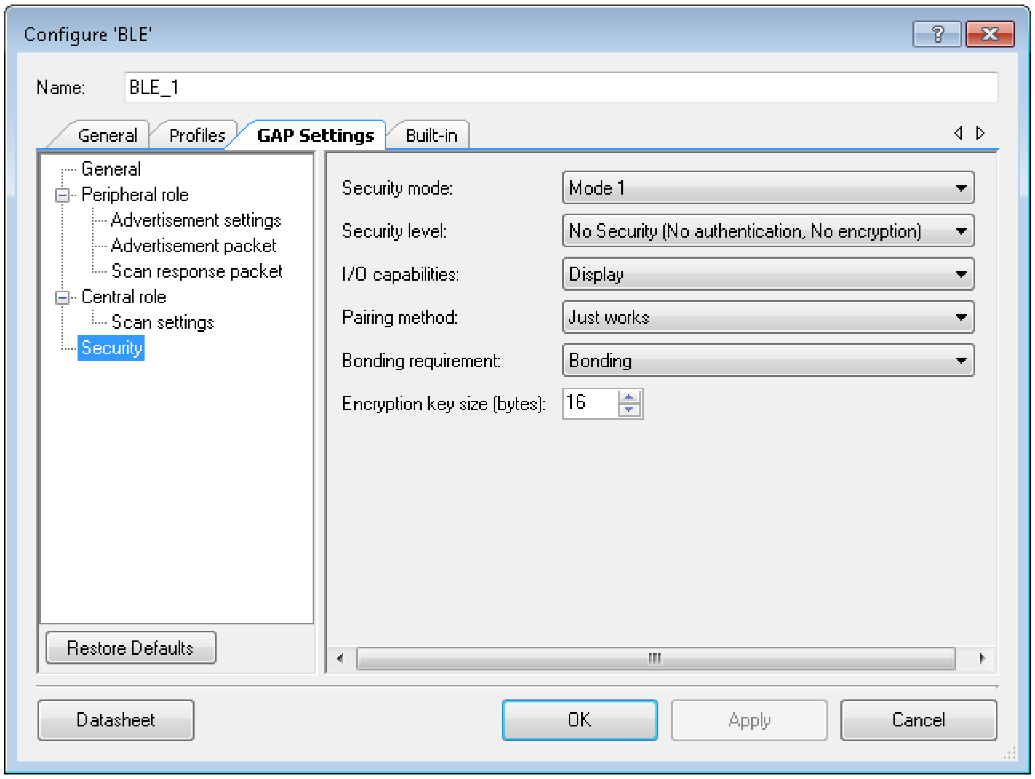
- **Slow scan parameters** – This connection results in a slower connection between the GATT Client and GATT Server devices than is possible using a normal connection. However this method consumes less power.
  - **Scan Window**: This parameter defines the scan window when operating in **Slow Connection**. The parameter is configured to increment in multiples of 0.625ms. Valid range is from 2.5 ms to 10240 ms. **Scan Window** must be less than the **Scan Interval**. Default: 11.25 ms.
  - **Scan Interval**: This parameter defines the scan interval when operating in **Slow Connection**. The parameter is configured to increment in multiples of 0.625 ms. Valid range is from 2.5 ms to 10240 ms. Default: 1280 ms.
  - **Scan Timeout**: The timeout value of scanning with slow scan parameters. Default: 150 s.

Connection Parameters

This section is the same as Connection Parameters for Advertisement Settings. The only difference is that Scan connection parameters will not be shown on the **Peripheral Preferred Connection** parameters on the **Profile** tab.

GAP Settings Tab – Security

This section contains several parameters to configure the global security options for the Component. If the device is configured as a GATT Server, you can optionally set each Characteristic using its own unique security setting in the **Profile Tree**.



Security mode: Mode 1
Security level: No Security (No authentication, No encryption)
I/O capabilities: Display
Pairing method: Just works
Bonding requirement: Bonding
Encryption key size (bytes): 16

Security mode

Defines GAP security modes for the Component. Both available modes may support authentication.

- Mode 1 – Used in designs where data encryption is required.
- Mode 2 – Used in designs where data signing is required.

Security level

Enables different levels of security depending on the selected **Security mode**:

- If Mode1 is selected, then the following security levels are available.
  - No Security – With this level of security, the device will not use encryption or authentication.
  - Unauthenticated pairing with encryption – With this level of security, the device will send encrypted data after establishing a connection with the remote device.

    - Authenticated pairing with encryption – With this level of security, the device will send encrypted data after establishing a connection with the remote device. To establish a connection, devices should perform the authenticated paring procedure.

- If Mode 2 is selected, then the following security levels are available.
    - Unauthenticated pairing with data signing – With this level of security, the device will perform data signing prior to sending it to the remote device after they establish a connection.
    - Authenticated pairing with data signing – With this level of security, the device will perform data signing prior to sending it to the remote device after they establish a connection. To establish a connection, the devices should perform the authenticated paring procedure.

I/O capabilities

This parameter refers to the device's input and output capability that can enable or restrict a particular pairing method or security level.

- No Input No Output – Used in devices that don't have any capability to enter and display the authentication key data. Used in mouse-like devices. No GAP authentication is required.
- Display Only – Used in devices with display capability and may display authentication key data. GAP authentication is required.
- Keyboard Only – Used in devices with numeric keypad. GAP authentication is required.
- Display Yes/No – Used in devices with display and at least two input keys for Yes/No action. GAP authentication is required.
- Keyboard and Display – Used in devices like PCs and tablets. GAP authentication is required.

Pairing Method

This parameter is used to explicitly configure the pairing method for the Component.

- Just Works – The device will use the simple paring procedure without authentication. With this method, the transferred data would be vulnerable to "man in the middle" attacks.
- Passkey Entry – This uses six numeric digits generated for a Short Term Key (STK) passed by the user between the devices.
- OOB (Out of Band) pairing – Uses an external means of communication to exchange the device pairing information. Pairing itself is performed using the BLE radio.

Bonding Requirement:

This parameter is used to configure the bonding requirements. The purpose of bonding is to create a relation between two Bluetooth devices based on a common link key (a bond). The link key is created and exchanged (pairing) during the bonding procedure and is expected to be stored by both Bluetooth devices, to be used for future authentication.

- **Bonding:** The device will store the link key of a connection after paring with the remote device and if a connection will be lost and re-established, the devices will use the previously stored key for the connection.

  Note Boding information is stored in RAM and should be written to Flash if it needs to be retained during shutdown. Refer to the Functional Description section for details on bonding and Flash write usage.

- **No Bonding:** The pairing process will be performed on each connection establishment.

Encryption Key Size

This parameter defines the encryption key size based on the Profile requirement. The valid values of encryption key size are 7 to 16 bytes.

BLE Component APIs

The BLE Component contains a comprehensive API list to allow you to configure the BLE stack, the underlying chip hardware and the BLE service specific configuration using software. You may access the GAP, GATT and L2CAP layers of the stack using these.

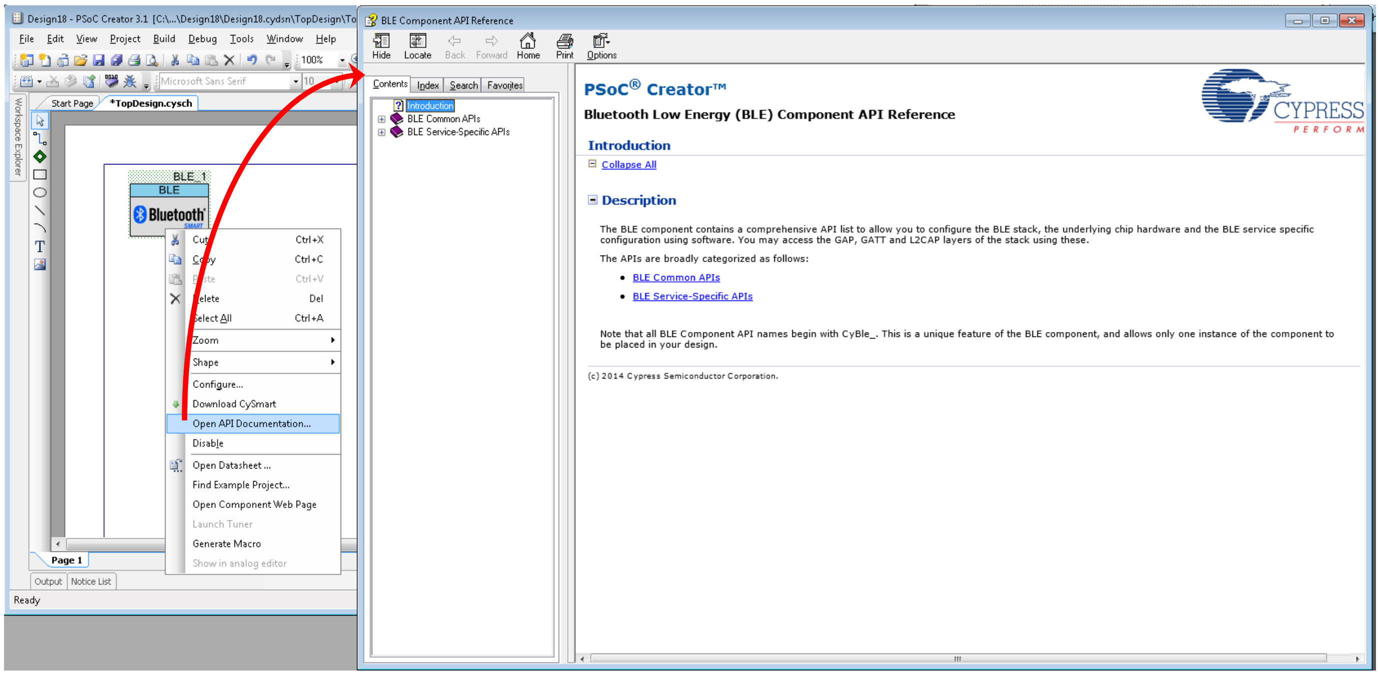
The APIs are broadly categorized as follows:

- BLE Common APIs
- BLE Service-Specific APIs

Note: All BLE Component API names begin with CyBle_. This is a unique feature of the BLE Component, and allows only one instance of the Component to be placed in your design.

HTML-Based API Document

Because the BLE Component has numerous APIs, Cypress has also provided a separate HTML-based API reference document (CHM file). To open this file, right-click on the BLE Component on the design canvas, and select **Open API Documentation…**



PSoC® Creator™

Bluetooth Low Energy (BLE) Component API Reference

Introduction

Collapse All

Description

The BLE component contains a comprehensive API list to allow you to configure the BLE stack, the underlying chip hardware and the BLE service specific configuration using software. You may access the GAP, GATT and L2CAP layers of the stack using these.

The APIs are broadly categorized as follows:

- BLE Common APIs
- BLE Service-Specific APIs

Note that all BLE Component API names begin with CyBle_. This is a unique feature of the BLE component, and allows only one instance of the component to be placed in your design.

(c) 2014 Cypress Semiconductor Corporation.

Sample Firmware Source Code

PSoC Creator provides numerous example projects that include schematics and example code in the Find Example Project dialog. For Component-specific examples, open the dialog from the Component Catalog or an instance of the Component in a schematic. For general examples, open the dialog from the Start Page or **File** menu. As needed, use the **Filter Options** in the dialog to narrow the list of projects available to select.

Refer to the "Find Example Project" topic in the PSoC Creator Help for more information.

MISRA Compliance

This section describes the MISRA-C:2004 compliance and deviations for the Component. There are three types of deviations defined:

- project deviations – deviations that are applicable for all PSoC Creator Components
- Component specific deviations – deviations that are applicable only for the common part of this Component
- Profile specific deviations – deviations that are applicable only for a specific Profile of the Component

This section provides information on Component-specific deviations. Project deviations are described in the MISRA Compliance section of the *System Reference Guide* along with information on the MISRA compliance verification environment.

The BLE Component has the following specific deviations.

MISRA-C:2004 Rule	Rule Class (Required/ Advisory)	Rule Description	Description of Deviation(s)
9.3	R	In an enumerator list, the '=' construct shall not be used to explicitly initialize members other than the first, unless all items are explicitly initialized.	Violated when a specific value needs to be assigned to an enumerator item.
10.1	R	The value of an expression of integer type shall not be implicitly converted to a different underlying type under some circumstances.	An operand of essentially enum type is being converted to unsigned type as a result of an arithmetic or conditional operation. The conversion does not have any unintended effect.
11.4	A	A cast should not be performed between a pointer to object type and a different pointer to object type.	A cast involving pointers is conducted with caution that the pointers are correctly aligned for the type of object being pointed to.
13.7	R	Boolean operations whose results are invariant shall not be permitted.	A Boolean operator can yields a result that can be proven to be always "true" or always "false" in some specific configurations because of generalized implementation approach.
17.4	R	Array indexing shall be the only allowed form of pointer arithmetic.	An array subscript operator is being used to subscript an expression which is not of array type. This is perfectly legitimate in the C language providing the pointer addresses an array element.
18.4	R	Unions shall not be used.	Deviated for constructing an efficient implementation.
19.7	A	A function should be used in preference to a function-like macro.	Deviated for more efficient code.

This Component has the following embedded Components: cy_isr, SCB. Refer to the corresponding Component datasheets for information on their MISRA compliance and specific deviations.

API Memory Usage

The Component memory usage varies significantly, depending on the compiler, device, number of APIs used and Component configuration. The following table provides the memory usage for all APIs available in the given Component configuration.

The measurements are done with the associated compiler configured in Release mode with optimization set for Size. For a specific design, the map file generated by the compiler can be analyzed to determine the memory usage.

The Component's BLE Stack is implemented in four libraries and therefore the Component memory usage is directly dependent on the library used. The libraries are:

- HCI Library (used in HCI mode)
- Peripheral (used when the Component is configured for GAP Peripheral or GAP Broadcaster role)
- Central (used when the Component is configured for GAP Central or GAP Observer role)
- Peripheral and Central (used when the Component is configured for GAP Peripheral and Central roles)

HCI Mode

Configuration	PSoC 4200-BL (GCC)		
	Flash Bytes	SRAM Bytes	Stack Bytes
HCI Mode	36444	5805	2048

Peripheral and Central Profile Mode

Configuration	PSoC 4200-BL (GCC)		
	Flash Bytes	SRAM Bytes	Stack Bytes
Alert Notification Profile (Server)	79262	9256	2048
Find Me Profile (Find Me Target role)	78668	9241	2048
Phone Alert Status	79274	9249	2048
Time	79808	9279	2048

Central Profile Mode

Configuration	PSoC 4200-BL (GCC)		
	Flash Bytes	SRAM Bytes	Stack Bytes
Alert Notification Profile (Server)	72594	9151	2048
Find Me Profile (Find Me Target role)	72012	9130	2048
HID over GATT Profile (Host)	77754	9338	2048
Phone Alert Status	72474	9136	2048
Proximity Profile (Proximity Reporter)	72778	9140	2048
Time	73008	9166	2048

Peripheral Profile Mode

Configuration	PSoC 4200-BL (GCC)		
	Flash Bytes	SRAM Bytes	Stack Bytes
Blood Pressure	71254	9137	2048
Cycling Power	71620	9136	2048
Cycling Speed and Cadence	71356	9119	2048
Find Me Profile (Find Me Target role)	70022	9084	2048
Glucose Profile (Glucose Sensor)	71404	9130	2048
Health Thermometer Profile (Server)	71520	9126	2048
Heart Rate Profile (Heart Rate Sensor)	71038	9105	2048
HID Over GATT Profile (HID Device)	73012	9157	2048
Location and Navigation	71006	9115	2048
Proximity Profile (Proximity Reporter)	71060	9096	2048
Running Speed and Cadence	71358	9122	2048
Scan Parameters Profile (Scan Server)	70534	9090	2048

BLE Common APIs

The common APIs act as a general interface between the BLE application and the BLE Stack module. The application may use these APIs to control the underlying hardware such as radio power, data encryption and device bonding via the stack. It may also access the GAP, GATT and L2CAP layers of the stack. These are divided into the following categories:

- BLE Common Core Functions
- GAP Functions
- GATT Functions
- L2CAP Functions

These APIs also use API specific definitions and data structures. Many of the APIs also rely on BLE Stack events. These are classified in the following subsets:

- BLE Common Events
- BLE Common Definitions and Data Structures

BLE Common Core Functions

The common core APIs are used for general BLE Component configuration. These include initialization, power management, and utilities.

Functions

Function	Description
CyBle_Start	This function initializes the BLE Stack. It takes care of initializing the... more
CyBle_Stop	This function stops any ongoing operation in the BLE Stack and forces the BLE Stack to shut down. The only function that can be called... more
CyBle_GetBleSsState	This function gets the BLE Subsystem's current operational mode. This state can be used to manage system level power modes based on return value.
CyBle_StoreAppData	This function instructs the Stack to backup application specific data into flash. This API must be called by application to backup application specific data. If... more
CyBle_StoreBondingData	This function writes the new bonding data from RAM to the dedicated Flash location as defined by the Component. It performs data comparing between RAM... more
CyBle_StoreStackData	This function instructs Stack to backup Stack internal RAM data into flash. This API must be called by application to backup stack data. If this... more
CyBle_SoftReset	This function resets the BLE Stack, including BLE sub-system hardware registers. BLE Stack transitions to idle mode. This function can be used to reset the... more
CyBle_EnterLPM	This function requests the underlying BLE modules to enter into one of the supported... more

Function	Description
CyBle_ExitLPM	Application can asynchronously wake up the BLE Stack from low power using this function. The wake up is not performed for the entire chip. This... more
CyBle_ProcessEvents	This function checks the internal task queue in the BLE Stack, and pending operation of the BLE Stack, if any. This needs to be called... more
CyBle_GetDeviceAddress	This API reads the BD device address from BLE Controller's memory. This address shall be used for BLE procedures unless explicitly indicated by BLE Host... more
CyBle_SetDeviceAddress	This function sets the Bluetooth device address into BLE Controller's memory. This address shall be used for BLE procedures unless explicitly indicated by BLE Host... more
CyBle_GetRssi	This function reads the recorded Received Signal Strength Indicator (RSSI) value for the last successfully received packet from the BLE radio sub-system. This is a... more
CyBle_GetTxPowerLevel	This function reads the transmit power of the BLE radio for the given BLE sub-system channel group. This is a blocking function. No event is... more
CyBle_SetTxPowerLevel	This function sets the transmit power of the BLE radio for given BLE sub-system channel group. This is a blocking function. No event is generated... more
CyBle_GetBleClockCfgParam	This function reads the clock configuration parameter of BLE sub-system. This is a blocking function. No event is generated on calling this function. The following... more
CyBle_SetBleClockCfgParam	This function sets the clock configuration parameter of BLE sub-system. This is a blocking function. No event is generated on calling this function. The following... more
CyBle_GenerateRandomNumber	This function generates 8-byte random number which complies with pseudo random number generation in accordance with [FIPS PUB 140-2]. Random number generation function is used... more
CyBle_AesEncrypt	This function uses BLE sub-system AES engine to encrypt 128-bit of plain text using the given AES key. The output of AES processing is copied... more
CyBle_SetCeLengthParam	This function sets the connection event duration related parameters that can result in extension or truncation of LE connection event based on more data (mdBit)... more
CyBle_WriteAuthPayloadTimeout	This function sets the Authentication Payload timeout in BLE Controller for LE_PING feature. Refer Bluetooth 4.1 core specification, Volume 6, Part B, section 4.6.5 for... more
CyBle_ReadAuthPayloadTimeout	This function reads the Authentication Payload timeout set in BLE Controller for LE_PING feature Refer Bluetooth 4.1 core specification, Volume 6, Part B, section 4.6.5... more
CyBle_GetStackLibraryVersion	This function retrieves the version information of the BLE Stack library. This is a blocking function. No event is generated on calling this function
CyBle_SetRxGainMode	This function configures the Rx gain mode for BLESS radio for Rx operation.
CyBle_SetTxGainMode	This function configures the Tx gain mode for BLESS radio for Tx operation.

Macros

Macro	Description
CyBle_GetState	This function is used to determine the current state of the Event Handler state machine.
CyBle_SetState	Used to set the Event Handler State Machine's state.

CyBle_Start

Prototype

```
CYBLE_API_RESULT_T CyBle_Start(CYBLE_CALLBACK_T callbackFunc);
```

Description

This function initializes the BLE Stack. It takes care of initializing the Profile layer, schedulers, Timer and other platform related resources required for the BLE Component. It also registers the callback function for BLE events that will be registered in the BLE stack.

Note that this function does not reset the BLE Stack.

For HCI-Mode of operation, this function will not initialize the BLE Host module.

Calling this function results in the generation of CYBLE_EVT_STACK_ON event on successful initialization of the BLE Stack.

Parameters

Parameters	Description
CYBLE_CALLBACK_T callbackFunc	Event callback function to receive events from BLE stack. CYBLE_CALLBACK_T is a function pointer type.

Returns

CYBLE_API_RESULT_T : Return value indicates if the function succeeded or failed. Following are the possible error codes.

Error codes	Description
CYBLE_ERROR_OK	On successful operation.
CYBLE_ERROR_INVALID_PARAMETER	On passing a NULL pointer to the function when the BLE stack is not built in HCI mode. CYBLE_ERROR_INVALID_PARAMETER is never returned in HCI mode.
CYBLE_ERROR_REPEATED_ATTEMPTS	On invoking this function more than once without calling CyBle_Shutdown() function between calls to this function.

CyBle_Stop

Prototype

```
void CyBle_Stop(void);
```

Description

This function stops any ongoing operation in the BLE Stack and forces the BLE Stack to shut down. The only function that can be called after calling this function is CyBle_Start().

Returns

None

CyBle_GetState

Prototype

```
#define CyBle_GetState (cyBle_state)
```

Description

This function is used to determine the current state of the Event Handler state machine.

Returns

CYBLE_STATE_T state - The current state.

CyBle_GetBleSsState

Prototype

```
CYBLE_BLESS_STATE_T CyBle_GetBleSsState(void);
```

Description

This function gets the BLE Subsystem's current operational mode. This state can be used to manage system level power modes based on return value.

Returns

CYBLE_BLESS_STATE_T bleStackMode: CYBLE_BLESS_STATE_T has one of the following modes

BLE Stack Mode	Description
CYBLE_BLESS_STATE_ACTIVE	BLE Sub System is in active mode, CPU can be in active mode or sleep mode.
CYBLE_BLESS_STATE_EVENT_CLOSE	BLE Sub System radio and Link Layer hardware finishes Tx/Rx. After this state application can try putting BLE to Deep Sleep State to save

	power in rest of the BLE transmission event.
CYBLE_BLESS_STATE_SLEEP	BLE Sub System is in sleep mode, CPU can be in sleep mode.
CYBLE_BLESS_STATE_ECO_ON	BLE Sub System is in process of wakeup from Deep Sleep Mode and ECO(XTAL) is turned on. CPU can be put in Deep Sleep Mode.
CYBLE_BLESS_STATE_ECO_STABLE	BLE Sub System is in process of wakeup from Deep Sleep Mode and ECO(XTAL) is stable. CPU can be put in sleep mode.
CYBLE_BLESS_STATE_DEEPSLEEP	BLE Sub System is in Deep Sleep Mode. CPU can be put in Deep Sleep Mode.
CYBLE_BLESS_STATE_HIBERNATE	BLE Sub System is in Hibernate Mode. CPU can be put in Deep Sleep Mode.

CyBle_SetState

Prototype

```
#define CyBle_SetState(state) (cyBle_state = (state))
```

Description

Used to set the Event Handler State Machine's state.

Parameters

Parameters	Description
state	The desired state that the event handler's state machine should be set to.

Returns

None

CyBle_StoreAppData

Prototype

```
CYBLE_API_RESULT_T CyBle_StoreAppData(uint8 * srcBuff, const uint8 destAddr[],
uint32 buffLen, uint8 isForceWrite);
```

Description

This function instructs the Stack to backup application specific data into flash. This API must be called by application to backup application specific data. If this API is not called appropriately, data will not be available on power cycle.

Parameters

Parameters	Description
uint8 * srcBuff	Source buffer
const uint8 destAddr[]	Destination address
uint32 buffLen	Length of srcData
uint8 isForceWrite	If value is set to 0, then stack will check if flash write is permissible.

Returns

CYBLE_API_RESULT_T : Return value indicates if the function succeeded or failed. Following are the possible error codes.

Error codes	Description
CYBLE_ERROR_OK	On successful operation
CYBLE_ERROR_FLASH_WRITE_NOT_PERMITED	Flash Write is not permitted

CyBle_StoreBondingData

Prototype

```
CYBLE_API_RESULT_T CyBle_StoreBondingData(uint8 isForceWrite);
```

Description

This function writes the new bonding data from RAM to the dedicated Flash location as defined by the Component. It performs data comparing between RAM and Flash before writing to Flash. If there is no change between RAM and Flash data, then no write is performed. It writes only one flash row in one call. Application should keep calling this function till API return CYBLE_ERROR_OK. This function is available only when Bonding requirement is selected in Security settings.

Parameters

Parameters	Description
uint8 isForceWrite	If value is set to 0, then stack will check if flash write is permissible.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - On successful operation
- CYBLE_ERROR_FLASH_WRITE_NOT_PERMITED - Flash Write is not complete

Side Effects

This API will automatically modify the clock settings for the device. Writing to flash requires changes to be done to the IMO (set to 48 MHz) and HFCLK (source set to IMO) settings. The configuration is restored before returning. This will impact the operation of most of the hardware in the device.

CyBle_StoreStackData

Prototype

```
CYBLE_API_RESULT_T CyBle_StoreStackData(uint8 isForceWrite);
```

Description

This function instructs Stack to backup Stack internal RAM data into flash. This API must be called by application to backup stack data. If this API is not called appropriately, stack internal data structure will not be available on power cycle.

Parameters

Parameters	Description
uint8 isForceWrite	If value is set to 0, then stack will check if flash write is permissible.

Returns

CYBLE_API_RESULT_T : Return value indicates if the function succeeded or failed. Following are the possible error codes.

Error codes	Description
CYBLE_ERROR_OK	On successful operation
CYBLE_ERROR_FLASH_WRITE_NOT_PERMITED	Flash Write is not permitted or not completely written

CyBle_SoftReset

Prototype

```
CYBLE_API_RESULT_T CyBle_SoftReset(void);
```

Description

This function resets the BLE Stack, including BLE sub-system hardware registers. BLE Stack transitions to idle mode. This function can be used to reset the BLE Stack if the BLE Stack turns unresponsive due to incomplete transfers with the peer BLE device.

This is a blocking function. No event is generated on calling this function.

Returns

CYBLE_API_RESULT_T : Return value indicates if the function succeeded or failed. Following are the possible error codes.

Error codes	Description
CYBLE_ERROR_OK	On successful operation
CYBLE_ERROR_INVALID_OPERATION	This error occurs if this function is invoked before invoking CyBle_StackInit function.

CyBle_EnterLPM

Prototype

```
CYBLE_LP_MODE_T CyBle_EnterLPM(CYBLE_LP_MODE_T pwrMode);
```

Description

This function requests the underlying BLE modules to enter into one of the supported low power modes. Application should use this function to put Bluetooth Low Energy Sub-System (BLESS) to Low Power Mode (LPM).

BLE Stack enters and exits low power modes based on its current state and hence the application should consider the BLE Stack LPM state before putting the CPU or the overall device into LPM. This function attempts to set the requested low power mode and if that is not possible, it tries to set the next higher low-power-mode. This behavior is due to the requirement that the application will always try to use the lowest power mode when there is nothing that it needs to process. Note that the CPU will not be able to access the BLESS registers when BLESS is in Deep Sleep Mode.

BLE Stack has the following power modes:

- Active
- Sleep (Low Power Mode)
- Deep Sleep with ECO Off (Low Power Mode)
- Hibernate (Low Power Mode)

Note that certain conditions may prevent BLE sub system from entering a particular low power mode.

Active Mode

Bluetooth Low Energy Sub System (BLESS) has three sub-modes in Active mode:

- Idle

- Transmit Mode, and
- Receive Mode

These modes draw full current from the device and the CPU has full access to its registers.

Sleep Mode

The clock to the link layer engine and digital modem is gated and the (External Crystal Oscillator) ECO continues to run to maintain the link layer timing. The application cannot enter sleep mode if a Transmit or Receive is in progress.

Deep Sleep with ECO Off Mode

The ECO is stopped and Watch Crystal Oscillator (WCO) is used to maintain link layer timing. All the regulators in the Radio Frequency (RF) transceiver are turned off to reduce leakage current and BLESS logic is kept powered ON from the System Resources Sub System (SRSS) Deep Sleep regulator for retention of current BLESS state information. This mode can be entered from either Idle (Active) or Sleep mode. It should be entered when the next scheduled activity instant in time domain is greater than the Deep Sleep total wakeup time (typically 2ms).

Hibernate mode

The application layer should invoke this function with the Hibernate Mode option to put the BLE Stack in to hibernate mode. If this mode is set, the micro-controller can be put in to Hibernate

Mode by the application layer. This mode ensures that BLE Sub-system is completely idle and no procedures such ADV, SCAN and CONNECTION are active.

The following table indicates the allowed sleep modes for the complete system (BLE Sub-system and the micro-controller). Modes marked In 'X' are the allowed combinations. The application layer should make sure that the invalid modes are not entered in to:

BLE Stack LPM Modes	PSoC4A-BLE Micro-controller Low Power Modes			
	Active	Sleep	DeepSleep	Hibernate
Active	X			
Sleep	X	X		
DeepSleep (ECO OFF)	X	X	X	
Hibernate				X

The application layer is responsible for putting the BLE Sub-system and the micro-controller in to the desired sleep modes. Upon entering the requested sleep mode combination, the BLE Sub-system and the micro-controller are woken up by an interrupt every advertisement interval(in case of a GAP Peripheral) or connection interval (in case of GAP Central). On wakeup, if the application needs to transmit some data, appropriate function(s) including the Stack functions need to be invoked. This needs to be followed by a call to the function CyBle_ProcessEvents, which handles all pending transmit and receive operations. The application can now put the complete system back in to one of the sleep modes. The application should ensure that the above invalid states are never encountered.

This is a blocking function. No event is generated on calling this function. Based on the return code from this function, the application layer should decide on the sleep mode for the complete system. For example, if the return code is CYBLE_BLESS_DEEPSLEEP, the application can choose to call system wide Deep Sleep Mode function.

Parameters

Parameters	Description
CYBLE_LP_MODE_T pwrMode	The power mode that the Component is intended to enter. The allowed values are, • CYBLE_BLESS_SLEEP • CYBLE_BLESS_DEEPSLEEP

Returns

CYBLE_LP_MODE_T: The actual power mode that BLE stack is now set to.

CyBle_ExitLPM

Prototype

```
CYBLE_LP_MODE_T CyBle_ExitLPM(void);
```

Description

Application can asynchronously wake up the BLE Stack from low power using this function. The wake up is not performed for the entire chip. This is a blocking call and returns when BLE Stack has come out of LPM. No event is generated on calling this function. It has no effect if it is invoked when the BLE Stack is already in active mode.

Returns

CYBLE_LP_MODE_T: The actual power mode that BLE stack is now set to. Expected return value is CYBLE_BLESS_ACTIVE.

CyBle_ProcessEvents

Prototype

```
void CyBle_ProcessEvents(void);
```

Description

This function checks the internal task queue in the BLE Stack, and pending operation of the BLE Stack, if any. This needs to be called at least once every interval 't' where:

- 't' is equal to connection interval or scan interval, whichever is smaller, if the device is in GAP Central mode of operation, or
- 't' is equal to connection interval or advertisement interval, whichever is smaller, if the device is in GAP Peripheral mode of operation.

On calling every interval 't', all pending operations of the BLE Stack are processed. This is a blocking function and returns only after processing all pending events of the BLE Stack Care should be taken to prevent this call from any kind of starvation; on starvation, events may be dropped by the stack. All the events generated will be propagated to higher layers of the BLE Stack and to the Application layer only after making a call to this function.

Returns

None

CyBle_GetDeviceAddress

Prototype

```
CYBLE_API_RESULT_T CyBle_GetDeviceAddress(CYBLE_GAP_BD_ADDR_T* bdAddr);
```

Description

This API reads the BD device address from BLE Controller's memory. This address shall be used for BLE procedures unless explicitly indicated by BLE Host through HCI commands. This is a blocking function and it returns immediately with the required value.

Parameters

Parameters	Description
CYBLE_GAP_BD_ADDR_T* bdAddr	Pointer to the CYBLE_GAP_BD_ADDR_T structure variable. It has two fields where, • bdAddr.addr: Bluetooth Device address buffer that is populated with the device address data from BLE stack. • bdAddr.type: Caller function should fill the "address type" to retrieve appropriate address. Caller function should use bdAddr.type = 0x00 to get the "Public Device Address" which is currently set. Caller function use bdAddr.type = 0x01 to get the "Random Device Address" which is currently set.

Returns

CYBLE_API_RESULT_T : Return value indicates if the function succeeded or failed. Following are the possible error codes.

Error codes	Description
CYBLE_ERROR_OK	On successful operation.
CYBLE_ERROR_INVALID_PARAMETER	On specifying NULL as input parameter.

CyBle_SetDeviceAddress

Prototype

```
CYBLE_API_RESULT_T CyBle_SetDeviceAddress(CYBLE_GAP_BD_ADDR_T* bdAddr);
```

Description

This function sets the Bluetooth device address into BLE Controller's memory. This address shall be used for BLE procedures unless explicitly indicated by BLE Host through HCI commands. The application layer needs to call this function every time an address change is required. Bluetooth 4.1 Core specification [3.12] specifies that the bluetooth device can change

its private address periodically, with the period being decided by the application; there are no limits specified on this period. The application layer should maintain its own timers in order to do this.

This is a blocking function. No event is generated on calling this function.

Parameters

Parameters	Description
CYBLE_GAP_BD_ADDR_T* bdAddr	Bluetooth Device address retrieved from the BLE stack gets stored to a variable pointed to by this pointer. The variable is of type CYBLE_GAP_BD_ADDR_T.

Returns

CYBLE_API_RESULT_T : Return value indicates if the function succeeded or failed. Following are the possible error codes.

Error codes	Description
CYBLE_ERROR_OK	On successful operation.
CYBLE_ERROR_INVALID_PARAMETER	On specifying NULL as input parameter.

CyBle_GetRssi

Prototype

```
int8 CyBle_GetRssi(void);
```

Description

This function reads the recorded Received Signal Strength Indicator (RSSI) value for the last successfully received packet from the BLE radio sub-system. This is a blocking function. No event is generated on calling this function.

Returns

int8: The RSSI value of the responding device.

Information	Description
Range	-85 <= N <= 5
Note	The value is in dBm.

CyBle_GetTxPowerLevel

Prototype

```
CYBLE_API_RESULT_T CyBle_GetTxPowerLevel(CYBLE_BLESS_PWR_IN_DB_T * bleSsPwrLvl);
```

Description

This function reads the transmit power of the BLE radio for the given BLE sub-system channel group. This is a blocking function. No event is generated on calling this function.

Parameters

Parameters	Description
CYBLE_BLESS_PWR_IN_DB_T * bleSsPwrLvl	Pointer to a variable of type CYBLE_BLESS_PWR_IN_DB_T where, • bleSsPwrLvl -> blePwrLevelInDbm indicates Output Power level in dBm returned by the function. • bleSsPwrLvl -> bleSsChId indicates Channel group for which power level is to be read. This needs to be set before calling the function. The value can be advertisement channels (CYBLE_LL_ADV_CH_TYPE) or data channels (CYBLE_LL_CONN_CH_TYPE).

Returns

CYBLE_API_RESULT_T: Return value indicates if the function succeeded or failed. Following are the possible error codes.

Error codes	Description
CYBLE_ERROR_OK	On successful operation
CYBLE_ERROR_INVALID_PARAMETER	On specifying NULL as input parameter

CyBle_SetTxPowerLevel

Prototype

```
CYBLE_API_RESULT_T CyBle_SetTxPowerLevel(CYBLE_BLESS_PWR_IN_DB_T * bleSsPwrLvl);
```

Description

This function sets the transmit power of the BLE radio for given BLE sub-system channel group. This is a blocking function. No event is generated on calling this function.

Parameters

Parameters	Description
CYBLE_BLESS_PWR_IN_DB_T * bleSsPwrLvl	Pointer to a variable of type 'CYBLE_BLESS_PWR_IN_DB_T' where, • bleSsPwrLvl -> blePwrLevelInDbm indicates Output Power level in dBm to be set by the function. • bleSsPwrLvl -> bleSsChId indicates Channel group for which power level is to be set. The value can be advertisement channels (CYBLE_LL_ADV_CH_TYPE) or data channels (CYBLE_LL_CONN_CH_TYPE).

Returns

CYBLE_API_RESULT_T : Return value indicates if the function succeeded or failed. Following are the possible error codes.

Error codes	Description
CYBLE_ERROR_OK	On successful operation.
CYBLE_ERROR_INVALID_PARAMETER	On specifying NULL as input parameter.

CyBle_GetBleClockCfgParam

Prototype

```
CYBLE_API_RESULT_T CyBle_GetBleClockCfgParam(CYBLE_BLESS_CLK_CFG_PARAMS_T *
bleSsClockConfig);
```

Description

This function reads the clock configuration parameter of BLE sub-system. This is a blocking function. No event is generated on calling this function. The following parameters related to the BLE sub-system clock are set by this function:

Sleep Clock accuracy

Sleep clock accuracy (SCA)in PPM. This parameter indicates the sleep clock accuracy in PPM as described in the following table. It is set in the BLE Stack and is used for BLE Connection operation while creating LE connection with the peer device.

Sleep Clock Accuracy Enum Field	PPM Range Translation (PPM)
CYBLE_LL_SCA_251_TO_500_PPM	251 - 500
CYBLE_LL_SCA_151_TO_250_PPM	151 – 250
CYBLE_LL_SCA_101_TO_150_PPM	101 - 150
CYBLE_LL_SCA_076_TO_100_PPM	76 - 100
CYBLE_LL_SCA_051_TO_075_PPM	51 - 75
CYBLE_LL_SCA_031_TO_050_PPM	31 - 50
CYBLE_LL_SCA_021_TO_030_PPM	21 - 30
CYBLE_LL_SCA_000_TO_020_PPM	0 - 20

Refer to Bluetooth Core Specification 4.1 Volume 6, Chapter 4.5.7 for more details on how the SCA is used.

Link Layer clock divider

This input decides the frequency of the clock to the link layer. A lower clock frequency results in lower power consumption. Default clock frequency for the operation is 24MHz. BLESS supports 24MHz, 12MHz and 8MHz clock configurations. Based on the end application requirement (how frequent the communication is expected to be), this parameter needs to be set.

Parameters

Parameters	Description
CYBLE_BLESS_CLK_CFG_PARAMS_T * bleSsClockConfig	Pointer to a variable of type CYBLE_BLESS_CLK_CFG_PARAMS_T to which the existing clock configuration is stored.

Returns

CYBLE_API_RESULT_T : Return value indicates if the function succeeded or failed. Following are the possible error codes.

Error codes	Description
CYBLE_ERROR_OK	On successful operation.
CYBLE_ERROR_INVALID_PARAMETER	On specifying NULL as input parameter.

CyBle_SetBleClockCfgParam

Prototype

```
CYBLE_API_RESULT_T CyBle_SetBleClockCfgParam(CYBLE_BLESS_CLK_CFG_PARAMS_T *
bleSsClockConfig);
```

Description

This function sets the clock configuration parameter of BLE sub-system. This is a blocking function. No event is generated on calling this function. The following parameters related to the BLE sub-system clock are set by this function:

Sleep Clock accuracy

Sleep clock accuracy (SCA) in PPM. This parameter indicates the sleep clock accuracy in PPM as described in the following table. It is set in the BLE Stack and is used for BLE Connection operation while creating LE connection with the peer device.

Sleep Clock Accuracy Enum Field	PPM Range Translation (PPM)
CYBLE_LL_SCA_251_TO_500_PPM	251 - 500
CYBLE_LL_SCA_151_TO_250_PPM	151 - 250
CYBLE_LL_SCA_101_TO_150_PPM	101 - 150
CYBLE_LL_SCA_076_TO_100_PPM	76 - 100
CYBLE_LL_SCA_051_TO_075_PPM	51 - 75
CYBLE_LL_SCA_031_TO_050_PPM	31 - 50
CYBLE_LL_SCA_021_TO_030_PPM	21 - 30
CYBLE_LL_SCA_000_TO_020_PPM	0 - 20

Refer to Bluetooth Core Specification 4.1 Volume 6, Chapter 4.5.7 for more details on how the SCA is used.

Link Layer clock divider

This input decides the frequency of the clock to the link layer. A lower clock frequency results in lower power consumption. Default clock frequency for the operation is 24MHz. BLESS supports 24MHz, 12MHz and 8MHz clock configurations. Based on the end application requirement (how frequent the communication is expected to be), this parameter needs to be set.

Parameters

Parameters	Description
CYBLE_BLESS_CLK_CFG_PARAMS_T * bleSsClockConfig	Pointer to a variable of type CYBLE_BLESS_CLK_CFG_PARAMS_T from which the existing clock configuration is taken.

Returns

CYBLE_API_RESULT_T : Return value indicates if the function succeeded or failed. Following are the possible error codes.

Error codes	Description
CYBLE_ERROR_OK	On successful operation.
CYBLE_ERROR_INVALID_PARAMETER	On specifying NULL as input parameter.

CyBle_GenerateRandomNumber

Prototype

```
CYBLE_API_RESULT_T CyBle_GenerateRandomNumber(uint8 * randomNumber);
```

Description

This function generates 8-byte random number which complies with pseudo random number generation in accordance with [FIPS PUB 140-2]. Random number generation function is used during security procedure documented in Bluetooth 4.1 core specification, Volume 3, Part H.

This is a blocking function. No event is generated on calling this function.

Parameters

Parameters	Description
uint8 * randomNumber	Pointer to a buffer of size 8 bytes in which the generated random number gets stored.

Returns

CYBLE_API_RESULT_T : Return value indicates if the function succeeded or failed. Following are the possible error codes.

Error codes	Description
CYBLE_ERROR_OK	On successful operation.
CYBLE_ERROR_INVALID_PARAMETER	On specifying NULL as input parameter.

CyBle_AesEncrypt

Prototype

```
CYBLE_API_RESULT_T CyBle_AesEncrypt(uint8 * plainData, uint8 * aesKey, uint8 *
encryptedData);
```

Description

This function uses BLE sub-system AES engine to encrypt 128-bit of plain text using the given AES key. The output of AES processing is copied to encryptedData buffer. Refer Bluetooth 4.1 core specification, Volume 3, Part H, section 2.2 for more details on usage of AES key.

This is a blocking function. No event is generated on calling this function.

Parameters

Parameters	Description
uint8 * plainData	Pointer to the data containing plain text (128-bit) that is to be encrypted.
uint8 * aesKey	Pointer to the AES Key (128-bit) that is to be used for AES encryption.
uint8 * encryptedData	Pointer to the encrypted data (128-bit) that is output of AES module for given plainData and aesKey.

Returns

CYBLE_API_RESULT_T : Return value indicates if the function succeeded or failed. Following are the possible error codes.

Error codes	Description
CYBLE_ERROR_OK	On successful operation
CYBLE_ERROR_INVALID_PARAMETER	On specifying NULL as input parameter

CyBle_SetCeLengthParam

Prototype

```
CYBLE_API_RESULT_T CyBle_SetCeLengthParam(uint8 bdHandle, uint8 mdBit, uint16
ceLength);
```

Description

This function sets the connection event duration related parameters that can result in extension or truncation of LE connection event based on more data (mdBit) bit status and 'ceLength' duration. Refer Bluetooth 4.1 core specification, Volume 6, Part B, section 4.5 for more details on connection states of BLE Link Layer.

This is a blocking function. No event is generated on calling this function.

BLE Stack uses the BLESS hardware (AES module) to encrypt/decrypt the data. BLESS must be initialized before using this function. This function can safely be used by the application in "single thread/task system" which is the case with the current implementation of the BLE Stack. For multitasking systems, this function must be used within the BLE task to ensure atomic operation.

Parameters

Parameters	Description
uint8 bdHandle	Peer device bdHandle.
uint8 mdBit	'More Data' bit to select more number of data packets in BLE Stack buffer. A value of 0x01 indicates extension and a value of 0x00 indicates truncation.
uint16 ceLength	CE length of connection event that can extend the connection event. Details on this parameter are as given below, • Value Range = 0x0000 to 0xFFFF • Time Calculation = N x 0.625 ms • Time Range = 0 ms to 40.959 ms

Returns

CYBLE_API_RESULT_T : Return value indicates if the function succeeded or failed. Following are the possible error codes.

Error codes	Description
CYBLE_ERROR_OK	On successful operation
CYBLE_ERROR_INVALID_PARAMETER	One of the input parameters is invalid
CYBLE_ERROR_NO_CONNECTION	Connection does not exist

CyBle_WriteAuthPayloadTimeout

Prototype

```
CYBLE_API_RESULT_T CyBle_WriteAuthPayloadTimeout(uint8 bdHandle, uint16
authPayloadTimeout);
```

Description

This function sets the Authentication Payload timeout in BLE Controller for LE_PING feature. Refer Bluetooth 4.1 core specification, Volume 6, Part B, section 4.6.5 for LE Ping operation.

This is a blocking function. No event is generated on calling this function.

Parameters

Parameters	Description
uint8 bdHandle	Peer device handle.
uint16 authPayloadTimeout	Variable containing authentication timeout value to be written to BLE Controller. Details on this parameter are as given below, • Value Range = 0x0001 to 0xFFFF • Default Value (N) = 3000 (30 seconds) • Time Calculation = N x 10 ms • Time Range = 10 ms to 655,350 ms

Returns

CYBLE_API_RESULT_T: Return value indicates if the function succeeded or failed. Following are the possible error codes.

Error codes	Description
CYBLE_ERROR_OK	On successful operation
CYBLE_ERROR_INVALID_PARAMETER	One of the input parameters is invalid
CYBLE_ERROR_INVALID_OPERATION	Operation is not permitted
CYBLE_ERROR_NO_CONNECTION	Connection does not exist

CyBle_ReadAuthPayloadTimeout

Prototype

```
CYBLE_API_RESULT_T CyBle_ReadAuthPayloadTimeout(uint8 bdHandle, uint16 *
authPayloadTimeout);
```

Description

This function reads the Authentication Payload timeout set in BLE Controller for LE_PING feature Refer Bluetooth 4.1 core specification, Volume 6, Part B, section 4.6.5 for LE Ping operation.

This is a blocking function. No event is generated on calling this function.

Parameters

Parameters	Description
uint8 bdHandle	Peer device handle
uint16 * authPayloadTimeout	Pointer to a variable to which authentication timeout value, read from BLE Controller, is written.

Returns

CYBLE_API_RESULT_T: Return value indicates if the function succeeded or failed. Following are the possible error codes.

Error codes	Description
CYBLE_ERROR_OK	On successful operation.
CYBLE_ERROR_INVALID_PARAMETER	One of the input parameters is invalid.
CYBLE_ERROR_INVALID_OPERATION	Operation is not permitted.
CYBLE_ERROR_NO_CONNECTION	Connection does not exist.

CyBle_SetRxGainMode

Prototype

```
void CyBle_SetRxGainMode(uint8 bleSsGainMode);
```

Description

This function configures the Rx gain mode for BLESS radio for Rx operation.

Parameters

Parameters	Description
uint8 bleSsGainMode	Gain mode setting for the output power

Returns

none

Notes

BLESS RD Gain Mode	Description
CYBLE_BLESS_NORMAL_GAIN_MODE	0x00u - BLESS Normal Gain Mode • Tx Pwr Range -18dBm to 0 dBm • Normal Rx Sensitivity
CYBLE_BLESS_HIGH_GAIN_MODE	0x01u - BLESS High Gain Mode • Tx Pwr Range -18dBm to 3 dBm • 3 dBm Additional Rx Sensitivity

CyBle_SetTxGainMode

Prototype

```
void CyBle_SetTxGainMode(uint8 bleSsGainMode);
```

Description

This function configures the Tx gain mode for BLESS radio for Tx operation.

Parameters

Parameters	Description
uint8 bleSsGainMode	Gain mode setting for the output power

Returns

none

Notes

BLESS RD Gain Mode	Description
CYBLE_BLESS_NORMAL_GAIN_MODE	0x00u - BLESS Normal Gain Mode • Tx Pwr Range -18dBm to 0 dBm • Normal Rx Sensitivity
CYBLE_BLESS_HIGH_GAIN_MODE	0x01u - BLESS High Gain Mode • Tx Pwr Range -18dBm to 3 dBm • 3 dBm Additional Rx Sensitivity

CyBle_GetStackLibraryVersion

Prototype

```
CYBLE_API_RESULT_T CyBle_GetStackLibraryVersion(CYBLE_STACK_LIB_VERSION_T*
stackVersion);
```

Description

This function retrieves the version information of the BLE Stack library. This is a blocking function. No event is generated on calling this function.

Parameters

Parameters	Description
CYBLE_STACK_LIB_VERSION_T* stackVersion	Pointer to a variable of type CYBLE_STACK_LIB_VERSION_T containing the version information of the CYBLE Stack library.

Returns

CYBLE_API_RESULT_T: Return value indicates if the function succeeded or failed. Following are the possible error codes.

Error codes	Description
CYBLE_ERROR_OK	On successful operation.
CYBLE_ERROR_INVALID_PARAMETER	stackVersion is NULL.

GAP Functions

The GAP APIs allow access to the Generic Access Profile (GAP) layer of the BLE stack. Depending on the chosen GAP role in the GUI, you may use a subset of the supported APIs.

The GAP API names begin with CyBle_Gap. In addition to this, the APIs also append the GAP role initial letter in the API name.

GAP Central and Peripheral Functions

These are APIs common to both GAP Central role and GAP Peripheral role. You may use them in either roles.

No letter is appended to the API name: CyBle_Gap

Functions

Function	Description
CyBle_GapSetIoCap	This function sets the input and output capability of the BLE Device that is used during authentication procedure. This is a blocking function. No event... more
CyBle_GapSetOobData	This function sets OOB presence flag and data. This function should be used by the application layer if it wants to enable OOB bonding procedure... more
CyBle_GapGetPeerBdAddr	This function reads the peer Bluetooth device address which has already been fetched by the BLE Stack. 'peerBdAddr' stores the peer's Bluetooth device address identified... more
CyBle_GapGetPeerBdHandle	This function reads the device handle of the remote Bluetooth device using 'peerBdAddr', which has already been fetched by the BLE Stack. 'bdHandle' stores the... more
CyBle_GapGetPeerDevSecurity	This API enables the application to get the device security of the peer device identified by the bdHandle, when in the trusted list.
CyBle_GapDisconnect	This function disconnects the peer device. It is to be used by the device in GAP Central mode and may be used by a GAP... more
CyBle_GapGetPeerDevSecurityKeyInfo	This function enables the application to know the keys shared by a given peer device upon completion of the security sequence (already fetched by the... more

Function	Description
CyBle_GapGenerateDeviceAddress	This function generates either public or random address based on 'type' field of CYBLE_GAP_BD_ADDR_T structure. It uses BLE Controller's random number generator to generate the... more
CyBle_GapAuthReq	This function starts authentication/pairing procedure with the peer device. It is a non-blocking function. If the local device is a GAP Central, the pairing request... more
CyBle_GapAuthPassKeyReply	This function sends passkey for authentication. It is a non-blocking function. It should be invoked in reply to the authentication request event CYBLE_EVT_GAP_PASSKEY_ENTRY_REQUEST received by... more
CyBle_GapRemoveDeviceFromWhiteList	This function removes the bonding information of the device and removes it from the white list. More details on 'bonding' and 'trusted devices' is available... more
CyBle_GapRemoveOldestDeviceFromBondedList	This function removes the oldest device from the bonded list.
CyBle_GapAddDeviceToWhiteList	This function adds the device to the whitelist. Maximum number of devices that can be added to the whitelist is eight. Refer to Bluetooth 4.1... more
CyBle_GapGetBondedDevicesList	This function returns the count and bluetooth device address of the devices in the bonded device list. This is a blocking function. No event is... more
CyBle_GapGenerateKeys	This function generates the security keys that are to be exchanged with peer device during key exchange stage and sets it in the BLE Stack.... more
CyBle_GapSetSecurityKeys	This function sets the security keys that are to be exchanged with peer device during key exchange stage and sets it in the BLE Stack.... more
CyBle_GapGetLocalName	This API is used to read the local device name - a Characteristic of the GAP Service.
CyBle_GapSetLocalName	This API is used to set the local device name - a Characteristic of the GAP Service. If the Characteristic length entered in the Component... more
CyBle_GapUpdateAdvData	This function allows changing the ADV data and SCAN response data while advertising is going on. Application shall preserve Bluetooth Spec 4.1 mandated AD flags... more
CyBle_GapGetDevSecurityKeyInfo	This function gets the local device's Keys and key flags. The IRK received from this function should be used as the input IRK for the... more

CyBle_GapSetIoCap

Prototype

```
CYBLE_API_RESULT_T CyBle_GapSetIoCap(CYBLE_GAP_IOCAP_T ioCap);
```

Description

This function sets the input and output capability of the BLE Device that is used during authentication procedure. This is a blocking function. No event is generated on calling this function. The input capabilities are described in the following table:

Capability	Description
No input	Device does not have the ability to indicate "yes" or "no"
Yes/No	Device has at least two buttons that can be easily mapped to "yes" and "no" or the device has a mechanism whereby the user can indicate either "yes" or "no".
Keyboard	Device has a numeric keyboard that can input the numbers "0" through "9" and a confirmation. Device also has at least two buttons that can be easily mapped to "yes" and "no" or the device has a mechanism whereby the user can indicate either "yes" or "no".

The output capabilities are described in the following table:

Capability	Description
No output	Device does not have the ability to display or communicate a 6 digit decimal number.
Numeric output	Device has the ability to display or communicate a 6 digit decimal number.

Combined capability is defined in the following table:

Input Capability	No Output	Numeric Output
No input	NoInputNoOutput	DisplayOnly
Yes/No	NoInputNoOutput	DisplayYesNo
Keyboard	KeyboardOnly	KeyboardDisplay

Refer Bluetooth 4.1 core specification, Volume 3, Part C, section 5.2.2.4 for more details on the IO capabilities. IO capabilities of the BLE devices are used to determine the pairing method. Please refer Bluetooth 4.1 core specification, Volume 3, Part H, section 2.3.5.1 for more details on the impact of IO capabilities on the pairing method chosen.

Parameters

Parameters	Description
io_cap	IO Capability of type CYBLE_GAP_IOCAP_T.

Returns

CYBLE_API_RESULT_T : Return value indicates if the function succeeded or failed. Following are the possible error codes.

Error codes	Description
CYBLE_ERROR_OK	On successful operation

CYBLE_ERROR_INVALID_PARAMETER	On specifying NULL as input parameter

CyBle_GapSetOobData

Prototype

```
CYBLE_API_RESULT_T CyBle_GapSetOobData(uint8 bdHandle, uint8 oobFlag, uint8 * key,
uint8 * oobData, uint8 * oobDataLen);
```

Description

This function sets OOB presence flag and data. This function should be used by the application layer if it wants to enable OOB bonding procedure for any specific device identified by "bdHandle". This function should be called before initiating authentication or before responding to authentication request to set OOB flag and data. For more details on OOB, please refer Bluetooth 4.1 core specification, Volume 1, Part A, section 5.2.4.3. This is a blocking function. No event is generated on calling this function.

Parameters

Parameters	Description
uint8 bdHandle	Peer device for which the Out Of Band signalling (OOB) configuration is to be used.
uint8 oobFlag	OOB data presence flag. Allowed value are, • CYBLE_GAP_OOB_DISABLE • CYBLE_GAP_OOB_ENABLE
uint8 * key	16 Octet Temporary Key, to be used for OOB authentication.
uint8 * oobData	Pointer to OOB data.
uint8 * oobDataLen	Pointer to a variable to store OOB data length.

Returns

CYBLE_API_RESULT_T : Return value indicates if the function succeeded or failed. Following are the possible error codes.

Error codes	Description
CYBLE_ERROR_OK	On successful operation
CYBLE_ERROR_INVALID_PARAMETER	On specifying NULL as input parameter
CYBLE_ERROR_NO_DEVICE_ENTITY	'bdHandle' does not represent known device entity

CyBle_GapGetPeerBdAddr

Prototype

```
CYBLE_API_RESULT_T CyBle_GapGetPeerBdAddr(uint8 bdHandle, CYBLE_GAP_BD_ADDR_T*
peerBdAddr);
```

Description

This function reads the peer Bluetooth device address which has already been fetched by the BLE Stack. 'peerBdAddr' stores the peer's Bluetooth device address identified with 'bdHandle'. This is a blocking function. No event is generated on calling this function.

Parameters

Parameters	Description
uint8 bdHandle	Peer device handle.
CYBLE_GAP_BD_ADDR_T* peerBdAddr	Empty buffer where the Bluetooth device address gets stored.

Returns

CYBLE_API_RESULT_T : Return value indicates if the function succeeded or failed. Following are the possible error codes.

Error codes	Description
CYBLE_ERROR_OK	On successful operation.
CYBLE_ERROR_INVALID_PARAMETER	On specifying NULL as input parameter for 'peerBdAddr'.
CYBLE_ERROR_NO_DEVICE_ENTITY	Specified device handle does not map to any device handle entry in BLE stack.

CyBle_GapGetPeerBdHandle

Prototype

```
CYBLE_API_RESULT_T CyBle_GapGetPeerBdHandle(uint8* bdHandle, CYBLE_GAP_BD_ADDR_T*
peerBdAddr);
```

Description

This function reads the device handle of the remote Bluetooth device using 'peerBdAddr', which has already been fetched by the BLE Stack. 'bdHandle' stores the peer device handle. This is a blocking function. No event is generated on calling this function.

Parameters

Parameters	Description
uint8* bdHandle	Pointer to a variable to store peer device handle
CYBLE_GAP_BD_ADDR_T* peerBdAddr	Pointer to Bluetooth device address of peer device of type CYBLE_GAP_BD_ADDR_T, to be provided to this function as an input

Returns

CYBLE_API_RESULT_T : Return value indicates if the function succeeded or failed. Following are the possible error codes.

Error codes	Description
CYBLE_ERROR_OK	On successful operation.
CYBLE_ERROR_INVALID_PARAMETER	On specifying NULL as input parameter for 'peerBdAddr' or 'bdHandle'.
CYBLE_ERROR_MEMORY_ALLOCATION_FAILED	Memory allocation failed.
CYBLE_ERROR_NO_DEVICE_ENTITY	Specified device handle does not map to any device handle entry in BLE stack.

CyBle_GapGetPeerDevSecurity

Prototype

```
CYBLE_API_RESULT_T CyBle_GapGetPeerDevSecurity(uint8 bdHandle, CYBLE_GAP_AUTH_INFO_T*
security);
```

Description

This API enables the application to get the device security of the peer device identified by the bdHandle, when in the trusted list.

Security

The security requirement of a device is expressed in terms of a security mode and security level. A physical connection between two devices shall operate in only one security mode.

There are two LE security modes. For details refer to section Part C, 10.2 of BLE Spec 4.0.

- LE security mode 1
- LE security mode 2

Security of the device is set as,

- Security = Le security Mode (X) | Security level (level(mode X))

- X = mode 1 or mode 2
- level(mode 1)) = Security level 1 or Security level 2 or Security level 3
- level(mode 2)) = Security level 1 or Security level 2

Bonding

Bonding will be set to 1 if bonding is required for the device. The purpose of bonding (Bonding = 1) is to create a relation between two Bluetooth devices based on a common link key (a bond). The link key is created and exchanged (pairing) during the bonding procedure and is expected to be stored by both Bluetooth devices, to be used for future authentication.

ekeySize

Each device shall have maximum and minimum encryption key length parameters which defines the maximum and minimum size of the encryption key allowed in octets. The maximum and minimum encryption key length parameters is between 7 octets (56 bits) and 16 octets (128 bits). This is defined by the profile or device application.

The smaller value of the initiating and responding devices' maximum encryption key length parameters is used as the encryption key size. Both the initiating and responding devices will check that the resultant encryption key size is not smaller than the minimum key size parameter for that device and if it is, the device will send the Pairing Failed event. i.e. Host stack IP will send CYBLE_EVT_PAIRING_FAILED event to the profile.

Parameters

Parameters	Description
uint8 bdHandle	Peer device handle.
CYBLE_GAP_AUTH_INFO_T* security	Buffer to where Security information will be written.

Returns

CYBLE_API_RESULT_T : Return value indicates if the function succeeded or failed. Following are the possible error codes.

Error codes	Description
CYBLE_ERROR_OK	On successful operation.
CYBLE_ERROR_INVALID_PARAMETER	On specifying NULL as input parameter for 'bdAddr' or 'irk'.
CYBLE_ERROR_MEMORY_ALLOCATION_FAILED	Memory allocation failed.
CYBLE_ERROR_NO_DEVICE_ENTITY	Specified device handle does not map to any device handle entry in BLE stack.

CyBle_GapDisconnect

Prototype

```
CYBLE_API_RESULT_T CyBle_GapDisconnect(uint8 bdHandle);
```

Description

This function disconnects the peer device. It is to be used by the device in GAP Central mode and may be used by a GAP Peripheral device to send a disconnect request. This is a non-blocking function. On disconnection, the following events are generated, in order.

- CYBLE_EVT_GATT_DISCONNECT_IND
- CYBLE_EVT_GAP_DEVICE_DISCONNECTED

Parameters

Parameters	Description
uint8 bdHandle	Peer device handle

Returns

CYBLE_API_RESULT_T : Return value indicates if the function succeeded or failed. Following are the possible error codes.

Error codes	Description
CYBLE_ERROR_OK	On successful operation.
CYBLE_ERROR_INVALID_PARAMETER	No device to be disconnected. The specified device handle does not map to any device entry in the BLE Stack.
CYBLE_ERROR_MEMORY_ALLOCATION_FAILED	Memory allocation failed.

CyBle_GapGetPeerDevSecurityKeyInfo

Prototype

```
CYBLE_API_RESULT_T CyBle_GapGetPeerDevSecurityKeyInfo(uint8 bdHandle, uint8 *
keysFlag, CYBLE_GAP_SMP_KEY_DIST_T * keyInfo);
```

Description

This function enables the application to know the keys shared by a given peer device upon completion of the security sequence (already fetched by the BLE Stack). The keys are shared by the peer device on initiation of authentication which is performed using the CyBle_GapAuthReq() or CyBle_GappAuthReqReply() function.

This is a blocking function. No event is generated on calling this function.

Parameters

Parameters	Description
uint8 bdHandle	Peer device handle.
uint8 * keysFlag	Indicates the keys to be retrieved from peer device. The following bit fields indicate the presence or absence of the keys distributed. **Negotiated Local/Peer Key distribution** • Bit 0. Encryption information (LTK and MID Information) • Bit 1. Identity information • Bit 2. Signature Key • Bit 3-7. Reserved
key_info	Pointer to variable of type CYBLE_GAP_SMP_KEY_DIST_T to copy the stored keys of the peer device identified by 'bdHandle'

Returns

CYBLE_API_RESULT_T : Return value indicates if the function succeeded or failed. Following are the possible error codes.

Error codes	Description
CYBLE_ERROR_OK	On successful operation.
CYBLE_ERROR_INVALID_PARAMETER	On specifying NULL as input parameter for 'keyInfo'.
CYBLE_ERROR_INVALID_OPERATION	An error occurred in BLE stack.

CyBle_GapGenerateDeviceAddress

Prototype

```
CYBLE_API_RESULT_T CyBle_GapGenerateDeviceAddress(CYBLE_GAP_BD_ADDR_T* bdAddr,
CYBLE_GAP_ADDR_TYPE_T addrType, uint8 * irk);
```

Description

This function generates either public or random address based on 'type' field of CYBLE_GAP_BD_ADDR_T structure. It uses BLE Controller's random number generator to generate the random part of the Bluetooth device address.

The parameter 'addrType' specifies further sub-classification within the public and random address types.

This is a blocking function. No event is generated on calling this function.

Parameters

Parameters	Description
CYBLE_GAP_BD_ADDR_T* bdAddr	Bluetooth device address is generated and populated in the structure pointed to by this pointer. The structure is of type CYBLE_GAP_BD_ADDR_T.
CYBLE_GAP_ADDR_TYPE_T addrType	Specifies the type of address. This can take one of the values from the enumerated data type CYBLE_GAP_ADDR_TYPE_T.
uint8 * irk	Pointer to buffer containing 128-bit 'IRK' data. This parameter is only used when CYBLE_GAP_RANDOM_PRIV_RESOLVABLE_ADDR is the value set to 'addrType'. For other values of 'addrType', this parameter is not used.

Returns

CYBLE_API_RESULT_T : Return value indicates if the function succeeded or failed. Following are the possible error codes.

Error codes	Description
CYBLE_ERROR_OK	On successful operation.
CYBLE_ERROR_INVALID_PARAMETER	On specifying NULL as input parameter.

CyBle_GapAuthReq

Prototype

```
CYBLE_API_RESULT_T CyBle_GapAuthReq(uint8 bdHandle, CYBLE_GAP_AUTH_INFO_T * authInfo);
```

Description

This function starts authentication/pairing procedure with the peer device. It is a non-blocking function.

If the local device is a GAP Central, the pairing request is sent to the GAP Peripheral device. On receiving CYBLE_EVT_GAP_AUTH_REQ event, the GAP Peripheral is expected to respond by invoking the CyBle_GappAuthReqReply() function.

If the local device is GAP Peripheral, a Security Request is sent to GAP Central device. On receiving CYBLE_EVT_GAP_AUTH_REQ event, the GAP Central device is expected to respond by invoking 'CyBle_GapAuthReq ()' function.

Parameters

Parameters	Description
uint8 bdHandle	Peer device handle
CYBLE_GAP_AUTH_INFO_T * authInfo	Pointer to security information of the device of type CYBLE_GAP_AUTH_INFO_T. The 'authErr' parameter in CYBLE_GAP_AUTH_INFO_T should be ignored as it is not used in this function.

Returns

CYBLE_API_RESULT_T : Return value indicates if the function succeeded or failed. Following are the possible error codes.

Error codes	Description
CYBLE_ERROR_OK	On successful operation.
CYBLE_ERROR_INVALID_PARAMETER	On specifying NULL as input parameter for 'authInfo' or assigning an invalid value to one of the elements of 'authInfo'.
CYBLE_ERROR_MEMORY_ALLOCATION_FAILED	Memory allocation failed.
CYBLE_ERROR_NO_DEVICE_ENTITY	No device entry in the BLE stack to run this operation.
CYBLE_ERROR_INSUFFICIENT_RESOURCES	On bonded device is full and application tries to initiate pairing with bonding enable.

CyBle_GapAuthPassKeyReply

Prototype

```
CYBLE_API_RESULT_T CyBle_GapAuthPassKeyReply(uint8 bdHandle, uint32 passkey, uint8
accept);
```

Description

This function sends passkey for authentication. It is a non-blocking function.

It should be invoked in reply to the authentication request event CYBLE_EVT_GAP_PASSKEY_ENTRY_REQUEST received by the BLE Stack. This function is used to accept the passkey request and send the passkey or reject the passkey request.

- If the authentication operation succeeds, CYBLE_EVT_GAP_AUTH_COMPLETE is generated. If the authentication process times out, CYBLE_EVT_TIMEOUT event is generated.
- If the authentication fails, CYBLE_EVT_GAP_AUTH_FAILED event is generated.

Parameters

Parameters	Description
uint8 bdHandle	Peer device handle
uint32 passkey	6-digit decimal number (authentication passkey)
uint8 accept	Accept or reject passkey entry request. Allowed values are, CYBLE_GAP_REJECT_PASSKEY_REQ CYBLE_GAP_ACCEPT_PASSKEY_REQ

Returns

CYBLE_API_RESULT_T : Return value indicates if the function succeeded or failed. Following are the possible error codes.

Error codes	Description
CYBLE_ERROR_OK	On successful operation.
CYBLE_ERROR_INVALID_PARAMETER	Invalid parameter.
CYBLE_ERROR_MEMORY_ALLOCATION_FAILED	Memory allocation failed.
CYBLE_ERROR_NO_DEVICE_ENTITY	Device identified using 'bdHandle' does not exist.

CyBle_GapRemoveDeviceFromWhiteList

Prototype

```
CYBLE_API_RESULT_T CyBle_GapRemoveDeviceFromWhiteList(CYBLE_GAP_BD_ADDR_T* bdAddr);
```

Description

This function marks the device specified by the handle as untrusted. It removes the bonding information of the device and removes it from the white list. More details on 'bonding' and 'trusted devices' is available in Bluetooth 4.1 core specification, Volume 3, Part C, section 9.4.4.

This is a blocking function. No event is generated on calling this function.

Parameters

Parameters	Description
CYBLE_GAP_BD_ADDR_T* bdAddr	Pointer to peer device address, of type CYBLE_GAP_BD_ADDR_T. If device address is set to 0, then all devices shall be removed from trusted list and white list.

Returns

CYBLE_API_RESULT_T : Return value indicates if the function succeeded or failed. Following are the possible error codes.

Error codes	Description
CYBLE_ERROR_OK	On successful operation.
CYBLE_ERROR_INVALID_PARAMETER	On specifying NULL as input parameter for 'bdAddr'.
CYBLE_ERROR_INVALID_OPERATION	Whitelist is already in use.
CYBLE_ERROR_NO_DEVICE_ENTITY	Device does not exist in the whitelist.

CyBle_GapRemoveOldestDeviceFromBondedList

Prototype

```
CYBLE_API_RESULT_T CyBle_GapRemoveOldestDeviceFromBondedList(void);
```

Description

This function removes the oldest device from the bonded list.

Returns

CYBLE_API_RESULT_T : Return value indicates if the function succeeded (0x0000) or failed. Following are the possible error codes returned -

Error codes	Description
CYBLE_ERROR_OK	On successful operation.
CYBLE_ERROR_NO_DEVICE_ENTITY	If no device is present bonded list

CyBle_GapAddDeviceToWhiteList

Prototype

```
CYBLE_API_RESULT_T CyBle_GapAddDeviceToWhiteList(CYBLE_GAP_BD_ADDR_T* bdAddr);
```

Description

This function adds the device to the whitelist. Maximum number of devices that can be added to the whitelist is eight including CYBLE_GAP_MAX_BONDED_DEVICE. Refer to Bluetooth 4.1 core specification, Volume 3, Part C, section 9.3.5 for more details on whitelist.

This is a blocking function. No event is generated on calling this function.

Parameters

Parameters	Description
CYBLE_GAP_BD_ADDR_T* bdAddr	Peer device address, of type CYBLE_GAP_BD_ADDR_T.

Returns

CYBLE_API_RESULT_T : Return value indicates if the function succeeded or failed. Following are the possible error codes.

Error codes	Description
CYBLE_ERROR_OK	On successful operation
CYBLE_ERROR_INVALID_PARAMETER	On specifying NULL as input parameter for 'bdAddr' or

	'bdAddr->type' has an invalid value
CYBLE_ERROR_INVALID_OPERATION	Whitelist is already in use
CYBLE_ERROR_INSUFFICIENT_RESOURCES	WhitelistMemory is full
CYBLE_ERROR_DEVICE_ALREADY_EXISTS	Matching device already exists in the whitelist

CyBle_GapGetBondedDevicesList

Prototype

```
CYBLE_API_RESULT_T CyBle_GapGetBondedDevicesList(CYBLE_GAP_BONDED_DEV_ADDR_LIST_T*
bondedDevList);
```

Description

This function returns the count and bluetooth device address of the devices in the bonded device list. This is a blocking function. No event is generated on calling this function.

Application invoking this function should allocate sufficientMemory for the structure CYBLE_GAP_BONDED_DEV_ADDR_LIST_T, where the complete list of bonded devices along with count can be written. Maximum devices bonded are specified by CYBLE_GAP_MAX_BONDED_DEVICE, which is a pre-processing parameter for the BLE Stack. Hence, the bonded device count will be less than or equal to CYBLE_GAP_MAX_BONDED_DEVICE.

Refer Bluetooth 4.1 core specification, Volume 3, Part C, section 9.4.4 for details on bonded devices.

Parameters

Parameters	Description
CYBLE_GAP_BONDED_DEV_ADDR_LIST_T* bondedDevList	Buffer to which list of bonded device list will be stored of type CYBLE_GAP_BONDED_DEV_ADDR_LIST_T.

Returns

CYBLE_API_RESULT_T : Return value indicates if the function succeeded or failed. Following are the possible error codes.

Error codes	Description
CYBLE_ERROR_OK	On successful operation.
CYBLE_ERROR_INVALID_PARAMETER	On specifying NULL as input parameter.

CyBle_GapGenerateKeys

Prototype

```
CYBLE_API_RESULT_T CyBle_GapGenerateKeys(uint8 keysFlag, CYBLE_GAP_SMP_KEY_DIST_T *
keyInfo);
```

Description

This function generates the security keys that are to be exchanged with peer device during key exchange stage of authentication procedure and sets it in the BLE Stack. This is a blocking function. No event is generated on calling this function.

Parameters

Parameters	Description
uint8 keysFlag	This parameter indicates which keys get exchanged with peer device. The following is the bit field mapping for the keys. **First 4 bits. Initiator's Key distribution** • Bit 0. Encryption information (LTK and MID Information) • Bit 1. Identity information • Bit 2. Signature Key • Bit 3. Reserved **Next 4 bits. Responder's Key distribution** • Bit 4. Encryption information (LTK and MID Information) • Bit 5. Identity information • Bit 6. Signature Key • Bit 7. Reserved
CYBLE_GAP_SMP_KEY_DIST_T * keyInfo	Pointer to a variable containing the returned keys, of type 'CYBLE_GAP_SMP_KEY_DIST_T'

Returns

CYBLE_API_RESULT_T : Return value indicates if the function succeeded or failed. Following are the possible error codes.

Error codes	Description
CYBLE_ERROR_OK	On successful operation.
CYBLE_ERROR_INVALID_PARAMETER	On specifying NULL as input parameter for 'keyInfo'

CyBle_GapGetLocalName

Prototype

```
CYBLE_API_RESULT_T CyBle_GapGetLocalName(char8 name[]);
```

Description

This API is used to read the local device name - a Characteristic of the GAP Service.

Parameters

Parameters	Description
char8 name[]	The local device name string. Used to read the local name to the given string array. It represents a UTF-8 encoded User Friendly Descriptive Name for the device. The length of the local device string is entered into the Component customizer and it can be set to a value from 0 to 248 bytes. If the name contained in the parameter is shorter than the length from the customizer, the end of the name is indicated by a NULL octet (0x00).

Returns

CYBLE_API_RESULT_T : Return value indicates if the function succeeded or failed. Following are the possible error codes.

Error codes	Description
CYBLE_ERROR_OK	Function completed successfully.
CYBLE_ERROR_INVALID_PARAMETER	On specifying NULL as input parameter

CyBle_GapSetLocalName

Prototype

```
CYBLE_API_RESULT_T CyBle_GapSetLocalName(const char8 name[]);
```

Description

This API is used to set the local device name - a Characteristic of the GAP Service. If the Characteristic length entered in the Component customizer is shorter than the string specified by the "name" parameter, the local device name will be cut to the length specified in the customizer.

Parameters

Parameters	Description
const char8 name[]	The local device name string. The name string to be written as the local device name. It represents a UTF-8 encoded User Friendly Descriptive Name for the device. The length of the local device string is entered into the Component customizer and it can be set to a value from 0 to 248 bytes. If the name contained in the parameter is shorter than the length from the customizer, the end of the name is indicated by a NULL octet (0x00).

Returns

CYBLE_API_RESULT_T : Return value indicates if the function succeeded or failed. Following are the possible error codes.

Error codes	Description
CYBLE_ERROR_OK	Function completed successfully.
CYBLE_ERROR_INVALID_PARAMETER	On specifying NULL as input parameter

CyBle_GapSetSecurityKeys

Prototype

```
CYBLE_API_RESULT_T CyBle_GapSetSecurityKeys(uint8 keysFlag, CYBLE_GAP_SMP_KEY_DIST_T *
keyInfo);
```

Description

This function sets the security keys that are to be exchanged with peer device during key exchange stage of authentication procedure and sets it in the BLE Stack. This is a blocking function. No event is generated on calling this function.

Parameters

Parameters	Description
uint8 keysFlag	This parameter indicates which keys get exchanged with peer device. The following is the bit field mapping for the keys. **First 4 bits. Initiator's Key distribution** • Bit 0. Encryption information (LTK and MID Information) • Bit 1. Identity information • Bit 2. Signature Key • Bit 3. Reserved **Next 4 bits. Responder's Key distribution** • Bit 4. Encryption information (LTK and MID Information) • Bit 5. Identity information • Bit 6. Signature Key • Bit 7. Reserved
CYBLE_GAP_SMP_KEY_DIST_T * keyInfo	Pointer to a variable containing the keys to be set, of type 'CYBLE_GAP_SMP_KEY_DIST_T'. idAddrInfo param of 'CYBLE_GAP_SMP_KEY_DIST_T' will be ignored. 'CyBle_SetDeviceAddress' api needs to be used to set bd address.

Returns

CYBLE_API_RESULT_T : Return value indicates if the function succeeded or failed. Following are the possible error codes.

Error codes	Description
CYBLE_ERROR_OK	On successful operation.

CYBLE_ERROR_INVALID_PARAMETER	On specifying NULL as input parameter for 'keyInfo'

CyBle_GapGetDevSecurityKeyInfo

Prototype

```
CYBLE_API_RESULT_T CyBle_GapGetDevSecurityKeyInfo(uint8 * keyFlags,
CYBLE_GAP_SMP_KEY_DIST_T * keys);
```

Description

This function gets the local device's Keys and key flags. The IRK received from this function should be used as the input IRK for the function 'CyBle_GapGenerateDeviceAddress' to generate Random Private Resolvable address. This is a blocking function. No event is generated on calling this function.

Parameters

Parameters	Description
uint8 * keyFlags	Pointer to a byte where the key flags are stored. Based on the flag bits, the calling application can determine if the returned value is valid (1) or not (0). **Key distribution flag** • Bit 0: Local Encryption information • Bit 1: Local Identity information • Bit 2: Local Signature Key • Bit 3 - Bit 7: Reserved
CYBLE_GAP_SMP_KEY_DIST_T * keys	Pointer to a structure of type CYBLE_GAP_SMP_KEY_DIST_T where the keys get stored

Returns

CYBLE_API_RESULT_T : Return value indicates if the function succeeded or failed. Following are the possible error codes.

Error codes	Description
CYBLE_ERROR_OK	On successful operation.
CYBLE_ERROR_INVALID_PARAMETER	On specifying NULL as input parameters

CyBle_GapUpdateAdvData

Prototype

```
CYBLE_API_RESULT_T CyBle_GapUpdateAdvData(CYBLE_GAPP_DISC_DATA_T * advDiscData,
CYBLE_GAPP_SCAN_RSP_DATA_T * advScanRespData);
```

Description

This function allows changing the ADV data and SCAN response data while advertising is going on. Application shall preserve Bluetooth Spec 4.1 mandated AD flags fields corresponding to the type of GAP discovery mode and only change the rest of the data. When the data is set, there is possible race condition that the device might be in process of transmitting ADV data present in FIFO and during that time firmware overwrites the data in FIFO. So in that particular ADV event adv payload may not be correct. This API must be called after checking the state of BLE SS using CyBle_GetBleSsState() API, It can safely be called when BLESS state is CYBLE_BLESS_STATE_EVENT_CLOSE. If this API is called in ADV event where actual Tx or Rx is going on then it may have catastrophic effect with respect to power on ADV timing.

Parameters

Parameters	Description
advData	Pointer to a structure of CYBLE_GAPP_DISC_DATA_T. It has two fields advData field representing the data and advDataLen indicating the length of present data. Application can pass this parameter as NULL for if the ADV data doesn't need to be changed.
scanRespData	Pointer to a structure of type CYBLE_GAPP_SCAN_RSP_DATA_T. It has two fields scanRspData field representing the data and scanRspDataLen indicating the the length of present data. Application can pass this parameter as NULL if the SCAN RESP data doesn't need to be changed.

Returns

CYBLE_API_RESULT_T : Return value indicates if the function succeeded or failed. Following are the possible error codes.

Error codes	Description
CYBLE_ERROR_OK	On successful operation.
CYBLE_ERROR_INVALID_PARAMETER	Data length in input parameter exceeds 31 bytes.

GAP Central Functions

APIs unique to designs configured as a GAP Central role.

A letter 'c' is appended to the API name: CyBle_Gapc

Functions

Function	Description
CyBle_GapcStartScan	This function is used for discovering GAP peripheral devices that are available for connection. It performs the scanning routine using the parameters entered in the... more
CyBle_GapcStopScan	This function used to stop the discovery of devices. On stopping discovery operation, CYBLE_EVT_GAPC_SCAN_START_STOP

	event is generated. Application layer needs to keep track of the... more
CyBle_GapcStartDiscovery	This function starts the discovery of devices which are advertising. This is a non-blocking function. As soon as the discovery operation starts, CYBLE_EVT_GAPC_SCAN_START_STOP event is... more
CyBle_GapcStopDiscovery	This function stops the discovery of devices. This is a non-blocking function. On stopping discovery operation, CYBLE_EVT_GAPC_SCAN_START_STOP event is generated. Application layer needs to keep... more
CyBle_GapcConnectDevice	This function is used to send a connection request to the remote device with the connection parameters set in the Component customizer. This function needs... more
CyBle_GapcInitConnection	This function sends a connection request to the remote device with required connection parameters. On successful connection, the following events are generated at the GAP... more
CyBle_GapcCancelConnection	This function cancels a previously initiated connection with the peer device. This is a blocking function. No event is generated on calling this function. If... more
CyBle_GapcConnectionParamUpdateRequest	This function sends 'connection parameter update' command to BLE Controller. This function can only be used from a device connected in GAP Central role. It... more
CyBle_GapcResolveDevice	This function enables the application to start resolution procedure for a device that is connected using resolvable private address. This is a non-blocking function. The... more
CyBle_GapcSetHostChannelClassification	This function sets channel classification for data channels. This classification persists until it is overwritten by a subsequent call to this function or the controller... more
CyBle_GapcSetRemoteAddr	This function allows application to set the new address of remote device identified by bdHandle. This API should be used when- If peer device is... more

CyBle_GapcStartScan

Prototype

```
CYBLE_API_RESULT_T CyBle_GapcStartScan(uint8 scanningIntervalType);
```

Description

This function is used for discovering GAP peripheral devices that are available for connection. It performs the scanning routine using the parameters entered in the Component's customizer.

As soon as the discovery operation starts, CYBLE_EVT_GAPC_SCAN_START_STOP event is generated. The CYBLE_EVT_GAPC_SCAN_PROGRESS_RESULT event is generated when a

GAP peripheral device is located. There are three discovery procedures can be selected in the customizer's GUI:

- Observation procedure: A device performing the observer role receives only advertisement data from devices irrespective of their discoverable mode settings. Advertisement data received is provided by the event, CYBLE_EVT_GAPC_SCAN_PROGRESS_RESULT. This procedure requires the scanType sub parameter to be passive scanning.
- Limited Discovery procedure: A device performing the limited discovery procedure receives advertisement data and scan# response data from devices in the limited discoverable mode only. Received data is provided by the event,
- CYBLE_EVT_GAPC_SCAN_PROGRESS_RESULT. This procedure requires the scanType sub-parameter to be active scanning.
- General Discovery procedure: A device performing the general discovery procedure receives the advertisement data and scan response data from devices in both limited discoverable mode and the general discoverable mode. Received data is provided by the event, CYBLE_EVT_GAPC_SCAN_PROGRESS_RESULT. This procedure requires the scanType sub-parameter to be active scanning.

Every Advertisement / Scan response packet received results in a new event, CYBLE_EVT_GAPC_SCAN_PROGRESS_RESULT. If 'scanTo' sub-parameter is a non-zero value, then upon commencement of discovery procedure and elapsed time = 'scanTo', CYBLE_EVT_TIMEOUT event is generated with the event parameter indicating CYBLE_GAP_SCAN_TO. Possible generated events are:

- CYBLE_EVT_GAPC_SCAN_START_STOP: If a device started or stopped scanning. Use CyBle_GetState() to determine the state. Sequential scanning could be started when CYBLE_STATE_DISCONNECTED state is returned.
- CYBLE_EVT_GAPC_SCAN_PROGRESS_RESULT
- CYBLE_EVT_TIMEOUT (CYBLE_GAP_SCAN_TO)

Parameters

Parameters	Description
uint8 scanningIntervalType	Fast or slow scanning interval with timings entered in Scan settings section of the customiser. • CYBLE_SCANNING_FAST 0x00u • CYBLE_SCANNING_SLOW 0x01u • CYBLE_SCANNING_CUSTOM 0x02u

Returns

CYBLE_API_RESULT_T : Return value indicates if the function succeeded or failed. Following are the possible error codes.

Error codes	Values	Description
CYBLE_ERROR_OK	0x0000	On successful operation.
CYBLE_ERROR_STACK_INTERNAL	0x0003	An error occurred in the BLE stack.

CyBle_GapcStopScan

Prototype

```
void CyBle_GapcStopScan(void);
```

Description

This function used to stop the discovery of devices. On stopping discovery operation, CYBLE_EVT_GAPC_SCAN_START_STOP event is generated. Application layer needs to keep track of the function call made before receiving this event to associate this event with either the start or stop discovery function.

Possible events generated are:

- CYBLE_EVT_GAPC_SCAN_START_STOP

Returns

None

CyBle_GapcStartDiscovery

Prototype

```
CYBLE_API_RESULT_T CyBle_GapcStartDiscovery(CYBLE_GAPC_DISC_INFO_T* scanInfo);
```

Description

This function starts the discovery of devices which are advertising. This is a non-blocking function. As soon as the discovery operation starts, CYBLE_EVT_GAPC_SCAN_START_STOP event is generated.

Every Advertisement / Scan response packet received results in a new event, CYBLE_EVT_GAPC_SCAN_PROGRESS_RESULT. If 'scanInfo->scanTo' is a non-zero value, upon commencement of discovery procedure and elapsed time = 'scanInfo->scanTo', CYBLE_EVT_TIMEOUT event is generated with the event parameter indicating CYBLE_GAP_SCAN_TO.

If 'scanInfo->scanTo' is equal to zero, the scanning operation is performed until the CyBle_GapcStopDiscovery() function is invoked.

There are three discovery procedures that can be specified as a parameter to this function.

Observation procedure

A device performing the observer role receives only advertisement data from devices irrespective of their discoverable mode settings. Advertisement data received is provided by the event,

CYBLE_EVT_GAPC_SCAN_PROGRESS_RESULT

'scanInfo->scanType' should be set as passive scanning (0x00).

Limited Discovery procedure

A device performing the limited discovery procedure receives advertisement data and scan response data from devices in the limited discoverable mode only. Received data is provided by the event,

CYBLE_EVT_GAPC_SCAN_PROGRESS_RESULT

'scanInfo->scanType' should be set as active scanning (0x01).

General Discovery procedure

A device performing the general discovery procedure receives the advertisement data and scan response data from devices in both limited discoverable mode and the general discoverable mode. Received data is provided by the event,

CYBLE_EVT_GAPC_SCAN_PROGRESS_RESULT

'scanInfo->scanType' should be set as active scanning (0x01).

Parameters

Parameters	Description
CYBLE_GAPC_DISC_INFO_T* scanInfo	Pointer to a variable of type CYBLE_GAPC_DISC_INFO_T

Returns

CYBLE_API_RESULT_T : Return value indicates if the function succeeded or failed. Following are the possible error codes.

Error codes	Description
CYBLE_ERROR_OK	On successful operation.
CYBLE_ERROR_INVALID_PARAMETER	On specifying NULL as input parameter for 'scanInfo' or if any element within 'scanInfo' has an invalid value.

CYBLE_ERROR_MEMORY_ALLOCATION_FAILED	Memory allocation failed.

CyBle_GapcStopDiscovery

Prototype

```
void CyBle_GapcStopDiscovery(void);
```

Description

This function stops the discovery of devices. This is a non-blocking function. On stopping discovery operation, CYBLE_EVT_GAPC_SCAN_START_STOP event is generated.

Application layer needs to keep track of the function call made before receiving this event to associate this event with either the start or stop discovery function.

Returns

None

CyBle_GapcConnectDevice

Prototype

```
CYBLE_API_RESULT_T CyBle_GapcConnectDevice(const CYBLE_GAP_BD_ADDR_T * address);
```

Description

This function is used to send a connection request to the remote device with the connection parameters set in the Component customizer. This function needs to be called only once after the target device is discovered by CyBle_GapcStartScan() and further scanning has stopped. Scanning is successfully stopped on invoking CyBle_GapcStopScan() and then receiving the event CYBLE_EVT_GAPC_SCAN_START_STOP with sub-parameter 'success' = 0x01u.

On successful connection, the following events are generated at the GAP Central device (as well as the GAP Peripheral device), in the following order.

- CYBLE_EVT_GATT_CONNECT_IND
- CYBLE_EVT_GAP_DEVICE_CONNECTED

A procedure is considered to have timed out if a connection response packet is not received within time set by cyBle_connectingTimeout global variable (30 seconds by default).

CYBLE_EVT_TIMEOUT event with CYBLE_GENERIC_TO parameter will indicate about connection procedure timeout. Connection will automatically be cancelled and state will be changed to CYBLE_STATE_DISCONNECTED.

Parameters

Parameters	Description
const CYBLE_GAP_BD_ADDR_T * address	The device address of the remote device to connect to.
timeout	Timeout for which timer to be started in seconds.

Returns

CYBLE_API_RESULT_T : Return value indicates if the function succeeded or failed. Following are the possible error codes.

Error codes	Description
CYBLE_ERROR_OK	On successful operation.
CYBLE_ERROR_STACK_INTERNAL	On error occurred in the BLE stack.
CYBLE_ERROR_INVALID_PARAMETER	On specifying NULL as input parameter for 'scanInfo' or if any element with in 'scanInfo' has an invalid value.

CyBle_GapcInitConnection

Prototype

```
CYBLE_API_RESULT_T CyBle_GapcInitConnection(CYBLE_GAPC_CONN_PARAM_T* connParam);
```

Description

This function sends a connection request to the remote device with required connection parameters. On successful connection, the following events are generated at the GAP Central end (as well as the GAP Peripheral end), in order.

- CYBLE_EVT_GATT_CONNECT_IND
- CYBLE_EVT_GAP_DEVICE_CONNECTED

This is a non-blocking function. This function needs to be called after successfully stopping scanning. Scanning is successfully stopped on invoking the CyBle_GapcStopDiscovery() function and receiving the event CYBLE_EVT_GAPC_SCAN_START_STOP with the event data of '0x01', indicating success.

For details related to connection modes and procedures, refer to Bluetooth 4.1 Core Specification, Volume 3, Part C, Section 9.3.

Parameters

Parameters	Description
CYBLE_GAPC_CONN_PARAM_T* connParam	Structure of type 'CYBLE_GAPC_CONN_PARAM_T' which contains the connection parameters.

	Note Any parameter of structure type CYBLE_GAPC_CONN_PARAM_T, if not required by a specific Bluetooth Low Energy profile, may be ignored.

Returns

CYBLE_API_RESULT_T : Return value indicates if the function succeeded or failed. Following are the possible error codes.

Error codes	Description
CYBLE_ERROR_OK	On successful operation.
CYBLE_ERROR_INVALID_PARAMETER	On specifying NULL as input parameter for 'connParam'or if any element within 'connParam' has an invalid value.
CYBLE_ERROR_INVALID_OPERATION	Device already connected.
CYBLE_ERROR_MEMORY_ALLOCATION_FAILED	Memory allocation failed.

CyBle_GapcCancelConnection

Prototype

```
CYBLE_API_RESULT_T CyBle_GapcCancelConnection(void);
```

Description

This function cancels a previously initiated connection with the peer device. This is a blocking function. No event is generated on calling this function.

If the devices are already connected, then this function should not be used. To disconnect from an existing connection, use the function CyBle_GapDisconnect().

Returns

CYBLE_API_RESULT_T : Return value indicates if the function succeeded or failed. Following are the possible error codes.

Error codes	Description
CYBLE_ERROR_OK	On successful operation.
CYBLE_ERROR_INVALID_OPERATION	Device already connected.
CYBLE_ERROR_MEMORY_ALLOCATION_FAILED	Memory allocation failed.

CyBle_GapcConnectionParamUpdateRequest

Prototype

```
CYBLE_API_RESULT_T CyBle_GapcConnectionParamUpdateRequest(uint8 bdHandle,
CYBLE_GAP_CONN_UPDATE_PARAM_T * connParam);
```

Description

This function sends the connection parameter update command to the local Controller. This function can only be used from device connected in GAP Central role.

Parameters

Parameters	Description
uint8 bdHandle	Peer device handle
CYBLE_GAP_CONN_UPDATE_PARAM_T * connParam	Pointer to a structure of type CYBLE_GAP_CONN_UPDATE_PARAM_T containing connection parameter updates

Returns

CYBLE_API_RESULT_T : Return value indicates if the function succeeded or failed. Following are the possible error codes.

Error codes	Description
CYBLE_ERROR_OK	On successful operation
CYBLE_ERROR_INVALID_PARAMETER	'connParam' is NULL
CYBLE_ERROR_MEMORY_ALLOCATION_FAILED	Memory allocation failed

Notes

Connection parameter update procedure, defined as part of Bluetooth spec 4.1, is not supported.

This function will allow GAP Central application to update connection parameter for local controller and local controller will follow the procedure as defined in Bluetooth Core specification 4.0.

CyBle_GapcResolveDevice

Prototype

```
CYBLE_API_RESULT_T CyBle_GapcResolveDevice(const uint8 * bdAddr, const uint8 * irk);
```

Description

This function enables the application to start resolution procedure for a device that is connected using resolvable private address. This is a blocking function. The application should use this function when in GAP Central mode.

Refer to Bluetooth 4.1 Core specification, Volume 3, Part C, section 10.8.2.3 Resolvable Private Address Resolution Procedure to understand the usage of Private addresses.

Parameters

Parameters	Description
const uint8 * bdAddr	Pointer to peer bluetooth device address of length 6 bytes, not NULL terminated.
const uint8 * irk	Pointer to 128-bit IRK to be used for resolving the peer's private resolvable address.

Returns

CYBLE_API_RESULT_T : Return value indicates if the function succeeded or failed. Following are the possible error codes.

Error codes	Description
CYBLE_ERROR_OK	On successful operation.
CYBLE_ERROR_INVALID_PARAMETER	On specifying NULL as input parameter for 'bdAddr' or 'irk'.
CYBLE_ERROR_INVALID_OPERATION	No device to be resolved. The specified device handle does not map to any device entry in the BLE Stack.

CyBle_GapcSetHostChannelClassification

Prototype

```
    CYBLE_API_RESULT_T CyBle_GapcSetHostChannelClassification(uint8* channelMap);
```

Description

This function sets channel classification for data channels. This classification persists until it is overwritten by a subsequent call to this function or the controller is reset. If this command is used, updates should be sent within 10 seconds of the BLE Host knowing that the channel classification has changed. The interval between two successive commands sent will be at least one second. This command will only be used when the local device supports the Master role.

For details, refer to Bluetooth core specification 4.1, Volume 2, part E, section 7.8.19.

This is a blocking function. No event is generated on calling this function.

Parameters

Parameters	Description
uint8* channelMap	This parameter contains five octet byte stream (Least Significant Byte having the bit fields 0 to 7, most significant byte having the bit fields 32 to 36). The nth such field (in the range 0 to 36) contains the value for the link layer channel index n. Allowed values and their interpretation are, • Channel 'n' is bad = 0x00u • Channel 'n' is unknown = 0x01u The most significant bits (37 to 39) are reserved and will be set to 0. At least one channel will be marked as unknown.

Returns

CYBLE_API_RESULT_T : Return value indicates if the function succeeded or failed. Following are the possible error codes.

Error codes	Description
CYBLE_ERROR_OK	On successful operation.
CYBLE_ERROR_INVALID_PARAMETER	On specifying NULL as input parameter for 'channelMap'.
CYBLE_ERROR_MEMORY_ALLOCATION_FAILED	Memory allocation failed.

CyBle_GapcSetRemoteAddr

Prototype

```
CYBLE_API_RESULT_T CyBle_GapcSetRemoteAddr(uint8 bdHandle, CYBLE_GAP_BD_ADDR_T
remoteAddr);
```

Description

This function allows application to set the new address of remote device identified by bdHandle.

This API should be used when-

- If peer device is previously bonded with public address and changes its bd address to
- resolvable private address. Application should resolve the device by calling 'CyBle_GapcResolveDevice()' api and set the new address if successfully resolved.
- If device is previously bonded with random, application should call this api to set the new address(public/random).

Parameters

Parameters	Description
uint8 bdHandle	Peer device handle
CYBLE_GAP_BD_ADDR_T remoteAddr	Peer device address, of type CYBLE_GAP_BD_ADDR_T.

Returns

CYBLE_API_RESULT_T : Return value indicates if the function succeeded or failed. Following are the possible error codes.

Error codes	Description
CYBLE_ERROR_OK	On successful operation.
CYBLE_ERROR_INVALID_PARAMETER	On invalid bdHandle

GAP Peripheral Functions

APIs unique to designs configured as a GAP Peripheral role.

A letter 'p' is appended to the API name: CyBle_Gapp

Functions

Function	Description
CyBle_GappStartAdvertisement	This function is used to start the advertisement using the advertisement data set in the Component customizer's GUI. After invoking this API, the device will... more
CyBle_GappStopAdvertisement	This function can be used to exit from discovery mode. After the execution of this function, there will no longer be any advertisements. On stopping... more
CyBle_GappEnterDiscoveryMode	This function sets the device into discoverable mode. In the discoverable mode, based on the parameters passed to this function, the BLE Device starts advertisement... more
CyBle_GappExitDiscoveryMode	This function is used to exit from discoverable mode. This is a non-blocking function. After the execution of this function, the device stops advertising. On... more
CyBle_GappAuthReqReply	This function is used to pass security information for authentication in reply to an authentication request from the master device. It should be invoked on... more

CyBle_GappStartAdvertisement

Prototype

```
CYBLE_API_RESULT_T CyBle_GappStartAdvertisement(uint8 advertisingIntervalType);
```

Description

This function is used to start the advertisement using the advertisement data set in the Component customizer's GUI. After invoking this API, the device will be available for connection by the devices configured for GAP central role. It is only included if the device is configured for GAP Peripheral or GAP Peripheral + Central role.

On start of advertisement, GAP Peripheral receives the CYBLE_EVT_GAPP_ADVERTISEMENT_START_STOP event. The following events are possible on invoking this function:

- CYBLE_EVT_GAP_DEVICE_CONNECTED: If the device connects to remote GAP Central device
- CYBLE_EVT_TIMEOUT: If no device in GAP Central mode connects to this device within the specified timeout limit. Stack automatically initiate stop advertising when Slow advertising was initiated, or starts Slow advertising after Fast advertising timeout occur.

- CYBLE_EVT_GAPP_ADVERTISEMENT_START_STOP: If device started or stopped advertising. Use CyBle_GetState() to determine the state. Sequential advertising could be started when CYBLE_STATE_DISCONNECTED state is returned.

Parameters

Parameters	Description
uint8 advertisingIntervalType	Fast or slow advertising interval with timings entered in Advertising settings section of the customizer. • CYBLE_ADVERTISING_FAST 0x00u • CYBLE_ADVERTISING_SLOW 0x01u • CYBLE_ADVERTISING_CUSTOM 0x02u

Returns

CYBLE_API_RESULT_T : Return value indicates if the function succeeded or failed. Following are the possible error codes.

Error codes	Description
CYBLE_ERROR_OK	On successful operation.
CYBLE_ERROR_INVALID_PARAMETER	On passing an invalid parameter.

CyBle_GappStopAdvertisement

Prototype

```
void CyBle_GappStopAdvertisement(void);
```

Description

This function can be used to exit from discovery mode. After the execution of this function, there will no longer be any advertisements. On stopping advertising, GAP Peripheral receives CYBLE_EVT_GAPP_ADVERTISEMENT_START_STOP event. It is expected that the application layer tracks the function call performed before occurrence of this event as this event can occur on making a call to Cy_BleGappStartAdvertisement(), CyBle_GappEnterDiscoveryMode(), or CyBle_GappStartAdvertisement() functions as well.

The following event occurs on invoking this function:

- CYBLE_EVT_GAPP_ADVERTISEMENT_START_STOP

Returns

None

CyBle_GappEnterDiscoveryMode

Prototype

```
CYBLE_API_RESULT_T CyBle_GappEnterDiscoveryMode(CYBLE_GAPP_DISC_MODE_INFO_T* advInfo);
```

Description

This function sets the device into discoverable mode. In the discoverable mode, based on the parameters passed to this function, the BLE Device starts advertisement and can respond to scan requests. This is a non-blocking function. It is to be used by the device in 'GAP Peripheral' mode of operation to set parameters essential for starting advertisement procedure.

On start of advertisement, the GAP Peripheral receives CYBLE_EVT_GAPP_ADVERTISEMENT_START_STOP event. The following two events can occur on invoking this function.

- CYBLE_EVT_GAP_DEVICE_CONNECTED - If the device connects to a GAP Central.
- CYBLE_EVT_TIMEOUT - If no device in 'GAP Central' mode connects to this device within the
- specified timeout limit. This event can occur if 'advInfo ->discMode' is equal to CYBLE_GAPP_LTD_DISC_MODE or CYBLE_GAPP_GEN_DISC_MODE. 'advInfo-> advTo' specifies the timeout duration. Set the 'advInfo-> advTo' to 0 when 'advInfo -> discMode' is set to CYBLE_GAPP_GEN_DISC_MODE so that the timeout event does not occur and the advertisement continues until the CyBle_GappExitDiscoveryMode() function is invoked.

Parameters

Parameters	Description
CYBLE_GAPP_DISC_MODE_INFO_T* advInfo	Structure of type CYBLE_GAPP_DISC_MODE_INFO_T, which contains the advertisement parameters

Returns

CYBLE_API_RESULT_T : Return value indicates if the function succeeded or failed. Following are the possible error codes.

Error codes	Description
CYBLE_ERROR_OK	On successful operation.
CYBLE_ERROR_INVALID_PARAMETER	On specifying null pointer for 'advInfo' or if any of the elements of this structure have invalid values.

CyBle_GappExitDiscoveryMode

Prototype

```
void CyBle_GappExitDiscoveryMode(void);
```

Description

This function is used to exit from discoverable mode. This is a non-blocking function. After the execution of this function, the device stops advertising.

On stopping advertising, GAP Peripheral receives CYBLE_EVT_GAPP_ADVERTISEMENT_START_STOP event. It is expected that the application layer keeps track of the function call performed before occurrence of this event, as this event can occur on making a call to the CyBle_GappEnterDiscoveryMode () function as well.

Returns

None

CyBle_GappAuthReqReply

Prototype

```
CYBLE_API_RESULT_T CyBle_GappAuthReqReply(uint8 bdHandle, CYBLE_GAP_AUTH_INFO_T *
authInfo);
```

Description

This function is used to pass security information for authentication in reply to an authentication request from the master device. It should be invoked on receiving CYBLE_EVT_GAP_AUTH_REQ event. Events shown in the following table may be received by the application based on the authentication result.

Event Parameter	Description
CYBLE_EVT_TIMEOUT .	With error code CYBLE_GAP_PAIRING_PROCESS_TO on invoking CyBle_GappAuthReqReply() or CyBle_GapAuthReq() if there is no response from the peer device
CYBLE_EVT_GAP_AUTH_COMPLETE	Pointer to structure of type 'CYBLE_GAP_AUTH_INFO_T' is returned as parameter to both the peer devices on successful authentication.
CYBLE_EVT_GAP_AUTH_FAILED	Received by both GAP Central and Peripheral devices (peers) on authentication failure. Data is of type CYBLE_GAP_AUTH_FAILED_REASON_T.
CYBLE_ERROR_INSUFFICIENT_RESOURCES	On bonded device is full and application tries to initiate pairing with bonding enable.

Parameters

Parameters	Description
uint8 bdHandle	Peer device handle.
CYBLE_GAP_AUTH_INFO_T * authInfo	Pointer to a variable containing security information of the device of type CYBLE_GAP_AUTH_INFO_T.

Returns

CYBLE_API_RESULT_T : Return value indicates if the function succeeded or failed. Following are the possible error codes.

Error codes	Description
CYBLE_ERROR_OK	On successful operation.
CYBLE_ERROR_INVALID_PARAMETER	On specifying null pointer for 'advInfo' or if any of the element of this structure has an invalid value.
CYBLE_ERROR_MEMORY_ALLOCATION_FAILED	Memory allocation failed
CYBLE_ERROR_NO_DEVICE_ENTITY	Device identified using 'bdHandle' does not exist.

GAP Definitions and Data Structures

Contains the GAP specific definitions and data structures used in the GAP APIs.

Enumerations

Enumeration	Description
CYBLE_GAP_SEC_LEVEL_T	Security Levels
CYBLE_GAP_ADDR_TYPE_T	GAP address type
CYBLE_GAP_ADV_ASSIGN_NUMBERS	Advertisement SIG assigned numbers
CYBLE_GAP_AUTH_FAILED_REASON_T	Authentication Failed Error Codes
CYBLE_GAP_IOCAP_T	IO capability
CYBLE_GAPC_ADV_EVENT_T	Advertisement event type
CYBLE_GAPP_ADV_T	Advertisement type

Structures

Structure	Description
CYBLE_GAP_SMP_KEY_DIST_T	Security Manager Key Distribution data
CYBLE_GAP_AUTH_INFO_T	Authentication Parameters Information

Structure	Description
CYBLE_GAP_BD_ADDR_T	Bluetooth Device Address
CYBLE_GAP_BONDED_DEV_ADDR_LIST_T	Bluetooth Bonded Device Address list
CYBLE_GAP_CONN_PARAM_UPDATED_IN_CONTROLLER_T	Current Connection Parameters used by controller
CYBLE_GAP_CONN_UPDATE_PARAM_T	GAP Connection Update parameters
CYBLE_GAP_PASSKEY_DISP_INFO_T	Passkey display information
CYBLE_GAPC_ADV_REPORT_T	Advertisement report received by GAP Central
CYBLE_GAPC_CONN_PARAM_T	Connection parameters at the GAP Central end
CYBLE_GAPC_DISC_INFO_T	Discovery information collected by Client
CYBLE_GAPC_T	GAP Service Characteristics server's GATT DB handles structure type
CYBLE_GAPP_DISC_DATA_T	Advertising data
CYBLE_GAPP_DISC_MODE_INFO_T	Advertising information
CYBLE_GAPP_DISC_PARAM_T	Advertising parameters
CYBLE_GAPP_SCAN_RSP_DATA_T	Scan response data

CYBLE_GAP_SEC_LEVEL_T

Prototype

```
typedef enum {
  CYBLE_GAP_SEC_LEVEL_1 = 0x00u,
  CYBLE_GAP_SEC_LEVEL_2,
  CYBLE_GAP_SEC_LEVEL_3,
  CYBLE_GAP_SEC_LEVEL_MASK = 0x0Fu
} CYBLE_GAP_SEC_LEVEL_T;
```

Description

Security Levels

Members

Members	Description
CYBLE_GAP_SEC_LEVEL_1 = 0x00u	Level 1 Mode 1 - No Security (No Authentication & No Encryption) Mode 2 - N/A
CYBLE_GAP_SEC_LEVEL_2	Level 2

	Mode 1 - Unauthenticated pairing with encryption (No MITM) Mode 2 - Unauthenticated pairing with data signing (No MITM)
CYBLE_GAP_SEC_LEVEL_3	Level 3 Mode 1 - Authenticated pairing with encryption (With MITM) Mode 2 - Authenticated pairing with data signing (With MITM)
CYBLE_GAP_SEC_LEVEL_MASK = 0x0Fu	LE Security Level Mask

CYBLE_GAP_SMP_KEY_DIST_T

Prototype

```
typedef struct {
  uint8 ltkInfo[CYBLE_GAP_SMP_LTK_SIZE];
  uint8 midInfo[CYBLE_GAP_SMP_MID_INFO_SIZE];
  uint8 irkInfo[CYBLE_GAP_SMP_IRK_SIZE];
  uint8 idAddrInfo[CYBLE_GAP_SMP_IDADDR_DATA_SIZE];
  uint8 csrkInfo[CYBLE_GAP_SMP_CSRK_SIZE];
} CYBLE_GAP_SMP_KEY_DIST_T;
```

Description

Security Manager Key Distribution data

Members

Members	Description
uint8 ltkInfo[CYBLE_GAP_SMP_LTK_SIZE];	Long Term Key
uint8 midInfo[CYBLE_GAP_SMP_MID_INFO_SIZE];	Encrypted Diversifier and Randdom Number
uint8 irkInfo[CYBLE_GAP_SMP_IRK_SIZE];	Identity Resolving Key
uint8 idAddrInfo[CYBLE_GAP_SMP_IDADDR_DATA_SIZE];	Public device/Static Random address type
uint8 csrkInfo[CYBLE_GAP_SMP_CSRK_SIZE];	Connection Signature Resolving Key

CYBLE_GAP_ADDR_TYPE_T

Prototype

```
typedef enum {
  CYBLE_GAP_RANDOM_PRIV_NON_RESOLVABLE_ADDR = 0x00u,
  CYBLE_GAP_RANDOM_PRIV_RESOLVABLE_ADDR = 0x01u,
  CYBLE_GAP_PUBLIC_ADDR = 0x02u,
  CYBLE_GAP_RANDOM_STATIC_ADDR = 0x03u
} CYBLE_GAP_ADDR_TYPE_T;
```

Description

GAP address type

Members

Members	Description
CYBLE_GAP_RANDOM_PRIV_NON_RESOLVABLE_ADDR = 0x00u	Random private non-resolvable address
CYBLE_GAP_RANDOM_PRIV_RESOLVABLE_ADDR = 0x01u	Random private resolvable address
CYBLE_GAP_PUBLIC_ADDR = 0x02u	Public address
CYBLE_GAP_RANDOM_STATIC_ADDR = 0x03u	Random static address

CYBLE_GAP_ADV_ASSIGN_NUMBERS

Prototype

```
typedef enum {
  CYBLE_GAP_ADV_FLAGS = 0x01u,
  CYBLE_GAP_ADV_INCOMPL_16UUID,
  CYBLE_GAP_ADV_COMPL_16UUID,
  CYBLE_GAP_ADV_INCOMPL_32_UUID,
  CYBLE_GAP_ADV_COMPL_32_UUID,
  CYBLE_GAP_ADV_INCOMPL_128_UUID,
  CYBLE_GAP_ADV_COMPL_128_UUID,
  CYBLE_GAP_ADV_SHORT_NAME,
  CYBLE_GAP_ADV_COMPL_NAME,
  CYBLE_GAP_ADV_TX_PWR_LVL,
  CYBLE_GAP_ADV_CLASS_OF_DEVICE = 0x0Du,
  CYBLE_GAP_ADV_SMPL_PAIR_HASH_C,
  CYBLE_GAP_ADV_SMPL_PAIR_RANDOM_R,
  CYBLE_GAP_ADV_DEVICE_ID,
  CYBLE_GAP_ADV_SCRT_MNGR_TK_VAL = 0x10u,
  CYBLE_GAP_ADV_SCRT_MNGR_OOB_FLAGS,
  CYBLE_GAP_ADV_SLAVE_CONN_INTRV_RANGE,
  CYBLE_GAP_ADV_SOLICIT_16UUID = 0x14u,
  CYBLE_GAP_ADV_SOLICIT_128UUID,
  CYBLE_GAP_ADV_SRVC_DATA_16UUID,
  CYBLE_GAP_ADV_PUBLIC_TARGET_ADDR,
  CYBLE_GAP_ADV_RANDOM_TARGET_ADDR,
  CYBLE_GAP_ADV_APPEARANCE,
  CYBLE_GAP_ADV_ADVERT_INTERVAL,
  CYBLE_GAP_ADV_LE_BT_DEVICE_ADDR,
  CYBLE_GAP_ADV_LE_ROLE,
  CYBLE_GAP_ADV_SMPL_PAIR_HASH_C256,
  CYBLE_GAP_ADV_SMPL_PAIR_RANDOM_R256,
  CYBLE_GAP_ADV_SOLICIT_32UUID,
  CYBLE_GAP_ADV_SRVC_DATA_32UUID,
  CYBLE_GAP_ADV_SRVC_DATA_128UUID,
  CYBLE_GAP_ADV_3D_INFO_DATA = 0x3D
} CYBLE_GAP_ADV_ASSIGN_NUMBERS;
```

Description

Advertisement SIG assigned numbers

Members

Members	Description
CYBLE_GAP_ADV_FLAGS = 0x01u	Flags
CYBLE_GAP_ADV_INCOMPL_16UUID	Incomplete List of 16-bit Service Class UUIDs
CYBLE_GAP_ADV_COMPL_16UUID	Complete List of 16-bit Service Class UUIDs
CYBLE_GAP_ADV_INCOMPL_32_UUID	Incomplete List of 32-bit Service Class UUIDs
CYBLE_GAP_ADV_COMPL_32_UUID	Complete List of 32-bit Service Class UUIDs
CYBLE_GAP_ADV_INCOMPL_128_UUID	Incomplete List of 128-bit Service Class UUIDs
CYBLE_GAP_ADV_COMPL_128_UUID	Complete List of 128-bit Service Class UUIDs
CYBLE_GAP_ADV_SHORT_NAME	Shortened Local Name
CYBLE_GAP_ADV_COMPL_NAME	Complete Local Name
CYBLE_GAP_ADV_TX_PWR_LVL	Tx Power Level
CYBLE_GAP_ADV_CLASS_OF_DEVICE = 0x0Du	Class of Device
CYBLE_GAP_ADV_SMPL_PAIR_HASH_C	Simple Pairing Hash C
CYBLE_GAP_ADV_SMPL_PAIR_RANDOM_R	Simple Pairing Randomizer R
CYBLE_GAP_ADV_DEVICE_ID	Device ID
CYBLE_GAP_ADV_SCRT_MNGR_TK_VAL = 0x10u	Security Manager TK Value
CYBLE_GAP_ADV_SCRT_MNGR_OOB_FLAGS	Security Manager Out of Band Flags
CYBLE_GAP_ADV_SLAVE_CONN_INTRV_RANGE	Slave Connection Interval Range
CYBLE_GAP_ADV_SOLICIT_16UUID = 0x14u	List of 16-bit Service Solicitation UUIDs
CYBLE_GAP_ADV_SOLICIT_128UUID	List of 128-bit Service Solicitation UUIDs
CYBLE_GAP_ADV_SRVC_DATA_16UUID	Service Data - 16-bit UUID
CYBLE_GAP_ADV_PUBLIC_TARGET_ADDR	Public Target Address
CYBLE_GAP_ADV_RANDOM_TARGET_ADDR	Random Target Address
CYBLE_GAP_ADV_APPEARANCE	Appearance
CYBLE_GAP_ADV_ADVERT_INTERVAL	Advertising Interval
CYBLE_GAP_ADV_LE_BT_DEVICE_ADDR	LE Bluetooth Device Address
CYBLE_GAP_ADV_LE_ROLE	LE Role

CYBLE_GAP_ADV_SMPL_PAIR_HASH_C256	Simple Pairing Hash C-256
CYBLE_GAP_ADV_SMPL_PAIR_RANDOM_R256	Simple Pairing Randomizer R-256
CYBLE_GAP_ADV_SOLICIT_32UUID	List of 32-bit Service Solicitation UUIDs
CYBLE_GAP_ADV_SRVC_DATA_32UUID	Service Data - 32-bit UUID
CYBLE_GAP_ADV_SRVC_DATA_128UUID	Service Data - 128-bit UUID
CYBLE_GAP_ADV_3D_INFO_DATA = 0x3D	3D Information Data

CYBLE_GAP_AUTH_FAILED_REASON_T

Prototype

```
typedef enum {
  CYBLE_GAP_AUTH_ERROR_NONE = 0x00u,
  CYBLE_GAP_AUTH_ERROR_PASSKEY_ENTRY_FAILED,
  CYBLE_GAP_AUTH_ERROR_OOB_DATA_NOT_AVAILABLE,
  CYBLE_GAP_AUTH_ERROR_AUTHENTICATION_REQ_NOT_MET,
  CYBLE_GAP_AUTH_ERROR_CONFIRM_VALUE_NOT_MATCH,
  CYBLE_GAP_AUTH_ERROR_PAIRING_NOT_SUPPORTED,
  CYBLE_GAP_AUTH_ERROR_INSUFFICIENT_ENCRYPTION_KEY_SIZE,
  CYBLE_GAP_AUTH_ERROR_COMMAND_NOT_SUPPORTED,
  CYBLE_GAP_AUTH_ERROR_UNSPECIFIED_REASON,
  CYBLE_GAP_AUTH_ERROR_REPEATED_ATTEMPTS,
  CYBLE_GAP_AUTH_ERROR_INVALID_PARAMETERS = 0x0Au,
  CYBLE_GAP_AUTH_ERROR_AUTHENTICATION_TIMEOUT = 0x15u,
  CYBLE_GAP_AUTH_ERROR_LINK_DISCONNECTED = 0x18u
} CYBLE_GAP_AUTH_FAILED_REASON_T;
```

Description

Authentication Failed Error Codes

Members

Members	Description
CYBLE_GAP_AUTH_ERROR_NONE = 0x00u	No Error
CYBLE_GAP_AUTH_ERROR_PASSKEY_ENTRY_FAILED	User input of passkey failed, for example, the user cancelled the operation
CYBLE_GAP_AUTH_ERROR_OOB_DATA_NOT_AVAILABLE	Out Of Band data is not available, applicable if NFC is supported
CYBLE_GAP_AUTH_ERROR_AUTHENTICATION_REQ_NOT_MET	Pairing procedure cannot be performed as authentication requirements cannot be met due to IO capabilities of one or both devices.

CYBLE_GAP_AUTH_ERROR_CONFIRM_VALUE_NOT_MATCH	Confirm value does not match the calculated compare value
CYBLE_GAP_AUTH_ERROR_PAIRING_NOT_SUPPORTED	Pairing is not supported by the device
CYBLE_GAP_AUTH_ERROR_INSUFFICIENT_ENCRYPTION_KEY_SIZE	Insufficient key size for the security requirements of this device
CYBLE_GAP_AUTH_ERROR_COMMAND_NOT_SUPPORTED	command received is not supported
CYBLE_GAP_AUTH_ERROR_UNSPECIFIED_REASON	Pairing failed due to an unspecified reason
CYBLE_GAP_AUTH_ERROR_REPEATED_ATTEMPTS	Pairing or authentication procedure is disallowed because too little time has elapsed since last pairing request or security request.
CYBLE_GAP_AUTH_ERROR_INVALID_PARAMETERS = 0x0Au	Invalid Parameters in Request – Invalid Command length and Parameter value outside range
CYBLE_GAP_AUTH_ERROR_AUTHENTICATION_TIMEOUT = 0x15u	Authentication process timeout, if pairing timeout happens for first time, application can choose to re-initiate the pairing procedure. If timeout occurs again, app may choose to disconnect peer device.
CYBLE_GAP_AUTH_ERROR_LINK_DISCONNECTED = 0x18u	Link disconnected

CYBLE_GAP_AUTH_INFO_T

Prototype

```
typedef struct {
  uint8 security;
  uint8 bonding;
  uint8 ekeySize;
  CYBLE_GAP_AUTH_FAILED_REASON_T authErr;
} CYBLE_GAP_AUTH_INFO_T;
```

Description

Authentication Parameters Information

Members

Members	Description
uint8 security;	Security Mode setting will be as follows: (CYBLE_GAP_SEC_MODE_1 | CYBLE_GAP_SEC_LEVEL_1)

	(CYBLE_GAP_SEC_MODE_1 | CYBLE_GAP_SEC_LEVEL_2) (CYBLE_GAP_SEC_MODE_1 | CYBLE_GAP_SEC_LEVEL_3) (CYBLE_GAP_SEC_MODE_2 | CYBLE_GAP_SEC_LEVEL_2) (CYBLE_GAP_SEC_MODE_2 | CYBLE_GAP_SEC_LEVEL_3)
uint8 bonding;	Bonding type setting: CYBLE_GAP_BONDING_NONE CYBLE_GAP_BONDING

uint8 ekeySize;	Encryption Key Size (octets)
	Minimum = 7
	maximum = 16

CYBLE_GAP_AUTH_FAILED_REASON_T authErr;	Parameter to say it authentication is accepted or rejected with reason. accepted = CYBLE_GAP_AUTH_ERROR_NONE or error code CYBLE_GAP_AUTH_FAILED_REASON_T.

CYBLE_GAP_BD_ADDR_T

Prototype

```
typedef struct {
  uint8 bdAddr[CYBLE_GAP_BD_ADDR_SIZE];
  uint8 type;
} CYBLE_GAP_BD_ADDR_T;
```

Description

Bluetooth Device Address

Members

Members	Description
uint8 bdAddr[CYBLE_GAP_BD_ADDR_SIZE];	Bluetooth device address
uint8 type;	public = 0, Random = 1

CYBLE_GAP_BONDED_DEV_ADDR_LIST_T

Prototype

```
typedef struct {
  uint8 count;
  CYBLE_GAP_BD_ADDR_T bdAddrList[CYBLE_GAP_MAX_BONDED_DEVICE]; }
CYBLE_GAP_BONDED_DEV_ADDR_LIST_T;
```

Description

Bluetooth Bonded Device Address list

Members

Members	Description
uint8 count;	Number of bonded devices
CYBLE_GAP_BD_ADDR_T bdAddrList[CYBLE_GAP_MAX_BONDED_DEVICE];	Pointer to list of bluetooth device addresses of bonded devices, of type 'CYBLE_GAP_BD_ADDR_T'. 'CYBLE_GAP_MAX_BONDED_DEVICE' is a '#define' to be defined during build-time.

CYBLE_GAP_CONN_PARAM_UPDATED_IN_CONTROLLER_T

Prototype

```
typedef struct {
  uint8 status;
  uint16 connIntv;
  uint16 connLatency;
  uint16 supervisionTO;
} CYBLE_GAP_CONN_PARAM_UPDATED_IN_CONTROLLER_T;
```

Description

Current Connection Parameters used by controller

Members

Members	Description
uint8 status;	status corresponding to this event will be HCI error code as defined in BLE spec 4.1
uint16 connIntv;	Connection interval used on this connection. Range: 0x0006 to 0x0C80 Time Range: 7.5 ms to 4 sec
uint16 connLatency;	Slave latency for the connection in number of connection events. Range: 0x0000 to 0x01F4
uint16 supervisionTO;	Supervision timeout for the LE Link. Supervision timeout will be supervisionTO * 10 ms Time Range: 100 msec to 32 secs

CYBLE_GAP_CONN_UPDATE_PARAM_T

Prototype

```
typedef struct {
  uint16 connIntvMin;
  uint16 connIntvMax;
  uint16 connLatency;
  uint16 supervisionTO;
} CYBLE_GAP_CONN_UPDATE_PARAM_T;
```

Description

GAP Connection Update parameters

Members

Members	Description
uint16 connIntvMin;	Minimum value for the connection event interval. This shall be less than or equal to conn_Interval_Max. Minimum connection interval will be connIntvMin * 1.25 ms Time Range: 7.5 ms to 4 sec
uint16 connIntvMax;	Maximum value for the connection event interval. This shall be greater than or equal to conn_Interval_Min. Maximum connection interval will be connIntvMax * 1.25 ms Time Range: 7.5 ms to 4 sec
uint16 connLatency;	Slave latency for the connection in number of connection events. Range: 0x0000 to 0x01F4
uint16 supervisionTO;	Supervision timeout for the LE Link. Supervision timeout will be supervisionTO * 10 ms Time Range: 100 msec to 32 secs

CYBLE_GAP_IOCAP_T

Prototype

```
typedef enum {
  CYBLE_GAP_IOCAP_DISPLAY_ONLY = 0x00u,
  CYBLE_GAP_IOCAP_DISPLAY_YESNO,
  CYBLE_GAP_IOCAP_KEYBOARD_ONLY,
  CYBLE_GAP_IOCAP_NOINPUT_NOOUTPUT,
  CYBLE_GAP_IOCAP_KEYBOARD_DISPLAY
} CYBLE_GAP_IOCAP_T;
```

Description

IO capability

Members

Members	Description
CYBLE_GAP_IOCAP_DISPLAY_ONLY = 0x00u	Platform supports only a mechanism to display or convey only 6 digit number to user.
CYBLE_GAP_IOCAP_DISPLAY_YESNO	The device has a mechanism whereby the user can indicate 'yes' or 'no'.
CYBLE_GAP_IOCAP_KEYBOARD_ONLY	Platform supports a numeric keyboard that can input the

	numbers '0' through '9' and a confirmation key(s) for 'yes' and 'no'.
CYBLE_GAP_IOCAP_NOINPUT_NOOUTPUT	Platform does not have the ability to display or communicate a 6 digit decimal number.
CYBLE_GAP_IOCAP_KEYBOARD_DISPLAY	Platform supports a mechanism through which 6 digit numeric value can be displayed and numeric keyboard that can input the numbers '0' through '9'.

CYBLE_GAP_PASSKEY_DISP_INFO_T

Prototype

```
typedef struct {
  uint8 bdHandle;
  uint32 passkey;
} CYBLE_GAP_PASSKEY_DISP_INFO_T;
```

Description

Passkey display information

Members

Members	Description
uint8 bdHandle;	bd handle of the remote device
uint32 passkey;	size = 6, not null terminated

CYBLE_GAPC_ADV_EVENT_T

Prototype

```
typedef enum {
  CYBLE_GAPC_CONN_UNDIRECTED_ADV = 0x00u,
  CYBLE_GAPC_CONN_DIRECTED_ADV,
  CYBLE_GAPC_SCAN_UNDIRECTED_ADV,
  CYBLE_GAPC_NON_CONN_UNDIRECTED_ADV,
  CYBLE_GAPC_SCAN_RSP
} CYBLE_GAPC_ADV_EVENT_T;
```

Description

Advertisement event type

Members

Members	Description
CYBLE_GAPC_CONN_UNDIRECTED_ADV = 0x00u	Connectable undirected advertising

CYBLE_GAPC_CONN_DIRECTED_ADV	Connectable directed advertising
CYBLE_GAPC_SCAN_UNDIRECTED_ADV	Scannable undirected advertising
CYBLE_GAPC_NON_CONN_UNDIRECTED_ADV	Non connectable undirected advertising
CYBLE_GAPC_SCAN_RSP	Scan Response

CYBLE_GAPC_ADV_REPORT_T

Prototype

```
typedef struct {
  CYBLE_GAPC_ADV_EVENT_T eventType;
  uint8 peerAddrType;
  uint8* peerBdAddr;
  uint8 dataLen;
  uint8* data;
  int8 rssi;
} CYBLE_GAPC_ADV_REPORT_T;
```

Description

Advertisement report received by GAP Central

Members

Members	Description
CYBLE_GAPC_ADV_EVENT_T eventType;	Advertisement event type Connectable undirected advertising = 0x00 Connectable directed advertising = 0x01 Scannable undirected advertising = 0x02 Non connectable undirected advertising = 0x03 Scan Response = 0x04
uint8 peerAddrType;	bd address type of the device advertising. CYBLE_GAP_ADDR_TYPE_PUBLIC (Public device address) CYBLE_GAP_ADDR_TYPE_RANDOM (Random device address)
uint8* peerBdAddr;	Public Device Address or Random Device Address for each device which responded to scanning.
uint8 dataLen;	length of the data for each device that responded to scanning
uint8* data;	Pointer to advertising or scan response data
int8 rssi;	Rssi of the responding device. Range: -85 <= N <= 0 Units: dBm

CYBLE_GAPC_CONN_PARAM_T

Prototype

```
typedef struct {
  uint16 scanIntv;
  uint16 scanWindow;
  uint8 initiatorFilterPolicy;
  uint8 peerBdAddr[CYBLE_GAP_BD_ADDR_SIZE];
  uint8 peerAddrType;
  uint8 ownAddrType;
  uint16 connIntvMin;
  uint16 connIntvMax;
  uint16 connLatency;
  uint16 supervisionTO;
  uint16 minCeLength;
  uint16 maxCeLength;
} CYBLE_GAPC_CONN_PARAM_T;
```

Description

Connection parameters at the GAP Central end

Members

Members	Description
uint16 scanIntv;	The time interval from when last LE scan is started until next subsequent LE scan. Time Range: 2.5 ms to 10.24 sec.
uint16 scanWindow;	The time duration of scanning to be performed Time Range: 2.5 ms to 10.24 sec
uint8 initiatorFilterPolicy;	Filter policies to be applied during connection procedure CYBLE_GAPC_CONN_ALL (White list is not used to determine which advertiser to connect. Peer address is used) CYBLE_GAPC_CONN_WHITELIST (White list is used to determine which advertiser to connect to. Peer address shall be ignored)
uint8 peerBdAddr[CYBLE_GAP_BD_ADDR_SIZE];	Peer's bd address with whom connection to be established
uint8 peerAddrType;	Peer's bd address type CYBLE_GAP_ADDR_TYPE_PUBLIC (Public device address) CYBLE_GAP_ADDR_TYPE_RANDOM (Random device address)
uint8 ownAddrType;	Own bd address type CYBLE_GAP_ADDR_TYPE_PUBLIC (Public device address) CYBLE_GAP_ADDR_TYPE_RANDOM (Random device address)

uint16 connIntvMin;	Minimum value for the connection event interval. This shall be less than or equal to conn_Interval_Max. Minimum connection interval will be connIntvMin * 1.25 ms Time Range: 7.5 ms to 4 sec
uint16 connIntvMax;	Maximum value for the connection event interval. This shall be greater than or equal to conn_Interval_Min. Maximum connection interval will be connIntvMax * 1.25 ms Time Range: 7.5 ms to 4 sec
uint16 connLatency;	Slave latency for the connection in number of connection events. Range: 0x0000 to 0x01F4
uint16 supervisionTO;	Supervision timeout for the LE Link. Supervision timeout will be supervisionTO * 10 ms Time Range: 100 msec to 32 secs
uint16 minCeLength;	Minimum length of connection needed for this LE connection. Range: 0x0000 - 0xFFFF
uint16 maxCeLength;	Maximum length of connection needed for this LE connection. Range: 0x0000 - 0xFFFF

CYBLE_GAPC_DISC_INFO_T

Prototype

```
typedef struct {
  uint8 discProcedure;
  uint8 scanType;
  uint16 scanIntv;
  uint16 scanWindow;
  uint8 ownAddrType;
  uint8 scanFilterPolicy;
  uint16 scanTo;
  uint8 filterDuplicates;
} CYBLE_GAPC_DISC_INFO_T;
```

Description

Discovery information collected by Client

Members

Members	Description
uint8 discProcedure;	Observation and discovery procedure. CYBLE_GAPC_OBSER_PROCEDURE (Observation procedure) CYBLE_GAPC_LTD_DISC_PROCEDURE (Limited discovery procedure) CYBLE_GAPC_GEN_DISC_PROCEDURE (General discovery procedure)

uint8 scanType;	Type of scan to perform CYBLE_GAPC_PASSIVE_SCANNING (Passive Scanning) CYBLE_GAPC_ACTIVE_SCANNING (Active scanning)
uint16 scanIntv;	The time interval from when last LE scan is started until next subsequent LE scan. Time Range: 2.5 ms to 10.24 sec.
uint16 scanWindow;	The time duration of scanning to be performed Time Range: 2.5 ms to 10.24 sec
uint8 ownAddrType;	Own BD Address Type CYBLE_GAP_ADDR_TYPE_PUBLIC (Public device address) CYBLE_GAP_ADDR_TYPE_RANDOM (Random device address)
uint8 scanFilterPolicy;	Filter policies to be applied during scanning procedure CYBLE_GAPC_ADV_ACCEPT_ALL_PKT (Accept all advertisement packets) CYBLE_GAPC_ADV_ACCEPT_WHITELIST_PKT (Ignore advertisement packets from devices not in the White List)
uint16 scanTo;	Scan timeout. Timeout is in seconds and none zero. If timeout is set as 0, then there will not be any timeout scanTo can be used for all GAP timeouts related to Central operation.
uint8 filterDuplicates;	Filter Duplicate Advertisement. The Filter Duplicates parameter controls whether the Link Layer shall filter duplicate advertising reports to the Host, or if the Link Layer should generate advertising reports for each packet received. CYBLE_GAPC_FILTER_DUP_DISABLE (Duplicate filtering disabled) CYBLE_GAPC_FILTER_DUP_ENABLE (Duplicate filtering enabled) By default, duplicate filtering is enabled

CYBLE_GAPC_T

Prototype

```
typedef struct {
  CYBLE_GATT_DB_ATTR_HANDLE_T deviceNameCharHandle;
  CYBLE_GATT_DB_ATTR_HANDLE_T appearanceCharHandle;
  CYBLE_GATT_DB_ATTR_HANDLE_T periphPrivacyCharHandle;
  CYBLE_GATT_DB_ATTR_HANDLE_T reconnAddrCharHandle;
  CYBLE_GATT_DB_ATTR_HANDLE_T prefConnParamCharHandle;
} CYBLE_GAPC_T;
```

Description

GAP Service Characteristics server's GATT DB handles structure type

Members

Members	Description
CYBLE_GATT_DB_ATTR_HANDLE_T	Handle of the GAPS Device Name Characteristic

deviceNameCharHandle;	
CYBLE_GATT_DB_ATTR_HANDLE_T appearanceCharHandle;	Handle of the GAPS Appearance Characteristic
CYBLE_GATT_DB_ATTR_HANDLE_T periphPrivacyCharHandle;	Handle of the GAPS Peripheral Privacy Flag Parameters Characteristic
CYBLE_GATT_DB_ATTR_HANDLE_T reconnAddrCharHandle;	Handle of the GAPS Reconnection Address Characteristic
CYBLE_GATT_DB_ATTR_HANDLE_T prefConnParamCharHandle;	Handle of the GAPS Peripheral Preferred Connection Parameters Characteristic

CYBLE_GAPP_ADV_T

Prototype

```
typedef enum {
  CYBLE_GAPP_CONNECTABLE_UNDIRECTED_ADV = 0x00u,
  CYBLE_GAPP_CONNECTABLE_HIGH_DC_DIRECTED_ADV,
  CYBLE_GAPP_SCANNABLE_UNDIRECTED_ADV,
  CYBLE_GAPP_NON_CONNECTABLE_UNDIRECTED_ADV,
  CYBLE_GAPP_CONNECTABLE_LOW_DC_DIRECTED_ADV
} CYBLE_GAPP_ADV_T;
```

Description

Advertisement type

Members

Members	Description
CYBLE_GAPP_CONNECTABLE_UNDIRECTED_ADV = 0x00u	Connectable undirected advertising
CYBLE_GAPP_CONNECTABLE_HIGH_DC_DIRECTED_ADV	Connectable high duty cycle directed advertising
CYBLE_GAPP_SCANNABLE_UNDIRECTED_ADV	Scannable undirected advertising
CYBLE_GAPP_NON_CONNECTABLE_UNDIRECTED_ADV	Non connectable undirected advertising
CYBLE_GAPP_CONNECTABLE_LOW_DC_DIRECTED_ADV	Connectable low duty cycle directed advertising

CYBLE_GAPP_DISC_DATA_T

Prototype

```
typedef struct {
  uint8 advData[CYBLE_GAP_MAX_ADV_DATA_LEN];
  uint8 advDataLen;
} CYBLE_GAPP_DISC_DATA_T;
```

Description

Advertising data

Members

Members	Description
uint8 advData[CYBLE_GAP_MAX_ADV_DATA_LEN];	GAP Advertisement Parameters which includes Flags, Service UUIDs and short name
uint8 advDataLen;	Length of the advertising data. This should be made zero if there is no data

CYBLE_GAPP_DISC_MODE_INFO_T

Prototype

```
typedef struct {
  uint8 discMode;
  CYBLE_GAPP_DISC_PARAM_T* advParam;
  CYBLE_GAPP_DISC_DATA_T* advData;
  CYBLE_GAPP_SCAN_RSP_DATA_T* scanRspData;
  uint16 advTo;
} CYBLE_GAPP_DISC_MODE_INFO_T;
```

Description

Advertising information

Members

Members	Description
uint8 discMode;	Broadcaster and discoverable mode CYBLE_GAPP_NONE_DISC_BROADCAST_MODE (Applicable for Broadcaster or non-discoverable mode) CYBLE_GAPP_LTD_DISC_MODE (Limited discovery mode) CYBLE_GAPP_GEN_DISC_MODE (General discovery mode)
CYBLE_GAPP_DISC_PARAM_T* advParam;	Advertisement parameters
CYBLE_GAPP_DISC_DATA_T* advData;	Advertisement data
CYBLE_GAPP_SCAN_RSP_DATA_T* scanRspData;	Scan Response data
uint16 advTo;	Advertisement timeout is in seconds. If timeout is set to 0, then there will not be any timeout. Parameter 'advTo' can be used for all GAP timeouts related to peripheral operation. For General discoverable mode, this timer will be ignored. Application is expected to exit from discoverable

	mode explicitly by calling CyBle_GappExitDiscoveryMode() function. For Limited discoverable mode, 'advTo' should not exceed 180 Sec.

CYBLE_GAPP_DISC_PARAM_T

Prototype

```
typedef struct {
  uint16 advIntvMin;
  uint16 advIntvMax;
  CYBLE_GAPP_ADV_T advType;
  uint8 ownAddrType;
  uint8 directAddrType;
  uint8 directAddr[CYBLE_GAP_BD_ADDR_SIZE];
  uint8 advChannelMap;
  uint8 advFilterPolicy;
} CYBLE_GAPP_DISC_PARAM_T;
```

Description

Advertising parameters

Members

Members	Description
uint16 advIntvMin;	Minimum advertising interval for undirected and low duty cycle directed advertising. Time Range: 20 ms to 10.24 sec
uint16 advIntvMax;	Maximum advertising interval for undirected and low duty cycle directed advertising. Time Range: 20 ms to 10.24 sec
CYBLE_GAPP_ADV_T advType;	Type of advertisement Connectable undirected advertising (0x00) Connectable high duty cycle directed advertising (0x01) Scannable undirected advertising (0x02) Non connectable undirected advertising (0x03) Connectable low duty cycle directed advertising (0x04)
uint8 ownAddrType;	Own BD Address Type CYBLE_GAP_ADDR_TYPE_PUBLIC (Public device address) CYBLE_GAP_ADDR_TYPE_RANDOM (Random device address)
uint8 directAddrType;	Address type of the Bluetooth device address being used for directed advertising, not applicable otherwise CYBLE_GAP_ADDR_TYPE_PUBLIC (Public device address) CYBLE_GAP_ADDR_TYPE_RANDOM (Random device address)

uint8 directAddr[CYBLE_GAP_BD_ADDR_SIZE];	This parameter specifies Bluetooth device address of the device to be connected while using directed advertising. In case of none direct advertising, parameter will be 0
uint8 advChannelMap;	Advertising channels that shall be used when transmitting advertising packets. Channel map selection: Enable channel 37 = bitmask. xxxxxxx1b Enable channel 38 = bitmask. xxxxxx1xb Enable channel 39 = bitmask. xxxxx1xxb
uint8 advFilterPolicy;	Advertising Filter Policy CYBLE_SCAN_ANY_CONN_ANY (Allow Scan Request from Any, Allow Connect Request from Any (Default)) CYBLE_SCAN_WHITELIST_CONN_ANY (Allow Scan Request from White List Only, Allow Connect Request) CYBLE_SCAN_ANY_CONN_WHITELIST (Allow Scan Request from Any, Allow Connect Request from White List Only) CYBLE_SCAN_WHITELIST_CONN_ANY (Allow Scan Request from White List Only, Allow Connect Request from White List Only)

CYBLE_GAPP_SCAN_RSP_DATA_T

Prototype

```
typedef struct {
  uint8 scanRspData[CYBLE_GAP_MAX_SCAN_RSP_DATA_LEN];
  uint8 scanRspDataLen;
} CYBLE_GAPP_SCAN_RSP_DATA_T;
```

Description

Scan response data

Members

Members	Description
uint8 scanRspData[CYBLE_GAP_MAX_SCAN_RSP_DATA_LEN];	Static user data transmitted in scan response. This should be made NULL if there is no data. Maximum length of the data is equal to 31 bytes
uint8 scanRspDataLen;	Length of the scan response data. This should be made zero if there is no data

GATT Functions

The GATT APIs allow access to the Generic Attribute Profile (GATT) layer of the BLE stack. Depending on the chosen GATT role in the GUI, you may use a subset of the supported APIs.

The GATT API names begin with CyBle_Gatt. In addition to this, the APIs also append the GATT role initial letter in the API name.

GATT Client and Server Functions

These are APIs common to both GATT Client role and GATT Server role. You may use them in either roles.

No letter is appended to the API name: CyBle_Gatt

Functions

Function	Description
CyBle_GattGetMtuSize	This function provides the correct MTU used by BLE stack. If function is called after MTU configuration procedure, it will provide the final negotiated MTU... more

CyBle_GattGetMtuSize

Prototype

```
CYBLE_API_RESULT_T CyBle_GattGetMtuSize(uint16* mtu);
```

Description

This function provides the correct MTU used by BLE stack. If function is called after MTU configuration procedure, it will provide the final negotiated MTU else default MTU (23 Bytes).

Parameters

Parameters	Description
uint16* mtu	buffer where Size of MTU will be stored.

Returns

CYBLE_API_RESULT_T : Return value indicates if the function succeeded or failed. Following are the possible error codes.

Error codes	Description
CYBLE_ERROR_OK	On successful operation
CYBLE_ERROR_INVALID_PARAMETER	If invalid parameter passed

GATT Server Functions

APIs unique to designs configured as a GATT Server role.

A letter 's' is appended to the API name: CyBle_Gatts

Functions

Function	Description
CyBle_GattsReInitGattDb	Reinitializes the GATT database.
CyBle_GattsWriteAttributeValue	This function is used to write to the value field of the specified attribute in the GATT database of a GATT Server. This is a... more
CyBle_GattsReadAttributeValue	This function is used to read the value field of the specified attribute from the GATT database in a GATT Server. This is a blocking... more
CyBle_GattsEnableAttribute	This function enables the attribute entry for service or Characteristic logical group in the GATT database registered in BLE Stack. This is a blocking function.... more
CyBle_GattsDisableAttribute	This function disables the attribute entry for service or Characteristic logical group in the GATT database registered in the BLE Stack. This is a blocking... more
CyBle_GattsNotification	This function sends a notification to the peer device when the GATT Server is configured to notify a Characteristic Value to the GATT Client without... more
CyBle_GattsIndication	This function sends an indication to the peer device when the GATT Server is configured to indicate a Characteristic Value to the GATT Client and... more
CyBle_GattsErrorRsp	This function sends an error response to the peer device. The Error Response is used to state that a given request cannot be performed, and... more
CyBle_GattsExchangeMtuRsp	This function sends the GATT Server's MTU size to the GATT Client. This function has to be invoked in response to an Exchange MTU Request... more
CyBle_GattsWriteRsp	This function sends a Write Response from a GATT Server to the GATT Client. This is a non-blocking function. This function has to be invoked... more
CyBle_GattsPrepWriteReqSupport	This function needs to be called after getting CYBLE_EVT_GATTS_PREP_WRITE_REQ event from the BLE Stack to perform necessary initialization in the BLE stack to support prepare... more

CyBle_GattsReInitGattDb

Prototype

```
CYBLE_API_RESULT_T CyBle_GattsReInitGattDb(void);
```

Description

Reinitializes the GATT database.

Returns

CYBLE_API_RESULT_T: An API result states if the API succeeded or failed with error codes:

- CYBLE_ERROR_OK: GATT database was reinitialized successfully
- CYBLE_ERROR_INVALID_STATE: If the function is called in any state except

- CYBLE_STATE_DISCONNECTED.
- Any of the CyBle_GattsDbRegister() stack API function return values.

CyBle_GattsWriteAttributeValue

Prototype

```
CYBLE_GATT_ERR_CODE_T CyBle_GattsWriteAttributeValue(CYBLE_GATT_HANDLE_VALUE_PAIR_T *
handleValuePair, uint16 offset, CYBLE_CONN_HANDLE_T * connHandle, uint8 flags);
```

Description

This function is used to write to the value field of the specified attribute in the GATT database of a GATT Server. This is a blocking function. No event is generated on calling this function.

If a peer device connected to the GATT Server initiates a write operation, this function is executed on the GATT Server. During such a call, the function checks for the attribute permissions (flags) before executing the write operation.

Parameters

Parameters	Description
CYBLE_GATT_HANDLE_VALUE_PAIR_T * handleValuePair	Pointer to handle value pair of type CYBLE_GATT_HANDLE_VALUE_PAIR_T. • 'handleValuePair.attrHandle' is an input for which value has to be written. • 'handleValuePair.value.len' is an input parameter for the length to be written. • 'handleValuePair.value.val' is an input parameter for data buffer. • 'handleValuePair.actualLen' has to be ignored as it is unused in this function.
uint16 offset	Offset at which the data (length in number of bytes) is written.
CYBLE_CONN_HANDLE_T * connHandle	Pointer to the attribute instance handle, of type CYBLE_CONN_HANDLE_T.
uint8 flags	Attribute permissions. Allowed values are, • CYBLE_GATT_DB_LOCALLY_INITIATED • CYBLE_GATT_DB_PEER_INITIATED

Returns

CYBLE_GATT_ERR_CODE_T : Return value indicates if the function succeeded or failed. Following are the possible error codes.

Error codes	Description
CYBLE_GATT_ERR_NONE	On successful operation
CYBLE_GATT_ERR_INVALID_HANDLE	'handleValuePair.attrHandle' is not valid
CYBLE_GATT_ERR_WRITE_NOT_PERMITTED	Write operation is not permitted on this attribute
CYBLE_GATT_ERR_INVALID_OFFSET	Offset value is invalid
CYBLE_GATT_ERR_UNLIKELY_ERROR	Some other error occurred

CyBle_GattsReadAttributeValue

Prototype

```
CYBLE_GATT_ERR_CODE_T CyBle_GattsReadAttributeValue(CYBLE_GATT_HANDLE_VALUE_PAIR_T*
handleValuePair, CYBLE_CONN_HANDLE_T* connHandle, uint8 flags);
```

Description

This function is used to read the value field of the specified attribute from the GATT database in a GATT Server. This is a blocking function. No event is generated on calling this function.

Peer initiated call to this function results in the function checking for attribute permissions before performing this operation.

Parameters

Parameters	Description
CYBLE_GATT_HANDLE_VALUE_PAIR_T* handleValuePair	Pointer to handle value pair of type CYBLE_GATT_HANDLE_VALUE_PAIR_T. • 'handleValuePair.attrHandle' is an input for which value has to be read. • 'handleValuePair.value.len' is an input parameter for the length to be read. • 'handleValuePair.value.val' is an output parameter for data buffer. • 'handleValuePair.actualLen' has to be ignored as it is unused in this function.
CYBLE_CONN_HANDLE_T* connHandle	Pointer to the attribute instance handle, of type CYBLE_CONN_HANDLE_T.
uint8 flags	Attribute permissions. Allowed values are, • CYBLE_GATT_DB_LOCALLY_INITIATED • CYBLE_GATT_DB_PEER_INITIATED

Returns

CYBLE_GATT_ERR_CODE_T : Return value indicates if the function succeeded or failed. Following are the possible error codes.

Error codes	Description
CYBLE_GATT_ERR_NONE	On successful operation
CYBLE_GATT_ERR_INVALID_HANDLE	'handleValuePair.attrHandle' is not valid
CYBLE_GATT_ERR_READ_NOT_PERMITTED	Read operation is not permitted on this attribute
CYBLE_GATT_ERR_INVALID_OFFSET	Offset value is invalid
CYBLE_GATT_ERR_UNLIKELY_ERROR	Some other error occurred

CyBle_GattsEnableAttribute

Prototype

```
CYBLE_GATT_ERR_CODE_T CyBle_GattsEnableAttribute(CYBLE_GATT_DB_ATTR_HANDLE_T
attrHandle);
```

Description

This function enables the attribute entry for service or Characteristic logical group in the GATT database registered in BLE Stack. This is a blocking function. No event is generated on calling this function.

This function returns an error if the attribute does not belong to any service or Characteristic logical group. If the attribute entry is already enabled, then this function returns status CYBLE_GATT_ERR_NONE.

Parameters

Parameters	Description
CYBLE_GATT_DB_ATTR_HANDLE_T attrHandle	Attribute handle of the registered GATT Database to enable particular attribute entry, of type CYBLE_GATT_DB_ATTR_HANDLE_T.

Returns

CYBLE_GATT_ERR_CODE_T : Return value indicates if the function succeeded or failed. Following are the possible error codes.

Error codes	Description
CYBLE_GATT_ERR_NONE	On successful operation
CYBLE_GATT_ERR_INVALID_HANDLE	'attrHandle' is not valid

CyBle_GattsDisableAttribute

Prototype

```
CYBLE_GATT_ERR_CODE_T CyBle_GattsDisableAttribute(CYBLE_GATT_DB_ATTR_HANDLE_T
attrHandle);
```

Description

This function disables the attribute entry for service or Characteristic logical group in the GATT database registered in the BLE Stack. This is a blocking function. No event is generated on calling this function.

This function returns error if the attribute does not belong to a service or a Characteristic logical group. If attribute entry is already disabled then it returns CYBLE_GATT_ERR_NONE as status. All the attribute entries are enabled in GATT database during stack initialization.

Parameters

Parameters	Description
CYBLE_GATT_DB_ATTR_HANDLE_T attrHandle	Attribute handle of the registered GATT Database to disable particular attribute entry, of type 'CYBLE_GATT_DB_ATTR_HANDLE_T'

Returns

CYBLE_GATT_ERR_CODE_T : Return value indicates if the function succeeded or failed. Following are the possible error codes.

Error codes	Description
CYBLE_GATT_ERR_NONE	On successful operation
CYBLE_GATT_ERR_INVALID_HANDLE	'attrHandle' is not valid

CyBle_GattsNotification

Prototype

```
CYBLE_API_RESULT_T CyBle_GattsNotification(CYBLE_CONN_HANDLE_T connHandle,
CYBLE_GATTS_HANDLE_VALUE_NTF_T * ntfParam);
```

Description

This function sends a notification to the peer device when the GATT Server is configured to notify a Characteristic Value to the GATT Client without expecting any Attribute Protocol layer acknowledgement that the notification was successfully received. This is a non-blocking function.

On enabling notification successfully for a specific attribute, if the GATT server has an updated value to be notified to the GATT Client, it sends out a 'Handle Value Notification' which results in CYBLE_EVT_GATTC_HANDLE_VALUE_NTF event at the GATT Client's end.

Refer to Bluetooth 4.1 core specification, Volume 3, Part G, section 4.10 for more details on notifications.

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	Connection handle to identify the peer GATT entity, of type CYBLE_CONN_HANDLE_T.
CYBLE_GATTS_HANDLE_VALUE_NTF_T * ntfParam	Pointer to structure of type CYBLE_GATTS_HANDLE_VALUE_NTF_T.

Returns

CYBLE_API_RESULT_T : Return value indicates if the function succeeded or failed. Following are the possible error codes.

Error codes	Description
CYBLE_ERROR_OK	On successful operation
CYBLE_ERROR_INVALID_PARAMETER	'connHandle' value does not represent any existing entry in the Stack
CYBLE_ERROR_INVALID_OPERATION	This operation is not permitted
CYBLE_ERROR_MEMORY_ALLOCATION_FAILED	Memory allocation failed

CyBle_GattsIndication

Prototype

```
CYBLE_API_RESULT_T CyBle_GattsIndication(CYBLE_CONN_HANDLE_T connHandle,
CYBLE_GATTS_HANDLE_VALUE_IND_T * indParam);
```

Description

This function sends an indication to the peer device when the GATT Server is configured to indicate a Characteristic Value to the GATT Client and expects an Attribute Protocol layer acknowledgement that the indication was successfully received. This is a non-blocking function.

On enabling indication successfully, if the GATT server has an updated value to be indicated to the GATT Client, it sends out a 'Handle Value Indication' which results in CYBLE_EVT_GATTC_HANDLE_VALUE_IND event at the GATT Client's end.

Refer to Bluetooth 4.1 core specification, Volume 3, Part G, section 4.11 for more details on Indications.

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	Connection handle to identify the peer GATT entity, of type CYBLE_CONN_HANDLE_T.
CYBLE_GATTS_HANDLE_VALUE_IND_T * indParam	Pointer to structure of type CYBLE_GATTS_HANDLE_VALUE_IND_T.

Returns

CYBLE_API_RESULT_T : Return value indicates if the function succeeded or failed. Following are the possible error codes.

Error codes	Description
CYBLE_ERROR_OK	On successful operation
CYBLE_ERROR_INVALID_PARAMETER	'connHandle' value does not represent any existing entry in the Stack
CYBLE_ERROR_INVALID_OPERATION	This operation is not permitted
CYBLE_ERROR_MEMORY_ALLOCATION_FAILED	Memory allocation failed

CyBle_GattsErrorRsp

Prototype

```
CYBLE_API_RESULT_T CyBle_GattsErrorRsp(CYBLE_CONN_HANDLE_T connHandle,
CYBLE_GATTS_ERR_PARAM_T * errRspParam);
```

Description

This function sends an error response to the peer device. The Error Response is used to state that a given request cannot be performed, and to provide the reason as defined in 'CYBLE_GATT_ERR_CODE_T'. This is a non-blocking function.

Note that the 'Write Command' initiated by GATT Client does not generate an 'Error Response' from the GATT Server's end. The GATT Client gets CYBLE_EVT_GATTC_ERROR_RSP event on receiving error response.

Refer Bluetooth 4.1 core specification, Volume 3, Part F, section 3.4.1.1 for more details on Error Response operation.

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	Connection handle to identify the peer GATT entity, of type CYBLE_CONN_HANDLE_T.

CYBLE_GATTS_ERR_PARAM_T * errRspParam	Pointer to structure of type CYBLE_GATTS_ERR_PARAM_T.

Returns

CYBLE_API_RESULT_T : Return value indicates if the function succeeded or failed. Following are the possible error codes.

Error codes	Description
CYBLE_ERROR_OK	On successful operation
CYBLE_ERROR_INVALID_PARAMETER	'connHandle' value does not represent any existing entry in the Stack
CYBLE_ERROR_INVALID_OPERATION	This operation is not permitted
CYBLE_ERROR_MEMORY_ALLOCATION_FAILED	Memory allocation failed

CyBle_GattsExchangeMtuRsp

Prototype

```
CYBLE_API_RESULT_T CyBle_GattsExchangeMtuRsp(CYBLE_CONN_HANDLE_T connHandle, uint16
mtu);
```

Description

This function sends the GATT Server's MTU size to the GATT Client. This function has to be invoked in response to an Exchange MTU Request received from the GATT Client. The GATT Server's MTU size should be greater than or equal to the default MTU size (23 bytes). This is a non-blocking function.

The peer GATT Client receives CYBLE_EVT_GATTC_XCHNG_MTU_RSP event on executing this function on the GATT Server.

Refer to Bluetooth 4.1 core specification, Volume 3, Part G, section 4.3.1 for more details on exchange of MTU.

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	Connection handle to identify the peer GATT entity, of type CYBLE_CONN_HANDLE_T.
uint16 mtu	Size of MTU, of type uint16

Returns

CYBLE_API_RESULT_T : Return value indicates if the function succeeded or failed. Following are the possible error codes.

Error codes	Description
CYBLE_ERROR_OK	On successful operation
CYBLE_ERROR_INVALID_PARAMETER	'connHandle' value does not represent any existing entry in the Stack or, 'mtu' has a value which is greater than that set on calling CyBle_StackInit function
CYBLE_ERROR_INVALID_OPERATION	This operation is not permitted
CYBLE_ERROR_MEMORY_ALLOCATION_FAILED	Memory allocation failed

CyBle_GattsWriteRsp

Prototype

```
CYBLE_API_RESULT_T CyBle_GattsWriteRsp(CYBLE_CONN_HANDLE_T connHandle);
```

Description

This function sends a Write Response from a GATT Server to the GATT Client. This is a non-blocking function. This function has to be invoked in response to a valid Write Request event from the GATT Client (CYBLE_EVT_GATTS_WRITE_REQ) to acknowledge that the attribute has been successfully written.

The Write Response has to be sent after the attribute value is written or saved by the GATT Server. Write Response results in CYBLE_EVT_GATTC_WRITE_RSP event at the GATT Client's end.

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	Connection handle to identify the peer GATT entity, of type CYBLE_CONN_HANDLE_T.

Returns

CYBLE_API_RESULT_T : Return value indicates if the function succeeded or failed. Following are the possible error codes.

Error codes	Description
CYBLE_ERROR_OK	On successful operation
CYBLE_ERROR_INVALID_PARAMETER	'connHandle' value does not represent any existing entry in the Stack

CYBLE_ERROR_INVALID_OPERATION	This operation is not permitted
CYBLE_ERROR_MEMORY_ALLOCATION_FAILED	Memory allocation failed

CyBle_GattsPrepWriteReqSupport

Prototype

```
void CyBle_GattsPrepWriteReqSupport(uint8 prepWriteSupport);
```

Description

This function needs to be called after getting CYBLE_EVT_GATTS_PREP_WRITE_REQ event from the BLE Stack to perform necessary initialization in the BLE stack to support prepare write request operation. This needs to be called from the same event call back context. This is a non-blocking function.

On receiving CYBLE_EVT_GATTS_PREP_WRITE_REQ, returning from the event handler without calling this function will result in prepare write response being sent to the peer device rejecting the prepare write operation. CYBLE_GATT_ERR_REQUEST_NOT_SUPPORTED error code will be sent to client.

Parameters

Parameters	Description
uint8 prepWriteSupport	If prepare write operation is supported by the application then the application layer should set this variable to CYBLE_GATTS_PREP_WRITE_SUPPORT. Any other value will result in the device rejecting the prepare write operation. Allowed values for this parameter CYBLE_GATTS_PREP_WRITE_SUPPORT CYBLE_GATTS_PREP_WRITE_NOT_SUPPORT

Returns

None

GATT Client Functions

APIs unique to designs configured as a GATT Client role.

A letter 'c' is appended to the API name: CyBle_Gattc

Functions

Function	Description
CyBle_GattcStopCmd	This function is used by the GATT Client to stop any of the following ongoing GATT procedures: CyBle_GattcDiscoverAllPrimaryServices

	CyBle_GattcDiscoverPrimaryServiceByUuid CyBle_GattcFindIncludedServices CyBle_GattcDiscoverAllCharacteristics CyBle_GattcDiscoverCharacteristicByUuid CyBle_GattcDiscoverAllCharacteristicDescriptors CyBle_GattcReadLongCharacteristicValues CyBle_GattcWriteLongCharacteristicValues... more
CyBle_GattcExchangeMtuReq	This function is used by the GATT Client to send Maximum Transmitted Unit (MTU) supported by the GATT Client. This is a non-blocking function. Default... more
CyBle_GattcDiscoverAllPrimaryServices	This function is used by the GATT Client to discover all the primary services on a GATT Server to which it is connected. This is... more
CyBle_GattcDiscoverPrimaryServiceByUuid	This function is used by the GATT Client to discover a specific primary service on a GATT Server, to which it is connected, when only... more
CyBle_GattcFindIncludedServices	This function is used by the GATT Client to find Included Service declarations within a GATT Service to which it is connected. This is a... more
CyBle_GattcDiscoverAllCharacteristics	This function is used by the GATT Client to find all Characteristic declarations within a service definition on a GATT Server connect to it when... more
CyBle_GattcDiscoverCharacteristicByUuid	This function is used by the GATT Client to discover service Characteristics on a GATT Server when only the service handle ranges are known and... more
CyBle_GattcDiscoverAllCharacteristicDescriptors	This function is used by the GATT Client to find all the Characteristic Descriptors. This is a non-blocking function. Internally, multiple Find Information Requests are... more
CyBle_GattcReadCharacteristicValue	This function reads a Characteristic Value from a GATT Server when the GATT Client knows the Characteristic Value Handle. This is a non-blocking function. Internally,... more
CyBle_GattcReadUsingCharacteristicUuid	This function reads a Characteristic Value from the GATT Server when the GATT Client only knows the Characteristic UUID and does not know the handle... more
CyBle_GattcReadLongCharacteristicValues	This function reads a Characteristic Value from the GATT Server when the GATT Client knows the Characteristic Value Handle and the length of the Characteristic... more
CyBle_GattcReadMultipleCharacteristicValues	This function reads multiple Characteristic Values from a GATT Server when the GATT Client knows the Characteristic Value Handles. This is a non-blocking function. Internally,... more
CyBle_GattcWriteWithoutResponse	This function writes a Characteristic Value to a GATT Server when the GATT Client knows the Characteristic Value Handle and the client does not need... more

CyBle_GattcSignedWriteWithoutRsp	This function writes a Characteristic Value to a server when the client knows the Characteristic Value Handle and the ATT Bearer is not encrypted. This... more
CyBle_GattcWriteCharacteristicValue	This function writes a Characteristic Value to a GATT Server when the GATT Client knows the Characteristic Value Handle. This is a non-blocking function. Internally,... more
CyBle_GattcWriteLongCharacteristicValues	This function writes a Characteristic Value to a GATT Server when the GATT Client knows the Characteristic Value Handle but the length of the Characteristic... more
CyBle_GattcReliableWrites	This function writes a Characteristic Value to a GATT Server when the GATT Client knows the Characteristic Value Handle, and assurance is required that the... more
CyBle_GattcConfirmation	This function sends confirmation to the GATT Server on receiving Handle Value Indication event CYBLE_EVT_GATTC_HANDLE_VALUE_IND at the GATT Client's end. This is a non-blocking function.... more
CyBle_GattcReadCharacteristicDescriptors	This function reads a Characteristic Descriptor from a GATT Server when the GATT Client knows the Attribute handle from the Characteristic Descriptor declaration. This is... more
CyBle_GattcReadLongCharacteristicDescriptors	This function reads a Characteristic Descriptor from a GATT Server when the GATT Client knows the Attribute handle from the Characteristic Descriptor declaration and the... more
CyBle_GattcWriteCharacteristicDescriptors	This function writes a Characteristic Descriptor value to a GATT Server when the GATT Client knows the Characteristic Descriptor handle. This is a non-blocking function.... more
CyBle_GattcWriteLongCharacteristicDescriptors	This function writes a Characteristic Descriptor value to a GATT Server when the GATT Client knows the Characteristic Descriptor handle but the length of the... more
CyBle_GattcStartDiscovery	Starts the automatic server discovery process. Two events may be generated after calling this function - CYBLE_EVT_GATTC_DISCOVERY_COMPLETE or CYBLE_EVT_GATTC_ERROR_RSP. The CYBLE_EVT_GATTC_DISCOVERY_COMPLETE event is generated when... more

CyBle_GattcStopCmd

Prototype

```
void CyBle_GattcStopCmd(void);
```

Description

This function is used by the GATT Client to stop any of the following ongoing GATT procedures:

- CyBle_GattcDiscoverAllPrimaryServices
- CyBle_GattcDiscoverPrimaryServiceByUuid
- CyBle_GattcFindIncludedServices
- CyBle_GattcDiscoverAllCharacteristics
- CyBle_GattcDiscoverCharacteristicByUuid
- CyBle_GattcDiscoverAllCharacteristicDescriptors
- CyBle_GattcReadLongCharacteristicValues
- CyBle_GattcWriteLongCharacteristicValues
- CyBle_GattcReliableWrites
- CyBle_GattcReadLongCharacteristicDescriptors
- CyBle_GattcWriteLongCharacteristicDescriptors

If none of the above procedures is ongoing, then this command will be ignored. This function has no effect on ATT procedures other than those listed above.

If the user intends to start a new GATT procedure including those listed above and there is an ongoing GATT procedure (any one from the above list), the user needs to call this function to stop the ongoing GATT procedure and then invoke the desired GATT procedure. This is a blocking function. No event is generated on calling this function.

Returns

None

CyBle_GattcExchangeMtuReq

Prototype

```
CYBLE_API_RESULT_T CyBle_GattcExchangeMtuReq(CYBLE_CONN_HANDLE_T connHandle, uint16
mtu);
```

Description

This function is used by the GATT Client to send Maximum Transmitted Unit (MTU) supported by the GATT Client. This is a non-blocking function.

Default MTU size as per Bluetooth 4.1 core specification is 23 bytes. If the GATT Client supports a size greater than the default, it has to invoke this function with the desired MTU size. This function should only be initiated once during a connection.

Refer to Bluetooth 4.1 core specification, Volume 3, Part G, section 4.3.1 for more details on MTU exchange operation.

This function call results in CYBLE_EVT_GATTS_XCNHG_MTU_REQ event at the GATT Server's end in response to which the GATT Server is expected to send its MTU size.

The CYBLE_EVT_GATTC_XCHNG_MTU_RSP event is generated at the GATT Client's end on receiving MTU response from the GATT Server.

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	Connection handle to identify the peer GATT entity of type CYBLE_CONN_HANDLE_T.
uint16 mtu	Size of MTU. Max MTU supported by BLE stack is 256 Bytes.

Returns

CYBLE_API_RESULT_T : Return value indicates if the function succeeded or failed. Following are the possible error codes.

Error codes	Description
CYBLE_ERROR_OK	On successful operation
CYBLE_ERROR_INVALID_PARAMETER	'connHandle' value does not represent any existing entry in the Stack or, 'mtu' has a value which is greater than that set on calling CyBle_StackInit function
CYBLE_ERROR_INVALID_OPERATION	This operation is not permitted
CYBLE_ERROR_MEMORY_ALLOCATION_FAILED	Memory allocation failed

CyBle_GattcDiscoverAllPrimaryServices

Prototype

```
CYBLE_API_RESULT_T CyBle_GattcDiscoverAllPrimaryServices(CYBLE_CONN_HANDLE_T
connHandle);
```

Description

This function is used by the GATT Client to discover all the primary services on a GATT Server to which it is connected. This is a non-blocking function.

Internally, this function initiates multiple Read By Group Type Requests to the peer device in response to which it receives Read By Group Type Responses. Each Read By Group Type Response results in CYBLE_EVT_GATTC_READ_BY_GROUP_TYPE_RSP event, which is propagated to the application layer for handling.

Primary service discovery is complete when Error Response (CYBLE_EVT_GATTC_ERROR_RSP) is received and the Error Code is set to Attribute Not Found or when the End Group Handle in the Read by Group Type Response is 0xFFFF. Completion of this operation is notified to the upper layer(s) using CYBLE_EVT_GATTC_ERROR_RSP with error code updated appropriately.

It is permitted to end the above stated sequence of operations early if the desired primary service is found prior to discovering all the primary services on the GATT Server. This can be achieved by calling the CyBle_GattcStopCmd() function.

Refer to Bluetooth 4.1 core specification, Volume 3, Part G, section 4.4.1 for more details on this sequence of operations.

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	Connection handle to identify the peer GATT entity of type CYBLE_CONN_HANDLE_T.

Returns

CYBLE_API_RESULT_T : Return value indicates if the function succeeded or failed. Following are the possible error codes.

Error codes	Description
CYBLE_ERROR_OK	On successful operation
CYBLE_ERROR_INVALID_PARAMETER	'connHandle' value does not represent any existing entry in the Stack
CYBLE_ERROR_INVALID_OPERATION	This operation is not permitted
CYBLE_ERROR_MEMORY_ALLOCATION_FAILED	Memory allocation failed

CyBle_GattcDiscoverPrimaryServiceByUuid

Prototype

```
CYBLE_API_RESULT_T CyBle_GattcDiscoverPrimaryServiceByUuid(CYBLE_CONN_HANDLE_T
connHandle, CYBLE_GATT_VALUE_T value);
```

Description

This function is used by the GATT Client to discover a specific primary service on a GATT Server, to which it is connected, when only the Service UUID is known. This is a non-blocking function.

Internally, this function initiates multiple Find By Type Value Requests with the Attribute Type parameter set to the UUID for Primary Service and the Attribute Value set to the 16-bit Bluetooth

UUID or 128-bit UUID for the specific primary service. Each Find By Type Value Response received from the peer device is passed to the application as CYBLE_EVT_GATTC_FIND_BY_TYPE_VALUE_RSP event.

The sequence of operations is complete when the Error Response is received and the Error Code is set to Attribute Not Found or when the End Group Handle in the Find By Type Value Response is 0xFFFF. Completion of this function is notified to upper layer using CYBLE_EVT_GATTC_ERROR_RSP event with the error code updated appropriately.

It is permitted to end the function early by calling the CyBle_GattcStopCmd() function if a desired primary service is found prior to discovery of all the primary services of the specified service UUID supported on the GATT Server.

Refer to Bluetooth 4.1 core specification, Volume 3, Part G, section 4.4.2 for more details on this sequence of operations.

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	Connection handle to identify the peer GATT entity of type CYBLE_CONN_HANDLE_T.
CYBLE_GATT_VALUE_T value	Parameter is of type 'CYBLE_GATT_VALUE_T', where, • 'value.val' should point to uint8 array containing the UUID to look for. UUID can be 16 or 128 bit. • 'value.len' should be set to 2 if the 16 bit UUID is to be found. The length should be set to 16 if 128 bit UUID is to be found. • 'value.actualLen' is an unused parameter and should be ignored as it is unused.

Returns

CYBLE_API_RESULT_T : Return value indicates if the function succeeded or failed. Following are the possible error codes.

Error codes	Description
CYBLE_ERROR_OK	On successful operation
CYBLE_ERROR_INVALID_PARAMETER	'connHandle' value does not represent any existing entry in the Stack
CYBLE_ERROR_INVALID_OPERATION	This operation is not permitted
CYBLE_ERROR_MEMORY_ALLOCATION_FAILED	Memory allocation failed

CyBle_GattcFindIncludedServices

Prototype

```
CYBLE_API_RESULT_T CyBle_GattcFindIncludedServices(CYBLE_CONN_HANDLE_T connHandle,
CYBLE_GATT_ATTR_HANDLE_RANGE_T * range);
```

Description

This function is used by the GATT Client to find Included Service declarations within a GATT Service to which it is connected. This is a non-blocking function.

Internally, multiple Read By Type Requests are sent to the peer device in response to which Read By Type Responses are received (CYBLE_EVT_GATTC_READ_BY_TYPE_RSP) and passed to the application layer.

When Read By Type Response data does not contain the service UUID, indicating the service UUID is a 128-bit UUID, the application layer can choose to get the service UUID by performing the following steps:

- Stop ongoing GATT operation by invoking CyBle_GattcStopCmd()
- Send Read Request by invoking the function CyBle_GattcReadCharacteristicValue() with the read request handle set to the attribute handle of the included service. Handle associated events.
- Re-initiate CyBle_GattcFindIncludedServices function, setting the start handle to the attribute handle which is placed next to the one used in the above step.

It is permitted to end the function early if a desired included service is found prior to discovering all the included services of the specified service supported on the server by calling the CyBle_GattcStopCmd() function. If the CyBle_GattcStopCmd() function is not invoked, completion of this function is notified to the upper layer using CYBLE_EVT_GATTC_ERROR_RSP.

Refer to Bluetooth 4.1 core specification, Volume 3, Part G, section 4.5.1 for more details on the sequence of operations.

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	Connection handle to identify the peer GATT entity of type CYBLE_CONN_HANDLE_T.
CYBLE_GATT_ATTR_HANDLE_RANGE_T * range	Pointer to the handle range of type CYBLE_GATT_ATTR_HANDLE_RANGE_T for which relationship discovery has to be performed

Returns

CYBLE_API_RESULT_T : Return value indicates if the function succeeded or failed. Following are the possible error codes.

Error codes	Description
CYBLE_ERROR_OK	On successful operation
CYBLE_ERROR_INVALID_PARAMETER	'connHandle' value does not represent any existing entry in the Stack
CYBLE_ERROR_INVALID_OPERATION	This operation is not permitted
CYBLE_ERROR_MEMORY_ALLOCATION_FAILED	Memory allocation failed

CyBle_GattcDiscoverAllCharacteristics

Prototype

```
CYBLE_API_RESULT_T CyBle_GattcDiscoverAllCharacteristics(CYBLE_CONN_HANDLE_T
connHandle, CYBLE_GATT_ATTR_HANDLE_RANGE_T range);
```

Description

This function is used by the GATT Client to find all Characteristic declarations within a service definition on a GATT Server connect to it when only the service handle range is known. This is a non-blocking function.

Internally, multiple Read By Type Requests are sent to the GATT Server in response to which Read By Type Responses are received. Each response results in the event CYBLE_EVT_GATTC_READ_BY_TYPE_RSP, which is passed to the application layer for handling.

It is permitted to end the function early by calling the CyBle_GattcStopCmd() function if a desired Characteristic is found prior to discovering all the Characteristics of the specified service supported on the GATT Server. Completion of this function is notified to upper layer using CYBLE_EVT_GATTC_ERROR_RSP event.

Refer to Bluetooth 4.1 core specification, Volume 3, Part G, section 4.6.1 for more details on the sequence of operations.

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	Connection handle to identify the peer GATT entity of type CYBLE_CONN_HANDLE_T.
CYBLE_GATT_ATTR_HANDLE_RANGE_T range	Parameter is of type CYBLE_GATT_ATTR_HANDLE_RANGE_T where, • 'range.startHandle' can be set to the start handle of the desired primary service. • 'range.endHandle' can be set to the end handle of the desired primary service.

Returns

CYBLE_API_RESULT_T : Return value indicates if the function succeeded or failed. Following are the possible error codes.

Error codes	Description
CYBLE_ERROR_OK	On successful operation
CYBLE_ERROR_INVALID_PARAMETER	'connHandle' value does not represent any existing entry in the Stack
CYBLE_ERROR_INVALID_OPERATION	This operation is not permitted
CYBLE_ERROR_MEMORY_ALLOCATION_FAILED	Memory allocation failed

CyBle_GattcDiscoverCharacteristicByUuid

Prototype

```
CYBLE_API_RESULT_T CyBle_GattcDiscoverCharacteristicByUuid(CYBLE_CONN_HANDLE_T
connHandle, CYBLE_GATTC_READ_BY_TYPE_REQ_T * readByTypeReqParam);
```

Description

This function is used by the GATT Client to discover service Characteristics on a GATT Server when only the service handle ranges are known and the Characteristic UUID is known. This is a non-blocking function.

Internally, multiple Read By Type Requests are sent to the peer device in response to which Read By Type Responses are received. Each of these responses results in the event CYBLE_EVT_GATTC_READ_BY_TYPE_RSP, which is passed to the application layer for further processing.

It is permitted to end the function early by calling the CyBle_GattcStopCmd() function if a desired Characteristic is found prior to discovering all the Characteristics for the specified service supported on the GATT Server. Completion of this function is notified to upper layer using CYBLE_EVT_GATTC_ERROR_RSP event.

Refer to Bluetooth 4.1 core specification, Volume 3, Part G, section 4.6.2 for more details on the sequence of operations.

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	Connection handle to identify the peer GATT entity of type CYBLE_CONN_HANDLE_T.
CYBLE_GATTC_READ_BY_TYPE_REQ_T * readByTypeReqParam	Pointer to a variable of type CYBLE_GATTC_READ_BY_TYPE_REQ_T.

Returns

CYBLE_API_RESULT_T : Return value indicates if the function succeeded or failed. Following are the possible error codes.

Error codes	Description
CYBLE_ERROR_OK	On successful operation
CYBLE_ERROR_INVALID_PARAMETER	'connHandle' value does not represent any existing entry in the Stack
CYBLE_ERROR_INVALID_OPERATION	This operation is not permitted
CYBLE_ERROR_MEMORY_ALLOCATION_FAILED	Memory allocation failed

CyBle_GattcDiscoverAllCharacteristicDescriptors

Prototype

```
CYBLE_API_RESULT_T
CyBle_GattcDiscoverAllCharacteristicDescriptors(CYBLE_CONN_HANDLE_T connHandle,
CYBLE_GATTC_FIND_INFO_REQ_T * findInfoReqParam);
```

Description

This function is used by the GATT Client to find all the Characteristic Descriptors. This is a non-blocking function.

Internally, multiple Find Information Requests are sent to the peer device in response to which Find Information Responses are received by the GATT Client. Each of these responses generate CYBLE_EVT_GATTC_FIND_INFO_RSP event at the GATT Client end which is propagated to the application layer for further processing.

It is permitted to end the function early by calling the CyBle_GattcStopCmd() function if desired Characteristic Descriptor is found prior to discovering all the Characteristic Descriptors of the specified Characteristic. Completion of this function is notified to upper layer using CYBLE_EVT_GATTC_ERROR_RSP event.

Refer to Bluetooth 4.1 core specification, Volume 3, Part G, section 4.7.1 for more details on the sequence of operations.

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	Connection handle to identify the peer GATT entity of type CYBLE_CONN_HANDLE_T.
CYBLE_GATTC_FIND_INFO_REQ_T * findInfoReqParam	Pointer to a variable of type CYBLE_GATTC_FIND_INFO_REQ_T.

Returns

CYBLE_API_RESULT_T : Return value indicates if the function succeeded or failed. Following are the possible error codes.

Error codes	Description
CYBLE_ERROR_OK	On successful operation
CYBLE_ERROR_INVALID_PARAMETER	'connHandle' value does not represent any existing entry in the Stack
CYBLE_ERROR_INVALID_OPERATION	This operation is not permitted
CYBLE_ERROR_MEMORY_ALLOCATION_FAILED	Memory allocation failed

CyBle_GattcReadCharacteristicValue

Prototype

```
CYBLE_API_RESULT_T CyBle_GattcReadCharacteristicValue(CYBLE_CONN_HANDLE_T connHandle,
CYBLE_GATTC_READ_REQ_T readReqParam);
```

Description

This function reads a Characteristic Value from a GATT Server when the GATT Client knows the Characteristic Value Handle. This is a non-blocking function.

Internally, Read Request is sent to the peer device in response to which Read Response is received. This response results in CYBLE_EVT_GATTC_READ_RSP event which is propagated to the application for handling the event data. An Error Response (CYBLE_EVT_GATTC_ERROR_RSP event at the GATT Client's end) is sent by the GATT Server in response to the Read Request on insufficient authentication or insufficient authorization or insufficient encryption key size is caused by the GATT Client, or if a read operation is not permitted on the Characteristic Value. The Error Code parameter is set as specified in the Attribute Protocol.

Refer to Bluetooth 4.1 core specification, Volume 3, Part G, section 4.8.1 for more details on the sequence of operations.

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	Connection handle to identify the peer GATT entity of type CYBLE_CONN_HANDLE_T.
CYBLE_GATTC_READ_REQ_T readReqParam	Pointer to a variable of type CYBLE_GATTC_READ_REQ_T.

Returns

CYBLE_API_RESULT_T : Return value indicates if the function succeeded or failed. Following are the possible error codes.

Error codes	Description
CYBLE_ERROR_OK	On successful operation
CYBLE_ERROR_INVALID_PARAMETER	'connHandle' value does not represent any existing entry in the Stack
CYBLE_ERROR_INVALID_OPERATION	This operation is not permitted
CYBLE_ERROR_MEMORY_ALLOCATION_FAILED	Memory allocation failed

CyBle_GattcReadUsingCharacteristicUuid

Prototype

```
CYBLE_API_RESULT_T CyBle_GattcReadUsingCharacteristicUuid(CYBLE_CONN_HANDLE_T
connHandle, CYBLE_GATTC_READ_BY_TYPE_REQ_T * readByTypeReqParam);
```

Description

This function reads a Characteristic Value from the GATT Server when the GATT Client only knows the Characteristic UUID and does not know the handle of the Characteristic. This is a non-blocking function.

Internally, Read By Type Request is sent to the peer device in response to which Read By Type Response is received by the GATT Client. This results in CYBLE_EVT_GATTC_READ_BY_TYPE_RSP event, which is propagated to the application layer for further handling.

Refer to Bluetooth 4.1 core specification, Volume 3, Part G, section 4.8.2 for more details on the sequence of operations.

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	Connection handle to identify the peer GATT entity of type CYBLE_CONN_HANDLE_T.
CYBLE_GATTC_READ_BY_TYPE_REQ_T * readByTypeReqParam	Parameter is of type CYBLE_GATTC_READ_BY_TYPE_REQ_T.

Returns

CYBLE_API_RESULT_T : Return value indicates if the function succeeded or failed. Following are the possible error codes.

Error codes	Description
CYBLE_ERROR_OK	On successful operation
CYBLE_ERROR_INVALID_PARAMETER	'connHandle' value does not represent any existing entry in the Stack
CYBLE_ERROR_INVALID_OPERATION	This operation is not permitted
CYBLE_ERROR_MEMORY_ALLOCATION_FAILED	Memory allocation failed

CyBle_GattcReadLongCharacteristicValues

Prototype

```
CYBLE_API_RESULT_T CyBle_GattcReadLongCharacteristicValues(CYBLE_CONN_HANDLE_T
connHandle, CYBLE_GATTC_READ_BLOB_REQ_T * readBlobReqParam);
```

Description

This function reads a Characteristic Value from the GATT Server when the GATT Client knows the Characteristic Value Handle and the length of the Characteristic Value is longer than can be sent in a single Read Response Attribute Protocol message. This is a non-blocking function.

Internally multiple Read Blob Requests are sent to the peer device in response to which Read Blob Responses are received. For each Read Blob Request, a Read Blob Response event is received (CYBLE_EVT_GATTC_READ_BLOB_RSP) with a portion of the Characteristic Value contained in the Part Attribute Value parameter. These events are propagated to the application layer for further processing. Each read blob response will return up to (MTU-1) bytes of data. If the size of Characteristic value field is an integral multiple of (MTU-1) then the operation

terminates with an error response event, where the error code is CYBLE_GATT_ERR_INVALID_OFFSET. If the size of the Characteristic value field is not an integral multiple of (MTU-1), the last read blob response will return data bytes which are less than (MTU-1). The application needs to monitor these two conditions before proceeding with the initiation of any other GATT operation.

An Error Response event (CYBLE_EVT_GATTC_ERROR_RSP) is sent by the GATT Server in response to the Read Blob Request if insufficient authentication, insufficient authorization, insufficient encryption key size is used by the client, or if a read operation is not permitted on the Characteristic Value. The Error Code parameter is set as specified in the Attribute Protocol.

If the Characteristic Value is not longer than (MTU - 1), an Error Response with the Error Code set to Attribute Not Long is received by the GATT Client on the first Read Blob Request.

Refer to Bluetooth 4.1 core specification, Volume 3, Part G, section 4.8.3 for more details on the sequence of operations.

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	Connection handle to identify the peer GATT entity, of type CYBLE_CONN_HANDLE_T.
readblobReqParam	Pointer to a variable of type CYBLE_GATTC_READ_BLOB_REQ_T.

Returns

CYBLE_API_RESULT_T : Return value indicates if the function succeeded or failed. Following are the possible error codes.

Error codes	Description
CYBLE_ERROR_OK	On successful operation
CYBLE_ERROR_INVALID_PARAMETER	'connHandle' value does not represent any existing entry in the Stack
CYBLE_ERROR_INVALID_OPERATION	This operation is not permitted
CYBLE_ERROR_MEMORY_ALLOCATION_FAILED	Memory allocation failed

CyBle_GattcReadMultipleCharacteristicValues

Prototype

```
CYBLE_API_RESULT_T CyBle_GattcReadMultipleCharacteristicValues(CYBLE_CONN_HANDLE_T
connHandle, CYBLE_GATTC_READ_MULT_REQ_T * readMultiReqParam);
```

Description

This function reads multiple Characteristic Values from a GATT Server when the GATT Client knows the Characteristic Value Handles. This is a non-blocking function.

Internally, Read Multiple Request is sent to the peer device in response to which Read Multiple Response is received. This results in C YBLE_EVT_GATTC_READ_MULTI_RSP event, which is propagated to the application layer.

An Error Response event is sent by the server (CYBLE_EVT_GATTC_ERROR_RSP) in response to the Read Multiple Request if insufficient authentication, insufficient authorization, insufficient encryption key size is used by the client, or if a read operation is not permitted on any of the Characteristic Values. The Error Code parameter is set as specified in the Attribute Protocol.

Refer to Bluetooth 4.1 core specification, Volume 3, Part G, section 4.8.4 for more details on the sequence of operations.

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	Connection handle to identify the peer GATT entity, of type CYBLE_CONN_HANDLE_T.
CYBLE_GATTC_READ_MULT_REQ_T * readMultiReqParam	Pointer to a variable of type CYBLE_GATTC_READ_MULT_REQ_T.

Returns

CYBLE_API_RESULT_T : Return value indicates if the function succeeded or failed. Following are the possible error codes.

Error codes	Description
CYBLE_ERROR_OK	On successful operation
CYBLE_ERROR_INVALID_PARAMETER	'connHandle' value does not represent any existing entry in the Stack
CYBLE_ERROR_INVALID_OPERATION	This operation is not permitted
CYBLE_ERROR_MEMORY_ALLOCATION_FAILED	Memory allocation failed

CyBle_GattcWriteWithoutResponse

Prototype

```
CYBLE_API_RESULT_T CyBle_GattcWriteWithoutResponse(CYBLE_CONN_HANDLE_T connHandle,
CYBLE_GATTC_WRITE_CMD_REQ_T * writeCmdReqParam);
```

Description

This function writes a Characteristic Value to a GATT Server when the GATT Client knows the Characteristic Value Handle and the client does not need an acknowledgement that the write was successfully performed. This is a blocking function. No event is generated on calling this function.

Internally, Write Command is sent to the GATT Server and nothing is received in response from the GATT Server.

Refer Bluetooth 4.1 core specification, Volume 3, Part G, section 4.9.1 for more details on the sequence of operations.

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	Connection handle to identify the peer GATT entity, of type CYBLE_CONN_HANDLE_T.

CYBLE_GATTC_WRITE_CMD_REQ_T * writeCmdReqParam	Pointer to a variable of type CYBLE_GATTC_WRITE_CMD_REQ_T.

Returns

CYBLE_API_RESULT_T : Return value indicates if the function succeeded or failed. Following are the possible error codes.

Error codes	Description
CYBLE_ERROR_OK	On successful operation
CYBLE_ERROR_INVALID_PARAMETER	'connHandle' value does not represent any existing entry in the Stack
CYBLE_ERROR_INVALID_OPERATION	This operation is not permitted
CYBLE_ERROR_MEMORY_ALLOCATION_FAILED	Memory allocation failed

CyBle_GattcSignedWriteWithoutRsp

Prototype

```
CYBLE_API_RESULT_T CyBle_GattcSignedWriteWithoutRsp(CYBLE_CONN_HANDLE_T connHandle,
CYBLE_GATTC_SIGNED_WRITE_CMD_REQ_T * signedWriteWithoutRspParam);
```

Description

This function writes a Characteristic Value to a server when the client knows the Characteristic Value Handle and the ATT Bearer is not encrypted. This sub-procedure shall only be used if the Characteristic Properties authenticated bit is enabled and the client and server device share a bond as defined in Bluetooth Spec4.1 [Vol. 3] Part C, Generic Access Profile.

This function only writes the first (ATT_MTU 15) octets of an Attribute Value. This function cannot be used to write a long Attribute.

Internally, Signed Write Command is used. Refer to Bluetooth Spec4.1 Security Manager [Vol. 3] Part H, Section 2.4.5.

If the authenticated Characteristic Value that is written is the wrong size, has an invalid value as defined by the profile, or the signed value does not authenticate the client, then the write shall not succeed and no error shall be generated by the server.

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	Connection handle to identify the peer GATT entity, of type CYBLE_CONN_HANDLE_T.
CYBLE_GATTC_SIGNED_WRITE_CMD_REQ_T * signedWriteWithoutRspParam	Pointer to a variable of type CYBLE_GATTC_SIGNED_WRITE_CMD_REQ_T

Returns

CYBLE_API_RESULT_T : Return value indicates if the function succeeded or failed. Following are the possible error codes.

Error codes	Description
CYBLE_ERROR_OK	On successful operation
CYBLE_ERROR_INVALID_PARAMETER	'connHandle' value does not represent any existing entry in the Stack
CYBLE_ERROR_INVALID_OPERATION	This operation is not permitted
CYBLE_ERROR_INSUFFICIENT_RESOURCES	BLE stack out of resource

CyBle_GattcWriteCharacteristicValue

Prototype

```
CYBLE_API_RESULT_T CyBle_GattcWriteCharacteristicValue(CYBLE_CONN_HANDLE_T connHandle,
CYBLE_GATTC_WRITE_REQ_T * writeReqParam);
```

Description

This function writes a Characteristic Value to a GATT Server when the GATT Client knows the Characteristic Value Handle. This is a non-blocking function.

Internally, Write Request is sent to the GATT Server in response to which Write Response is received. This results in the event CYBLE_EVT_GATTC_WRITE_RSP, which indicates that the write operation succeeded.

An Error Response event (CYBLE_EVT_GATTC_ERROR_RSP) is sent by the server in response to the Write Request if insufficient authentication, insufficient authorization, insufficient encryption key size is used by the client, or if a write operation is not permitted on the Characteristic Value. The Error Code parameter is set as specified in the Attribute Protocol.

Refer to Bluetooth 4.1 core specification, Volume 3, Part G, section 4.9.3 for more details on the sequence of operations.

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	Connection handle to identify the peer GATT entity, of type CYBLE_CONN_HANDLE_T.
CYBLE_GATTC_WRITE_REQ_T * writeReqParam	Pointer to a variable of type CYBLE_GATTC_WRITE_REQ_T.

Returns

CYBLE_API_RESULT_T : Return value indicates if the function succeeded or failed. Following are the possible error codes.

Error codes	Description
CYBLE_ERROR_OK	On successful operation
CYBLE_ERROR_INVALID_PARAMETER	'connHandle' value does not represent any existing entry in the Stack
CYBLE_ERROR_INVALID_OPERATION	This operation is not permitted
CYBLE_ERROR_MEMORY_ALLOCATION_FAILED	Memory allocation failed

CyBle_GattcWriteLongCharacteristicValues

Prototype

```
CYBLE_API_RESULT_T CyBle_GattcWriteLongCharacteristicValues(CYBLE_CONN_HANDLE_T
connHandle, CYBLE_GATTC_PREP_WRITE_REQ_T * writePrepReqParam);
```

Description

This function writes a Characteristic Value to a GATT Server when the GATT Client knows the Characteristic Value Handle but the length of the Characteristic Value is longer than MTU size and cannot be sent in a single Write Request Attribute Protocol message. This is a non-blocking function.

Internally, multiple Prepare Write Requests are sent to the GATT Server in response to which Prepare Write Responses are received. No events are generated by the BLE Stack during these operations.

Prepare Write Requests are repeated until the complete Characteristic Value has been transferred to the GATT Server, after which an Execute Write Request is sent to the GATT Server to write the initially transferred value at the GATT Server's end. This generates CYBLE_EVT_GATTS_EXEC_WRITE_REQ at the GATT Server's end.

Once the GATT Server responds, CYBLE_EVT_GATTC_EXEC_WRITE_RSP event is generated at the GATT Client's end. The value associated with this event has to be checked by the application layer to confirm that the long write operation succeeded.

An Error Response event CYBLE_EVT_GATTC_ERROR_RSP is received by the GATT Client in response to the Prepare Write Request if insufficient authentication, insufficient authorization, insufficient encryption key size is used by the client, or if a write operation is not permitted on the Characteristic Value. The Error Code parameter is set as specified in the Attribute Protocol.

Refer to Bluetooth 4.1 core specification, Volume 3, Part G, section 4.9.4 for more details on the sequence of operations.

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	Connection handle to identify the peer GATT entity, of type CYBLE_CONN_HANDLE_T.
CYBLE_GATTC_PREP_WRITE_REQ_T * writePrepReqParam	Pointer to a variable of type CYBLE_GATTC_PREP_WRITE_REQ_T,
val	points to the actual data to be written. 'writePrepReqParam' and all associated variables need to be retained inMemory by the calling application until the GATT Write Long Characteristic Value operation is completed successfully.

Returns

CYBLE_API_RESULT_T : Return value indicates if the function succeeded or failed. Following are the possible error codes.

Error codes	Description
CYBLE_ERROR_OK	On successful operation
CYBLE_ERROR_INVALID_PARAMETER	'connHandle' value does not represent any existing entry in the Stack
CYBLE_ERROR_INVALID_OPERATION	This operation is not permitted
CYBLE_ERROR_MEMORY_ALLOCATION_FAILED	Memory allocation failed

CyBle_GattcReliableWrites

Prototype

```
CYBLE_API_RESULT_T CyBle_GattcReliableWrites(CYBLE_CONN_HANDLE_T connHandle,
CYBLE_GATTC_PREP_WRITE_REQ_T * writePrepReqParam, uint8 numOfRequests);
```

Description

This function writes a Characteristic Value to a GATT Server when the GATT Client knows the Characteristic Value Handle, and assurance is required that the correct Characteristic Value is going to be written by transferring the Characteristic Value to be written in both directions before the write is performed. This is a non-blocking function.

Internally, multiple Prepare Write Requests are sent to the GATT Server in response to which Prepare Write Responses are received. No events are generated by the BLE Stack during these operations.

Prepare Write Requests are repeated until the complete Characteristic Value has been transferred to the GATT Server, after which an Execute Write Request is sent to the GATT Server to write the initially transferred value at the GATT Server's end. This generates CYBLE_EVT_GATTS_EXEC_WRITE_REQ at the GATT Server's end.

Once the GATT Server responds, a CYBLE_EVT_GATTC_EXEC_WRITE_RSP event is generated at the GATT Client's end. The value associated with this event has to be checked by the application layer to confirm that the long write operation succeeded. An Error Response event CYBLE_EVT_GATTC_ERROR_RSP is received by the GATT Client in response to the Prepare Write Request if insufficient authentication, insufficient authorization, insufficient encryption key size is used by the client, or if a write operation is not permitted on the Characteristic Value. The Error Code parameter is set as specified in the Attribute Protocol.

Refer to Bluetooth 4.1 core specification, Volume 3, Part G, section 4.9.5 for more details on the sequence of operations.

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	Connection handle to identify the peer GATT entity, of type CYBLE_CONN_HANDLE_T.
CYBLE_GATTC_PREP_WRITE_REQ_T * writePrepReqParam	Pointer to a variable of type CYBLE_GATTC_PREP_WRITE_REQ_T. Since more than one writes are performed as part of this function, the first array element of the array of type CYBLE_GATTC_PREP_WRITE_REQ_T, which contains the values to be written, has to be specified. 'writePrepReqParam' and all associated variables need to be retained inMemory by the calling application until the GATT Reliable Write operation is completed successfully.
uint8 numOfRequests	Number of requests. That is, the count of array of structures of type CYBLE_GATTC_PREP_WRITE_REQ_T. Each array element represents a value and the attribute to which the value has to be written.

Returns

CYBLE_API_RESULT_T : Return value indicates if the function succeeded or failed. Following are the possible error codes.

Error codes	Description
CYBLE_ERROR_OK	On successful operation
CYBLE_ERROR_INVALID_PARAMETER	'connHandle' value does not represent any existing entry in the Stack
CYBLE_ERROR_INVALID_OPERATION	This operation is not permitted
CYBLE_ERROR_MEMORY_ALLOCATION_FAILED	Memory allocation failed

CyBle_GattcConfirmation

Prototype

```
CYBLE_API_RESULT_T CyBle_GattcConfirmation(CYBLE_CONN_HANDLE_T connHandle);
```

Description

This function sends confirmation to the GATT Server on receiving Handle Value Indication event CYBLE_EVT_GATTC_HANDLE_VALUE_IND at the GATT Client's end. This is a non-blocking function.

This function call results in CYBLE_EVT_GATTS_HANDLE_VALUE_CNF event at the GATT Server's end.

Refer to Bluetooth 4.1 core specification, Volume 3, Part G, section 4.11.1 for more details on the sequence of operations.

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	Connection handle to identify the peer GATT entity, of type CYBLE_CONN_HANDLE_T.

Returns

CYBLE_API_RESULT_T : Return value indicates if the function succeeded or failed. Following are the possible error codes.

Error codes	Description
CYBLE_ERROR_OK	On successful operation
CYBLE_ERROR_INVALID_PARAMETER	'connHandle' value does not represent any existing entry in the Stack
CYBLE_ERROR_INVALID_OPERATION	This operation is not permitted
CYBLE_ERROR_MEMORY_ALLOCATION_FAILED	Memory allocation failed

CyBle_GattcReadCharacteristicDescriptors

Prototype

```
CYBLE_API_RESULT_T CyBle_GattcReadCharacteristicDescriptors(CYBLE_CONN_HANDLE_T
connHandle, CYBLE_GATTC_READ_REQ_T readReqParam);
```

Description

This function reads a Characteristic Descriptor from a GATT Server when the GATT Client knows the Attribute handle from the Characteristic Descriptor declaration. This is a non-blocking function.

Internally, Read Request is sent to the peer device in response to which Read Response is received. This response results in CYBLE_EVT_GATTC_READ_RSP event, which is propagated to the application for handling the event data.

An Error Response (CYBLE_EVT_GATTC_ERROR_RSP event at the GATT Client's end) is sent by the GATT Server in response to the Read Request on insufficient authentication or insufficient authorization or insufficient encryption key size is caused by the GATT Client, or if a read operation is not permitted on the Characteristic Value. The Error Code parameter is set as specified in the Attribute Protocol.

Refer to Bluetooth 4.1 core specification, Volume 3, Part G, section 4.12.1 for more details on the sequence of operations.

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	Connection handle to identify the peer GATT entity, of type CYBLE_CONN_HANDLE_T.
CYBLE_GATTC_READ_REQ_T readReqParam	Pointer to a variable of type CYBLE_GATTC_READ_REQ_T.

Returns

CYBLE_API_RESULT_T : Return value indicates if the function succeeded or failed. Following are the possible error codes.

Error codes	Description
CYBLE_ERROR_OK	On successful operation
CYBLE_ERROR_INVALID_PARAMETER	'connHandle' value does not represent any existing entry in the Stack
CYBLE_ERROR_INVALID_OPERATION	This operation is not permitted
CYBLE_ERROR_MEMORY_ALLOCATION_FAILED	Memory allocation failed

CyBle_GattcReadLongCharacteristicDescriptors

Prototype

```
CYBLE_API_RESULT_T CyBle_GattcReadLongCharacteristicDescriptors(CYBLE_CONN_HANDLE_T
connHandle, CYBLE_GATTC_READ_BLOB_REQ_T * readBlobReqParam);
```

Description

This function reads a Characteristic Descriptor from a GATT Server when the GATT Client knows the Attribute handle from the Characteristic Descriptor declaration and the length of the Characteristic Descriptor declaration is longer than what can be sent in a single Read Response Attribute Protocol message. This is a non-blocking function.

Internally multiple Read Blob Requests are sent to the peer device in response to which Read Blob Responses are received. For each Read Blob Request, a Read Blob Response event is received (CYBLE_EVT_GATTC_READ_BLOB_RSP) with a portion of the Characteristic Value contained in the Part Attribute Value parameter. These events are propagated to the application layer for further processing. Each read blob response will return up to (MTU-1) bytes of data. If the size of Characteristic Descriptor field is an integral multiple of (MTU-1) then the operation terminates with an error response event, where the error code is CYBLE_GATT_ERR_INVALID_OFFSET. If the size of the Characteristic Descriptor field is not an integral multiple of (MTU-1), the last read blob response will return data bytes which are less than (MTU-1). The application needs to monitor these two conditions before proceeding with the initiation of any other GATT operation.

An Error Response event (CYBLE_EVT_GATTC_ERROR_RSP) is sent by the GATT Server in response to the Read Blob Request if insufficient authentication, insufficient authorization, insufficient encryption key size is used by the client, or if a read operation is not permitted on the Characteristic Value. The Error Code parameter is set as specified in the Attribute Protocol. If the Characteristic Value is not longer than (MTU - 1) an Error Response with the Error Code set to Attribute Not Long is received by the GATT Client on the first Read Blob Request.

Refer to Bluetooth 4.1 core specification, Volume 3, Part G, section 4.12.2 for more details on the sequence of operations.

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	Connection handle to identify the peer GATT entity, of type CYBLE_CONN_HANDLE_T.
readBlonReqParam	Pointer to a variable of type CYBLE_GATTC_READ_BLOB_REQ_T

Returns

CYBLE_API_RESULT_T : Return value indicates if the function succeeded or failed. Following are the possible error codes.

Error codes	Description
CYBLE_ERROR_OK	On successful operation
CYBLE_ERROR_INVALID_PARAMETER	'connHandle' value does not represent any existing entry in the Stack
CYBLE_ERROR_INVALID_OPERATION	This operation is not permitted
CYBLE_ERROR_MEMORY_ALLOCATION_FAILED	Memory allocation failed

CyBle_GattcWriteCharacteristicDescriptors

Prototype

```
CYBLE_API_RESULT_T CyBle_GattcWriteCharacteristicDescriptors(CYBLE_CONN_HANDLE_T
connHandle, CYBLE_GATTC_WRITE_REQ_T * writeReqParam);
```

Description

This function writes a Characteristic Descriptor value to a GATT Server when the GATT Client knows the Characteristic Descriptor handle. This is a non-blocking function.

Internally, Write Request is sent to the GATT Server in response to which Write Response is received. This results in the event CYBLE_EVT_GATTC_WRITE_RSP, which indicates that the write operation succeeded.

An Error Response event (CYBLE_EVT_GATTC_ERROR_RSP) is sent by the server in response to the Write Request if insufficient authentication, insufficient authorization, insufficient encryption key size is used by the client, or if a write operation is not permitted on the Characteristic Value. The Error Code parameter is set as specified in the Attribute Protocol.

Refer to Bluetooth 4.1 core specification, Volume 3, Part G, section 4.12.3 for more details on the sequence of operations.

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	Connection handle to identify the peer GATT entity, of type CYBLE_CONN_HANDLE_T.
CYBLE_GATTC_WRITE_REQ_T * writeReqParam	Pointer to a variable of type CYBLE_GATTC_WRITE_REQ_T

Returns

CYBLE_API_RESULT_T : Return value indicates if the function succeeded or failed. Following are the possible error codes.

Error codes	Description
CYBLE_ERROR_OK	On successful operation
CYBLE_ERROR_INVALID_PARAMETER	'connHandle' value does not represent any existing entry in the Stack
CYBLE_ERROR_INVALID_OPERATION	This operation is not permitted
CYBLE_ERROR_MEMORY_ALLOCATION_FAILED	Memory allocation failed

CyBle_GattcWriteLongCharacteristicDescriptors

Prototype

```
CYBLE_API_RESULT_T CyBle_GattcWriteLongCharacteristicDescriptors(CYBLE_CONN_HANDLE_T
connHandle, CYBLE_GATTC_PREP_WRITE_REQ_T * writePrepReqParam);
```

Description

This function writes a Characteristic Descriptor value to a GATT Server when the GATT Client knows the Characteristic Descriptor handle but the length of the Characteristic Descriptor value is longer than what can be sent in a single Write Request Attribute Protocol message. This is a non-blocking function.

Internally, multiple Prepare Write Requests are sent to the GATT Server in response to which Prepare Write Responses are received. No events are generated by the BLE Stack during these operations.

Prepare Write Requests are repeated until the complete Characteristic Descriptor Value has been transferred to the GATT Server, after which an Execute Write Request is sent to the GATT Server to write the initially transferred value at the GATT Server's end. This generates CYBLE_EVT_GATTS_EXEC_WRITE_REQ at the GATT Server's end.

Once the GATT Server responds, CYBLE_EVT_GATTC_EXEC_WRITE_RSP' event is generated at the GATT Client's end. The value associated with this event has to be checked by the application layer to confirm that the long write operation succeeded.

An Error Response event CYBLE_EVT_GATTC_ERROR_RSP is received by the GATT Client in response to the Prepare Write Request if insufficient authentication, insufficient authorization, insufficient encryption key size is used by the client, or if a write operation is not permitted on the Characteristic Value. The Error Code parameter is set as specified in the Attribute Protocol.

Refer Bluetooth 4.1 core specification, Volume 3, Part G, section 4.12.4 for more details on the sequence of operations.

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	Connection handle to identify the peer GATT entity, of type CYBLE_CONN_HANDLE_T.
CYBLE_GATTC_PREP_WRITE_REQ_T * writePrepReqParam	Pointer to a variable of type CYBLE_GATTC_PREP_WRITE_REQ_T,
val	points to the actual data to be written. 'writePrepReqParam' and all associated variables need to be retained inMemory by the calling application until the GATT Write Long Characteristic Descriptor operation is completed successfully.

Returns

CYBLE_API_RESULT_T : Return value indicates if the function succeeded or failed. Following are the possible error codes.

Error codes	Description
CYBLE_ERROR_OK	On successful operation
CYBLE_ERROR_INVALID_PARAMETER	'connHandle' value does not represent any existing entry in

	the Stack
CYBLE_ERROR_INVALID_OPERATION	This operation is not permitted
CYBLE_ERROR_MEMORY_ALLOCATION_FAILED	Memory allocation failed

CyBle_GattcStartDiscovery

Prototype

```
CYBLE_API_RESULT_T CyBle_GattcStartDiscovery(CYBLE_CONN_HANDLE_T connHandle);
```

Description

Starts the automatic server discovery process. Two events may be generated after calling this function - CYBLE_EVT_GATTC_DISCOVERY_COMPLETE or CYBLE_EVT_GATTC_ERROR_RSP. The CYBLE_EVT_GATTC_DISCOVERY_COMPLETE event is generated when the remote device was successfully discovered. The CYBLE_EVT_GATTC_ERROR_RSP is generated if the device discovery is failed.

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	The handle which consists of the device ID and ATT connection ID.

Returns

CYBLE_API_RESULT_T : Return value indicates if the function succeeded or failed. Following are the possible error codes. CYBLE_ERROR_OK - On successful operation CYBLE_ERROR_INVALID_PARAMETER - 'connHandle' value does not represent any existing entry. in the Stack CYBLE_ERROR_INVALID_OPERATION - The operation is not permitted. CYBLE_ERROR_MEMORY_ALLOCATION_FAILED -Memory allocation failed.

GATT Definitions and Data Structures

Contains the GATT specific definitions and data structures used in the GATT APIs.

Enumerations

Enumeration	Description
CYBLE_GATT_ERR_CODE_T	GATT profile error codes
CYBLE_GATT_PDU_T	Opcode which has resulted in error

Structures

Structure	Description
CYBLE_GATT_ATTR_HANDLE_RANGE_T	GATT Attribute Handle Range type
CYBLE_GATT_HANDLE_VALUE_OFFSET_PARAM_T	GATT Handle Value Pair along with offset type
CYBLE_GATT_HANDLE_VALUE_PAIR_T	GATT handle - value pair type
CYBLE_GATT_VALUE_T	Abstracts Variable Length Values for GATT. Apart from data, and length, 'actual length' is needed so that GATT can indicate to the application actual length... more
CYBLE_GATT_XCHG_MTU_PARAM_T	MTU exchange parameter type
CYBLE_GATTC_ERR_RSP_PARAM_T	Error Response parameter type received from Server For error codes that are received during gatt discovery procedure, Client may choose to disconnect the link. i.e.... more
CYBLE_GATTC_FIND_BY_TYPE_RSP_PARAM_T	GATT find by type value response received from server
CYBLE_GATTC_FIND_BY_TYPE_VALUE_REQ_T	GATT find by type value request to be sent to Server
CYBLE_GATTC_FIND_INFO_RSP_PARAM_T	GATT find info response received from Server
CYBLE_GATTC_GRP_ATTR_DATA_LIST_T	Data Element for Group Response
CYBLE_GATTC_HANDLE_LIST_T	GATT handle list type
CYBLE_GATTC_HANDLE_UUID_LIST_PARAM_T	GATT list of Handle UUID pair parameter type
CYBLE_GATTC_HANDLE_VALUE_NTF_PARAM_T	Handle value notification data received from server
CYBLE_GATTC_READ_BLOB_REQ_T	Read blob request to be sent to Server
CYBLE_GATTC_READ_BY_GRP_RSP_PARAM_T	Read By Group Response received from Server
CYBLE_GATTC_READ_BY_TYPE_REQ_T	GATT read by type request to be sent to Server
CYBLE_GATTC_READ_RSP_PARAM_T	Read response parameter type received from server
CYBLE_GATTC_T	Structure with discovered attributes information of Generic Attribute Service (GATTS)
CYBLE_GATTC_EXEC_WRITE_RSP_T	Execute Write result
CYBLE_GATTS_EXEC_WRITE_REQ_T	Execute Write result
CYBLE_GATTS_ATT_GENERIC_VAL_T	Attribute value type used in GATT database
CYBLE_GATTS_DB_T	GATT database structure used in the GAP Server
CYBLE_GATTS_ERR_PARAM_T	GATT Server Error Response parameter type
CYBLE_GATTS_PREP_WRITE_REQ_PARAM_T	Prepare write request parameter received from Client

CYBLE_GATTS_WRITE_REQ_PARAM_T	Write request parameter received from Client
CYBLE_GATTS_T	Structure with Generic Attribute Service (GATTS) attribute handles
CYBLE_DISC_CHAR_INFO_T	Characteristic data received with read by type response during discovery process
CYBLE_DISC_DESCR_INFO_T	Characteristic Descriptor data received with find info response during discovery process
CYBLE_DISC_INCL_INFO_T	Included service data received with read by type response during discovery process
CYBLE_DISC_SRVC_INFO_T	CYBLE_GATT_ROLE_SERVER

Types

Type	Description
CYBLE_GATT_DB_ATTR_HANDLE_T	GATT BD Attribute Handle Type
CYBLE_GATTC_FIND_INFO_REQ_T	GATT find info request to be sent to Server
CYBLE_GATTC_HANDLE_VALUE_IND_PARAM_T	GATT handle value indication parameter received from server type
CYBLE_GATTC_PREP_WRITE_REQ_T	Prepare write request to be sent to Server
CYBLE_GATTC_READ_BY_TYPE_RSP_PARAM_T	GATT read by type response received from server
CYBLE_GATTC_READ_MULT_REQ_T	Read multiple request to be sent to Server
CYBLE_GATTC_READ_REQ_T	Read request to be sent to Server
CYBLE_GATTC_WRITE_CMD_REQ_T	Write command request to be sent to Server
CYBLE_GATTC_WRITE_REQ_T	Write request to be sent to Server
CYBLE_GATTC_SIGNED_WRITE_CMD_REQ_T	Signed Write command request to be sent to Server
CYBLE_GATTS_SIGNED_WRITE_CMD_REQ_PARAM_T	Signed Write command request parameter received from Client
CYBLE_GATTS_HANDLE_VALUE_IND_T	GATT handle value indication parameter type
CYBLE_GATTS_HANDLE_VALUE_NTF_T	Handle value notification data to be sent to Client
CYBLE_GATTS_PREP_WRITE_RSP_PARAM_T	Prepare write response parameter to be sent to Client
CYBLE_GATTS_READ_RSP_PARAM_T	Read response parameter to be sent to Client
CYBLE_GATTS_WRITE_CMD_REQ_PARAM_T	Write command request parameter received

	from Client

Unions

Union	Description
CYBLE_GATTS_ATT_VALUE_T	Attribute value type used in GATT database

CYBLE_GATT_ATTR_HANDLE_RANGE_T

Prototype

```
typedef struct {
  CYBLE_GATT_DB_ATTR_HANDLE_T startHandle;
  CYBLE_GATT_DB_ATTR_HANDLE_T endHandle;
} CYBLE_GATT_ATTR_HANDLE_RANGE_T;
```

Description

GATT Attribute Handle Range type

Members

Members	Description
CYBLE_GATT_DB_ATTR_HANDLE_T startHandle;	Start Handle
CYBLE_GATT_DB_ATTR_HANDLE_T endHandle;	End Handle

CYBLE_GATT_DB_ATTR_HANDLE_T

Prototype

```
typedef uint16 CYBLE_GATT_DB_ATTR_HANDLE_T;
```

Description

GATT BD Attribute Handle Type

CYBLE_GATT_ERR_CODE_T

Prototype

```
typedef enum {
  CYBLE_GATT_ERR_NONE = 0x00u,
  CYBLE_GATT_ERR_INVALID_HANDLE,
  CYBLE_GATT_ERR_READ_NOT_PERMITTED,
  CYBLE_GATT_ERR_WRITE_NOT_PERMITTED,
  CYBLE_GATT_ERR_INVALID_PDU,
  CYBLE_GATT_ERR_INSUFFICIENT_AUTHENTICATION,
  CYBLE_GATT_ERR_REQUEST_NOT_SUPPORTED,
    CYBLE_GATT_ERR_INVALID_OFFSET,
    CYBLE_GATT_ERR_INSUFFICIENT_AUTHORIZATION,
    CYBLE_GATT_ERR_PREPARE_WRITE_QUEUE_FULL,
    CYBLE_GATT_ERR_ATTRIBUTE_NOT_FOUND,
    CYBLE_GATT_ERR_ATTRIBUTE_NOT_LONG,
    CYBLE_GATT_ERR_INSUFFICIENT_ENC_KEY_SIZE,
    CYBLE_GATT_ERR_INVALID_ATTRIBUTE_LEN,
    CYBLE_GATT_ERR_UNLIKELY_ERROR,
    CYBLE_GATT_ERR_INSUFFICIENT_ENCRYPTION,
    CYBLE_GATT_ERR_UNSUPPORTED_GROUP_TYPE,
    CYBLE_GATT_ERR_INSUFFICIENT_RESOURCE = 0x11,
    CYBLE_GATT_ERR_HEART_RATE_CONTROL_POINT_NOT_SUPPORTED = 0x80u,
    CYBLE_GATT_ERR_CPS_INAPPROPRIATE_CONNECTION_PARAMETERS = 0x80u,
    CYBLE_GATTS_ERR_PROCEDURE_ALREADY_IN_PROGRESS = 0x80u,
    CYBLE_GATTS_ERR_CCCD_IMPROPERLY_CONFIGURED = 0x81u,
    CYBLE_GATT_ERR_ANS_COMMAND_NOT_SUPPORTED = 0xA0u,
    CYBLE_GATT_ERR_CCCD_IMPROPERLY_CONFIGURED = 0xFDu,
    CYBLE_GATT_ERR_PROCEDURE_ALREADY_IN_PROGRESS = 0xFEu,
    CYBLE_GATT_ERR_OUT_OF_RANGE = 0xFFu
  } CYBLE_GATT_ERR_CODE_T;
```

Description

GATT profile error codes

Members

Members	Description
CYBLE_GATT_ERR_NONE = 0x00u	No Error
CYBLE_GATT_ERR_INVALID_HANDLE	Invalid Handle error code is used in the case when the ATT handle in the ATT request PDU is invalid.
CYBLE_GATT_ERR_READ_NOT_PERMITTED	Read Not Permitted error code is used in the case when the permission to read the value of an ATT handle is not permitted on the ATT server.
CYBLE_GATT_ERR_WRITE_NOT_PERMITTED	Write Not Permitted error code is used in the case when the permission to write the value of an ATT handle is not permitted on the ATT server.
CYBLE_GATT_ERR_INVALID_PDU	Invalid PDU error code is used in the case when the format of the PDU sent from the ATT Client is incorrect.
CYBLE_GATT_ERR_INSUFFICIENT_AUTHENTICATION	Insufficient Authentication error code is used in the case when

	an access to a handle is attempted on a un-authenticated link but the attribute requires that the link be authenticated before any client can access it.
CYBLE_GATT_ERR_REQUEST_NOT_SUPPORTED	Request not supported error code is used in the case when the server does not support the processing of an ATT request sent from the client.
CYBLE_GATT_ERR_INVALID_OFFSET	Invalid Offset error code is used in the case when the offset sent by the client in the Read blob/Prepare Write Request is invalid with respect to the length of the value in the server.
CYBLE_GATT_ERR_INSUFFICIENT_AUTHORIZATION	Insufficient Authorization error code is used in the case when the ATT server does not Authorize the client and hence prohibiting the client from reading the handle value.
CYBLE_GATT_ERR_PREPARE_WRITE_QUEUE_FULL	Write queue full error code is used when there is no more space left in the prepare write queue on the server to entertain any more prepare writes from a client.
CYBLE_GATT_ERR_ATTRIBUTE_NOT_FOUND	Attribute not found error is used when the ATT server cannot find any handles that belong to the Attribute type in the given range of handles that the client specified in its request. This error code can be sent to the client in response to the following request PDUs – Find Information, Find by Type Value, Read by Type, Read by Group Type requests.
CYBLE_GATT_ERR_ATTRIBUTE_NOT_LONG	Attribute Not Long error code is used when the client tries to read or write a Attribute handle's value which cannot be read or written through Read Blob or multiple prepare write requests.
CYBLE_GATT_ERR_INSUFFICIENT_ENC_KEY_SIZE	Insufficient encryption key size error code is used when the

	client tries to access an Attribute Handle's Value for which the link need to be encrypted with a key of certain minimum key size and the current link is encrypted with a key of lesser size than the minimum required.
CYBLE_GATT_ERR_INVALID_ATTRIBUTE_LEN	Invalid Attribute length error code is used when the Attribute value's length is not correct to process the request containing the value.
CYBLE_GATT_ERR_UNLIKELY_ERROR	Unlikely error is used when the processing of the Attribute request has encountered an error that is not covered by any other error code.
CYBLE_GATT_ERR_INSUFFICIENT_ENCRYPTION	Insufficient encryption error code is used when the client tries to read or write an Attribute handle which requires the link to be encrypted and the link is currently not encrypted.
CYBLE_GATT_ERR_UNSUPPORTED_GROUP_TYPE	Unsupported Group Type error code is used when the Attribute type requested in the Read by Group Type request is not a valid grouping attribute on the server.
CYBLE_GATT_ERR_INSUFFICIENT_RESOURCE = 0x11	Insufficient Resources error code is used when the ATT server does not have enough resources such as memory etc. to process the request from the client.
CYBLE_GATT_ERR_HEART_RATE_CONTROL_POINT_NOT_SUPPORTED = 0x80u	Heart Rate Control Point Not Supported error code is used when a unsupported code is written into Heart Rate service Control Point Characteristic.
CYBLE_GATT_ERR_CPS_INAPPROPRIATE_CONNECTION_PARAMETERS = 0x80u	The notifications of the Cycling Power Vector Characteristic cannot be sent due to inappropriate connection parameters.
CYBLE_GATTS_ERR_PROCEDURE_ALREADY_IN_PROGRESS = 0x80u	Procedure Already in Progress error code is used when a profile or service request cannot

	be serviced because an operation that has been previously triggered is still in progress.
CYBLE_GATTS_ERR_CCCD_IMPROPERLY_CONFIGURED = 0x81u	Client Characteristic Configuration Descriptor Improperly Configured error code is used when a Client Characteristic Configuration Descriptor is not configured according to the requirements of the profile or service.
CYBLE_GATT_ERR_ANS_COMMAND_NOT_SUPPORTED = 0xA0u	Command Not Supported used by the Alert Notification Server when the Client sends incorrect value of the Command ID or Category ID of to the Alert Notification Control Point Characteristic.
CYBLE_GATT_ERR_CCCD_IMPROPERLY_CONFIGURED = 0xFDu	Client Characteristic Configuration Descriptor Improperly Configured error code is used when a Client Characteristic Configuration Descriptor is not configured according to the requirements of the profile or service.
CYBLE_GATT_ERR_PROCEDURE_ALREADY_IN_PROGRESS = 0xFEu	The Procedure Already in Progress error code is used when a profile or service request cannot be serviced because an operation that has been previously triggered is still in progress.
CYBLE_GATT_ERR_OUT_OF_RANGE = 0xFFu	Out of Range error code is used when an attribute value is out of range as defined by a profile or service specification.

CYBLE_GATT_HANDLE_VALUE_OFFSET_PARAM_T

Prototype

```
typedef struct {
  CYBLE_GATT_HANDLE_VALUE_PAIR_T handleValuePair;
  uint16 offset;
} CYBLE_GATT_HANDLE_VALUE_OFFSET_PARAM_T;
```

Description

GATT Handle Value Pair along with offset type

Members

Members	Description
CYBLE_GATT_HANDLE_VALUE_PAIR_T handleValuePair;	Attribute Handle & Value to be Written
uint16 offset;	Offset at which Write is to be performed

CYBLE_GATT_HANDLE_VALUE_PAIR_T

Prototype

```
typedef struct {
  CYBLE_GATT_VALUE_T value;
  CYBLE_GATT_DB_ATTR_HANDLE_T attrHandle;
} CYBLE_GATT_HANDLE_VALUE_PAIR_T;
```

Description

GATT handle - value pair type

Members

Members	Description
CYBLE_GATT_VALUE_T value;	Attribute Value
CYBLE_GATT_DB_ATTR_HANDLE_T attrHandle;	Attribute Handle

CYBLE_GATT_PDU_T

Prototype

```
typedef enum {
  CYBLE_GATT_ERROR_RSP = 0x01u,
  CYBLE_GATT_XCNHG_MTU_REQ,
  CYBLE_GATT_XCHNG_MTU_RSP,
  CYBLE_GATT_FIND_INFO_REQ,
  CYBLE_GATT_FIND_INFO_RSP,
  CYBLE_GATT_FIND_BY_TYPE_VALUE_REQ,
  CYBLE_GATT_FIND_BY_TYPE_VALUE_RSP,
  CYBLE_GATT_READ_BY_TYPE_REQ,
  CYBLE_GATT_READ_BY_TYPE_RSP,
  CYBLE_GATT_READ_REQ,
  CYBLE_GATT_READ_RSP,
  CYBLE_GATT_READ_BLOB_REQ,
  CYBLE_GATT_READ_BLOB_RSP,
  CYBLE_GATT_READ_MULTIPLE_REQ,
  CYBLE_GATT_READ_MULTIPLE_RSP,
    CYBLE_GATT_READ_BY_GROUP_REQ,
    CYBLE_GATT_READ_BY_GROUP_RSP,
    CYBLE_GATT_WRITE_REQ,
    CYBLE_GATT_WRITE_RSP,
    CYBLE_GATT_WRITE_CMD = 0x52u,
    CYBLE_GATT_PREPARE_WRITE_REQ = 0x16u,
    CYBLE_GATT_PREPARE_WRITE_RSP,
    CYBLE_GATT_EXECUTE_WRITE_REQ,
    CYBLE_GATT_EXECUTE_WRITE_RSP,
    CYBLE_GATT_HANDLE_VALUE_NTF = 0x1Bu,
    CYBLE_GATT_HANDLE_VALUE_IND = 0x1Du,
    CYBLE_GATT_HANDLE_VALUE_CNF = 0x1Eu,
    CYBLE_GATT_SIGNED_WRITE_CMD = 0xD2,
    CYBLE_GATT_UNKNOWN_PDU_IND = 0xFFu
  } CYBLE_GATT_PDU_T;
```

Description

Opcode which has resulted in error

Members

Members	Description
CYBLE_GATT_ERROR_RSP = 0x01u	Error Response PDU
CYBLE_GATT_XCNHG_MTU_REQ	Exchange MTU Request PDU
CYBLE_GATT_XCHNG_MTU_RSP	Exchange MTU Response PDU
CYBLE_GATT_FIND_INFO_REQ	Find Information Request PDU
CYBLE_GATT_FIND_INFO_RSP	Find Information Response PDU
CYBLE_GATT_FIND_BY_TYPE_VALUE_REQ	Find By Type Value Request PDU
CYBLE_GATT_FIND_BY_TYPE_VALUE_RSP	Find By Type Value Response PDU
CYBLE_GATT_READ_BY_TYPE_REQ	Read By Type Request PDU
CYBLE_GATT_READ_BY_TYPE_RSP	Read By Type Response PDU
CYBLE_GATT_READ_REQ	Read Request PDU
CYBLE_GATT_READ_RSP	Read Response PDU
CYBLE_GATT_READ_BLOB_REQ	Read Blob Request PDU
CYBLE_GATT_READ_BLOB_RSP	Read Blob Response PDU
CYBLE_GATT_READ_MULTIPLE_REQ	Read Multiple Request PDU
CYBLE_GATT_READ_MULTIPLE_RSP	Read Multiple Response PDU
CYBLE_GATT_READ_BY_GROUP_REQ	Read Group Type Request PDU
CYBLE_GATT_READ_BY_GROUP_RSP	Read Group Type Response PDU

CYBLE_GATT_WRITE_REQ	Write Request PDU
CYBLE_GATT_WRITE_RSP	Write Response PDU
CYBLE_GATT_WRITE_CMD = 0x52u	Write Command PDU
CYBLE_GATT_PREPARE_WRITE_REQ = 0x16u	Prepare Write Request PDU
CYBLE_GATT_PREPARE_WRITE_RSP	Prepare Write Response PDU
CYBLE_GATT_EXECUTE_WRITE_REQ	Execute Write Request PDU
CYBLE_GATT_EXECUTE_WRITE_RSP	Execute Write Response PDU
CYBLE_GATT_HANDLE_VALUE_NTF = 0x1Bu	Handle Value Notification PDU
CYBLE_GATT_HANDLE_VALUE_IND = 0x1Du	Handle Value Indication PDU
CYBLE_GATT_HANDLE_VALUE_CNF = 0x1Eu	Handle Value Confirmation PDU
CYBLE_GATT_SIGNED_WRITE_CMD = 0xD2	Signed Write Command PDU
CYBLE_GATT_UNKNOWN_PDU_IND = 0xFFu	Unknown or Unhandled PDU

CYBLE_GATT_VALUE_T

Prototype

```
typedef struct {
  uint8* val;
  uint16 len;
  uint16 actualLen;
} CYBLE_GATT_VALUE_T;
```

Description

Abstracts Variable Length Values for GATT.

Apart from data, and length, 'actual length' is needed so that GATT can indicate to the application actual length of data processed for a PDU.

Is used in multiple commands - see CYBLE_GATT_READ_RSP, CYBLE_GATT_FIND_BY_TYPE_VALUE_REQ, CYBLE_GATT_READ_BLOB_RSP etc.

In GATT Read Response for example, if the attribute length is 30 octects and the MTU is 23 octets, then only first 22 octets can be sent by GATT, therefore actual length will be 22 (MTU-1). However, if the GATT MTU is configured to be 54 for example, all 30 octets can be transmitted and the actual length will be 30.

Actual length should be derived as - actualLen = length>(MTU-1) ? (MTU-1):len

In case multiple values are being packed, the actual length processed will depend on the available MTU.

Members

Members	Description
uint8* val;	Pointer to the value to be packed
uint16 len;	Length of Value to be packed
uint16 actualLen;	Out Parameter Indicating Actual Length Packed. Actual length can be less than or equal to the 'len' parameter value.

CYBLE_GATT_XCHG_MTU_PARAM_T

Prototype

```
typedef struct {
  CYBLE_CONN_HANDLE_T connHandle;
  uint16 mtu;
} CYBLE_GATT_XCHG_MTU_PARAM_T;
```

Description

MTU exchange parameter type

Members

Members	Description
CYBLE_CONN_HANDLE_T connHandle;	Connection handle
uint16 mtu;	Client/Server Rx/Tx MTU Size

CYBLE_GATTC_ERR_RSP_PARAM_T

Prototype

```
typedef struct {
  CYBLE_CONN_HANDLE_T connHandle;
  CYBLE_GATT_PDU_T opCode;
  CYBLE_GATT_DB_ATTR_HANDLE_T attrHandle;
  CYBLE_GATT_ERR_CODE_T errorCode;
} CYBLE_GATTC_ERR_RSP_PARAM_T;
```

Description

Error Response parameter type received from Server For error codes that are received during gatt discovery procedure, Client may choose to disconnect the link. i.e. if client did not get the service of its choice, client may choose to disconnect. the link.

Members

Members	Description
CYBLE_CONN_HANDLE_T connHandle;	Connection handle
CYBLE_GATT_PDU_T opCode;	Opcode which has resulted in Error
CYBLE_GATT_DB_ATTR_HANDLE_T attrHandle;	Attribute Handle in which error is generated
CYBLE_GATT_ERR_CODE_T errorCode;	Error Code describing cause of error

CYBLE_GATTC_FIND_BY_TYPE_RSP_PARAM_T

Prototype

```
typedef struct {
  CYBLE_CONN_HANDLE_T connHandle;
  CYBLE_GATT_ATTR_HANDLE_RANGE_T * range;
  uint8 count;
} CYBLE_GATTC_FIND_BY_TYPE_RSP_PARAM_T;
```

Description

GATT find by type value response received from server

Members

Members	Description
CYBLE_CONN_HANDLE_T connHandle;	Connection handle
CYBLE_GATT_ATTR_HANDLE_RANGE_T * range;	Handle Range List
uint8 count;	Size of List

CYBLE_GATTC_FIND_BY_TYPE_VALUE_REQ_T

Prototype

```
typedef struct {
  CYBLE_GATT_VALUE_T value;
  CYBLE_GATT_ATTR_HANDLE_RANGE_T range;
  CYBLE_UUID16 uuid;
} CYBLE_GATTC_FIND_BY_TYPE_VALUE_REQ_T;
```

Description

GATT find by type value request to be sent to Server

Members

Members	Description
CYBLE_GATT_VALUE_T value;	Attribute Value to Find
CYBLE_GATT_ATTR_HANDLE_RANGE_T range;	Handle Range - Start and End Handle
CYBLE_UUID16 uuid;	16-bit UUID to Find

CYBLE_GATTC_FIND_INFO_RSP_PARAM_T

Prototype

```
typedef struct {
  CYBLE_CONN_HANDLE_T connHandle;
  CYBLE_GATTC_HANDLE_UUID_LIST_PARAM_T handleValueList;
  uint8 uuidFormat;
} CYBLE_GATTC_FIND_INFO_RSP_PARAM_T;
```

Description

GATT find info response received from Server

Members

Members	Description
CYBLE_CONN_HANDLE_T connHandle;	Connection handle
CYBLE_GATTC_HANDLE_UUID_LIST_PARAM_T handleValueList;	Handle Value List
uint8 uuidFormat;	Format indicating, 16 bit (0x01) or 128 bit (0x02) UUIDs

CYBLE_GATTC_GRP_ATTR_DATA_LIST_T

Prototype

```
typedef struct {
  uint8 * attrValue;
  uint16 length;
  uint16 attrLen;
} CYBLE_GATTC_GRP_ATTR_DATA_LIST_T;
```

Description

Data Element for Group Response

Members

Members	Description
uint8 * attrValue;	atribute handle value pair
uint16 length;	Length of each Attribute Data Element including the Handle Range
uint16 attrLen;	Total Length of Attribute Data

CYBLE_GATTC_HANDLE_LIST_T

Prototype

```
typedef struct {
  uint16 * handleList;
  uint16 listCount;
  uint16 actualCount;
} CYBLE_GATTC_HANDLE_LIST_T;
```

Description

GATT handle list type

Members

Members	Description
uint16 * handleList;	Handle list where the UUID with value Indicated is found
uint16 listCount;	Number of Handles in the list
uint16 actualCount;	Actual Number of Handles Packed. This is a output parameter

CYBLE_GATTC_HANDLE_UUID_LIST_PARAM_T

Prototype

```
typedef struct {
  uint8 * list;
  uint16 byteCount;
} CYBLE_GATTC_HANDLE_UUID_LIST_PARAM_T;
```

Description

GATT list of Handle UUID pair parameter type

Members

Members	Description
uint8 * list;	Handle - UUID Pair list This is a packed byte stream, hence it needs to be unpacked and decoded.

uint16 byteCount;	Number of elements in the list in bytes

CYBLE_GATTC_HANDLE_VALUE_NTF_PARAM_T

Prototype

```
typedef struct {
  CYBLE_CONN_HANDLE_T connHandle;
  CYBLE_GATT_HANDLE_VALUE_PAIR_T handleValPair; }
CYBLE_GATTC_HANDLE_VALUE_NTF_PARAM_T;
```

Description

Handle value notification data received from server

Members

Members	Description
CYBLE_CONN_HANDLE_T connHandle;	Connection handle
CYBLE_GATT_HANDLE_VALUE_PAIR_T handleValPair;	handle value pair, actual length files needs to be ignored

CYBLE_GATTC_READ_BLOB_REQ_T

Prototype

```
typedef struct {
  CYBLE_GATT_DB_ATTR_HANDLE_T attrHandle;
  uint16 offset;
} CYBLE_GATTC_READ_BLOB_REQ_T;
```

Description

Read blob request to be sent to Server

Members

Members	Description
CYBLE_GATT_DB_ATTR_HANDLE_T attrHandle;	Handle on which Read Blob is requested
uint16 offset;	Value Offset from which the Read is Requested

CYBLE_GATTC_READ_BY_GRP_RSP_PARAM_T

Prototype

```
typedef struct {
    CYBLE_CONN_HANDLE_T connHandle;
    CYBLE_GATTC_GRP_ATTR_DATA_LIST_T attrData; } CYBLE_GATTC_READ_BY_GRP_RSP_PARAM_T;
```

Description

Read By Group Response received from Server

Members

Members	Description
CYBLE_CONN_HANDLE_T connHandle;	Connection handle
CYBLE_GATTC_GRP_ATTR_DATA_LIST_T attrData;	group attribute data list

CYBLE_GATTC_READ_BY_TYPE_REQ_T

Prototype

```
  typedef struct {
    CYBLE_GATT_ATTR_HANDLE_RANGE_T range;
    CYBLE_UUID_T uuid;
    uint8 uuidFormat;
  } CYBLE_GATTC_READ_BY_TYPE_REQ_T;
```

Description

GATT read by type request to be sent to Server

Members

Members	Description
CYBLE_GATT_ATTR_HANDLE_RANGE_T range;	Handle Range
CYBLE_UUID_T uuid;	UUID
uint8 uuidFormat;	Format indicating, 16 bit or 128 bit UUIDs For 16bits UUID format - CYBLE_GATT_16_BIT_UUID_FORMAT (0x01) For 128bits UUID format - CYBLE_GATT_128_BIT_UUID_FORMAT (0x02)

CYBLE_GATTC_READ_RSP_PARAM_T

Prototype

```
  typedef struct {
    CYBLE_CONN_HANDLE_T connHandle;
    CYBLE_GATT_VALUE_T value;
  } CYBLE_GATTC_READ_RSP_PARAM_T;
```

Description

Read response parameter type received from server

Members

Members	Description
CYBLE_CONN_HANDLE_T connHandle;	Connection handle
CYBLE_GATT_VALUE_T value;	Attribute Value

CYBLE_GATTC_T

Prototype

```
typedef struct {
  CYBLE_SRVR_CHAR_INFO_T serviceChanged;
  CYBLE_GATT_DB_ATTR_HANDLE_T cccdHandle;
} CYBLE_GATTC_T;
```

Description

Structure with discovered attributes information of Generic Attribute Service (GATTS)

Members

Members	Description
CYBLE_SRVR_CHAR_INFO_T serviceChanged;	Handle of the Service Changed Characteristic
CYBLE_GATT_DB_ATTR_HANDLE_T cccdHandle;	Client Characteristic Configuration Descriptor handle

CYBLE_GATTC_FIND_INFO_REQ_T

Prototype

```
typedef CYBLE_GATT_ATTR_HANDLE_RANGE_T CYBLE_GATTC_FIND_INFO_REQ_T;
```

Description

GATT find info request to be sent to Server

CYBLE_GATTC_HANDLE_VALUE_IND_PARAM_T

Prototype

```
typedef CYBLE_GATTC_HANDLE_VALUE_NTF_PARAM_T CYBLE_GATTC_HANDLE_VALUE_IND_PARAM_T;
```

Description

GATT handle value indication parameter received from server type

CYBLE_GATTC_PREP_WRITE_REQ_T

Prototype

```
typedef CYBLE_GATT_HANDLE_VALUE_OFFSET_PARAM_T CYBLE_GATTC_PREP_WRITE_REQ_T;
```

Description

Prepare write request to be sent to Server

CYBLE_GATTC_READ_BY_TYPE_RSP_PARAM_T

Prototype

```
typedef CYBLE_GATTC_READ_BY_GRP_RSP_PARAM_T CYBLE_GATTC_READ_BY_TYPE_RSP_PARAM_T;
```

Description

GATT read by type response received from server

CYBLE_GATTC_READ_MULT_REQ_T

Prototype

```
typedef CYBLE_GATTC_HANDLE_LIST_T CYBLE_GATTC_READ_MULT_REQ_T;
```

Description

Read multiple request to be sent to Server

CYBLE_GATTC_READ_REQ_T

Prototype

```
typedef CYBLE_GATT_DB_ATTR_HANDLE_T CYBLE_GATTC_READ_REQ_T;
```

Description

Read request to be sent to Server

CYBLE_GATTC_WRITE_CMD_REQ_T

Prototype

```
typedef CYBLE_GATT_HANDLE_VALUE_PAIR_T CYBLE_GATTC_WRITE_CMD_REQ_T;
```

Description

Write command request to be sent to Server

CYBLE_GATTC_WRITE_REQ_T

Prototype

```
typedef CYBLE_GATT_HANDLE_VALUE_PAIR_T CYBLE_GATTC_WRITE_REQ_T;
```

Description

Write request to be sent to Server

CYBLE_GATTC_EXEC_WRITE_RSP_T

Prototype

```
typedef struct {
  CYBLE_CONN_HANDLE_T connHandle;
  uint8 result;
} CYBLE_GATTC_EXEC_WRITE_RSP_T;
```

Description

Execute Write result

Members

Members	Description
CYBLE_CONN_HANDLE_T connHandle;	Connection handle
uint8 result;	Result of the execute write request

CYBLE_GATTC_SIGNED_WRITE_CMD_REQ_T

Prototype

```
typedef CYBLE_GATT_HANDLE_VALUE_PAIR_T CYBLE_GATTC_SIGNED_WRITE_CMD_REQ_T;
```

Description

Signed Write command request to be sent to Server

CYBLE_GATTS_SIGNED_WRITE_CMD_REQ_PARAM_T

Prototype

```
typedef CYBLE_GATTS_WRITE_REQ_PARAM_T CYBLE_GATTS_SIGNED_WRITE_CMD_REQ_PARAM_T;
```

Description

Signed Write command request parameter received from Client

CYBLE_GATTS_EXEC_WRITE_REQ_T

Prototype

```
typedef struct {
  CYBLE_CONN_HANDLE_T connHandle;
  CYBLE_GATT_DB_ATTR_HANDLE_T attrHandle;
  uint16 length;
  uint16 offset;
  uint8 result;
} CYBLE_GATTS_EXEC_WRITE_REQ_T;
```

Description

Execute Write result

Members

Members	Description
CYBLE_CONN_HANDLE_T connHandle;	Connection handle
CYBLE_GATT_DB_ATTR_HANDLE_T attrHandle;	Attribute Handle
uint16 length;	Total length written as part of prepare write request
uint16 offset;	Offset at which prepare write is started
uint8 result;	Result of the execute write request

CYBLE_GATTS_ATT_GENERIC_VAL_T

Prototype

```
typedef struct {
  uint16 length;
  void * attGenericVal;
} CYBLE_GATTS_ATT_GENERIC_VAL_T;
```

Description

Attribute value type used in GATT database

Members

Members	Description
uint16 length;	Length in number of bytes for attGenericVal

void * attGenericVal;	Buffer to the store generic Characteristic value based on length or complete UUID value if the attribute is of type 128-bit UUID and 32-bit UUID type.

CYBLE_GATTS_ATT_VALUE_T

Prototype

```
typedef union {
  CYBLE_GATTS_ATT_GENERIC_VAL_T attFormatValue;
  uint16 attValueUuid;
} CYBLE_GATTS_ATT_VALUE_T;
```

Description

Attribute value type used in GATT database

Members

Members	Description
CYBLE_GATTS_ATT_GENERIC_VAL_T attFormatValue;	Buffer containing 32-bit or 128-bit UUID values for Service and Characteristic declaration. Attribute format structure: if entry is for Characteristic value format, then it has the "attribute format value" of pointer type to represent generic structure to cater wide formats of available list of Characteristic formats.
uint16 attValueUuid;	Attribute UUID value

CYBLE_GATTS_DB_T

Prototype

```
typedef struct {
  uint16 attHandle;
  uint16 attType;
  uint32 permission;
  uint16 attEndHandle;
  CYBLE_GATTS_ATT_VALUE_T attValue;
} CYBLE_GATTS_DB_T;
```

Description

GATT database structure used in the GAP Server

Members

Members	Description
uint16 attHandle;	Start Handle: Act as an index for querying BLE GATT database
uint16 attType;	UUID: 16-bit UUID type for an attribute entry, for 32-bit and 128-bit UUIDs the last 16 bits should be stored in this entry. GATT DB access layer shall retrieve

	complete 128-bit UUID from CYBLE_GATTS_ATT_GENERIC_VAL_T structure.
uint32 permission;	The permission bits are clubbed in to a 32-bit field. These 32-bits can be grouped in to 4 bytes. The lowest significant byte is byte 0 (B0) and the most significant byte is byte 3 (B3). The bytes where the permissions have been grouped is as given below. Attribute permissions (B0) Characteristic permissions (B1) Implementation specific permission (B3, B2)
uint16 attEndHandle;	Attribute end handle, indicating logical boundary of given attribute.
CYBLE_GATTS_ATT_VALUE_T attValue;	Attribute value format, it can be one of following: uint16 16bit - UUID for 16bit service & Characteristic declaration CYBLE_GATTS_ATT_GENERIC_VAL_T attFormatValue - Buffer containing 32-bit or 128-bit UUID values for service & charactertistic declaration CYBLE_GATTS_ATT_GENERIC_VAL_T attFormatValue - Buffer containing generic char definition value, or generic Descriptor values

CYBLE_GATTS_ERR_PARAM_T

Prototype

```
typedef struct {
  CYBLE_GATT_DB_ATTR_HANDLE_T attrHandle;
  uint8 opcode;
  CYBLE_GATT_ERR_CODE_T errorCode;
} CYBLE_GATTS_ERR_PARAM_T;
```

Description

GATT Server Error Response parameter type

Members

Members	Description
CYBLE_GATT_DB_ATTR_HANDLE_T attrHandle;	Handle in which error is generated
uint8 opcode;	Opcode which has resulted in Error Information on ATT/GATT opcodes is available in the Bluetooth specification.
CYBLE_GATT_ERR_CODE_T errorCode;	Error Code describing cause of error

CYBLE_GATTS_PREP_WRITE_REQ_PARAM_T

Prototype

```
typedef struct {
  CYBLE_CONN_HANDLE_T connHandle;
  CYBLE_GATT_DB_ATTR_HANDLE_T attrHandle;
} CYBLE_GATTS_PREP_WRITE_REQ_PARAM_T;
```

Description

Prepare write request parameter received from Client

Members

Members	Description
CYBLE_CONN_HANDLE_T connHandle;	Connection handle
CYBLE_GATT_DB_ATTR_HANDLE_T attrHandle;	Attribute Handle

CYBLE_GATTS_WRITE_REQ_PARAM_T

Prototype

```
typedef struct {
  CYBLE_CONN_HANDLE_T connHandle;
  CYBLE_GATT_HANDLE_VALUE_PAIR_T handleValPair; } CYBLE_GATTS_WRITE_REQ_PARAM_T;
```

Description

Write request parameter received from Client

Members

Members	Description
CYBLE_CONN_HANDLE_T connHandle;	Connection handle
CYBLE_GATT_HANDLE_VALUE_PAIR_T handleValPair;	Handle value pair

CYBLE_GATTS_HANDLE_VALUE_IND_T

Prototype

```
typedef CYBLE_GATT_HANDLE_VALUE_PAIR_T CYBLE_GATTS_HANDLE_VALUE_IND_T;
```

Description

GATT handle value indication parameter type

CYBLE_GATTS_HANDLE_VALUE_NTF_T

Prototype

```
typedef CYBLE_GATT_HANDLE_VALUE_PAIR_T CYBLE_GATTS_HANDLE_VALUE_NTF_T;
```

Description

Handle value notification data to be sent to Client

CYBLE_GATTS_PREP_WRITE_RSP_PARAM_T

Prototype

```
typedef CYBLE_GATT_HANDLE_VALUE_OFFSET_PARAM_T CYBLE_GATTS_PREP_WRITE_RSP_PARAM_T;
```

Description

Prepare write response parameter to be sent to Client

CYBLE_GATTS_READ_RSP_PARAM_T

Prototype

```
typedef CYBLE_GATT_VALUE_T CYBLE_GATTS_READ_RSP_PARAM_T;
```

Description

Read response parameter to be sent to Client

CYBLE_GATTS_WRITE_CMD_REQ_PARAM_T

Prototype

```
typedef CYBLE_GATTS_WRITE_REQ_PARAM_T CYBLE_GATTS_WRITE_CMD_REQ_PARAM_T;
```

Description

Write command request parameter received from Client

CYBLE_GATTS_T

Prototype

```
typedef struct {
  CYBLE_GATT_DB_ATTR_HANDLE_T serviceHandle;
  CYBLE_GATT_DB_ATTR_HANDLE_T serviceChangedHandle;
  CYBLE_GATT_DB_ATTR_HANDLE_T cccdHandle;
} CYBLE_GATTS_T;
```

Description

Structure with Generic Attribute Service (GATTS) attribute handles

Members

Members	Description
CYBLE_GATT_DB_ATTR_HANDLE_T serviceHandle;	Service handle
CYBLE_GATT_DB_ATTR_HANDLE_T	Handle of the Service Changed Characteristic

serviceChangedHandle;	
CYBLE_GATT_DB_ATTR_HANDLE_T cccdHandle;	Client Characteristic Configuration Descriptor handle

CYBLE_DISC_CHAR_INFO_T

Prototype

```
typedef struct {
  CYBLE_GATT_DB_ATTR_HANDLE_T charDeclHandle;
  uint8 properties;
  CYBLE_GATT_DB_ATTR_HANDLE_T valueHandle;
  CYBLE_UUID_T uuid;
  uint8 uuidFormat;
} CYBLE_DISC_CHAR_INFO_T;
```

Description

Characteristic data received with read by type response during discovery process

Members

Members	Description
CYBLE_GATT_DB_ATTR_HANDLE_T charDeclHandle;	Handle for Characteristic declaration
uint8 properties;	Properties for value field
CYBLE_GATT_DB_ATTR_HANDLE_T valueHandle;	Handle to server database attribute value entry
CYBLE_UUID_T uuid;	Characteristic UUID
uint8 uuidFormat;	UUID Format - 16-bit (0x01) or 128-bit (0x02)

CYBLE_DISC_DESCR_INFO_T

Prototype

```
typedef struct {
  CYBLE_CONN_HANDLE_T connHandle;
  CYBLE_GATT_DB_ATTR_HANDLE_T descrHandle;
  CYBLE_UUID_T uuid;
  uint8 uuidFormat;
} CYBLE_DISC_DESCR_INFO_T;
```

Description

Characteristic Descriptor data received with find info response during discovery process

Members

Members	Description
CYBLE_CONN_HANDLE_T connHandle;	Handle to server database attribute entry
CYBLE_GATT_DB_ATTR_HANDLE_T descrHandle;	Descriptor handle
CYBLE_UUID_T uuid;	Descriptor UUID
uint8 uuidFormat;	UUID Format - 16-bit (0x01) or 128-bit (0x02)

CYBLE_DISC_INCL_INFO_T

Prototype

```
typedef struct {
  CYBLE_GATT_DB_ATTR_HANDLE_T inclDefHandle;
  CYBLE_GATT_ATTR_HANDLE_RANGE_T inclHandleRange;
  CYBLE_UUID_T uuid;
  uint8 uuidFormat;
} CYBLE_DISC_INCL_INFO_T;
```

Description

Included service data received with read by type response during discovery process

Members

Members	Description
CYBLE_GATT_DB_ATTR_HANDLE_T inclDefHandle;	Included definition handle
CYBLE_GATT_ATTR_HANDLE_RANGE_T inclHandleRange;	Included declaration handle range
CYBLE_UUID_T uuid;	Included UUID
uint8 uuidFormat;	UUID Format - 16-bit (0x01) or 128-bit (0x02)

CYBLE_DISC_SRVC_INFO_T

Prototype

```
typedef struct {
  CYBLE_GATT_ATTR_HANDLE_RANGE_T range;
  uint16 uuid;
} CYBLE_DISC_SRVC_INFO_T;
```

Description

CYBLE_GATT_ROLE_SERVER

L2CAP Functions

The L2CAP APIs allow access to the Logical link control and adaptation protocol (L2CAP) layer of the BLE stack.

The L2CAP API names begin with CyBle_L2cap.

Functions

Function	Description
CyBle_L2capCbfcRegisterPsm	This function registers a new upper layer protocol or PSM to L2CAP, along with the set of callbacks for the L2CAP Credit Based Flow Control... more
CyBle_L2capCbfcUnregisterPsm	This function de-registers an upper layer protocol or LE_PSM from L2CAP for the L2CAP Credit Based Flow Control mode. This is a blocking function. No... more
CyBle_L2capCbfcConnectReq	This L2CAP function initiates L2CAP channel establishment procedure in Credit Based Flow Control (CBFC) mode. Connection establishment is initiated to the specified remote Bluetooth device,... more
CyBle_L2capCbfcConnectRsp	This L2CAP function enables an upper layer protocol to respond to L2CAP connection request for LE Credit Based Flow Control mode of the specified PSM... more
CyBle_L2capCbfcSendFlowControlCredit	This L2CAP function enables an upper layer protocol to send LE Flow Control Credit packet to peer Bluetooth device, when it is capable of receiving... more
CyBle_L2capChannelDataWrite	This function sends a data packet on the L2CAP CBFC channel. This is a non-blocking function. CYBLE_EVT_L2CAP_CBFC_DATA_READ event is generated at the peer device's end... more
CyBle_L2capDisconnectReq	This function initiates sending of an L2CAP Disconnect Request (CYBLE_EVT_L2CAP_CBFC_DISCONN_IND event received by the peer device) command to the remote L2CAP entity to initiate disconnection... more
CyBle_L2capLeConnectionParamUpdateRequest	This function sends the connection parameter update request to the Master of the link. This is a non-blocking function. This function can only be used... more
CyBle_L2capLeConnectionParamUpdateResponse	This API sends the connection parameter update response to slave. This API can only be used from device connected in LE master role.

CyBle_L2capCbfcRegisterPsm

Prototype

```
CYBLE_API_RESULT_T CyBle_L2capCbfcRegisterPsm(uint16 l2capPsm, uint16 creditLwm);
```

Description

This function registers a new upper layer protocol or PSM to L2CAP, along with the set of callbacks for the L2CAP Credit Based Flow Control mode. This is a blocking function. No event is generated on calling this function.

Refer Bluetooth 4.1 core specification, Volume 3, Part A, section 3.4 for more details about credit based flow control mode of operation.

Parameters

Parameters	Description
uint16 l2capPsm	PSM value of the higher-level protocol
uint16 creditLwm	Upper Layer defined Receive Credit Low Mark

Returns

CYBLE_API_RESULT_T : Return value indicates if the function succeeded or failed. Following are the possible error codes.

Error codes	Description
CYBLE_ERROR_OK	On successful operation
CYBLE_ERROR_INVALID_PARAMETER	If 'l2capPsm' is 0
CYBLE_ERROR_INSUFFICIENT_RESOURCES	Cannot register more than one PSM
CYBLE_ERROR_L2CAP_PSM_WRONG_ENCODING	PSM value must be an odd number and the Most Significant Byte must have Least Significant Bit value set to '0'. If PSM does not follow this guideline, this return code is generated.
CYBLE_ERROR_L2CAP_PSM_ALREADY_REGISTERED	PSM already Registered

CyBle_L2capCbfcUnregisterPsm

Prototype

```
CYBLE_API_RESULT_T CyBle_L2capCbfcUnregisterPsm(uint16 l2capPsm);
```

Description

This function de-registers an upper layer protocol or LE_PSM from L2CAP for the L2CAP Credit Based Flow Control mode. This is a blocking function. No event is generated on calling this function.

Parameters

Parameters	Description
uint16 l2capPsm	PSM value of the higher-level protocol

Returns

CYBLE_API_RESULT_T : Return value indicates if the function succeeded or failed. Following are the possible error codes.

Error codes	Description
CYBLE_ERROR_OK	On successful operation
CYBLE_ERROR_MEMORY_ALLOCATION_FAILED	Memory allocation failed
CYBLE_ERROR_L2CAP_PSM_WRONG_ENCODING	L2CAP PSM value specified is incorrect or does not exist

CyBle_L2capCbfcConnectReq

Prototype

```
CYBLE_API_RESULT_T CyBle_L2capCbfcConnectReq(uint8 bdHandle, uint16 remotePsm,
uint16 localPsm, CYBLE_L2CAP_CBFC_CONNECT_PARAM_T * param);
```

Description

This L2CAP function initiates L2CAP channel establishment procedure in Credit Based Flow Control (CBFC) mode. Connection establishment is initiated to the specified remote Bluetooth device, for the specified PSM representing an upper layer protocol above L2CAP. This is a non-blocking function.

At the receiver's end, CYBLE_EVT_L2CAP_CBFC_CONN_IND event is generated. In response to this call, CYBLE_EVT_L2CAP_CBFC_CONN_CNF event is generated at the sender's end.

Refer to Bluetooth 4.1 core specification, Volume 3, Part A, section 4.22 for more details about this operation.

Parameters

Parameters	Description
uint8 bdHandle	Peer device handle.
uint16 remotePsm	Remote PSM, representing the upper layer protocol above L2CAP.
uint16 localPsm	Local PSM, representing the upper layer protocol above L2CAP.
CYBLE_L2CAP_CBFC_CONNECT_PARAM_T * param	This parameter must be a pointer to the CYBLE_L2CAP_CBFC_CONNECT_PARAM_T variable containing

	the connection parameters for the L2CAP channel.

Returns

CYBLE_API_RESULT_T : Return value indicates if the function succeeded or failed. Following are the possible error codes.

Error codes	Description
CYBLE_ERROR_OK	On successful operation
CYBLE_ERROR_INVALID_PARAMETER	If "param" is NULL
CYBLE_ERROR_INSUFFICIENT_RESOURCES	Insufficient resources
CYBLE_L2CAP_PSM_NOT_REGISTERED	PSM not Registered

CyBle_L2capCbfcConnectRsp

Prototype

```
CYBLE_API_RESULT_T CyBle_L2capCbfcConnectRsp(uint16 localCid, uint16 response,
CYBLE_L2CAP_CBFC_CONNECT_PARAM_T * param);
```

Description

This L2CAP function enables an upper layer protocol to respond to L2CAP connection request for LE Credit Based Flow Control mode of the specified PSM from the specified remote Bluetooth device. This is a non-blocking function. It is mandatory that the upper layer PSM always responds back by calling this function upon receiving CBFC Connection Request (CYBLE_EVT_L2CAP_CBFC_CONN_IND) event.

The channel is established (opened) only when the PSM concerned responds back with an event indicating success (CYBLE_EVT_L2CAP_CBFC_CONN_CNF, at the peer device's end). Otherwise, the channel establishment request from the peer will be rejected by L2CAP with appropriate result and status as received from the upper layer PSM.

Refer to Bluetooth 4.1 core specification, Volume 3, Part A, section 4.23 for more details about this operation.

Parameters

Parameters	Description
uint16 localCid	This parameter specifies the local L2CAP channel end-point for this new L2CAP channel. On receipt of L2CAP Connect Request command from the peer, local L2CAP will temporarily create a channel. This parameter identifies the new channel. If the upper layer PSM chooses to reject this connection, this temporary channel will be closed.
uint16 response	This parameter specifies the response of the upper layer for the

	new L2CAP channel establishment request from the peer. It must be set to a value as specified in L2CAP Connect Result Codes. Refer to Bluetooth 4.1 core specification, Volume 3, Part A, section 4.23 for more details.
CYBLE_L2CAP_CBFC_CONNECT_PARAM_T * param	This parameter must be a pointer to the CYBLE_L2CAP_CBFC_CONNECT_PARAM_T variable containing the connection parameters for the L2CAP channel.

Returns

CYBLE_API_RESULT_T : Return value indicates if the function succeeded or failed. Following are the possible error codes.

Error codes	Description
CYBLE_ERROR_OK	On successful operation
CYBLE_ERROR_INVALID_PARAMETER	If "param" is NULL
CYBLE_ERROR_L2CAP_CONNECTION_ENTITY_NOT_FOUND	Connection entity is not found

CyBle_L2capCbfcSendFlowControlCredit

Prototype

```
CYBLE_API_RESULT_T CyBle_L2capCbfcSendFlowControlCredit(uint16 localCid, uint16
credit);
```

Description

This L2CAP function enables an upper layer protocol to send LE Flow Control Credit packet to peer Bluetooth device, when it is capable of receiving additional LE-frames. This is a non-blocking function.

This function is invoked when the device is expecting more data from the peer device and it gets an event indicating that the peer device is low on credits CYBLE_EVT_L2CAP_CBFC_RX_CREDIT_IND for which it needs to respond by sending credits by invoking this function. Once the peer device receives these credits, it gets CYBLE_EVT_L2CAP_CBFC_TX_CREDIT_IND event indicating the same. It is the responsibility of the application layer of the device sending the credit to keep track of the total number of credits and making sure that it does not exceed 65535.

Refer to Bluetooth 4.1 core specification, Volume 3, Part A, section 4.24 for more details about this operation.

Parameters

Parameters	Description
uint16	This parameter specifies the local channel end-point for the L2CAP channel. For the initiator of

localCid	L2CAP channel establishment, this must be set to the value indicated by the CYBLE_EVT_L2CAP_CBFC_CONN_CNF event. For the responder, the upper layer protocol obtains this value when it receives the event CYBLE_EVT_L2CAP_CBFC_CONN_IND.
uint16 credit	The credit value field represents number of credits the receiving frames that can be sent to the peer device sending the LE Flow Control Credit packet. The credit value field is a number between 1 and 65535.

Returns

CYBLE_API_RESULT_T : Return value indicates if the function succeeded or failed. Following are the possible error codes.

Error codes	Description
CYBLE_ERROR_OK	On successful operation
CYBLE_ERROR_MEMORY_ALLOCATION_FAILED	Memory allocation failed
CYBLE_L2CAP_CONNECTION_ENTITY_NOT_FOUND	L2CAP connection instance is not present

CyBle_L2capChannelDataWrite

Prototype

```
CYBLE_API_RESULT_T CyBle_L2capChannelDataWrite(uint8 bdHandle, uint16 localCid,
uint8* buffer, uint16 bufferLen);
```

Description

This function sends a data packet on the L2CAP CBFC channel. This is a non-blocking function. CYBLE_EVT_L2CAP_CBFC_DATA_READ event is generated at the peer device's end after invoking this function.

Refer to Bluetooth 4.1 core specification, Volume 3, Part A, section 3.4 for more details about this operation.

Parameters

Parameters	Description
uint8 bdHandle	Peer device handle.
uint16 localCid	This parameter specifies the local channel end-point for the L2CAP channel. For the initiator of L2CAP channel establishment, this must be set to the value indicated by the CYBLE_EVT_L2CAP_CBFC_CONN_CNF event. For the responder, the upper layer protocol obtains this value when it receives the event CYBLE_EVT_L2CAP_CBFC_CONN_IND.
uint8* buffer	Buffer containing packet to be sent.
uint16 bufferLen	Packet length.

Returns

CYBLE_API_RESULT_T : Return value indicates if the function succeeded or failed. Following are the possible error codes.

Error codes	Description
CYBLE_ERROR_OK	On successful operation
CYBLE_ERROR_INVALID_PARAMETER	If "buffer" is NULL
CYBLE_ERROR_MEMORY_ALLOCATION_FAILED	Memory allocation failed
CYBLE_ERROR_NO_CONNECTION	No Link Layer connection is present
CYBLE_L2CAP_CHANNEL_NOT_FOUND	No L2ACP channel found corresponding to CID
CYBLE_L2CAP_NOT_ENOUGH_CREDITS	Not Enough Credits to transfer data

CyBle_L2capDisconnectReq

Prototype

```
CYBLE_API_RESULT_T CyBle_L2capDisconnectReq(uint16 localCid);
```

Description

This function initiates sending of an L2CAP Disconnect Request (CYBLE_EVT_L2CAP_CBFC_DISCONN_IND event received by the peer device) command to the remote L2CAP entity to initiate disconnection of the referred L2CAP channel. This is a non-blocking function.

Disconnection of the L2CAP channel always succeeds - either by reception of the L2CAP Disconnect Response from the peer, or by timeout. In any case, L2CAP will confirm disconnection of the channel, by calling the CYBLE_EVT_L2CAP_CBFC_DISCONN_CNF event.

Refer to Bluetooth 4.1 core specification, Volume 3, Part A, section 4.6 for more details about this operation.

Parameters

Parameters	Description
uint16 localCid	This parameter specifies the local channel end-point for the L2CAP channel. For initiator of L2CAP channel establishment, this must be set to the value indicated by the event CYBLE_EVT_L2CAP_CBFC_CONN_CNF. For the responder, the upper layer protocol obtains this value when it receives the event CYBLE_EVT_L2CAP_CBFC_CONN_IND.

Returns

CYBLE_API_RESULT_T : Return value indicates if the function succeeded or failed. Following are the possible error codes.

Error codes	Description
CYBLE_ERROR_OK	On successful operation
CYBLE_ERROR_INVALID_OPERATION	No Link Layer connection is present
CYBLE_ERROR_MEMORY_ALLOCATION_FAILED	Memory allocation failed
CYBLE_L2CAP_CONNECTION_ENTITY_NOT_FOUND	No connection entity found which can be disconnected

CyBle_L2capLeConnectionParamUpdateRequest

Prototype

```
CYBLE_API_RESULT_T CyBle_L2capLeConnectionParamUpdateRequest(uint8 bdHandle,
CYBLE_GAP_CONN_UPDATE_PARAM_T * connParam);
```

Description

This function sends the connection parameter update request to the Master of the link. This is a non-blocking function. This function can only be used from device connected in LE slave role.

To send connection parameter update request from the master to the slave, use CyBle_GapcConnectionParamUpdateRequest() function. This function results in CYBLE_EVT_L2CAP_CONN_PARAM_UPDATE_REQ event at the Master's end.

Refer to Bluetooth 4.1 core specification, Volume 3, Part A, section 4.20 for more details about this operation.

Parameters

Parameters	Description
uint8 bdHandle	Peer device handle
CYBLE_GAP_CONN_UPDATE_PARAM_T * connParam	Pointer to a variable of type CYBLE_GAP_CONN_UPDATE_PARAM_T which indicates the response to the Connection Parameter Update Request

Returns

CYBLE_API_RESULT_T : Return value indicates if the function succeeded or failed. Following are the possible error codes.

Error codes	Description
CYBLE_ERROR_OK	On successful operation

CYBLE_ERROR_INVALID_PARAMETER	If "connParam" is NULL
CYBLE_ERROR_INVALID_OPERATION	Connection Parameter Update Request is not allowed
CYBLE_ERROR_MEMORY_ALLOCATION_FAILED	Memory allocation failed
CYBLE_ERROR_NO_CONNECTION	No Link Layer connection is present

CyBle_L2capLeConnectionParamUpdateResponse

Prototype

```
CYBLE_API_RESULT_T CyBle_L2capLeConnectionParamUpdateResponse(uint8 bdHandle, uint16
result);
```

Description

This API sends the connection parameter update response to slave. This API can only be used from device connected in LE master role.

Parameters

Parameters	Description
uint8 bdHandle	Peer device handle

Returns

This field indicates the response to the Connection Parameter Update Request

CYBLE_API_RESULT_T : Return value indicates if the function succeeded or failed. Following are the possible error codes.

Error codes	Description
CYBLE_ERROR_OK	On successful operation
CYBLE_ERROR_INVALID_PARAMETER	If 'result' is invalid (greater than connection parameter reject code i.e., 0x0001)
CYBLE_ERROR_MEMORY_ALLOCATION_FAILED	Memory allocation failed
CYBLE_ERROR_NO_CONNECTION	No Link Layer connection is present

L2CAP Definitions and Data Structures

Contains the L2CAP specific definitions and data structures used in the L2CAP APIs.

Enumerations

Enumeration	Description
CYBLE_L2CAP_RESULT_PARAM_T	The result code of call back structures for L2CAP
CYBLE_L2CAP_COMMAND_REJ_REASON_T	Reason for command reject event - CYBLE_EVT_L2CAP_COMMAND_REJ

Structures

Structure	Description
CYBLE_L2CAP_CBFC_CONN_CNF_PARAM_T	Connect confirmation parameter
CYBLE_L2CAP_CBFC_CONN_IND_PARAM_T	Connect indication parameter
CYBLE_L2CAP_CBFC_CONNECT_PARAM_T	L2CAP Credit based flow Connection parameter
CYBLE_L2CAP_CBFC_DATA_WRITE_PARAM_T	Data Write parameter
CYBLE_L2CAP_CBFC_DISCONN_CNF_PARAM_T	Disconnect confirmation parameter
CYBLE_L2CAP_CBFC_LOW_RX_CREDIT_PARAM_T	Rx credit info parameter
CYBLE_L2CAP_CBFC_LOW_TX_CREDIT_PARAM_T	Tx credit info parameter
CYBLE_L2CAP_CBFC_RX_PARAM_T	Receive Data parameter

CYBLE_L2CAP_CBFC_CONN_CNF_PARAM_T

Prototype

```
typedef struct {
  uint8 bdHandle;
  uint16 lCid;
  uint16 response;
  CYBLE_L2CAP_CBFC_CONNECT_PARAM_T connParam;
} CYBLE_L2CAP_CBFC_CONN_CNF_PARAM_T;
```

Description

Connect confirmation parameter

Members

Members	Description
uint8 bdHandle;	bd handle of the remote device
uint16 lCid;	Local CID
uint16 response;	Response codes for Connection parameter update request
CYBLE_L2CAP_CBFC_CONNECT_PARAM_T	L2CAP Credit based flow Connection parameter

connParam;	

CYBLE_L2CAP_CBFC_CONN_IND_PARAM_T

Prototype

```
typedef struct {
  uint8 bdHandle;
  uint16 lCid;
  uint16 psm;
  CYBLE_L2CAP_CBFC_CONNECT_PARAM_T connParam;
} CYBLE_L2CAP_CBFC_CONN_IND_PARAM_T;
```

Description

Connect indication parameter

Members

Members	Description
uint8 bdHandle;	bd handle of the remote device
uint16 lCid;	Local CID
uint16 psm;	PSM value for the Protocol
CYBLE_L2CAP_CBFC_CONNECT_PARAM_T connParam;	L2CAP Credit based flow Connection parameter

CYBLE_L2CAP_CBFC_CONNECT_PARAM_T

Prototype

```
typedef struct {
  uint16 mtu;
  uint16 mps;
  uint16 credit;
} CYBLE_L2CAP_CBFC_CONNECT_PARAM_T;
```

Description

L2CAP Credit based flow Connection parameter

Members

Members	Description
uint16 mtu;	MTU - Maximum SDU Size The MTU field specifies the maximum SDU size (in octets) that the L2CAP layer entity sending the LE Credit Based Connection Request can receive on this channel. L2CAP implementations shall support a minimum MTU size of 23 octets.

uint16 mps;	MPS - Maximum PDU Size The MPS field specifies the maximum payload size (in octets) that the L2CAP layer entity sending the LE Credit Based Connection Request is capable of receiving on this channel. L2CAP implementations shall support a minimum MPS of 23 octets and may support an MPS up to 65533 octets.
uint16 credit;	Initial number of Credits The initial credit value indicates the number of LE-frames that the peer device can send to the L2CAP layer entity sending the LE Credit Based Connection Request. The initial credit value shall be in the range of 0 to 65535.

CYBLE_L2CAP_CBFC_DATA_WRITE_PARAM_T

Prototype

```
typedef struct {
  uint16 lCid;
  CYBLE_L2CAP_RESULT_PARAM_T result;
  uint8 * buffer;
  uint16 bufferLength;
} CYBLE_L2CAP_CBFC_DATA_WRITE_PARAM_T;
```

Description

Data Write parameter

Members

Members	Description
uint16 lCid;	Local CID
CYBLE_L2CAP_RESULT_PARAM_T result;	The result field indicates the outcome of the connection request. The result value of 0x0000 indicates success while a non-zero value indicates the connection request failed or is pending.
uint8 * buffer;	Currently NULL. For future usage
uint16 bufferLength;	Currently 0. For future usage

CYBLE_L2CAP_CBFC_DISCONN_CNF_PARAM_T

Prototype

```
typedef struct {
  uint16 lCid;
  CYBLE_L2CAP_RESULT_PARAM_T result;
} CYBLE_L2CAP_CBFC_DISCONN_CNF_PARAM_T;
```

Description

Disconnect confirmation parameter

Members

Members	Description
uint16 lCid;	Local CID
CYBLE_L2CAP_RESULT_PARAM_T result;	The result field indicates the outcome of the connection request. The result value of 0x0000 indicates success while a non-zero value indicates the connection request failed or is pending.

CYBLE_L2CAP_CBFC_LOW_RX_CREDIT_PARAM_T

Prototype

```
typedef struct {
  uint16 lCid;
  uint16 credit;
} CYBLE_L2CAP_CBFC_LOW_RX_CREDIT_PARAM_T;
```

Description

Rx credit info parameter

Members

Members	Description
uint16 lCid;	Local CID
uint16 credit;	The number of credits (LE-frames)

CYBLE_L2CAP_CBFC_LOW_TX_CREDIT_PARAM_T

Prototype

```
typedef struct {
  uint16 lCid;
  CYBLE_L2CAP_RESULT_PARAM_T result;
  uint16 credit;
} CYBLE_L2CAP_CBFC_LOW_TX_CREDIT_PARAM_T;
```

Description

Tx credit info parameter

Members

Members	Description
uint16 lCid;	Local CID
CYBLE_L2CAP_RESULT_PARAM_T	A result value of 0x0000 indicates success, while a non-zero value indicates an error condition (e.g. credit overflow, if total number of

result;	credits crosses specification defined maximum limit of 0xFFFF)
uint16 credit;	The number of credits (LE-frames)

CYBLE_L2CAP_CBFC_RX_PARAM_T

Prototype

```
typedef struct {
  uint16 lCid;
  CYBLE_L2CAP_RESULT_PARAM_T result;
  uint8 * rxData;
  uint16 rxDataLength;
} CYBLE_L2CAP_CBFC_RX_PARAM_T;
```

Description

Receive Data parameter

Members

Members	Description
uint16 lCid;	Local CID
CYBLE_L2CAP_RESULT_PARAM_T result;	A result value of 0x0000 indicates success, while a non-zero value indicates an error condition (e.g. peer device violating credit flow, or MTU size limit)
uint8 * rxData;	Received L2cap Data
uint16 rxDataLength;	Received L2cap Data Length

CYBLE_L2CAP_RESULT_PARAM_T

Prototype

```
typedef enum {
  CYBLE_L2CAP_RESULT_SUCCESS = 0x0000u,
  CYBLE_L2CAP_RESULT_COMMAND_TIMEOUT = 0x2318u,
  CYBLE_L2CAP_RESULT_INCORRECT_SDU_LENGTH = 0x2347u,
  CYBLE_L2CAP_RESULT_NOT_ENOUGH_CREDITS = 0x2371u,
  CYBLE_L2CAP_RESULT_CREDIT_OVERFLOW = 0x2373u,
  CYBLE_L2CAP_RESULT_UNACCEPTABLE_CREDIT_VALUE = 0x2374u
} CYBLE_L2CAP_RESULT_PARAM_T;
```

Description

The result code of call back structures for L2CAP

Members

Members	Description
CYBLE_L2CAP_RESULT_SUCCESS = 0x0000u	Operation Successful
CYBLE_L2CAP_RESULT_COMMAND_TIMEOUT = 0x2318u	Command timeout, if l2cap signaling channel timeout occurs, app should disconnect.
CYBLE_L2CAP_RESULT_INCORRECT_SDU_LENGTH = 0x2347u	Invalid sdu length
CYBLE_L2CAP_RESULT_NOT_ENOUGH_CREDITS = 0x2371u	Not enough credit to perform this operation
CYBLE_L2CAP_RESULT_CREDIT_OVERFLOW = 0x2373u	Credit overflow. Total credit exceeded 65535 (maximum)
CYBLE_L2CAP_RESULT_UNACCEPTABLE_CREDIT_VALUE = 0x2374u	Invalid credit value, receive credit is Zero

CYBLE_L2CAP_COMMAND_REJ_REASON_T

Prototype

```
typedef enum {
  CYBLE_L2CAP_COMMAND_NOT_UNDERSTOOD = 0x0000u,
  CYBLE_L2CAP_SIGNALLING_MTU_EXCEEDED,
  CYBLE_L2CAP_INVALID_CID_IN_REQUEST
} CYBLE_L2CAP_COMMAND_REJ_REASON_T;
```

Description

Reason for command reject event - CYBLE_EVT_L2CAP_COMMAND_REJ

Members

Members	Description
CYBLE_L2CAP_COMMAND_NOT_UNDERSTOOD = 0x0000u	Command Not Understood
CYBLE_L2CAP_SIGNALLING_MTU_EXCEEDED	Signaling MTU exceeded
CYBLE_L2CAP_INVALID_CID_IN_REQUEST	Invalid Connection Identifier in request

BLE Common Events

The BLE stack generates events to notify the application on various status alerts concerning the stack. These can be generic stack events or can be specific to GAP, GATT or L2CAP layers. The service specific events are handled separately in BLE Service-Specific Events.

CYBLE_EVENT_T

Prototype

```
typedef enum {
  CYBLE_EVT_HOST_INVALID = 0x00u,
  CYBLE_EVT_STACK_ON = 0x01u,
  CYBLE_EVT_TIMEOUT,
  CYBLE_EVT_HARDWARE_ERROR,
  CYBLE_EVT_HCI_STATUS,
  CYBLE_EVT_STACK_BUSY_STATUS,
  CYBLE_EVT_GAPC_SCAN_PROGRESS_RESULT = 0x20u,
  CYBLE_EVT_GAP_AUTH_REQ,
  CYBLE_EVT_GAP_PASSKEY_ENTRY_REQUEST,
  CYBLE_EVT_GAP_PASSKEY_DISPLAY_REQUEST,
  CYBLE_EVT_GAP_AUTH_COMPLETE,
  CYBLE_EVT_GAP_AUTH_FAILED,
  CYBLE_EVT_GAPP_ADVERTISEMENT_START_STOP,
  CYBLE_EVT_GAP_DEVICE_CONNECTED,
  CYBLE_EVT_GAP_DEVICE_DISCONNECTED,
  CYBLE_EVT_GAP_ENCRYPT_CHANGE,
  CYBLE_EVT_GAP_CONNECTION_UPDATE_COMPLETE,
  CYBLE_EVT_GAPC_SCAN_START_STOP,
  CYBLE_EVT_GAP_KEYINFO_EXCHNGE_CMPLT,
  CYBLE_EVT_GATTC_ERROR_RSP = 0x40u,
  CYBLE_EVT_GATT_CONNECT_IND,
  CYBLE_EVT_GATT_DISCONNECT_IND,
  CYBLE_EVT_GATTS_XCNHG_MTU_REQ,
  CYBLE_EVT_GATTC_XCHNG_MTU_RSP,
  CYBLE_EVT_GATTC_READ_BY_GROUP_TYPE_RSP,
  CYBLE_EVT_GATTC_READ_BY_TYPE_RSP,
  CYBLE_EVT_GATTC_FIND_INFO_RSP,
  CYBLE_EVT_GATTC_FIND_BY_TYPE_VALUE_RSP,
  CYBLE_EVT_GATTC_READ_RSP,
  CYBLE_EVT_GATTC_READ_BLOB_RSP,
  CYBLE_EVT_GATTC_READ_MULTI_RSP,
  CYBLE_EVT_GATTS_WRITE_REQ,
  CYBLE_EVT_GATTC_WRITE_RSP,
  CYBLE_EVT_GATTS_WRITE_CMD_REQ,
  CYBLE_EVT_GATTS_PREP_WRITE_REQ,
  CYBLE_EVT_GATTS_EXEC_WRITE_REQ,
  CYBLE_EVT_GATTC_EXEC_WRITE_RSP,
  CYBLE_EVT_GATTC_HANDLE_VALUE_NTF,
  CYBLE_EVT_GATTC_HANDLE_VALUE_IND,
  CYBLE_EVT_GATTS_HANDLE_VALUE_CNF,
  CYBLE_EVT_GATTS_DATA_SIGNED_CMD_REQ,
  CYBLE_EVT_L2CAP_CONN_PARAM_UPDATE_REQ = 0x70u,
  CYBLE_EVT_L2CAP_CONN_PARAM_UPDATE_RSP,
  CYBLE_EVT_L2CAP_COMMAND_REJ,
  CYBLE_EVT_L2CAP_CBFC_CONN_IND,
  CYBLE_EVT_L2CAP_CBFC_CONN_CNF,
  CYBLE_EVT_L2CAP_CBFC_DISCONN_IND,
  CYBLE_EVT_L2CAP_CBFC_DISCONN_CNF,
  CYBLE_EVT_L2CAP_CBFC_DATA_READ,
  CYBLE_EVT_L2CAP_CBFC_RX_CREDIT_IND,
    CYBLE_EVT_L2CAP_CBFC_TX_CREDIT_IND,
    CYBLE_EVT_L2CAP_CBFC_DATA_WRITE_IND,
    CYBLE_EVT_PENDING_FLASH_WRITE = 0xFA,
    CYBLE_EVT_MAX = 0xFF
} CYBLE_EVENT_T;
```

Description

Host stack events.

- Generic events: 0x01 to 0x1F
- GAP events: 0x20 to 0x3F
- GATT events: 0x40 to 0x6F
- L2AP events: 0x70 to 0x7F
- Future use: 0x80 to 0xFF

Members

Members	Description
CYBLE_EVT_STACK_ON = 0x01u	This event is received when BLE stack is initialized and turned ON by invoking CyBle_StackInit () function.
CYBLE_EVT_TIMEOUT	This event is received when there is a timeout and application needs to handle the event. Timeout reason is defined by CYBLE_TO_REASON_CODE_T.
CYBLE_EVT_HARDWARE_ERROR	This event indicates that some internal hardware error has occurred. Reset of the hardware may be required.
CYBLE_EVT_HCI_STATUS	This event is triggered by 'Host Stack' if 'Controller' responds with an error code for any HCI command. Event parameter returned will be an HCI error code as defined in Bluetooth 4.1 core specification, Volume 2, Part D, section 1.3. This event will be received only if there is an error.
CYBLE_EVT_STACK_BUSY_STATUS	This event is triggered by host stack if BLE stack is busy or not busy. Parameter corresponding to this event will be the state of BLE stack. BLE stack busy = CYBLE_STACK_STATE_BUSY, BLE stack not busy = CYBLE_STACK_STATE_FREE
CYBLE_EVT_GAPC_SCAN_PROGRESS_RESULT = 0x20u	This event is triggered every time a device is discovered; pointer to structure of type CYBLE_GAPC_ADV_REPORT_T is returned as the event parameter.
CYBLE_EVT_GAP_AUTH_REQ	This event is received by Peripheral and Central devices. When it is received by Peripheral, peripheral

	needs to Call CyBle_GappAuthReqReply() to reply to authentication request from Central. When this event is received by Central, that means the slave has requested Central to initiate authentication procedure. Central needs to call CyBle_GappAuthReq() to initiate authentication procedure. Pointer to structure of type CYBLE_GAP_AUTH_INFO_T is returned as the event parameter.
CYBLE_EVT_GAP_PASSKEY_ENTRY_REQUEST	This event indicates that the device has to send passkey to be used during the pairing procedure. CyBle_GapAuthPassKeyReply() is required to be called with valid parameters on receiving this event. Refer to Bluetooth Core Spec. 4.1, Part H, Section 2.3.5.1 Selecting STK Generation Method. Nothing is returned as part of the event parameter.
CYBLE_EVT_GAP_PASSKEY_DISPLAY_REQUEST	This event indicates that the device needs to display passkey during the pairing procedure. Refer to Bluetooth Core Spec. 4.1, Part H, Section 2.3.5.1 Selecting STK Generation Method. Pointer to data of type 'uint32' is returned as part of the event parameter. Passkey can be any 6-decimal-digit value.
CYBLE_EVT_GAP_AUTH_COMPLETE	This event indicates that the authentication procedure has been completed. The event parameter contains the security information as defined by CYBLE_GAP_AUTH_INFO_T. This event is generated at the end of the following three operations: Authentication is initiated with a newly connected device Encryption is initiated with a connected device that is already bonded Re-Encryption is initiated with a connected device with link already encrypted During encryption/re-encryption, the Encryption Information exchanged during the pairing process is used to encrypt/re-encrypt the link. As this does not modify any of the authentication parameters with which the devices were paired, this event is generated with NULL event data and the result of the encryption operation.
CYBLE_EVT_GAP_AUTH_FAILED	Authentication process failed between two devices. The return value of type CYBLE_GAP_AUTH_FAILED_REASON_T indicates the reason for failure.
CYBLE_EVT_GAPP_ADVERTISEMENT_START_STOP	Peripheral device has started/stopped advertising. This event is generated after making a call to CyBle_GappEnterDiscoveryMode and CyBle_GappExitDiscoveryMode functions. The event parameter contains the status which is of type 'uint8'. If the data is '0x00', it indicates 'success'; Anything else indicates 'failure'.
CYBLE_EVT_GAP_DEVICE_CONNECTED	This event is generated at the GAP Central end after connection is completed with peer device. Event

	parameter is a pointer to a structure of type CYBLE_GAP_CONN_PARAM_UPDATED_IN_CONTROLLER_T. Disconnected from remote device. Parameter returned with the event contains pointer to the the reason for disconnection, which is of type uint8.
CYBLE_EVT_GAP_DEVICE_DISCONNECTED	
CYBLE_EVT_GAP_ENCRYPT_CHANGE	Encryption change event for active connection. 'evParam' can be decoded as evParam[0] = 0x00 -> Encryption OFF evParam[0] = 0x01 -> Encryption ON Any other value of evParam[0] -> Error This is an informative event for application when there is a change in encryption. Application may choose to ignore it.
CYBLE_EVT_GAP_CONNECTION_UPDATE_COMPLETE	This event is generated at the GAP Central and the Peripheral end after connection parameter update is requested from the host to the controller. Event parameter is a pointer to a structure of type CYBLE_GAP_CONN_PARAM_UPDATED_IN_CONTROL LER_T.
CYBLE_EVT_GAPC_SCAN_START_STOP	Central device has started/stopped scanning. This event is generated after making a call to CyBle_GapcStartDiscovery and CyBle_GapcStopDiscovery APIs. The event parameter contains the status, which is of type 'uint8'. If the data is '0x00', it indicates 'success'; Anything else indicates 'failure'.
CYBLE_EVT_GAP_KEYINFO_EXCHNGE_CMPLT	Indication that the SMP keys exchange with peer device is complete, the event handler is expected to store the peer device keys, especially IRK which is used to resolve the peer device after the connection establishment. Event parameter returns data of type CYBLE_GAP_SMP_KEY_DIST_T containing the peer device keys.
CYBLE_EVT_GATTC_ERROR_RSP = 0x40u	The event is received by the Client when the Server cannot perform the requested operation and sends out an error response. Event parameter is a pointer to a structure of type CYBLE_GATTC_ERR_RSP_PARAM_T.
CYBLE_EVT_GATT_CONNECT_IND	After completion of authentication events, peer and local GATT profiles are connected. On receiving this event, profile may initiate profile level operations.
CYBLE_EVT_GATT_DISCONNECT_IND	GATT is disconnected. Nothing is returned as part of the event parameter.
CYBLE_EVT_GATTS_XCNHG_MTU_REQ	'MTU Exchange Request' received from GATT client device. Event parameter contains the MTU size of type CYBLE_GATT_XCHG_MTU_PARAM_T.
CYBLE_EVT_GATTC_XCHNG_MTU_RSP	'MTU Exchange Response' received from server device. Event parameter is a pointer to a structure of type

	CYBLE_GATT_XCHG_MTU_PARAM_T.
CYBLE_EVT_GATTC_READ_BY_GROUP_TYPE_RSP	'Read by Group Type Response' received from server device. Event parameter is a pointer to a structure of type CYBLE_GATTC_READ_BY_GRP_RSP_PARAM_T.
CYBLE_EVT_GATTC_READ_BY_TYPE_RSP	'Read by Type Response' received from server device. Event parameter is a pointer to a structure of type CYBLE_GATTC_READ_BY_TYPE_RSP_PARAM_T.
CYBLE_EVT_GATTC_FIND_INFO_RSP	'Find Information Response' received from server device. Event parameter is a pointer to a structure of type 'CYBLE_GATTC_FIND_INFO_RSP_PARAM_T.
CYBLE_EVT_GATTC_FIND_BY_TYPE_VALUE_RSP	'Find by Type Value Response' received from server device. Event parameter is a pointer to a structure of type CYBLE_GATTC_FIND_BY_TYPE_RSP_PARAM_T.
CYBLE_EVT_GATTC_READ_RSP	'Read Response' from server device. Event parameter is a pointer to a structure of type CYBLE_GATTC_READ_RSP_PARAM_T.
CYBLE_EVT_GATTC_READ_BLOB_RSP	'Read Blob Response' from server. Event parameter is a pointer to a structure of type CYBLE_GATTC_READ_RSP_PARAM_T.
CYBLE_EVT_GATTC_READ_MULTI_RSP	'Read Multiple Responses' from server. Event parameter is a pointer to a structure of type CYBLE_GATTC_READ_RSP_PARAM_T. The 'actualLen' field should be ignored as it is unused in this event response.
CYBLE_EVT_GATTS_WRITE_REQ	'Write Request' from client device. Event parameter is a pointer to a structure of type CYBLE_GATTS_WRITE_REQ_PARAM_T
CYBLE_EVT_GATTC_WRITE_RSP	'Write Response' from server device. Event parameter is a pointer to a structure of type CYBLE_CONN_HANDLE_T.
CYBLE_EVT_GATTS_WRITE_CMD_REQ	'Write Command' Request from client device. Event parameter is a pointer to a structure of type CYBLE_GATTS_WRITE_CMD_REQ_PARAM_T.
CYBLE_EVT_GATTS_PREP_WRITE_REQ	'Prepare Write' Request from client device. Event parameter is a pointer to a structure of type CYBLE_GATTS_PREP_WRITE_REQ_PARAM_T.
CYBLE_EVT_GATTS_EXEC_WRITE_REQ	'Execute Write' response from client device. Event parameter is a pointer to a structure of type 'CYBLE_GATTS_EXEC_WRITE_REQ_T' This event will be triggered as soon as GATT DB is modified. If at any point of time 'CYBLE_GATT_EXECUTE_WRITE_CANCEL_FLAG' is received in result fields of 'CYBLE_GATTS_EXEC_WRITE_REQ_T' structure, then all

	previous writes are cancelled.
CYBLE_EVT_GATTC_EXEC_WRITE_RSP	'Execute Write' response from server device. Event parameter is a pointer to a structure of type CYBLE_GATTC_EXEC_WRITE_RSP_T.
CYBLE_EVT_GATTC_HANDLE_VALUE_NTF	Notification data received from server device. Event parameter is a pointer to a structure of type CYBLE_GATTC_HANDLE_VALUE_NTF_PARAM_T.
CYBLE_EVT_GATTC_HANDLE_VALUE_IND	Indication data received from server device. Event parameter is a pointer to a structure of type CYBLE_GATTC_HANDLE_VALUE_IND_PARAM_T.
CYBLE_EVT_GATTS_HANDLE_VALUE_CNF	Confirmation to indication response from client device. Event parameter is a pointer to a structure of type CYBLE_CONN_HANDLE_T.
CYBLE_EVT_GATTS_DATA_SIGNED_CMD_REQ	Confirmation to indication response from client device. Event parameter is a pointer to a structure of type CYBLE_GATTS_SIGNED_WRITE_CMD_REQ_PARAM_T . if value.val parameter is set to Zero, then signature is not matched and ignored by stack.
CYBLE_EVT_L2CAP_CONN_PARAM_UPDATE_REQ = 0x70u	This event indicates the connection parameter update received from the remote device. The application is expected to reply to L2CAP using the CyBle_L2capLeConnectionParamUpdateResponse() function to respond to the remote device, whether parameters are accepted or rejected. Event Parameter pointer points to data of type 'CYBLE_GAP_CONN_UPDATE_PARAM_T'
CYBLE_EVT_L2CAP_CONN_PARAM_UPDATE_RSP	This event indicates the connection parameter update response received from the master. Event Parameter pointer points to data with two possible values: • Accepted = 0x0000 • Rejected = 0x0001 Data is of type unit16.
CYBLE_EVT_L2CAP_COMMAND_REJ	This event indicates the connection parameter update request has been rejected. Event parameter is a pointer to a structure of type CYBLE_CONN_UPDATE_PARAM_REJ_REASON_T.
CYBLE_EVT_L2CAP_CBFC_CONN_IND	This event is used to inform application of the incoming L2CAP CBFC Connection Request. Event parameter is a pointer to a structure of type CYBLE_L2CAP_CBFC_CONN_IND_PARAM_T is returned.
CYBLE_EVT_L2CAP_CBFC_CONN_CNF	This event is used to inform application of the L2CAP CBFC Connection Response/Confirmation. Event parameter is a pointer to a structure of type CYBLE_L2CAP_CBFC_CONN_CNF_PARAM_T is returned.

CYBLE_EVT_L2CAP_CBFC_DISCONN_IND	This event is used to inform application of the L2CAP CBFC Disconnection Request received from the Peer device. Event parameter is a pointer to Local CID of type unit16.
CYBLE_EVT_L2CAP_CBFC_DISCONN_CNF	This event is used to inform application of the L2CAP CBFC Disconnection confirmation/Response received from the Peer device. Event parameter is a pointer to a structure of type CYBLE_L2CAP_CBFC_DISCONN_CNF_PARAM_T.
CYBLE_EVT_L2CAP_CBFC_DATA_READ	This event is used to inform application of data received over L2CAP CBFC channel. Event parameter is a pointer to a structure of type CYBLE_L2CAP_CBFC_RX_PARAM_T.
CYBLE_EVT_L2CAP_CBFC_RX_CREDIT_IND	This event is used to inform the application of receive credits reached low mark. After receiving L2CAP data/payload from peer device for a specification Channel, the available credits are calculated. If the credit count goes below the low mark, this event is called to inform the application of the condition, so that if the application wantsm it can send more credits to the peer device. Event parameter is a pointer to a structure of type CYBLE_L2CAP_CBFC_LOW_RX_CREDIT_PARAM_T.
CYBLE_EVT_L2CAP_CBFC_TX_CREDIT_IND	This event is used to inform application of having received transmit credits. This event is called on receiving LE Flow Control Credit from peer device. Event parameter is a pointer to a structure of type CYBLE_L2CAP_CBFC_LOW_TX_CREDIT_PARAM_T. If the 'result' field of the received data is non-zero, this indicates an error. If the sum of 'credit' field value and the previously available credit at the peer device receiving credit information exceeds 65535, it indicates a 'credit overflow' error. In case of error, the peer device receiving this event should initiate disconnection of the L2CAP channel by invoking CyBle_L2capDisconnectReq () function.
CYBLE_EVT_L2CAP_CBFC_DATA_WRITE_IND	This event is used to inform application of data transmission completion over L2CAP CBFC channel. Event parameter is of type 'CYBLE_L2CAP_CBFC_DATA_WRITE_PARAM_T'
CYBLE_EVT_PENDING_FLASH_WRITE = 0xFA	This event is used to inform application that flash write is pending Stack internal data structures are modified and require backup .

BLE Common Definitions and Data Structures

Contains definitions and structures that are common to all BLE common APIs. Note that some of these are also used in Service-specific APIs.

Enumerations

Enumeration	Description
CYBLE_API_RESULT_T	Common error codes received as API result. • Common error codes: 0x00 to 0x0C • L2CAP error codes: 0x0D to 0x13 • GATT DB error codes: 0x14 to... more
CYBLE_TO_REASON_CODE_T	BLE stack timeout. This is received with CYBLE_EVT_TIMEOUT event It is application's responsibility to disconnect or keep the channel on depends on type of timeouts.... more
CYBLE_BLESS_PWR_LVL_T	BLESS Power enum reflecting power level values supported by BLESS radio
CYBLE_BLESS_PHY_CH_GRP_ID_T	BLE channel group ID
CYBLE_BLESS_WCO_SCA_CFG_T	BLE WCO sleep clock accuracy configuration
CYBLE_BLESS_ECO_CLK_DIV_T	BLE ECO clock divider
CYBLE_LP_MODE_T	BLE power modes
CYBLE_CLIENT_STATE_T	Client State type
CYBLE_CONN_UPDATE_PARAM_REJ_REASON_T	Reason for command reject event - L2CAP_COMMAND_REJECTED_EVENT
CYBLE_STATE_T	Event handler state machine type
CYBLE_BLESS_STATE_T	BLESS Power enum reflecting power states supported by BLESS radio.

Structures

Structure	Description
CYBLE_BLESS_PWR_IN_DB_T	Structure to set/get BLE radio power
CYBLE_BLESS_CLK_CFG_PARAMS_T	BLE clock configuration parameters
CYBLE_CONN_HANDLE_T	Connection Handle
CYBLE_UUID128_T	GATT 128 Bit UUID type
CYBLE_STACK_LIB_VERSION_T	This structure is used to hold version information of the BLE Stack Library
CYBLE_SRVR_CHAR_INFO_T	Characteristic Attribute handle + properties structure

Types

Type	Description
CYBLE_APP_CB_T	Event callback function prototype to receive events from stack
CYBLE_CALLBACK_T	Event callback function prototype to receive events from BLE Component
CYBLE_UUID16	GATT 16 Bit UUID
CYBLE_CHAR_AGGREGATE_FMT_T	This is type CYBLE_CHAR_AGGREGATE_FMT_T.
CYBLE_CHAR_PRESENT_FMT_T	This is type CYBLE_CHAR_PRESENT_FMT_T.
CYBLE_CHAR_USER_DESCRIPTION_T	This is type CYBLE_CHAR_USER_DESCRIPTION_T.
CYBLE_CLIENT_CHAR_CONFIG_T	This is type CYBLE_CLIENT_CHAR_CONFIG_T.
CYBLE_SERVER_CHAR_CONFIG_T	This is type CYBLE_SERVER_CHAR_CONFIG_T.
CYBLE_STACK_EV_CB_PF	Event callback function prototype to receive events from stack

Unions

Union	Description
CYBLE_UUID_T	GATT UUID type

CYBLE_API_RESULT_T

Prototype

```
typedef enum {
CYBLE_ERROR_OK = 0x00u,
CYBLE_ERROR_INVALID_PARAMETER,
CYBLE_ERROR_INVALID_OPERATION,
CYBLE_ERROR_MEMORY_ALLOCATION_FAILED,
CYBLE_ERROR_INSUFFICIENT_RESOURCES,
CYBLE_ERROR_OOB_NOT_AVAILABLE,
CYBLE_ERROR_NO_CONNECTION,
CYBLE_ERROR_NO_DEVICE_ENTITY,
CYBLE_ERROR_REPEATED_ATTEMPTS,
CYBLE_ERROR_GAP_ROLE,
CYBLE_ERROR_TX_POWER_READ,
CYBLE_ERROR_BT_ON_NOT_COMPLETED,
CYBLE_ERROR_SEC_FAILED,
CYBLE_ERROR_L2CAP_PSM_WRONG_ENCODING = 0x0Du,
CYBLE_ERROR_L2CAP_PSM_ALREADY_REGISTERED,
CYBLE_ERROR_L2CAP_PSM_NOT_REGISTERED,
CYBLE_ERROR_L2CAP_CONNECTION_ENTITY_NOT_FOUND,
CYBLE_ERROR_L2CAP_CHANNEL_NOT_FOUND,
CYBLE_ERROR_L2CAP_NOT_ENOUGH_CREDITS,
CYBLE_ERROR_L2CAP_PSM_NOT_IN_RANGE,
CYBLE_ERROR_GATT_DB_INVALID_ATTR_HANDLE,
CYBLE_ERROR_DEVICE_ALREADY_EXISTS = 0x27u,
CYBLE_ERROR_FLASH_WRITE_NOT_PERMITED = 0x28u,
CYBLE_ERROR_MIC_AUTH_FAILED = 0x29u,
    CYBLE_ERROR_MAX = 0xFFu
    } CYBLE_API_RESULT_T;
```

Description

Common error codes received as API result

Members

Members	Description
CYBLE_ERROR_OK = 0x00u	No Error occurred
CYBLE_ERROR_INVALID_PARAMETER	At least one of the input parameters is invalid
CYBLE_ERROR_INVALID_OPERATION	Operation is not permitted
CYBLE_ERROR_MEMORY_ALLOCATION_FAILED	An internal error occurred in the stack
CYBLE_ERROR_INSUFFICIENT_RESOURCES	Insufficient resources to perform requested operation
CYBLE_ERROR_OOB_NOT_AVAILABLE	OOB data not available
CYBLE_ERROR_NO_CONNECTION	Connection is required to perform requested operation. Connection not present
CYBLE_ERROR_NO_DEVICE_ENTITY	No device entity to perform requested operation
CYBLE_ERROR_REPEATED_ATTEMPTS	Attempted repeat operation is not allowed
CYBLE_ERROR_GAP_ROLE	GAP role is incorrect
CYBLE_ERROR_TX_POWER_READ	Error reading TC power
CYBLE_ERROR_BT_ON_NOT_COMPLETED	BLE Initialization failed
CYBLE_ERROR_SEC_FAILED	Security operation failed
CYBLE_ERROR_L2CAP_PSM_WRONG_ENCODING = 0x0Du	L2CAP PSM encoding is incorrect
CYBLE_ERROR_L2CAP_PSM_ALREADY_REGISTERED	L2CAP PSM has already been registered
CYBLE_ERROR_L2CAP_PSM_NOT_REGISTERED	L2CAP PSM has not been registered
CYBLE_ERROR_L2CAP_CONNECTION_ENTITY_NOT_FOUND	L2CAPconnection entity not found
CYBLE_ERROR_L2CAP_CHANNEL_NOT_FOUND	L2CAP channel not found
CYBLE_ERROR_L2CAP_NOT_ENOUGH_CREDITS	L2CAP not enough credits
CYBLE_ERROR_L2CAP_PSM_NOT_IN_RANGE	Specified PSM is out of range
CYBLE_ERROR_GATT_DB_INVALID_ATTR_HANDLE	Invalid attribute handle
CYBLE_ERROR_DEVICE_ALREADY_EXISTS = 0x27u	Device cannot be added to whitelist as it has already been added

Members	Description
CYBLE_ERROR_FLASH_WRITE_NOT_PERMITED = 0x28u	Write to flash is not permitted
CYBLE_ERROR_MIC_AUTH_FAILED = 0x29u	MIC Authentication failure
CYBLE_ERROR_MAX = 0xFFu	All other errors not covered in the above list map to this error code

CYBLE_TO_REASON_CODE_T

Prototype

```
typedef enum {
  CYBLE_GAP_ADV_MODE_TO = 0x01u,
  CYBLE_GAP_SCAN_TO,
  CYBLE_GATT_RSP_TO,
  CYBLE_GENERIC_TO
} CYBLE_TO_REASON_CODE_T;
```

Description

BLE stack timeout. This is received with CYBLE_EVT_TIMEOUT event It is application's responsibility to disconnect or keep the channel on depends on type of timeouts. i.e. GATT procedure timeout: Application may choose to disconnect.

Members

Members	Description
CYBLE_GAP_ADV_MODE_TO = 0x01u	Advertisement time set by application has expired
CYBLE_GAP_SCAN_TO	Scan time set by application has expired
CYBLE_GATT_RSP_TO	GATT procedure timeout
CYBLE_GENERIC_TO	Generic timeout

CYBLE_BLESS_PWR_IN_DB_T

Prototype

```
typedef struct {
  CYBLE_BLESS_PWR_LVL_T blePwrLevelInDbm;
  CYBLE_BLESS_PHY_CH_GRP_ID_T bleSsChId;
} CYBLE_BLESS_PWR_IN_DB_T;
```

Description

Structure to set/get BLE radio power

Members

Members	Description
CYBLE_BLESS_PWR_LVL_T blePwrLevelInDbm;	Output Power level
CYBLE_BLESS_PHY_CH_GRP_ID_T bleSsChId;	Channel group ID for which power level is to be read/written

CYBLE_BLESS_PWR_LVL_T

Prototype

```
typedef enum {
  CYBLE_LL_PWR_LVL_NEG_18_DBM = 0x01u,
  CYBLE_LL_PWR_LVL_NEG_12_DBM,
  CYBLE_LL_PWR_LVL_NEG_6_DBM,
  CYBLE_LL_PWR_LVL_NEG_3_DBM,
  CYBLE_LL_PWR_LVL_NEG_2_DBM,
  CYBLE_LL_PWR_LVL_NEG_1_DBM,
  CYBLE_LL_PWR_LVL_3_DBM,
  CYBLE_LL_PWR_LVL_0_DBM,
  CYBLE_LL_PWR_LVL_MAX
} CYBLE_BLESS_PWR_LVL_T;
```

Description

BLESS Power enum reflecting power level values supported by BLESS radio

Members

Members	Description
CYBLE_LL_PWR_LVL_NEG_18_DBM = 0x01u	ABS PWR = -18dBm, PA_Gain = 0x01
CYBLE_LL_PWR_LVL_NEG_12_DBM	ABS PWR = -12dBm, PA_Gain = 0x02
CYBLE_LL_PWR_LVL_NEG_6_DBM	ABS PWR = -6dBm, PA_Gain = 0x03
CYBLE_LL_PWR_LVL_NEG_3_DBM	ABS PWR = -3dBm, PA_Gain = 0x04
CYBLE_LL_PWR_LVL_NEG_2_DBM	ABS PWR = -2dBm, PA_Gain = 0x05
CYBLE_LL_PWR_LVL_NEG_1_DBM	ABS PWR = -1dBm, PA_Gain = 0x06
CYBLE_LL_PWR_LVL_3_DBM	ABS PWR = 3dBm, PA_Gain = 0x07
CYBLE_LL_PWR_LVL_0_DBM	ABS PWR = 0dBm, PA_Gain = 0x07

CYBLE_BLESS_PHY_CH_GRP_ID_T

Prototype

```
typedef enum {
  CYBLE_LL_ADV_CH_TYPE = 0x00u,
    CYBLE_LL_CONN_CH_TYPE,
    CYBLE_LL_MAX_CH_TYPE
  } CYBLE_BLESS_PHY_CH_GRP_ID_T;
```

Description

BLE channel group ID

Members

Members	Description
CYBLE_LL_ADV_CH_TYPE = 0x00u	Advertisement channel type
CYBLE_LL_CONN_CH_TYPE	Connection channel type
CYBLE_LL_MAX_CH_TYPE	Maximum value of CYBLE_BLESS_PHY_CH_GRP_ID_T type

CYBLE_BLESS_CLK_CFG_PARAMS_T

Prototype

```
  typedef struct {
    CYBLE_BLESS_WCO_SCA_CFG_T bleLlSca;
    CYBLE_BLESS_ECO_CLK_DIV_T bleLlClockDiv;
    uint16 ecoXtalStartUpTime;
  } CYBLE_BLESS_CLK_CFG_PARAMS_T;
```

Description

BLE clock configuration parameters

Members

Members	Description
CYBLE_BLESS_WCO_SCA_CFG_T bleLlSca;	32 kHz Cycles Link Layer clock divider
uint16 ecoXtalStartUpTime;	ECO crystal startup time in micro seconds. The maximum allowed value for this field is 4000 (4 milliseconds)

CYBLE_BLESS_WCO_SCA_CFG_T

Prototype

```
  typedef enum {
    CYBLE_LL_SCA_251_TO_500_PPM = 0x00u,
    CYBLE_LL_SCA_151_TO_250_PPM,
    CYBLE_LL_SCA_101_TO_150_PPM,
    CYBLE_LL_SCA_076_TO_100_PPM,
    CYBLE_LL_SCA_051_TO_075_PPM,
    CYBLE_LL_SCA_031_TO_050_PPM,
    CYBLE_LL_SCA_021_TO_030_PPM,
    CYBLE_LL_SCA_000_TO_020_PPM,
    CYBLE_LL_SCA_IN_PPM_INVALID
  } CYBLE_BLESS_WCO_SCA_CFG_T;
```

Description

BLE WCO sleep clock accuracy configuration

CYBLE_BLESS_ECO_CLK_DIV_T

Prototype

```
  typedef enum {
    CYBLE_LL_ECO_CLK_DIV_1 = 0x00u,
    CYBLE_LL_ECO_CLK_DIV_2,
    CYBLE_LL_ECO_CLK_DIV_4,
    CYBLE_LL_ECO_CLK_DIV_8,
    CYBLE_LL_ECO_CLK_DIV_INVALID
  } CYBLE_BLESS_ECO_CLK_DIV_T;
```

Description

BLE ECO clock divider

CYBLE_APP_CB_T

Prototype

```
  typedef void (* CYBLE_APP_CB_T)(uint8 event, void* evParam);
```

Description

Event callback function prototype to receive events from stack

CYBLE_CALLBACK_T

Prototype

```
  typedef void (* CYBLE_CALLBACK_T)(uint32 eventCode, void *eventParam);
```

Description

Event callback function prototype to receive events from BLE Component

CYBLE_LP_MODE_T

Prototype

```
  typedef enum {
   CYBLE_BLESS_ACTIVE = 0x01u,
   CYBLE_BLESS_SLEEP,
   CYBLE_BLESS_DEEPSLEEP,
   CYBLE_BLESS_HIBERNATE,
   CYBLE_BLESS_INVALID = 0xFFu
   } CYBLE_LP_MODE_T;
```

Description

BLE power modes

Members

Members	Description
CYBLE_BLESS_ACTIVE = 0x01u	Link Layer engine and Digital modem clocked from ECO. The CPU can access the BLE Sub-System (BLESS) registers. This mode collectively denotes Tx Mode, Rx Mode, and Idle mode of BLESS.
CYBLE_BLESS_SLEEP	The clock to the link layer engine and digital modem is gated. The ECO continues to run to maintain the link layer timing.
CYBLE_BLESS_DEEPSLEEP	The ECO is stopped and WCO is used to maintain link layer timing. RF transceiver is turned off completely to reduce leakage current. BLESS logic is kept powered ON from the SRSS deep sleep regulator for retention.
CYBLE_BLESS_HIBERNATE	External power is available but all internal LDOs are turned off.
CYBLE_BLESS_INVALID = 0xFFu	Invalid mode

CYBLE_CONN_HANDLE_T

Prototype

```
   typedef struct {
     uint8 bdHandle;
     uint8 attId;
   } CYBLE_CONN_HANDLE_T;
```

Description

Connection Handle

Members

Members	Description
uint8 bdHandle;	Identifies the peer instance
uint8 attId;	Identifies the ATT Instance

CYBLE_UUID_T

Prototype

```
typedef union {
  CYBLE_UUID16 uuid16;
  CYBLE_UUID128_T uuid128;
} CYBLE_UUID_T;
```

Description

GATT UUID type

Members

Members	Description
CYBLE_UUID16 uuid16;	16-bit UUID
CYBLE_UUID128_T uuid128;	128-bit UUID

CYBLE_UUID16

Prototype

```
typedef uint16 CYBLE_UUID16;
```

Description

GATT 16-bit UUID

CYBLE_UUID128_T

Prototype

```
typedef struct {
  uint8 value[CYBLE_GATT_128_BIT_UUID_SIZE];
} CYBLE_UUID128_T;
```

Description

GATT 128-bit UUID type

CYBLE_STACK_LIB_VERSION_T

Prototype

```
typedef struct {
  uint8 majorVersion;
  uint8 minorVersion;
  uint8 patch;
  uint8 buildNumber;
} CYBLE_STACK_LIB_VERSION_T;
```

Description

This structure is used to hold version information of the BLE Stack Library

Members

Members	Description
uint8 majorVersion;	The major version of the library
uint8 minorVersion;	The minor version of the library
uint8 patch;	The patch number of the library
uint8 buildNumber;	The build number of the library

CYBLE_CLIENT_STATE_T

Prototype

```
typedef enum {
  CYBLE_CLIENT_STATE_CONNECTED,
  CYBLE_CLIENT_STATE_SRVC_DISCOVERING,
  CYBLE_CLIENT_STATE_INCL_DISCOVERING,
  CYBLE_CLIENT_STATE_CHAR_DISCOVERING,
  CYBLE_CLIENT_STATE_DESCR_DISCOVERING,
  CYBLE_CLIENT_STATE_DISCOVERED,
  CYBLE_CLIENT_STATE_DISCONNECTING,
  CYBLE_CLIENT_STATE_DISCONNECTED_DISCOVERED,
  CYBLE_CLIENT_STATE_DISCONNECTED
} CYBLE_CLIENT_STATE_T;
```

Description

Client State type

Members

Members	Description
CYBLE_CLIENT_STATE_CONNECTED	Server device is connected
CYBLE_CLIENT_STATE_SRVC_DISCOVERING	Server services are being discovered
CYBLE_CLIENT_STATE_INCL_DISCOVERING	Server included services are being discovered
CYBLE_CLIENT_STATE_CHAR_DISCOVERING	Server Characteristics are being discovered
CYBLE_CLIENT_STATE_DESCR_DISCOVERING	Server char. Descriptors are being discovered
CYBLE_CLIENT_STATE_DISCOVERED	Server is discovered

CYBLE_CLIENT_STATE_DISCONNECTING	Server is disconnecting
CYBLE_CLIENT_STATE_DISCONNECTED_DISCOVERED	Server is disconnected but discovered
CYBLE_CLIENT_STATE_DISCONNECTED	Essentially initial client state

CYBLE_SRVR_CHAR_INFO_T

Prototype

```
typedef struct {
  uint8 properties;
  CYBLE_GATT_DB_ATTR_HANDLE_T valueHandle;
} CYBLE_SRVR_CHAR_INFO_T;
```

Description

Characteristic Attribute handle + properties structure

Members

Members	Description
uint8 properties;	Properties for value field
CYBLE_GATT_DB_ATTR_HANDLE_T valueHandle;	Handle of server database attribute value entry

CYBLE_STATE_T

Prototype

```
typedef enum {
  CYBLE_STATE_STOPPED,
  CYBLE_STATE_INITIALIZING,
  CYBLE_STATE_CONNECTED,
  CYBLE_STATE_ADVERTISING,
  CYBLE_STATE_SCANNING,
  CYBLE_STATE_CONNECTING,
  CYBLE_STATE_DISCONNECTED
} CYBLE_STATE_T;
```

Description

Event handler state machine type

Members

Members	Description
CYBLE_STATE_STOPPED	BLE is turned off
CYBLE_STATE_INITIALIZING	Initializing state

CYBLE_STATE_CONNECTED	Peer device is connected
CYBLE_STATE_ADVERTISING	Advertising process CYBLE_GAP_ROLE_PERIPHERAL || CYBLE_GAP_ROLE_BROADCASTER
CYBLE_STATE_SCANNING	Scanning process CYBLE_GAP_ROLE_CENTRAL || CYBLE_GAP_ROLE_OBSERVER
CYBLE_STATE_CONNECTING	Connecting CYBLE_GAP_ROLE_CENTRAL
CYBLE_STATE_DISCONNECTED	Essentially idle state

CYBLE_CHAR_AGGREGATE_FMT_T

Prototype

```
typedef CYBLE_GATTS_ATT_VALUE_T CYBLE_CHAR_AGGREGATE_FMT_T;
```

Description

This is type CYBLE_CHAR_AGGREGATE_FMT_T.

CYBLE_CHAR_PRESENT_FMT_T

Prototype

```
typedef CYBLE_GATTS_ATT_VALUE_T CYBLE_CHAR_PRESENT_FMT_T;
```

Description

This is type CYBLE_CHAR_PRESENT_FMT_T.

CYBLE_CHAR_USER_DESCRIPTION_T

Prototype

```
typedef CYBLE_GATTS_ATT_VALUE_T CYBLE_CHAR_USER_DESCRIPTION_T;
```

Description

This is type CYBLE_CHAR_USER_DESCRIPTION_T.

CYBLE_CLIENT_CHAR_CONFIG_T

Prototype

```
typedef CYBLE_GATTS_ATT_VALUE_T CYBLE_CLIENT_CHAR_CONFIG_T;
```

Description

This is type CYBLE_CLIENT_CHAR_CONFIG_T.

CYBLE_SERVER_CHAR_CONFIG_T

Prototype

```
typedef CYBLE_GATTS_ATT_VALUE_T CYBLE_SERVER_CHAR_CONFIG_T;
```

Description

This is type CYBLE_SERVER_CHAR_CONFIG_T.

CYBLE_STACK_EV_CB_PF

Prototype

```
typedef void (* CYBLE_STACK_EV_CB_PF)(CYBLE_EVENT_T event, void* evParam);
```

Description

Event callback function prototype to receive events from stack

CYBLE_BLESS_STATE_T

Prototype

```
typedef enum {
  CYBLE_BLESS_STATE_ACTIVE = 0x01,
  CYBLE_BLESS_STATE_EVENT_CLOSE,
  CYBLE_BLESS_STATE_SLEEP,
  CYBLE_BLESS_STATE_ECO_ON,
  CYBLE_BLESS_STATE_ECO_STABLE,
  CYBLE_BLESS_STATE_DEEPSLEEP,
  CYBLE_BLESS_STATE_HIBERNATE,
  CYBLE_BLESS_STATE_INVALID = 0xFFu
} CYBLE_BLESS_STATE_T;
```

Description

BLESS Power enum reflecting power states supported by BLESS radio.

BLE Service-Specific APIs

This section describes BLE Service-specific APIs. The Service APIs are only included in the design if the Service is added to the selected Profile in the Component GUI. These are interfaces for the BLE application to use during BLE connectivity. The service specific APIs internally use the BLE Stack APIs to achieve the Service use case.

Refer to the Bluetooth Special Interest Group Web Site for links to the latest specifications and other documentation.

Many of the APIs will generate Service-specifc events. The events are also used in the Service-specific callback functions. These are documented in:

- BLE Service-Specific Events

Alert Notification Service (ANS)

The Alert Notification Service exposes alert information in a device. This information includes:

- Type of alert occurring in a device
- Additional text information such as the caller's ID or sender's ID
- Count of new alerts
- Count of unread alert items

Depending on the chosen GATT role in the GUI, you may use a subset of the supported APIs.

The ANS API names begin with CyBle_Ans. In addition to this, the APIs also append the GATT role initial letter in the API name.

ANS Server and Client Function

These are APIs common to both GATT Client role and GATT Server role. You may use them in either roles.

No letter is appended to the API name: CyBle_Ans

Functions

Function	Description
CyBle_AnsRegisterAttrCallback	Registers a callback function for Alert Notification Service specific attribute operations.

CyBle_AnsRegisterAttrCallback

Prototype

```
void CyBle_AnsRegisterAttrCallback(CYBLE_CALLBACK_T callbackFunc);
```

Description

Registers a callback function for Alert Notification Service specific attribute operations.

Parameters

Parameters	Description
CYBLE_CALLBACK_T callbackFunc	An application layer event callback function to receive service specific events from the BLE Component. The definition of CYBLE_CALLBACK_T for Alert Notification Service is, typedef void (* CYBLE_CALLBACK_T) (uint32 eventCode, void *eventParam) eventCode indicates the event that triggered this callback (e.g. CYBLE_EVT_ANSS_NOTIFICATION_ENABLED) eventParam contains the parameters corresponding to the current event (e.g. Pointer to CYBLE_ANS_CHAR_VALUE_T structure that contains details of the Characteristic for which notification enabled event was triggered).

Returns

None

Side Effects

The *eventParams in the callback function should not be used by the application once the callback function execution is finished. Otherwise this data may become corrupted.

ANS Server Functions

APIs unique to ANS designs configured as a GATT Server role.

A letter 's' is appended to the API name: CyBle_Anss

Functions

Function	Description
CyBle_AnssSetCharacteristicValue	Sets a Characteristic value of Alert Notification Service, which is a value identified by charIndex, to the local database.
CyBle_AnssGetCharacteristicValue	Gets a Characteristic value of Alert Notification Service. The value is identified by charIndex.
CyBle_AnssGetCharacteristicDescriptor	Gets a Characteristic Descriptor of the specified Characteristic of Alert Notification Service.
CyBle_AnssSendNotification	Sends a notification with the Characteristic value, as specified by its charIndex, to the Client device.

CyBle_AnssSetCharacteristicValue

Prototype

```
CYBLE_API_RESULT_T CyBle_AnssSetCharacteristicValue(CYBLE_ANS_CHAR_INDEX_T charIndex,
uint8 attrSize, uint8 * attrValue);
```

Description

Sets a Characteristic value of Alert Notification Service, which is a value identified by charIndex, to the local database.

Parameters

Parameters	Description
CYBLE_ANS_CHAR_INDEX_T charIndex	The index of the service Characteristic of type CYBLE_ANS_CHAR_INDEX_T. The valid values are, CYBLE_ANS_SUPPORTED_NEW_ALERT_CAT CYBLE_ANS_SUPPORTED_UNREAD_ALERT_CAT
uint8 attrSize	The size of the Characteristic value attribute.
uint8 * attrValue	The pointer to Characteristic value data that should be sent to the server device.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The request is handled successfully.
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameters failed.

CyBle_AnssGetCharacteristicValue

Prototype

```
CYBLE_API_RESULT_T CyBle_AnssGetCharacteristicValue(CYBLE_ANS_CHAR_INDEX_T charIndex,
uint8 attrSize, uint8 * attrValue);
```

Description

Gets a Characteristic value of Alert Notification Service. The value is identified by charIndex.

Parameters

Parameters	Description
CYBLE_ANS_CHAR_INDEX_T charIndex	The index of the service Characteristic of type CYBLE_ANS_CHAR_INDEX_T. The valid values are, CYBLE_ANS_NEW_ALERT CYBLE_ANS_UNREAD_ALERT_STATUS
uint8 attrSize	The size of the Characteristic value attribute.
uint8 * attrValue	The pointer to the location where Characteristic value data should be stored.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The request is handled successfully.
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameter failed.

CyBle_AnssGetCharacteristicDescriptor

Prototype

```
CYBLE_API_RESULT_T CyBle_AnssGetCharacteristicDescriptor(CYBLE_ANS_CHAR_INDEX_T
charIndex, CYBLE_ANS_DESCR_INDEX_T descrIndex, uint8 attrSize, uint8 * attrValue);
```

Description

Gets a Characteristic Descriptor of the specified Characteristic of Alert Notification Service.

Parameters

Parameters	Description
CYBLE_ANS_CHAR_INDEX_T charIndex	The index of the service Characteristic of type CYBLE_ANS_CHAR_INDEX_T. The valid values are, CYBLE_ANS_NEW_ALERT CYBLE_ANS_UNREAD_ALERT_STATUS
CYBLE_ANS_DESCR_INDEX_T descrIndex	The index of the service Characteristic Descriptor of type CYBLE_ANS_DESCR_INDEX_T. The valid value is, CYBLE_ANS_CCCD
uint8 attrSize	The size of the Characteristic Descriptor attribute.
uint8 * attrValue	The pointer to the location where Characteristic Descriptor value data should be stored.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The request is handled successfully.
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameter failed.

CyBle_AnssSendNotification

Prototype

```
CYBLE_API_RESULT_T CyBle_AnssSendNotification(CYBLE_CONN_HANDLE_T connHandle,
CYBLE_ANS_CHAR_INDEX_T charIndex, uint8 attrSize, uint8 * attrValue);
```

Description

Sends a notification with the Characteristic value, as specified by its charIndex, to the Client device.

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	The connection handle.
CYBLE_ANS_CHAR_INDEX_T charIndex	The index of the service Characteristic of type CYBLE_ANS_CHAR_INDEX_T. The valid values are, CYBLE_ANS_UNREAD_ALERT_STATUS CYBLE_ANS_NEW_ALERT
uint8 attrSize	The size of the Characteristic value attribute.
uint8 * attrValue	The pointer to the Characteristic value data that should be sent to the Client device.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The function completed successfully.
- CYBLE_ERROR_INVALID_PARAMETER - Validation of input parameter is failed.
- CYBLE_ERROR_INVALID_OPERATION - Operation is invalid for this. Characteristic.
- CYBLE_ERROR_INVALID_STATE - Connection with the client is not established.
- CYBLE_ERROR_MEMORY_ALLOCATION_FAILED -Memory allocation failed.
- CYBLE_ERROR_NTF_DISABLED - Notification is not enabled by the client.

ANS Client Functions

APIs unique to ANS designs configured as a GATT Client role.

A letter 'c' is appended to the API name: CyBle_Ansc

Functions

Function	Description
CyBle_AnscSetCharacteristicValue	Sends a request to the peer device to set the Characteristic value, as identified by its charIndex.
CyBle_AnscGetCharacteristicValue	Sends a request to the peer device to get a Characteristic value, as

	identified by its charIndex.
CyBle_AnscSetCharacteristicDescriptor	Sends a request to the peer device to set the Characteristic Descriptor of the specified Characteristic of Alert Notification Service.
CyBle_AnscGetCharacteristicDescriptor	Sends a request to the peer device to get the Characteristic Descriptor of the specified Characteristic of Alert Notification Service.

CyBle_AnscSetCharacteristicValue

Prototype

```
CYBLE_API_RESULT_T CyBle_AnscSetCharacteristicValue(CYBLE_CONN_HANDLE_T connHandle,
CYBLE_ANS_CHAR_INDEX_T charIndex, uint8 attrSize, uint8 * attrValue);
```

Description

Sends a request to the peer device to set the Characteristic value, as identified by its charIndex.

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	The connection handle.
CYBLE_ANS_CHAR_INDEX_T charIndex	The index of the service Characteristic.
uint8 attrSize	Size of the Characteristic value attribute.
uint8 * attrValue	Pointer to the Characteristic value data that should be sent to the server device.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The request was sent successfully.
- CYBLE_ERROR_INVALID_STATE - The Component is in invalid state for current operation.
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameters failed.
- CYBLE_ERROR_MEMORY_ALLOCATION_FAILED -Memory allocation failed.
- CYBLE_ERROR_INVALID_OPERATION - Operation is invalid for this Characteristic.

CyBle_AnscGetCharacteristicValue

Prototype

```
CYBLE_API_RESULT_T CyBle_AnscGetCharacteristicValue(CYBLE_CONN_HANDLE_T connHandle,
CYBLE_ANS_CHAR_INDEX_T charIndex);
```

Description

Sends a request to the peer device to get a Characteristic value, as identified by its charIndex.

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	The connection handle.
CYBLE_ANS_CHAR_INDEX_T charIndex	The index of the service Characteristic.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The request was sent successfully;
- CYBLE_ERROR_INVALID_STATE - The Component is in invalid state for current operation.
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameters failed.
- CYBLE_ERROR_MEMORY_ALLOCATION_FAILED -Memory allocation failed.
- CYBLE_ERROR_INVALID_OPERATION - Operation is invalid for this Characteristic.

CyBle_AnscSetCharacteristicDescriptor

Prototype

```
CYBLE_API_RESULT_T CyBle_AnscSetCharacteristicDescriptor(CYBLE_CONN_HANDLE_T
connHandle, CYBLE_ANS_CHAR_INDEX_T charIndex, CYBLE_ANS_DESCR_INDEX_T descrIndex, uint8
attrSize, uint8 * attrValue);
```

Description

Sends a request to the peer device to set the Characteristic Descriptor of the specified Characteristic of Alert Notification Service.

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	The BLE peer device connection handle.
CYBLE_ANS_CHAR_INDEX_T charIndex	The index of the ANS Characteristic.
CYBLE_ANS_DESCR_INDEX_T descrIndex	The index of the ANS Characteristic Descriptor.
uint8 attrSize	The size of the Characteristic Descriptor attribute.
uint8 * attrValue	The pointer to the Characteristic Descriptor value data that should be sent to the server device.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The request was sent successfully.
- CYBLE_ERROR_INVALID_STATE - The Component is in invalid state for current operation.
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameters failed.
- CYBLE_ERROR_MEMORY_ALLOCATION_FAILED -Memory allocation failed.
- CYBLE_ERROR_INVALID_OPERATION - Operation is invalid for this Characteristic.

CyBle_AnscGetCharacteristicDescriptor

Prototype

```
CYBLE_API_RESULT_T CyBle_AnscGetCharacteristicDescriptor(CYBLE_CONN_HANDLE_T
connHandle, CYBLE_ANS_CHAR_INDEX_T charIndex, uint8 descrIndex);
```

Description

Sends a request to the peer device to get the Characteristic Descriptor of the specified Characteristic of Alert Notification Service.

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	BLE peer device connection handle.
CYBLE_ANS_CHAR_INDEX_T charIndex	The index of the Service Characteristic.
uint8 descrIndex	The index of the Service Characteristic Descriptor.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - A request was sent successfully.
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameters failed.
- CYBLE_ERROR_INVALID_STATE - The Component is in invalid state for current operation.
- CYBLE_ERROR_MEMORY_ALLOCATION_FAILED -Memory allocation failed.
- CYBLE_ERROR_INVALID_OPERATION - Cannot process a request to send PDU due to invalid operation performed by the application.

ANS Definitions and Data Structures

Contains the ANS specific definitions and data structures used in the ANS APIs.

Enumerations

Enumeration	Description
CYBLE_ANS_CHAR_INDEX_T	ANS Characteristic indexes
CYBLE_ANS_DESCR_INDEX_T	ANS Characteristic Descriptors indexes

Structures

Structure	Description
CYBLE_ANS_CHAR_VALUE_T	Alert Notification Service Characteristic Value parameter structure
CYBLE_ANS_DESCR_VALUE_T	Alert Notification Service Characteristic Descriptor Value parameter structure
CYBLE_ANSC_T	Structure with discovered attributes information of Alert Notification Service
CYBLE_ANSS_CHAR_T	ANS Characteristic with Descriptors
CYBLE_ANSS_T	Structure with Alert Notification Service attribute handles

CYBLE_ANS_CHAR_INDEX_T

Prototype

```
typedef enum {
  CYBLE_ANS_SUPPORTED_NEW_ALERT_CAT,
  CYBLE_ANS_NEW_ALERT,
  CYBLE_ANS_SUPPORTED_UNREAD_ALERT_CAT,
  CYBLE_ANS_UNREAD_ALERT_STATUS,
  CYBLE_ANS_ALERT_NTF_CONTROL_POINT,
  CYBLE_ANS_CHAR_COUNT
} CYBLE_ANS_CHAR_INDEX_T;
```

Description

ANS Characteristic indexes

Members

Members	Description
CYBLE_ANS_SUPPORTED_NEW_ALERT_CAT	Supported New Alert Category Characteristic index
CYBLE_ANS_NEW_ALERT	New Alert Characteristic index
CYBLE_ANS_SUPPORTED_UNREAD_ALERT_CAT	Supported Unread Alert Category Characteristic index
CYBLE_ANS_UNREAD_ALERT_STATUS	Unread Alert Status Characteristic index
CYBLE_ANS_ALERT_NTF_CONTROL_POINT	Alert Notification Control Point Characteristic index
CYBLE_ANS_CHAR_COUNT	Total count of ANS Characteristics

CYBLE_ANS_CHAR_VALUE_T

Prototype

```
typedef struct {
  CYBLE_CONN_HANDLE_T connHandle;
  CYBLE_ANS_CHAR_INDEX_T charIndex;
  CYBLE_GATT_VALUE_T * value;
} CYBLE_ANS_CHAR_VALUE_T;
```

Description

Alert Notification Service Characteristic Value parameter structure

Members

Members	Description
CYBLE_CONN_HANDLE_T connHandle;	Peer device handle
CYBLE_ANS_CHAR_INDEX_T charIndex;	Index of Alert Notification Service Characteristic
CYBLE_GATT_VALUE_T * value;	Pointer to Characteristic value

CYBLE_ANS_DESCR_INDEX_T

Prototype

```
typedef enum {
  CYBLE_ANS_CCCD,
  CYBLE_ANS_DESCR_COUNT
} CYBLE_ANS_DESCR_INDEX_T;
```

Description

ANS Characteristic Descriptors indexes

Members

Members	Description
CYBLE_ANS_CCCD	Client Characteristic Configuration Descriptor index
CYBLE_ANS_DESCR_COUNT	Total count of Descriptors

CYBLE_ANS_DESCR_VALUE_T

Prototype

```
typedef struct {
  CYBLE_CONN_HANDLE_T connHandle;
  CYBLE_ANS_CHAR_INDEX_T charIndex;
  CYBLE_ANS_DESCR_INDEX_T descrIndex;
  CYBLE_GATT_VALUE_T * value;
} CYBLE_ANS_DESCR_VALUE_T;
```

Description

Alert Notification Service Characteristic Descriptor Value parameter structure

Members

Members	Description
CYBLE_CONN_HANDLE_T connHandle;	Connection handle
CYBLE_ANS_CHAR_INDEX_T charIndex;	Characteristic index of Service
CYBLE_ANS_DESCR_INDEX_T descrIndex;	Service Characteristic Descriptor index
CYBLE_GATT_VALUE_T * value;	Pointer to value of Service Characteristic Descriptor value

CYBLE_ANSC_T

Prototype

```
typedef struct {
  CYBLE_SRVR_FULL_CHAR_INFO_T Characteristics[CYBLE_ANS_CHAR_COUNT]; } CYBLE_ANSC_T;
```

Description

Structure with discovered attributes information of Alert Notification Service

Members

Members	Description
CYBLE_SRVR_FULL_CHAR_INFO_T Characteristics[CYBLE_ANS_CHAR_COUNT];	Structure with Characteristic handles + properties of Alert Notification Service

CYBLE_ANSS_CHAR_T

Prototype

```
typedef struct {
  CYBLE_GATT_DB_ATTR_HANDLE_T charHandle;
  CYBLE_GATT_DB_ATTR_HANDLE_T descrHandle[CYBLE_ANS_DESCR_COUNT]; }
CYBLE_ANSS_CHAR_T;
```

Description

ANS Characteristic with Descriptors

Members

Members	Description
CYBLE_GATT_DB_ATTR_HANDLE_T charHandle;	Handle of Characteristic value
CYBLE_GATT_DB_ATTR_HANDLE_T descrHandle[CYBLE_ANS_DESCR_COUNT];	Handle of Descriptor

CYBLE_ANSS_T

Prototype

```
typedef struct {
  CYBLE_GATT_DB_ATTR_HANDLE_T serviceHandle;
  CYBLE_ANSS_CHAR_T charInfo[CYBLE_ANS_CHAR_COUNT];
} CYBLE_ANSS_T;
```

Description

Structure with Alert Notification Service attribute handles

Members

Members	Description
CYBLE_GATT_DB_ATTR_HANDLE_T serviceHandle;	Alert Notification Service handle
CYBLE_ANSS_CHAR_T charInfo[CYBLE_ANS_CHAR_COUNT];	Array of Alert Notification Service Characteristics + Descriptors handles

Battery Service (BAS)

The Battery Service exposes the battery level of a single battery or set of batteries in a device. Depending on the chosen GATT role in the GUI, you may use a subset of the supported APIs.

The BAS API names begin with CyBle_Bas. In addition to this, the APIs also append the GATT role initial letter in the API name.

BAS Server and Client Function

These are APIs common to both GATT Client role and GATT Server role. You may use them in either roles.

No letter is appended to the API name: CyBle_Bas

Functions

Function	Description
CyBle_BasRegisterAttrCallback	Registers a callback function for service specific attribute operations.

CyBle_BasRegisterAttrCallback

Prototype

```
void CyBle_BasRegisterAttrCallback(CYBLE_CALLBACK_T callbackFunc);
```

Description

Registers a callback function for service specific attribute operations.

Parameters

Parameters	Description
CYBLE_CALLBACK_T callbackFunc	An application layer event callback function to receive battery service events from the BLE Component. The definition of CYBLE_CALLBACK_T for Battery Service is, typedef void (* CYBLE_CALLBACK_T) (uint32 eventCode, void *eventParam) eventCode indicates the event that triggered this callback (e.g. CYBLE_EVT_BASS_NOTIFICATION_ENABLED) eventParam contains the parameters corresponding to the current event (e.g.,pointer to CYBLE_BAS_CHAR_VALUE_T structure that contains details of the Characteristic for which notification enabled event was triggered)

Returns

None

Side Effects

The *eventParams in the callback function should not be used by the application once the callback function execution is finished. Otherwise this data may become corrupted.

BAS Server Functions

APIs unique to BAS designs configured as a GATT Server role.

A letter 's' is appended to the API name: CyBle_Bass

Functions

Function	Description
CyBle_BassSetCharacteristicValue	Sets a Characteristic value of the service in the local database.
CyBle_BassGetCharacteristicValue	Gets a Characteristic value of the Battery service, which is identified by charIndex.
CyBle_BassGetCharacteristicDescriptor	Gets a Characteristic Descriptor of a specified Characteristic of the Battery service from the local GATT database.
CyBle_BassSendNotification	This function updates the value of the Battery Level Characteristic in the GATT database. If the client has configured a notification on the Battery Level... more

CyBle_BassSetCharacteristicValue

Prototype

```
CYBLE_API_RESULT_T CyBle_BassSetCharacteristicValue(uint8 serviceIndex,
CYBLE_BAS_CHAR_INDEX_T charIndex, uint8 attrSize, uint8 * attrValue);
```

Description

Sets a Characteristic value of the service in the local database.

Parameters

Parameters	Description
uint8 serviceIndex	The index of the service instance.
CYBLE_BAS_CHAR_INDEX_T charIndex	The index of the service Characteristic of type CYBLE_BAS_CHAR_INDEX_T.
uint8 attrSize	The size of the Characteristic value attribute. A battery level Characteristic has 1 byte length.
uint8 * attrValue	The pointer to the Characteristic value data that should be sent to the server device.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The request handled successfully
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameter failed

CyBle_BassGetCharacteristicValue

Prototype

```
CYBLE_API_RESULT_T CyBle_BassGetCharacteristicValue(uint8 serviceIndex,
CYBLE_BAS_CHAR_INDEX_T charIndex, uint8 attrSize, uint8 * attrValue);
```

Description

Gets a Characteristic value of the Battery service, which is identified by charIndex.

Parameters

Parameters	Description
uint8 serviceIndex	The index of the service instance. e.g. If two Battery Services are supported in your design, then first service will be identified by serviceIndex of 0 and the second by serviceIndex of 1.
CYBLE_BAS_CHAR_INDEX_T charIndex	The index of a service Characteristic of type CYBLE_BAS_CHAR_INDEX_T.
uint8 attrSize	The size of the Characteristic value attribute. A battery level Characteristic has a 1 byte length.
uint8 * attrValue	The pointer to the location where Characteristic value data should be stored.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The request handled successfully
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameter failed

CyBle_BassGetCharacteristicDescriptor

Prototype

```
CYBLE_API_RESULT_T CyBle_BassGetCharacteristicDescriptor(uint8 serviceIndex,
CYBLE_BAS_CHAR_INDEX_T charIndex, CYBLE_BAS_DESCR_INDEX_T descrIndex, uint8 attrSize,
uint8 * attrValue);
```

Description

Gets a Characteristic Descriptor of a specified Characteristic of the Battery service from the local GATT database.

Parameters

Parameters	Description
uint8 serviceIndex	The index of the service instance. e.g. If two Battery Services are supported in your design, then first service will be identified by serviceIndex of 0 and the second by serviceIndex of 1.
CYBLE_BAS_CHAR_INDEX_T charIndex	The index of a service Characteristic of type CYBLE_BAS_CHAR_INDEX_T.
CYBLE_BAS_DESCR_INDEX_T descrIndex	The index of a service Characteristic Descriptor of type CYBLE_BAS_DESCR_INDEX_T.
uint8 attrSize	The size of the Characteristic Descriptor attribute.
uint8 * attrValue	The pointer to the location where Characteristic Descriptor value data should be stored.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The request handled successfully
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameter failed

CyBle_BassSendNotification

Prototype

```
CYBLE_API_RESULT_T CyBle_BassSendNotification(CYBLE_CONN_HANDLE_T connHandle, uint8
serviceIndex, CYBLE_BAS_CHAR_INDEX_T charIndex, uint8 attrSize, uint8 * attrValue);
```

Description

This function updates the value of the Battery Level Characteristic in the GATT database. If the client has configured a notification on the Battery Level Characteristic, the function additionally sends this value using a GATT Notification message.

The CYBLE_EVT_BASC_NOTIFICATION event is received by the peer device, on invoking this function.

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	The BLE peer device connection handle
uint8 serviceIndex	The index of the service instance. e.g. If two Battery Services are supported in your design, then first service will be identified by serviceIndex of 0 and the second by serviceIndex of 1.
CYBLE_BAS_CHAR_INDEX_T charIndex	The index of a service Characteristic of type CYBLE_BAS_CHAR_INDEX_T.
uint8 attrSize	The size of the Characteristic value attribute. A battery level Characteristic has 1 byte length.
uint8 * attrValue	The pointer to the Characteristic value data that should be sent to the Client device.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The request handled successfully
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameter failed
- CYBLE_ERROR_INVALID_OPERATION - This operation is not permitted
- CYBLE_ERROR_INVALID_STATE - Connection with the client is not established
- CYBLE_ERROR_MEMORY_ALLOCATION_FAILED -Memory allocation failed.
- CYBLE_ERROR_NTF_DISABLED - Notification is not enabled by the client.

BAS Client Functions

APIs unique to BAS designs configured as a GATT Client role.

A letter 'c' is appended to the API name: CyBle_Basc

Functions

Function	Description
CyBle_BascGetCharacteristicValue	This function is used to read the Characteristic value from a server which is identified by charIndex. This function call can result in generation of... more
CyBle_BascSetCharacteristicDescriptor	Sends a request to set Characteristic Descriptor of specified Battery Service Characteristic on the server device. This function call can result in the generation of... more

Function	Description
CyBle_BascGetCharacteristicDescriptor	Sends a request to get Characteristic Descriptor of specified Battery Service Characteristic from the server device. This function call can result in generation of the... more

CyBle_BascGetCharacteristicValue

Prototype

```
CYBLE_API_RESULT_T CyBle_BascGetCharacteristicValue(CYBLE_CONN_HANDLE_T connHandle,
uint8 serviceIndex, CYBLE_BAS_CHAR_INDEX_T charIndex);
```

Description

This function is used to read the Characteristic value from a server which is identified by charIndex.

This function call can result in generation of the following events based on the response from the server device.

- CYBLE_EVT_BASC_READ_CHAR_RESPONSE
- CYBLE_EVT_GATTC_ERROR_RSP

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	The BLE peer device connection handle.
uint8 serviceIndex	Index of the service instance. e.g. If two Battery Services are supported in your design, then first service will be identified by serviceIndex of 0 and the second by serviceIndex of 1.
CYBLE_BAS_CHAR_INDEX_T charIndex	The index of a service Characteristic of type CYBLE_BAS_CHAR_INDEX_T.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The read request was sent successfully
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameters failed
- CYBLE_ERROR_GATT_DB_INVALID_ATTR_HANDLE - The peer device doesn't have the particular Characteristic
- CYBLE_ERROR_MEMORY_ALLOCATION_FAILED -Memory allocation failed

- CYBLE_ERROR_INVALID_STATE - Connection with the server is not established
- CYBLE_ERROR_INVALID_OPERATION - Operation is invalid for this Characteristic

CyBle_BascSetCharacteristicDescriptor

Prototype

```
CYBLE_API_RESULT_T CyBle_BascSetCharacteristicDescriptor(CYBLE_CONN_HANDLE_T
connHandle, uint8 serviceIndex, CYBLE_BAS_CHAR_INDEX_T charIndex,
CYBLE_BAS_DESCR_INDEX_T descrIndex, uint8 attrSize, uint8 * attrValue);
```

Description

Sends a request to set Characteristic Descriptor of specified Battery Service Characteristic on the server device. This function call can result in the generation of the following events based on the response from the server device.

- CYBLE_EVT_BASC_WRITE_DESCR_RESPONSE
- CYBLE_EVT_GATTC_ERROR_RSP

One of the following events is received by the peer device, on invoking this function.

- CYBLE_EVT_BASS_NOTIFICATION_ENABLED
- CYBLE_EVT_BASS_NOTIFICATION_DISABLED

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	The BLE peer device connection handle.
uint8 serviceIndex	Index of the service instance. e.g. If two Battery Services are supported in your design, then first service will be identified by serviceIndex of 0 and the second by serviceIndex of 1.
CYBLE_BAS_CHAR_INDEX_T charIndex	The index of a service Characteristic of type CYBLE_BAS_CHAR_INDEX_T.
CYBLE_BAS_DESCR_INDEX_T descrIndex	The index of a service Characteristic Descriptor of type CYBLE_BAS_DESCR_INDEX_T.
uint8 attrSize	The size of the Characteristic Descriptor attribute.
uint8 * attrValue	Pointer to the Characteristic Descriptor value data that should be sent to the server device.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The request was sent successfully
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameters failed
- CYBLE_ERROR_INVALID_STATE - The state is not valid
- CYBLE_ERROR_MEMORY_ALLOCATION_FAILED -Memory allocation failed
- CYBLE_ERROR_INVALID_OPERATION - This operation is not permitted on the specified attribute

CyBle_BascGetCharacteristicDescriptor

Prototype

```
CYBLE_API_RESULT_T CyBle_BascGetCharacteristicDescriptor(CYBLE_CONN_HANDLE_T
connHandle, uint8 serviceIndex, CYBLE_BAS_CHAR_INDEX_T charIndex,
CYBLE_BAS_DESCR_INDEX_T descrIndex);
```

Description

Sends a request to get Characteristic Descriptor of specified Battery Service Characteristic from the server device. This function call can result in generation of the following events based on the response from the server device.

- CYBLE_EVT_BASC_READ_DESCR_RESPONSE
- CYBLE_EVT_GATTC_ERROR_RSP

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	The BLE peer device connection handle.
uint8 serviceIndex	Index of the service instance. e.g. If two Battery Services are supported in your design, then first service will be identified by serviceIndex of 0 and the second by serviceIndex of 1.
CYBLE_BAS_CHAR_INDEX_T charIndex	The index of a Battery service Characteristic of type CYBLE_BAS_CHAR_INDEX_T.
CYBLE_BAS_DESCR_INDEX_T descrIndex	The index of a Battery service Characteristic Descriptor of type CYBLE_BAS_DESCR_INDEX_T.

Returns

- CYBLE_ERROR_OK - The request was sent successfully
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameters failed
- CYBLE_ERROR_INVALID_STATE - The state is not valid
- CYBLE_ERROR_MEMORY_ALLOCATION_FAILED -Memory allocation failed
- CYBLE_ERROR_INVALID_OPERATION - This operation is not permitted on the specified attribute

BAS Definitions and Data Structures

Contains the BAS specific definitions and data structures used in the BAS APIs.

Enumerations

Enumeration	Description
CYBLE_BAS_CHAR_INDEX_T	This is type CYBLE_BAS_CHAR_INDEX_T.
CYBLE_BAS_DESCR_INDEX_T	BAS Characteristic Descriptors indexes

Structures

Structure	Description
CYBLE_BAS_CHAR_VALUE_T	Battery Service Characteristic Value parameter structure
CYBLE_BAS_DESCR_VALUE_T	Battery Service Characteristic Descriptor Value parameter structure
CYBLE_BASC_T	Structure with discovered attributes information of Battery Service
CYBLE_BASS_NOTIF_PAR_T	This is type CYBLE_BASS_NOTIF_PAR_T.
CYBLE_BASS_T	Structure with Battery Service attribute handles

CYBLE_BAS_CHAR_INDEX_T

Prototype

```
typedef enum {
  CYBLE_BAS_BATTERY_LEVEL,
  CYBLE_BAS_CHAR_COUNT
} CYBLE_BAS_CHAR_INDEX_T;
```

Description

This is type CYBLE_BAS_CHAR_INDEX_T.

Members

Members	Description
CYBLE_BAS_BATTERY_LEVEL	Battery Level Characteristic index
CYBLE_BAS_CHAR_COUNT	Total count of Characteristics

CYBLE_BAS_CHAR_VALUE_T

Prototype

```
typedef struct {
  CYBLE_CONN_HANDLE_T connHandle;
  uint8 serviceIndex;
  CYBLE_BAS_CHAR_INDEX_T charIndex;
  CYBLE_GATT_VALUE_T * value;
} CYBLE_BAS_CHAR_VALUE_T;
```

Description

Battery Service Characteristic Value parameter structure

Members

Members	Description
CYBLE_CONN_HANDLE_T connHandle;	Peer device handle
uint8 serviceIndex;	Service instance
CYBLE_BAS_CHAR_INDEX_T charIndex;	Index of a service Characteristic
CYBLE_GATT_VALUE_T * value;	Characteristic value

CYBLE_BAS_DESCR_INDEX_T

Prototype

```
typedef enum {
  CYBLE_BAS_BATTERY_LEVEL_CCCD,
  CYBLE_BAS_BATTERY_LEVEL_CPFD,
  CYBLE_BAS_DESCR_COUNT
} CYBLE_BAS_DESCR_INDEX_T;
```

Description

BAS Characteristic Descriptors indexes

Members

Members	Description
CYBLE_BAS_BATTERY_LEVEL_CCCD	Client Characteristic Configuration Descriptor index
CYBLE_BAS_BATTERY_LEVEL_CPFD	Characteristic Presentation Format Descriptor index
CYBLE_BAS_DESCR_COUNT	Total count of Descriptors

CYBLE_BAS_DESCR_VALUE_T

Prototype

```
typedef struct {
  CYBLE_CONN_HANDLE_T connHandle;
  uint8 serviceIndex;
  CYBLE_BAS_CHAR_INDEX_T charIndex;
  CYBLE_BAS_DESCR_INDEX_T descrIndex;
  CYBLE_GATT_VALUE_T * value;
} CYBLE_BAS_DESCR_VALUE_T;
```

Description

Battery Service Characteristic Descriptor Value parameter structure

Members

Members	Description
CYBLE_CONN_HANDLE_T connHandle;	Peer device handle
uint8 serviceIndex;	Service instance
CYBLE_BAS_CHAR_INDEX_T charIndex;	Index of service Characteristic
CYBLE_BAS_DESCR_INDEX_T descrIndex;	Index of service Characteristic Descriptor
CYBLE_GATT_VALUE_T * value;	Descriptor value

CYBLE_BASC_T

Prototype

```
typedef struct {
  CYBLE_CONN_HANDLE_T connHandle;
  CYBLE_SRVR_CHAR_INFO_T batteryLevel;
  CYBLE_GATT_DB_ATTR_HANDLE_T cpfdHandle;
  CYBLE_GATT_DB_ATTR_HANDLE_T cccdHandle;
  CYBLE_GATT_DB_ATTR_HANDLE_T rrdHandle;
} CYBLE_BASC_T;
```

Description

Structure with discovered attributes information of Battery Service

Members

Members	Description
CYBLE_CONN_HANDLE_T connHandle;	Peer device handle
CYBLE_SRVR_CHAR_INFO_T batteryLevel;	Battery Level Characteristic info
CYBLE_GATT_DB_ATTR_HANDLE_T cpfdHandle;	Characteristic Presentation Format Descriptor handle
CYBLE_GATT_DB_ATTR_HANDLE_T cccdHandle;	Client Characteristic Configuration Descriptor handle
CYBLE_GATT_DB_ATTR_HANDLE_T rrdHandle;	Report Reference Descriptor handle

CYBLE_BASS_NOTIF_PAR_T

Prototype

```
typedef struct {
  CYBLE_CONN_HANDLE_T connHandle;
  uint8 serviceIndex;
  CYBLE_BAS_CHAR_INDEX_T charIndex;
} CYBLE_BASS_NOTIF_PAR_T;
```

Description

This is type CYBLE_BASS_NOTIF_PAR_T.

Members

Members	Description
CYBLE_CONN_HANDLE_T connHandle;	Peer device handle
uint8 serviceIndex;	Service instance
CYBLE_BAS_CHAR_INDEX_T charIndex;	Index of a service Characteristic

CYBLE_BASS_T

Prototype

```
typedef struct {
  CYBLE_GATT_DB_ATTR_HANDLE_T serviceHandle;
  CYBLE_GATT_DB_ATTR_HANDLE_T batteryLevelHandle;
  CYBLE_GATT_DB_ATTR_HANDLE_T cpfdHandle;
  CYBLE_GATT_DB_ATTR_HANDLE_T cccdHandle;
} CYBLE_BASS_T;
```

Description

Structure with Battery Service attribute handles

Members

Members	Description
CYBLE_GATT_DB_ATTR_HANDLE_T serviceHandle;	Battery Service handle
CYBLE_GATT_DB_ATTR_HANDLE_T batteryLevelHandle;	Battery Level Characteristic handle
CYBLE_GATT_DB_ATTR_HANDLE_T cpfdHandle;	Characteristic Presentation Format Descriptor handle
CYBLE_GATT_DB_ATTR_HANDLE_T cccdHandle;	Client Characteristic Configuration Descriptor handle

Blood Pressure Service (BLS)

The Blood Pressure Service exposes blood pressure and other data related to a non-invasive blood pressure monitor for consumer and professional healthcare applications.

Depending on the chosen GATT role in the GUI, you may use a subset of the supported APIs.

The BLS API names begin with CyBle_Bls. In addition to this, the APIs also append the GATT role initial letter in the API name.

BLS Server and Client Function

These are APIs common to both GATT Client role and GATT Server role. You may use them in either roles.

No letter is appended to the API name: CyBle_Bls

Functions

Function	Description
CyBle_BlsRegisterAttrCallback	Registers a callback function for service specific attribute operations.

CyBle_BlsRegisterAttrCallback

Prototype

```
void CyBle_BlsRegisterAttrCallback(CYBLE_CALLBACK_T callbackFunc);
```

Description

Registers a callback function for service specific attribute operations.

Parameters

Parameters	Description
CYBLE_CALLBACK_T callbackFunc	An application layer event callback function to receive events from the BLE Component. The definition of CYBLE_CALLBACK_T for Blood Pressure Service is, typedef void (* CYBLE_CALLBACK_T) (uint32 eventCode, void *eventParam) eventCode indicates the event that triggered this callback (e.g. CYBLE_EVT_BASS_NOTIFICATION_ENABLED) eventParam contains the parameters corresponding to the current event (e.g. Pointer to CYBLE_BLS_CHAR_VALUE_T structure that contains details of the Characteristic for which notification enabled event was triggered).

Returns

None

BLS Server Functions

APIs unique to BLS designs configured as a GATT Server role.

A letter 's' is appended to the API name: CyBle_Blss

Functions

Function	Description
CyBle_BlssGetCharacteristicDescriptor	Gets a Characteristic Descriptor of a specified Characteristic of the Blood pressure service from the local GATT database.
CyBle_BlssGetCharacteristicValue	Gets a Characteristic value of the Blood pressure service, which is identified by charIndex.
CyBle_BlssSendIndication	Sends an indication of the specified Characteristic to the Client device.
CyBle_BlssSendNotification	Sends a notification of the specified Characteristic to the Client device.
CyBle_BlssSetCharacteristicValue	Sets the value of a Characteristic which is identified by charIndex.

CyBle_BlssGetCharacteristicDescriptor

Prototype

```
CYBLE_API_RESULT_T CyBle_BlssGetCharacteristicDescriptor(CYBLE_BLS_CHAR_INDEX_T
charIndex, CYBLE_BLS_DESCR_INDEX_T descrIndex, uint8 attrSize, uint8 * attrValue);
```

Description

Gets a Characteristic Descriptor of a specified Characteristic of the Blood pressure service from the local GATT database.

Parameters

Parameters	Description
CYBLE_BLS_CHAR_INDEX_T charIndex	The index of the Characteristic.
CYBLE_BLS_DESCR_INDEX_T descrIndex	The index of the Characteristic Descriptor.
uint8 attrSize	The size of the Characteristic Descriptor attribute.
uint8 * attrValue	The pointer to the location where Characteristic Descriptor value data should be stored.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The request handled successfully
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameter failed
- CYBLE_ERROR_GATT_DB_INVALID_ATTR_HANDLE - Optional Descriptor is absent

CyBle_BlssGetCharacteristicValue

Prototype

```
CYBLE_API_RESULT_T CyBle_BlssGetCharacteristicValue(CYBLE_BLS_CHAR_INDEX_T charIndex,
uint8 attrSize, uint8 * attrValue);
```

Description

Gets a Characteristic value of the Blood pressure service, which is identified by charIndex.

Parameters

Parameters	Description
CYBLE_BLS_CHAR_INDEX_T charIndex	The index of a service Characteristic.
uint8 attrSize	The size of the Characteristic value attribute.
uint8 * attrValue	The pointer to the Characteristic value data that should be in the GATT database.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The request handled successfully

- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameter failed
- CYBLE_ERROR_GATT_DB_INVALID_ATTR_HANDLE - Optional Characteristic is absent

CyBle_BlssSendIndication

Prototype

```
CYBLE_API_RESULT_T CyBle_BlssSendIndication(CYBLE_CONN_HANDLE_T connHandle,
CYBLE_BLS_CHAR_INDEX_T charIndex, uint8 attrSize, uint8 * attrValue);
```

Description

Sends an indication of the specified Characteristic to the Client device.

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	The connection handle which consist of the device ID and ATT connection ID.
CYBLE_BLS_CHAR_INDEX_T charIndex	The index of the service Characteristic.
uint8 attrSize	The size of the Characteristic value attribute.
uint8 * attrValue	The pointer to the Characteristic value data that should be sent to the client device.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The request handled successfully
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameter failed
- CYBLE_ERROR_INVALID_OPERATION - This operation is not permitted
- CYBLE_ERROR_INVALID_STATE - Connection with the client is not established
- CYBLE_ERROR_MEMORY_ALLOCATION_FAILED -Memory allocation failed
- CYBLE_ERROR_IND_DISABLED - Indication is not enabled by the client

CyBle_BlssSendNotification

Prototype

```
CYBLE_API_RESULT_T CyBle_BlssSendNotification(CYBLE_CONN_HANDLE_T connHandle,
CYBLE_BLS_CHAR_INDEX_T charIndex, uint8 attrSize, uint8 * attrValue);
```

Description

Sends a notification of the specified Characteristic to the Client device.

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	The connection handle which consist of the device ID and ATT connection ID.
CYBLE_BLS_CHAR_INDEX_T charIndex	The index of the service Characteristic.
uint8 attrSize	The size of the Characteristic value attribute.
uint8 * attrValue	The pointer to the Characteristic value data that should be sent to the client device.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The request handled successfully
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameter failed
- CYBLE_ERROR_INVALID_OPERATION - This operation is not permitted
- CYBLE_ERROR_GATT_DB_INVALID_ATTR_HANDLE - Optional Characteristic is absent
- CYBLE_ERROR_INVALID_STATE - Connection with the client is not established
- CYBLE_ERROR_MEMORY_ALLOCATION_FAILED -Memory allocation failed
- CYBLE_ERROR_NTF_DISABLED - Notification is not enabled by the client

CyBle_BlssSetCharacteristicValue

Prototype

```
CYBLE_API_RESULT_T CyBle_BlssSetCharacteristicValue(CYBLE_BLS_CHAR_INDEX_T charIndex,
uint8 attrSize, uint8 * attrValue);
```

Description

Sets the value of a Characteristic which is identified by charIndex.

Parameters

Parameters	Description
CYBLE_BLS_CHAR_INDEX_T charIndex	The index of a service Characteristic.
uint8 attrSize	The size of the Characteristic value attribute.
uint8 * attrValue	The pointer to the Characteristic value data that should be sent to the server device.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The request handled successfully.
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameter failed.
- CYBLE_ERROR_GATT_DB_INVALID_ATTR_HANDLE - Optional Characteristic is absent

BLS Client Functions

APIs unique to BLS designs configured as a GATT Client role.

A letter 'c' is appended to the API name: CyBle_Blsc

Functions

Function	Description
CyBle_BlscGetCharacteristicValue	This function is used to read the Characteristic Value from a server which is identified by charIndex.
CyBle_BlscSetCharacteristicDescriptor	Sends a request to set Characteristic Descriptor of specified Blood Pressure Service Characteristic on the server device.
CyBle_BlscGetCharacteristicDescriptor	Sends a request to get Characteristic Descriptor of specified Blood Pressure Service Characteristic from the server device. This function call can result in the generation... more

CyBle_BlscGetCharacteristicValue

Prototype

```
CYBLE_API_RESULT_T CyBle_BlscGetCharacteristicValue(CYBLE_CONN_HANDLE_T connHandle,
CYBLE_BLS_CHAR_INDEX_T charIndex);
```

Description

This function is used to read the Characteristic Value from a server which is identified by charIndex.

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	The connection handle.
CYBLE_BLS_CHAR_INDEX_T charIndex	The index of the service Characteristic.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The read request was sent successfully
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameters failed
- CYBLE_ERROR_GATT_DB_INVALID_ATTR_HANDLE - The peer device doesn't have the particular Characteristic
- CYBLE_ERROR_MEMORY_ALLOCATION_FAILED -Memory allocation failed
- CYBLE_ERROR_INVALID_STATE - Connection with the server is not established
- CYBLE_ERROR_INVALID_OPERATION - Operation is invalid for this Characteristic

CyBle_BlscSetCharacteristicDescriptor

Prototype

```
CYBLE_API_RESULT_T CyBle_BlscSetCharacteristicDescriptor(CYBLE_CONN_HANDLE_T
connHandle, CYBLE_BLS_CHAR_INDEX_T charIndex, CYBLE_BLS_DESCR_INDEX_T descrIndex, uint8
attrSize, uint8 * attrValue);
```

Description

Sends a request to set Characteristic Descriptor of specified Blood Pressure Service Characteristic on the server device.

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	The BLE peer device connection handle.
CYBLE_BLS_CHAR_INDEX_T charIndex	The index of the service Characteristic.

CYBLE_BLS_DESCR_INDEX_T descrIndex	The index of the service Characteristic Descriptor.
uint8 attrSize	The size of the Characteristic Descriptor value attribute.
uint8 * attrValue	Pointer to the Characteristic Descriptor value data that should be sent to the server device.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The request was sent successfully
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameters failed
- CYBLE_ERROR_INVALID_STATE - The state is not valid
- CYBLE_ERROR_MEMORY_ALLOCATION_FAILED -Memory allocation failed
- CYBLE_ERROR_GATT_DB_INVALID_ATTR_HANDLE - The peer device doesn't have the particular Characteristic
- CYBLE_ERROR_INVALID_OPERATION - This operation is not permitted on the specified attribute

CyBle_BlscGetCharacteristicDescriptor

Prototype

```
CYBLE_API_RESULT_T CyBle_BlscGetCharacteristicDescriptor(CYBLE_CONN_HANDLE_T
connHandle, CYBLE_BLS_CHAR_INDEX_T charIndex, CYBLE_BLS_DESCR_INDEX_T descrIndex);
```

Description

Sends a request to get Characteristic Descriptor of specified Blood Pressure Service Characteristic from the server device. This function call can result in the generation of the following events based on the response from the server device.

- CYBLE_EVT_BLSC_READ_DESCR_RESPONSE
- CYBLE_EVT_GATTC_ERROR_RSP

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	The BLE peer device connection handle.
CYBLE_BLS_CHAR_INDEX_T charIndex	The index of a service Characteristic.

CYBLE_BLS_DESCR_INDEX_T descrIndex	The index of a service Characteristic Descriptor.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The request was sent successfully
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameters failed
- CYBLE_ERROR_INVALID_STATE - The state is not valid
- CYBLE_ERROR_MEMORY_ALLOCATION_FAILED -Memory allocation failed
- CYBLE_ERROR_GATT_DB_INVALID_ATTR_HANDLE - The peer device doesn't have the particular Descriptor
- CYBLE_ERROR_INVALID_OPERATION - This operation is not permitted on the specified attribute

BLS Definitions and Data Structures

Contains the BLS specific definitions and data structures used in the BLS APIs.

Enumerations

Enumeration	Description
CYBLE_BLS_CHAR_INDEX_T	Service Characteristics indexes
CYBLE_BLS_DESCR_INDEX_T	Service Characteristic Descriptors indexes

Structures

Structure	Description
CYBLE_BLS_CHAR_VALUE_T	Blood Pressure Service Characteristic Value parameter structure
CYBLE_BLS_DESCR_VALUE_T	Blood Pressure Service Characteristic Descriptor Value parameter structure
CYBLE_BLSC_CHAR_T	Blood Pressure Client Server's Characteristic structure type
CYBLE_BLSC_T	Structure with discovered atributes information of Blood Pressure Service
CYBLE_BLSS_CHAR_T	Characteristic with Descriptors
CYBLE_BLSS_T	Structure with Blood Pressure Service attribute handles

CYBLE_BLS_CHAR_INDEX_T

Prototype

```
typedef enum {
  CYBLE_BLS_BPM,
  CYBLE_BLS_ICP,
  CYBLE_BLS_BPF,
  CYBLE_BLS_CHAR_COUNT
} CYBLE_BLS_CHAR_INDEX_T;
```

Description

Service Characteristics indexes

Members

Members	Description
CYBLE_BLS_BPM	Blood Pressure Measurement Characteristic index
CYBLE_BLS_ICP	Intermediate Cuff Pressure Context Characteristic index
CYBLE_BLS_BPF	Blood Pressure Feature Characteristic index
CYBLE_BLS_CHAR_COUNT	Total count of BLS Characteristics

CYBLE_BLS_CHAR_VALUE_T

Prototype

```
typedef struct {
  CYBLE_CONN_HANDLE_T connHandle;
  CYBLE_BLS_CHAR_INDEX_T charIndex;
  CYBLE_GATT_VALUE_T * value;
} CYBLE_BLS_CHAR_VALUE_T;
```

Description

Blood Pressure Service Characteristic Value parameter structure

Members

Members	Description
CYBLE_CONN_HANDLE_T connHandle;	Peer device handle
CYBLE_BLS_CHAR_INDEX_T charIndex;	Index of service Characteristic
CYBLE_GATT_VALUE_T * value;	Characteristic value

CYBLE_BLS_DESCR_INDEX_T

Prototype

```
typedef enum {
  CYBLE_BLS_CCCD,
  CYBLE_BLS_DESCR_COUNT
} CYBLE_BLS_DESCR_INDEX_T;
```

Description

Service Characteristic Descriptors indexes

Members

Members	Description
CYBLE_BLS_CCCD	Client Characteristic Configuration Descriptor index
CYBLE_BLS_DESCR_COUNT	Total count of BLS Descriptors

CYBLE_BLS_DESCR_VALUE_T

Prototype

```
typedef struct {
  CYBLE_CONN_HANDLE_T connHandle;
  CYBLE_BLS_CHAR_INDEX_T charIndex;
  CYBLE_BLS_DESCR_INDEX_T descrIndex;
  CYBLE_GATT_VALUE_T * value;
} CYBLE_BLS_DESCR_VALUE_T;
```

Description

Blood Pressure Service Characteristic Descriptor Value parameter structure

Members

Members	Description
CYBLE_CONN_HANDLE_T connHandle;	Peer device handle
CYBLE_BLS_CHAR_INDEX_T charIndex;	Index of service Characteristic
CYBLE_BLS_DESCR_INDEX_T descrIndex;	Index of service Characteristic Descriptor
CYBLE_GATT_VALUE_T * value;	Descriptor value

CYBLE_BLSC_CHAR_T

Prototype

```
typedef struct {
    uint8 properties;
    CYBLE_GATT_DB_ATTR_HANDLE_T valueHandle;
    CYBLE_GATT_DB_ATTR_HANDLE_T cccdHandle;
    CYBLE_GATT_DB_ATTR_HANDLE_T endHandle;
  } CYBLE_BLSC_CHAR_T;
```

Description

Blood Pressure Client Server's Characteristic structure type

Members

Members	Description
uint8 properties;	Properties for value field
CYBLE_GATT_DB_ATTR_HANDLE_T valueHandle;	Handle of server database attribute value entry
CYBLE_GATT_DB_ATTR_HANDLE_T cccdHandle;	Blood Pressure client char. config. Descriptor's handle
CYBLE_GATT_DB_ATTR_HANDLE_T endHandle;	Characteristic end handle

CYBLE_BLSC_T

Prototype

```
  typedef struct {
    CYBLE_BLSC_CHAR_T charInfo[CYBLE_BLS_CHAR_COUNT];
  } CYBLE_BLSC_T;
```

Description

Structure with discovered atributes information of Blood Pressure Service

Members

Members	Description
CYBLE_BLSC_CHAR_T charInfo[CYBLE_BLS_CHAR_COUNT];	Structure with Characteristic handles + properties of Blood Pressure Service

CYBLE_BLSS_CHAR_T

Prototype

```
  typedef struct {
    CYBLE_GATT_DB_ATTR_HANDLE_T charHandle;
    CYBLE_GATT_DB_ATTR_HANDLE_T cccdHandle;
  } CYBLE_BLSS_CHAR_T;
```

Description

Characteristic with Descriptors

Members

Members	Description
CYBLE_GATT_DB_ATTR_HANDLE_T charHandle;	Blood Pressure Service Characteristic's handle
CYBLE_GATT_DB_ATTR_HANDLE_T cccdHandle;	Blood Pressure Service char. Descriptor's handle

CYBLE_BLSS_T

Prototype

```
typedef struct {
  CYBLE_GATT_DB_ATTR_HANDLE_T serviceHandle;
  CYBLE_BLSS_CHAR_T charInfo[CYBLE_BLS_CHAR_COUNT];
} CYBLE_BLSS_T;
```

Description

Structure with Blood Pressure Service attribute handles

Members

Members	Description
CYBLE_GATT_DB_ATTR_HANDLE_T serviceHandle;	Blood Pressure Service handle
CYBLE_BLSS_CHAR_T charInfo[CYBLE_BLS_CHAR_COUNT];	Array of Blood Pressure Service Characteristics + Descriptors handles

Current Time Service (CTS)

Many Bluetooth devices have the ability to store and show time information. This Service defines how a Bluetooth device can expose time information to other Bluetooth devices.

Depending on the chosen GATT role in the GUI, you may use a subset of the supported APIs.

The CTS API names begin with CyBle_Cts. In addition to this, the APIs also append the GATT role initial letter in the API name.

CTS Server and Client Function

These are APIs common to both GATT Client role and GATT Server role. You may use them in either roles.

No letter is appended to the API name: CyBle_Cts

Functions

Function	Description
CyBle_CtsRegisterAttrCallback	Registers a callback function for service specific attribute operations.

CyBle_CtsRegisterAttrCallback

Prototype

```
void CyBle_CtsRegisterAttrCallback(CYBLE_CALLBACK_T callbackFunc);
```

Description

Registers a callback function for service specific attribute operations.

Parameters

Parameters	Description
CYBLE_CALLBACK_T callbackFunc	An application layer event callback function to receive events from the BLE Component. The definition of CYBLE_CALLBACK_T for Current Time Service is, typedef void (* CYBLE_CALLBACK_T) (uint32 eventCode, void *eventParam) eventCode indicates the event that triggered this callback (e.g. CYBLE_EVT_CTSS_NOTIFICATION_ENABLED) eventParam contains the parameters corresponding to the current event (e.g. Pointer to CYBLE_CTS_CHAR_VALUE_T structure that contains details of the Characteristic for which notification enabled event was triggered).

Returns

None

CTS Server Functions

APIs unique to CTS designs configured as a GATT Server role.

A letter 's' is appended to the API name: CyBle_Ctss

Functions

Function	Description
CyBle_CtssSetCharacteristicValue	Sets a value for one of three Characteristic values of the Current Time Service. The Characteristic is identified by charIndex.
CyBle_CtssGetCharacteristicValue	Gets a Characteristic value of the Current Time Service, which is identified by charIndex.
CyBle_CtssGetCharacteristicDescriptor	Gets a Characteristic Descriptor of a specified Characteristic of the Current Time Service.

CyBle_CtssSendNotification	Sends a notification to the Client device. A Characteristic value also gets written to the GATT database.

CyBle_CtssSetCharacteristicValue

Prototype

```
CYBLE_API_RESULT_T CyBle_CtssSetCharacteristicValue(CYBLE_CTS_CHAR_INDEX_T charIndex,
uint8 attrSize, uint8 * attrValue);
```

Description

Sets a value for one of three Characteristic values of the Current Time Service. The Characteristic is identified by charIndex.

Parameters

Parameters	Description
CYBLE_CTS_CHAR_INDEX_T charIndex	The index of the Current Time Service Characteristic.
uint8 attrSize	The size of the Characteristic value attribute.
uint8 * attrValue	The pointer to the Characteristic value data that should be sent to the server device.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The Characteristic value was written successfully
- CYBLE_ERROR_GATT_DB_INVALID_ATTR_HANDLE - Optional Characteristic is absent
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameters failed

CyBle_CtssGetCharacteristicValue

Prototype

```
CYBLE_API_RESULT_T CyBle_CtssGetCharacteristicValue(CYBLE_CTS_CHAR_INDEX_T charIndex,
uint8 attrSize, uint8 * attrValue);
```

Description

Gets a Characteristic value of the Current Time Service, which is identified by charIndex.

Parameters

Parameters	Description
CYBLE_CTS_CHAR_INDEX_T charIndex	The index of a Current Time Service Characteristic.
uint8 attrSize	The size of the Current Time Service Characteristic value attribute.
uint8 * attrValue	The pointer to the location where Characteristic value data should be stored.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The Characteristic value was read successfully
- CYBLE_ERROR_GATT_DB_INVALID_ATTR_HANDLE - Optional Characteristic is absent
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameters failed

CyBle_CtssGetCharacteristicDescriptor

Prototype

```
CYBLE_API_RESULT_T CyBle_CtssGetCharacteristicDescriptor(CYBLE_CTS_CHAR_INDEX_T
charIndex, CYBLE_CTS_CHAR_DESCRIPTORS_T descrIndex, uint8 attrSize, uint8 *
attrValue);
```

Description

Gets a Characteristic Descriptor of a specified Characteristic of the Current Time Service.

Parameters

Parameters	Description
CYBLE_CTS_CHAR_INDEX_T charIndex	The index of the Characteristic.
CYBLE_CTS_CHAR_DESCRIPTORS_T descrIndex	The index of the Descriptor.
uint8 attrSize	The size of the Descriptor value.
uint8 * attrValue	The pointer to the location where Characteristic Descriptor value data should be stored.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The request handled successfully
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameter failed
- CYBLE_ERROR_GATT_DB_INVALID_ATTR_HANDLE - Optional Descriptor is absent

CyBle_CtssSendNotification

Prototype

```
CYBLE_API_RESULT_T CyBle_CtssSendNotification(CYBLE_CONN_HANDLE_T connHandle,
CYBLE_CTS_CHAR_INDEX_T charIndex, uint8 attrSize, uint8 * attrValue);
```

Description

Sends a notification to the Client device. A Characteristic value also gets written to the GATT database.

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	The connection handle.
CYBLE_CTS_CHAR_INDEX_T charIndex	The index of a service Characteristic to be send as a notification to the Client device.
uint8 attrSize	The size of the Characteristic value attribute.
uint8 * attrValue	The pointer to the Characteristic value data that should be sent to the Client device.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The Characteristic notification was sent successfully.
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameters failed.
- CYBLE_ERROR_INVALID_OPERATION - Operation is invalid for this Characteristic.
- CYBLE_ERROR_INVALID_STATE - Connection with the client is not established.
- CYBLE_ERROR_MEMORY_ALLOCATION_FAILED -Memory allocation failed.
- CYBLE_ERROR_NTF_DISABLED - Notification is not enabled by the client.

CTS Client Functions

APIs unique to CTS designs configured as a GATT Client role.

A letter 'c' is appended to the API name: CyBle_Ctsc

Functions

Function	Description
CyBle_CtscGetCharacteristicValue	Gets a Characteristic value of the Current Time Service, which is identified by charIndex.
CyBle_CtscSetCharacteristicDescriptor	Sets a Characteristic Descriptor of the Current Time Characteristic of the Current Time Service.
CyBle_CtscGetCharacteristicDescriptor	Gets a Characteristic Descriptor of the Current Time Characteristic of the Current Time Service.

CyBle_CtscGetCharacteristicValue

Prototype

```
CYBLE_API_RESULT_T CyBle_CtscGetCharacteristicValue(CYBLE_CONN_HANDLE_T connHandle,
CYBLE_CTS_CHAR_INDEX_T charIndex);
```

Description

Gets a Characteristic value of the Current Time Service, which is identified by charIndex.

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	The connection handle.
CYBLE_CTS_CHAR_INDEX_T charIndex	The index of a service Characteristic.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The request was sent successfully.
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameters failed.
- CYBLE_ERROR_MEMORY_ALLOCATION_FAILED -Memory allocation failed.
- CYBLE_ERROR_INVALID_STATE - Connection with the server is not established.
- CYBLE_ERROR_INVALID_OPERATION - Operation is invalid for this Characteristic.

- CYBLE_ERROR_GATT_DB_INVALID_ATTR_HANDLE - Peer device doesn't have a particular Characteristic.

CyBle_CtscSetCharacteristicDescriptor

Prototype

```
CYBLE_API_RESULT_T CyBle_CtscSetCharacteristicDescriptor(CYBLE_CONN_HANDLE_T
connHandle, CYBLE_CTS_CHAR_INDEX_T charIndex, CYBLE_CTS_CHAR_DESCRIPTORS_T descrIndex,
uint8 attrSize, uint8 * attrValue);
```

Description

Sets a Characteristic Descriptor of the Current Time Characteristic of the Current Time Service.

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	The connection handle.
CYBLE_CTS_CHAR_INDEX_T charIndex	The index of the Current Time Service Characteristic.
CYBLE_CTS_CHAR_DESCRIPTORS_T descrIndex	The index of the Current Time Service Characteristic Descriptor.
uint8 attrSize	The size of the Characteristic Descriptor attribute.
uint8 * attrValue	The pointer to the Characteristic Descriptor value data that should be sent to the server device.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The request was sent successfully.
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameters failed.
- CYBLE_ERROR_INVALID_STATE - Connection with the server is not established.
- CYBLE_ERROR_INVALID_OPERATION - This operation is not permitted on specified attribute.
- CYBLE_ERROR_MEMORY_ALLOCATION_FAILED -Memory allocation failed.
- CYBLE_ERROR_GATT_DB_INVALID_ATTR_HANDLE - Peer device doesn't have a particular Descriptor.

CyBle_CtscGetCharacteristicDescriptor

Prototype

```
CYBLE_API_RESULT_T CyBle_CtscGetCharacteristicDescriptor(CYBLE_CONN_HANDLE_T
connHandle, CYBLE_CTS_CHAR_INDEX_T charIndex, uint8 descrIndex);
```

Description

Gets a Characteristic Descriptor of the Current Time Characteristic of the Current Time Service.

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	The connection handle.
CYBLE_CTS_CHAR_INDEX_T charIndex	The index of the service Characteristic.
uint8 descrIndex	The index of a service Characteristic Descriptor.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The request was sent successfully.
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameters failed.
- CYBLE_ERROR_INVALID_STATE - State is not valid.
- CYBLE_ERROR_MEMORY_ALLOCATION_FAILED -Memory allocation failed.
- CYBLE_ERROR_INVALID_OPERATION - This operation is not permitted on specified attribute.
- CYBLE_ERROR_GATT_DB_INVALID_ATTR_HANDLE - Peer device doesn't have a particular Descriptor.

CTS Definitions and Data Structures

Contains the CTS specific definitions and data structures used in the CTS APIs.

Enumerations

Enumeration	Description
CYBLE_CTS_CHAR_INDEX_T	Service Characteristics indexes
CYBLE_CTS_CHAR_DESCRIPTORS_T	Service Characteristic Descriptors indexes

Structures

Structure	Description
CYBLE_CTS_CURRENT_TIME_T	Current Time Characteristic structure
CYBLE_CTS_LOCAL_TIME_INFO_T	Local Time Information Characteristic structure
CYBLE_CTS_REFERENCE_TIME_INFO_T	Reference Time Information Characteristic structure
CYBLE_CTS_CHAR_VALUE_T	Current Time Service Characteristic Value parameter structure
CYBLE_CTS_DESCR_VALUE_T	Current Time Service Characteristic Descriptor Value parameter structure
CYBLE_CTSC_T	Structure with discovered attributes information of Current Time Service
CYBLE_CTSS_T	Structure with Current Time Service attribute handles

CYBLE_CTS_CHAR_INDEX_T

Prototype

```
typedef enum {
  CYBLE_CTS_CURRENT_TIME,
  CYBLE_CTS_LOCAL_TIME_INFO,
  CYBLE_CTS_REFERENCE_TIME_INFO,
  CYBLE_CTS_CHAR_COUNT
} CYBLE_CTS_CHAR_INDEX_T;
```

Description

Service Characteristics indexes

Members

Members	Description
CYBLE_CTS_CURRENT_TIME	Current Time Characteristic index
CYBLE_CTS_LOCAL_TIME_INFO	Local Time Information Characteristic index
CYBLE_CTS_REFERENCE_TIME_INFO	Reference Time Information Characteristic index
CYBLE_CTS_CHAR_COUNT	Total count of Current Time Service Characteristics

CYBLE_CTS_CHAR_DESCRIPTORS_T

Prototype

```
typedef enum {
  CYBLE_CTS_CURRENT_TIME_CCCD,
  CYBLE_CTS_COUNT
} CYBLE_CTS_CHAR_DESCRIPTORS_T;
```

Description

Service Characteristic Descriptors indexes

Members

Members	Description
CYBLE_CTS_CURRENT_TIME_CCCD	Current Time Client Characteristic configuration Descriptor index
CYBLE_CTS_COUNT	Total count of Current Time Service Characteristic Descriptors

CYBLE_CTS_CURRENT_TIME_T

Prototype

```
typedef struct {
  uint8 yearLow;
  uint8 yearHigh;
  uint8 month;
  uint8 day;
  uint8 hours;
  uint8 minutes;
  uint8 seconds;
  uint8 dayOfWeek;
  uint8 fractions256;
  uint8 adjustReason;
} CYBLE_CTS_CURRENT_TIME_T;
```

Description

Current Time Characteristic structure

Members

Members	Description
uint8 yearLow;	LSB of current year
uint8 yearHigh;	MSB of current year
uint8 month;	Current month
uint8 day;	Current day
uint8 hours;	Current time - hours
uint8 minutes;	Current time - minutes
uint8 seconds;	Current time – seconds
uint8 dayOfWeek;	Current day of week
uint8 fractions256;	The value of 1/256th of second

uint8 adjustReason;	Reason of Current Time service Characteristics change

CYBLE_CTS_LOCAL_TIME_INFO_T

Prototype

```
typedef struct {
  int8 timeZone;
  uint8 dst;
} CYBLE_CTS_LOCAL_TIME_INFO_T;
```

Description

Local Time Information Characteristic structure

Members

Members	Description
int8 timeZone;	Current Time Zone
uint8 dst;	Daylight Saving Time value

CYBLE_CTS_REFERENCE_TIME_INFO_T

Prototype

```
typedef struct {
  uint8 timeSource;
  uint8 timeAccuracy;
  uint8 daysSinceUpdate;
  uint8 hoursSinseUpdate;
} CYBLE_CTS_REFERENCE_TIME_INFO_T;
```

Description

Reference Time Information Characteristic structure

Members

Members	Description
uint8 timeSource;	Time update source
uint8 timeAccuracy;	Time accuracy
uint8 daysSinceUpdate;	Days since last time update
uint8 hoursSinseUpdate;	Hours since last time update

CYBLE_CTS_CHAR_VALUE_T

Prototype

```
typedef struct {
  CYBLE_CONN_HANDLE_T connHandle;
  CYBLE_CTS_CHAR_INDEX_T charIndex;
  CYBLE_GATT_VALUE_T * value;
} CYBLE_CTS_CHAR_VALUE_T;
```

Description

Current Time Service Characteristic Value parameter structure

Members

Members	Description
CYBLE_CONN_HANDLE_T connHandle;	Connection handle
CYBLE_CTS_CHAR_INDEX_T charIndex;	Characteristic index of Current Time Service
CYBLE_GATT_VALUE_T * value;	Pointer to value of Current Time Service Characteristic

CYBLE_CTS_DESCR_VALUE_T

Prototype

```
typedef struct {
  CYBLE_CONN_HANDLE_T connHandle;
  CYBLE_CTS_CHAR_INDEX_T charIndex;
  CYBLE_CTS_CHAR_DESCRIPTORS_T descrIndex;
  CYBLE_GATT_VALUE_T * value;
} CYBLE_CTS_DESCR_VALUE_T;
```

Description

Current Time Service Characteristic Descriptor Value parameter structure

Members

Members	Description
CYBLE_CONN_HANDLE_T connHandle;	Connection handle
CYBLE_CTS_CHAR_INDEX_T charIndex;	Characteristic index of Current Time Service
CYBLE_CTS_CHAR_DESCRIPTORS_T descrIndex;	Characteristic index Descriptor of Current Time Service
CYBLE_GATT_VALUE_T * value;	Pointer to value of Current Time Service Characteristic

CYBLE_CTSC_T

Prototype

```
typedef struct {
  CYBLE_SRVR_CHAR_INFO_T currTimeCharacteristics[CYBLE_CTS_CHAR_COUNT];
  CYBLE_GATT_DB_ATTR_HANDLE_T currTimeCccdHandle;
} CYBLE_CTSC_T;
```

Description

Structure with discovered attributes information of Current Time Service

Members

Members	Description
CYBLE_SRVR_CHAR_INFO_T currTimeCharacteristics[CYBLE_CTS_CHAR_COUNT];	Structure with Characteristic handles + properties of Current Time Service
CYBLE_GATT_DB_ATTR_HANDLE_T currTimeCccdHandle;	Current Time Client Characteristic Configuration handle of Current Time Service

CYBLE_CTSS_T

Prototype

```
typedef struct {
  CYBLE_GATT_DB_ATTR_HANDLE_T serviceHandle;
  CYBLE_GATT_DB_ATTR_HANDLE_T currTimeCharHandle;
  CYBLE_GATT_DB_ATTR_HANDLE_T currTimeCccdHandle;
  CYBLE_GATT_DB_ATTR_HANDLE_T localTimeInfCharHandle;
  CYBLE_GATT_DB_ATTR_HANDLE_T refTimeInfCharHandle;
} CYBLE_CTSS_T;
```

Description

Structure with Current Time Service attribute handles

Members

Members	Description
CYBLE_GATT_DB_ATTR_HANDLE_T serviceHandle;	Current Time Service handle
CYBLE_GATT_DB_ATTR_HANDLE_T currTimeCharHandle;	Current Time Characteristic handle
CYBLE_GATT_DB_ATTR_HANDLE_T currTimeCccdHandle;	Current Time Client Characteristic Configuration Characteristic handle
CYBLE_GATT_DB_ATTR_HANDLE_T	Local Time Information Characteristic handle

localTimeInfCharHandle;	
CYBLE_GATT_DB_ATTR_HANDLE_T refTimeInfCharHandle;	Reference Time Information Characteristic handle

Cycling Power Service (CPS)

The Cycling Power Service (CPS) exposes power- and force-related data and optionally speed- and cadence-related data from a Cycling Power sensor (GATT Server) intended for sports and fitness applications.

Depending on the chosen GATT role in the GUI, you may use a subset of the supported APIs.

The CPS API names begin with CyBle_Cps. In addition to this, the APIs also append the GATT role initial letter in the API name.

CPS Server and Client Function

These are APIs common to both GATT Client role and GATT Server role. You may use them in either roles.

No letter is appended to the API name: CyBle_Cps

Functions

Function	Description
CyBle_CpsRegisterAttrCallback	Registers a callback function for service specific attribute operations.

CyBle_CpsRegisterAttrCallback

Prototype

```
void CyBle_CpsRegisterAttrCallback(CYBLE_CALLBACK_T callbackFunc);
```

Description

Registers a callback function for service specific attribute operations.

Parameters

Parameters	Description
CYBLE_CALLBACK_T callbackFunc	An application layer event callback function to receive events from the BLE Component. The definition of CYBLE_CALLBACK_T for CPS is, typedef void (* CYBLE_CALLBACK_T) (uint32 eventCode, void *eventParam) • eventCode indicates the event that triggered this callback. • eventParam contains the parameters corresponding to the current event.

Returns

None.

CPS Server Functions

APIs unique to CPS designs configured as a GATT Server role.

A letter 's' is appended to the API name: CyBle_Cpss

Functions

Function	Description
CyBle_CpssGetCharacteristicDescriptor	Gets a Characteristic Descriptor of a specified Characteristic of the service.
CyBle_CpssGetCharacteristicValue	Gets a Characteristic value of the service, which is a value identified by charIndex.
CyBle_CpssSendIndication	Sends indication with a Characteristic value of the CPS,which is a value specified by charIndex, to the Client device.
CyBle_CpssSendNotification	Sends notification with a Characteristic value of the CPS, which is a value specified by charIndex, to the Client device.
CyBle_CpssSetCharacteristicDescriptor	Sets a Characteristic Descriptor of a specified Characteristic of the service.
CyBle_CpssSetCharacteristicValue	Sets a Characteristic value of the service in the local database.
CyBle_CpssStartBroadcast	This function is used to start broadcasting of the Cycling Power Measurement Characteristic or update broadcasting data when it was started before. It is available... more
CyBle_CpssStopBroadcast	This function is used to stop broadcasting of the Cycling Power Measurement Characteristic.

CyBle_CpssStopBroadcast

Prototype

```
void CyBle_CpssStopBroadcast(void);
```

Description

This function is used to stop broadcasting of the Cycling Power Measurement Characteristic.

Returns

None

CyBle_CpssStartBroadcast

Prototype

```
CYBLE_API_RESULT_T CyBle_CpssStartBroadcast(uint16 advInterval, uint8 attrSize,
uint8 * attrValue);
```

Description

This function is used to start broadcasting of the Cycling Power Measurement Characteristic or update broadcasting data when it was started before. It is available only in Broadcaster role.

Parameters

Parameters	Description
uint16 advInterval	Advertising interval in 625 us units. The valid range is from CYBLE_GAP_ADV_ADVERT_INTERVAL_NONCON_MIN to CYBLE_GAP_ADV_ADVERT_INTERVAL_MAX.
uint8 attrSize	The size of the Characteristic value attribute. This size is limited by maximum advertising packet length and advertising header size.
uint8 * attrValue	The pointer to the Cycling Power Measurement Characteristic that include the mandatory fields (e.g. the Flags field and the Instantaneous Power field) and depending on the Flags field, some optional fields in a non connectable undirected advertising event.

Returns

CYBLE_API_RESULT_T : Return value indicates if the function succeeded or failed. Following are the possible error codes.

Error codes	Description
CYBLE_ERROR_OK	On successful operation.
CYBLE_ERROR_INVALID_PARAMETER	On passing an invalid parameter.

CyBle_CpssSetCharacteristicValue

Prototype

```
CYBLE_API_RESULT_T CyBle_CpssSetCharacteristicValue(CYBLE_CPS_CHAR_INDEX_T
charIndex, uint8 attrSize, uint8 * attrValue);
```

Description

Sets a Characteristic value of the service in the local database.

Parameters

Parameters	Description
CYBLE_CPS_CHAR_INDEX_T charIndex	The index of a service Characteristic of type CYBLE_CPS_CHAR_INDEX_T.
uint8 attrSize	The size of the Characteristic value attribute.
uint8 * attrValue	The pointer to the Characteristic value data that should be sent to the server device.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The request is handled successfully
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameter failed

CyBle_CpssSetCharacteristicDescriptor

Prototype

```
CYBLE_API_RESULT_T CyBle_CpssSetCharacteristicDescriptor(CYBLE_CPS_CHAR_INDEX_T
charIndex, CYBLE_CPS_DESCR_INDEX_T descrIndex, uint8 attrSize, uint8 * attrValue);
```

Description

Sets a Characteristic Descriptor of a specified Characteristic of the service.

Parameters

Parameters	Description
CYBLE_CPS_CHAR_INDEX_T charIndex	The index of a service Characteristic of type CYBLE_CPS_CHAR_INDEX_T.
CYBLE_CPS_DESCR_INDEX_T descrIndex	The index of a service Characteristic Descriptor of type CYBLE_CPS_DESCR_INDEX_T.
uint8 attrSize	The size of the Characteristic Descriptor attribute.
uint8 * attrValue	The pointer to the Descriptor value data that should be stored in the GATT database.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The request is handled successfully
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameter failed

CyBle_CpssSendNotification

Prototype

```
CYBLE_API_RESULT_T CyBle_CpssSendNotification(CYBLE_CONN_HANDLE_T connHandle,
CYBLE_CPS_CHAR_INDEX_T charIndex, uint8 attrSize, uint8 * attrValue);
```

Description

Sends notification with a Characteristic value of the CPS, which is a value specified by charIndex, to the Client device.

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	The connection handle
CYBLE_CPS_CHAR_INDEX_T charIndex	The index of a service Characteristic of type CYBLE_CPS_CHAR_INDEX_T.
uint8 attrSize	The size of the Characteristic value attribute.
uint8 * attrValue	The pointer to the Characteristic value data that should be sent to the Client device.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The request handled successfully
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameter failed
- CYBLE_ERROR_INVALID_OPERATION - This operation is not permitted
- CYBLE_ERROR_GATT_DB_INVALID_ATTR_HANDLE - Optional Characteristic is absent
- CYBLE_ERROR_INVALID_STATE - Connection with the client is not established
- CYBLE_ERROR_MEMORY_ALLOCATION_FAILED -Memory allocation failed.
- CYBLE_ERROR_NTF_DISABLED - Notification is not enabled by the client.

CyBle_CpssSendIndication

Prototype

```
CYBLE_API_RESULT_T CyBle_CpssSendIndication(CYBLE_CONN_HANDLE_T connHandle,
CYBLE_CPS_CHAR_INDEX_T charIndex, uint8 attrSize, uint8 * attrValue);
```

Description

Sends indication with a Characteristic value of the CPS,which is a value specified by charIndex, to the Client device.

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	The connection handle
CYBLE_CPS_CHAR_INDEX_T charIndex	The index of a service Characteristic of type CYBLE_CPS_CHAR_INDEX_T.
uint8 attrSize	The size of the Characteristic value attribute.
uint8 * attrValue	The pointer to the Characteristic value data that should be sent to the Client device.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The request handled successfully
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameter failed
- CYBLE_ERROR_INVALID_OPERATION - This operation is not permitted
- CYBLE_ERROR_GATT_DB_INVALID_ATTR_HANDLE - Optional Characteristic is absent
- CYBLE_ERROR_INVALID_STATE - Connection with the client is not established
- CYBLE_ERROR_MEMORY_ALLOCATION_FAILED -Memory allocation failed
- CYBLE_ERROR_IND_DISABLED - Indication is not enabled by the client

CyBle_CpssGetCharacteristicValue

Prototype

```
CYBLE_API_RESULT_T CyBle_CpssGetCharacteristicValue(CYBLE_CPS_CHAR_INDEX_T charIndex,
uint8 attrSize, uint8 * attrValue);
```

Description

Gets a Characteristic value of the service, which is a value identified by charIndex.

Parameters

Parameters	Description
CYBLE_CPS_CHAR_INDEX_T charIndex	The index of a service Characteristic of type CYBLE_CPS_CHAR_INDEX_T.
uint8 attrSize	The size of the Characteristic value attribute.
uint8 * attrValue	The pointer to the location where Characteristic value data should be stored.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The request is handled successfully
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameter failed

CyBle_CpssGetCharacteristicDescriptor

Prototype

```
CYBLE_API_RESULT_T CyBle_CpssGetCharacteristicDescriptor(CYBLE_CPS_CHAR_INDEX_T
charIndex, CYBLE_CPS_DESCR_INDEX_T descrIndex, uint8 attrSize, uint8 * attrValue);
```

Description

Gets a Characteristic Descriptor of a specified Characteristic of the service.

Parameters

Parameters	Description
CYBLE_CPS_CHAR_INDEX_T charIndex	The index of a service Characteristic of type CYBLE_CPS_CHAR_INDEX_T.
CYBLE_CPS_DESCR_INDEX_T descrIndex	The index of a service Characteristic Descriptor of type CYBLE_CPS_DESCR_INDEX_T.
uint8 attrSize	The size of the Characteristic Descriptor attribute.
uint8 * attrValue	The pointer to the location where Characteristic Descriptor value data should be stored.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The request handled successfully
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameter failed

CPS Client Functions

APIs unique to CPS designs configured as a GATT Client role.

A letter 'c' is appended to the API name: CyBle_Cpsc

Functions

Function	Description
CyBle_CpscSetCharacteristicValue	Sends a request to set a Characteristic value of the service, which is a value identified by charIndex, to the server device.
CyBle_CpscGetCharacteristicValue	This function is used to read a Characteristic value, which is a value identified by charIndex, from the server. The Read Response returns the Characteristic... more
CyBle_CpscGetCharacteristicDescriptor	Gets a Characteristic Descriptor of a specified Characteristic of the service.
CyBle_CpscSetCharacteristicDescriptor	This function is used to write the Characteristic Descriptor to the server which is identified by charIndex
CyBle_CpscStartObserve	This function is used for observing GAP peripheral devices. A device performing the observer role receives only advertisement data from devices irrespective of their discoverable... more
CyBle_CpscStopObserve	This function used to stop the discovery of devices. On stopping discovery operation, CYBLE_EVT_GAPC_SCAN_START_STOP event is generated. Application layer needs to keep track of the... more

CyBle_CpscStopObserve

Prototype

```
void CyBle_CpscStopObserve(void);
```

Description

This function used to stop the discovery of devices. On stopping discovery operation, CYBLE_EVT_GAPC_SCAN_START_STOP event is generated. Application layer needs to keep track of the function call made before receiving this event to associate this event with either the start or stop discovery function.

Possible events generated are:

- CYBLE_EVT_GAPC_SCAN_START_STOP

Returns

None

CyBle_CpscStartObserve

Prototype

```
CYBLE_API_RESULT_T CyBle_CpscStartObserve(void);
```

Description

This function is used for observing GAP peripheral devices. A device performing the observer role receives only advertisement data from devices irrespective of their discoverable mode settings. Advertisement data received is provided by the event, CYBLE_EVT_CPSC_SCAN_PROGRESS_RESULT. This procedure sets the scanType sub parameter to passive scanning.

If 'scanTo' sub-parameter is set to zero value, then passive scanning procedure will continue until you call CyBle_GapcStopObserve API. Possible generated events are:

- CYBLE_EVT_CPSC_SCAN_PROGRESS_RESULT

Returns

CYBLE_API_RESULT_T : Return value indicates if the function succeeded or failed. Following are the possible error codes.

Error codes	Description
CYBLE_ERROR_OK	On successful operation.
CYBLE_ERROR_STACK_INTERNAL	An error occurred in the BLE stack.

CyBle_CpscSetCharacteristicValue

Prototype

```
CYBLE_API_RESULT_T CyBle_CpscSetCharacteristicValue(CYBLE_CONN_HANDLE_T connHandle,
CYBLE_CPS_CHAR_INDEX_T charIndex, uint8 attrSize, uint8 * attrValue);
```

Description

Sends a request to set a Characteristic value of the service, which is a value identified by charIndex, to the server device.

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	The connection handle.
CYBLE_CPS_CHAR_INDEX_T charIndex	The index of a service Characteristic of type CYBLE_CPS_CHAR_INDEX_T.
uint8 attrSize	The size of the Characteristic value attribute.

uint8 * attrValue	The pointer to the Characteristic value data that should be send to the server device.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The request was sent successfully
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameters failed
- CYBLE_ERROR_MEMORY_ALLOCATION_FAILED -Memory allocation failed
- CYBLE_ERROR_INVALID_STATE - Connection with the server is not established
- CYBLE_ERROR_GATT_DB_INVALID_ATTR_HANDLE - The peer device doesn't have the particular Characteristic
- CYBLE_ERROR_INVALID_OPERATION - Operation is invalid for this Characteristic

CyBle_CpscSetCharacteristicDescriptor

Prototype

```
CYBLE_API_RESULT_T CyBle_CpscSetCharacteristicDescriptor(CYBLE_CONN_HANDLE_T
connHandle, CYBLE_CPS_CHAR_INDEX_T charIndex, CYBLE_CPS_DESCR_INDEX_T descrIndex, uint8
attrSize, uint8 * attrValue);
```

Description

This function is used to write the Characteristic Descriptor to the server which is identified by charIndex

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	The connection handle.
CYBLE_CPS_CHAR_INDEX_T charIndex	The index of a service Characteristic of type CYBLE_CPS_CHAR_INDEX_T.
CYBLE_CPS_DESCR_INDEX_T descrIndex	The index of a service Characteristic Descriptor of type CYBLE_CPS_DESCR_INDEX_T.
uint8 attrSize	The size of the Characteristic value attribute.
uint8 * attrValue	Pointer to the Characteristic Descriptor value data that should be sent to the server device.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The request was sent successfully
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameters failed
- CYBLE_ERROR_INVALID_STATE - The state is not valid
- CYBLE_ERROR_MEMORY_ALLOCATION_FAILED -Memory allocation failed
- CYBLE_ERROR_INVALID_OPERATION - This operation is not permitted on the specified attribute

CyBle_CpscGetCharacteristicValue

Prototype

```
CYBLE_API_RESULT_T CyBle_CpscGetCharacteristicValue(CYBLE_CONN_HANDLE_T connHandle,
CYBLE_CPS_CHAR_INDEX_T charIndex);
```

Description

This function is used to read a Characteristic value, which is a value identified by charIndex, from the server.

The Read Response returns the Characteristic Value in the Attribute Value parameter.

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	The connection handle.
CYBLE_CPS_CHAR_INDEX_T charIndex	The index of a service Characteristic of type CYBLE_CPS_CHAR_INDEX_T.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The read request was sent successfully
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameters failed
- CYBLE_ERROR_GATT_DB_INVALID_ATTR_HANDLE - The peer device doesn't have the particular Characteristic
- CYBLE_ERROR_MEMORY_ALLOCATION_FAILED -Memory allocation failed

- CYBLE_ERROR_INVALID_STATE - Connection with the server is not established
- CYBLE_ERROR_INVALID_OPERATION - Operation is invalid for this Characteristic

CyBle_CpscGetCharacteristicDescriptor

Prototype

```
CYBLE_API_RESULT_T CyBle_CpscGetCharacteristicDescriptor(CYBLE_CONN_HANDLE_T
connHandle, CYBLE_CPS_CHAR_INDEX_T charIndex, CYBLE_CPS_DESCR_INDEX_T descrIndex);
```

Description

Gets a Characteristic Descriptor of a specified Characteristic of the service.

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	The connection handle.
CYBLE_CPS_CHAR_INDEX_T charIndex	The index of a service Characteristic of type CYBLE_CPS_CHAR_INDEX_T.
CYBLE_CPS_DESCR_INDEX_T descrIndex	The index of a service Characteristic Descriptor of type CYBLE_CPS_DESCR_INDEX_T.

Returns

- CYBLE_ERROR_OK - The request was sent successfully
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameters failed
- CYBLE_ERROR_INVALID_STATE - The state is not valid
- CYBLE_ERROR_MEMORY_ALLOCATION_FAILED -Memory allocation failed
- CYBLE_ERROR_INVALID_OPERATION - This operation is not permitted on the specified attribute

CPS Definitions and Data Structures

Contains the CPS specific definitions and data structures used in the CPS APIs.

Enumerations

Enumeration	Description
CYBLE_CPS_CHAR_INDEX_T	Characteristic indexes

CYBLE_CPS_CP_OC_T	Op Codes of the Cycling Power Control Point Characteristic
CYBLE_CPS_CP_RC_T	Response Code of the Cycling Power Control Point Characteristic
CYBLE_CPS_DESCR_INDEX_T	Characteristic descriptors indexes
CYBLE_CPS_SL_VALUE_T	Sensor Location Characteristic value

Structures

Structure	Description
CYBLE_CPS_CHAR_VALUE_T	CYBLE_CPS_CLIENT
CYBLE_CPS_DESCR_VALUE_T	This is type CYBLE_CPS_DESCR_VALUE_T.
CYBLE_CPSC_CHAR_T	Characteristic with Descriptors
CYBLE_CPSC_T	Structure with discovered attributes information of Cycling Power Service
CYBLE_CPSS_CHAR_T	Characteristic with Descriptors
CYBLE_CPSS_T	Structure with Cycling Power Service attribute handles
CYBLE_CPS_CP_ADJUSTMENT_T	This is type CYBLE_CPS_CP_ADJUSTMENT_T.
CYBLE_CPS_DATE_TIME_T	This is type CYBLE_CPS_DATE_TIME_T.

CYBLE_CPS_SL_VALUE_T

Prototype

```
typedef enum {
  CYBLE_CPS_SL_OTHER,
  CYBLE_CPS_SL_TOP_OF_SHOE,
  CYBLE_CPS_SL_IN_SHOE,
  CYBLE_CPS_SL_HIP,
  CYBLE_CPS_SL_FRONT_WHEEL,
  CYBLE_CPS_SL_LEFT_CRANK,
  CYBLE_CPS_SL_RIGHT_CRANK,
  CYBLE_CPS_SL_LEFT_PEDAL,
  CYBLE_CPS_SL_RIGHT_PEDAL,
  CYBLE_CPS_SL_FRONT_HUB,
  CYBLE_CPS_SL_REAR_DROPOUT,
  CYBLE_CPS_SL_CHAINSTAY,
  CYBLE_CPS_SL_REAR_WHEEL,
  CYBLE_CPS_SL_REAR_HUB,
  CYBLE_CPS_SL_CHEST,
  CYBLE_CPS_SL_COUNT
} CYBLE_CPS_SL_VALUE_T;
```

Description

Sensor Location Characteristic value

CYBLE_CPS_DESCR_VALUE_T

Prototype

```
typedef struct {
  CYBLE_CONN_HANDLE_T connHandle;
  CYBLE_CPS_CHAR_INDEX_T charIndex;
  CYBLE_CPS_DESCR_INDEX_T descrIndex;
  CYBLE_GATT_VALUE_T * value;
} CYBLE_CPS_DESCR_VALUE_T;
```

Description

This is type CYBLE_CPS_DESCR_VALUE_T.

Members

Members	Description
CYBLE_CONN_HANDLE_T connHandle;	Peer device handle
CYBLE_CPS_CHAR_INDEX_T charIndex;	Index of service Characteristic
CYBLE_CPS_DESCR_INDEX_T descrIndex;	Index of Descriptor
CYBLE_GATT_VALUE_T * value;	Characteristic value

CYBLE_CPS_DESCR_INDEX_T

Prototype

```
typedef enum {
  CYBLE_CPS_CCCD,
  CYBLE_CPS_SCCD,
  CYBLE_CPS_DESCR_COUNT
} CYBLE_CPS_DESCR_INDEX_T;
```

Description

Characteristic descriptors indexes

Members

Members	Description
CYBLE_CPS_CCCD	Client Characteristic Configuration Descriptor index
CYBLE_CPS_SCCD	Handle of the Server Characteristic Configuration Descriptor
CYBLE_CPS_DESCR_COUNT	Total count of Descriptors

CYBLE_CPS_DATE_TIME_T

Prototype

```
typedef struct {
  uint16 year;
  uint8 month;
  uint8 day;
  uint8 hours;
  uint8 minutes;
  uint8 seconds;
} CYBLE_CPS_DATE_TIME_T;
```

Description

This is type CYBLE_CPS_DATE_TIME_T.

CYBLE_CPS_CP_RC_T

Prototype

```
typedef enum {
  CYBLE_CPS_CP_RC_SUCCESS = 1u,
  CYBLE_CPS_CP_RC_NOT_SUPPORTED,
  CYBLE_CPS_CP_RC_INVALID_PARAMETER,
  CYBLE_CPS_CP_RC_OPERATION_FAILED
} CYBLE_CPS_CP_RC_T;
```

Description

Response Code of the Cycling Power Control Point Characteristic

Members

Members	Description
CYBLE_CPS_CP_RC_SUCCESS = 1u	Response for successful operation.
CYBLE_CPS_CP_RC_NOT_SUPPORTED	Response if unsupported Op Code is received
CYBLE_CPS_CP_RC_INVALID_PARAMETER	Response if Parameter received does not meet the requirements of the service or is outside of the supported range of the Sensor
CYBLE_CPS_CP_RC_OPERATION_FAILED	Response if the requested procedure failed

CYBLE_CPS_CP_OC_T

Prototype

```
typedef enum {
  CYBLE_CPS_CP_OC_SCV = 1u,
  CYBLE_CPS_CP_OC_USL,
  CYBLE_CPS_CP_OC_RSSL,
    CYBLE_CPS_CP_OC_SCRL,
    CYBLE_CPS_CP_OC_RCRL,
    CYBLE_CPS_CP_OC_SCHL,
    CYBLE_CPS_CP_OC_RCHL,
    CYBLE_CPS_CP_OC_SCHW,
    CYBLE_CPS_CP_OC_RCHW,
    CYBLE_CPS_CP_OC_SSL,
    CYBLE_CPS_CP_OC_RSL,
    CYBLE_CPS_CP_OC_SOC,
    CYBLE_CPS_CP_OC_MCPMCC,
    CYBLE_CPS_CP_OC_RSR,
    CYBLE_CPS_CP_OC_RFCD,
    CYBLE_CPS_CP_OC_RC = 32u
  } CYBLE_CPS_CP_OC_T;
```

Description

Op Codes of the Cycling Power Control Point Characteristic

Members

Members	Description
CYBLE_CPS_CP_OC_SCV = 1u	Set Cumulative Value
CYBLE_CPS_CP_OC_USL	Update Sensor Location
CYBLE_CPS_CP_OC_RSSL	Request Supported Sensor Locations
CYBLE_CPS_CP_OC_SCRL	Set Crank Length
CYBLE_CPS_CP_OC_RCRL	Request Crank Length
CYBLE_CPS_CP_OC_SCHL	Set Chain Length
CYBLE_CPS_CP_OC_RCHL	Request Chain Length
CYBLE_CPS_CP_OC_SCHW	Set Chain Weight
CYBLE_CPS_CP_OC_RCHW	Request Chain Weight
CYBLE_CPS_CP_OC_SSL	Set Span Length
CYBLE_CPS_CP_OC_RSL	Request Span Length
CYBLE_CPS_CP_OC_SOC	Start Offset Compensation
CYBLE_CPS_CP_OC_MCPMCC	Mask Cycling Power Measurement Characteristic Content
CYBLE_CPS_CP_OC_RSR	Request Sampling Rate
CYBLE_CPS_CP_OC_RFCD	Request Factory Calibration Date
CYBLE_CPS_CP_OC_RC = 32u	Response Code

CYBLE_CPS_CP_ADJUSTMENT_T

Prototype

```
typedef struct {
  uint16 crankLength;
  uint16 chainLength;
  uint16 chainWeight;
  uint16 spanLength;
  CYBLE_CPS_DATE_TIME_T factoryCalibrationDate;
  uint8 samplingRate;
  int16 offsetCompensation;
} CYBLE_CPS_CP_ADJUSTMENT_T;
```

Description

This is type CYBLE_CPS_CP_ADJUSTMENT_T.

Members

Members	Description
uint16 crankLength;	In millimeters with a resolution of 1/2 millimeter
uint16 chainLength;	In millimeters with a resolution of 1 millimeter
uint16 chainWeight;	In grams with a resolution of 1 gram
uint16 spanLength;	In millimeters with a resolution of 1 millimeter
CYBLE_CPS_DATE_TIME_T factoryCalibrationDate;	Use the same format as the Date Time Characteristic
uint8 samplingRate;	In Hertz with a resolution of 1 Hertz
int16 offsetCompensation;	either the raw force in Newton or the raw torque in 1/32 Newton meter based on the server capabilities. 0xFFFF means Not Available"

CYBLE_CPS_CHAR_VALUE_T

Prototype

```
typedef struct {
  CYBLE_CONN_HANDLE_T connHandle;
  CYBLE_CPS_CHAR_INDEX_T charIndex;
  CYBLE_GATT_VALUE_T * value;
} CYBLE_CPS_CHAR_VALUE_T;
```

Description

CYBLE_CPS_CLIENT

Members

Members	Description
CYBLE_CONN_HANDLE_T connHandle;	Peer device handle
CYBLE_CPS_CHAR_INDEX_T charIndex;	Index of service Characteristic
CYBLE_GATT_VALUE_T * value;	Characteristic value

CYBLE_CPS_CHAR_INDEX_T

Prototype

```
typedef enum {
  CYBLE_CPS_POWER_MEASURE,
  CYBLE_CPS_POWER_FEATURE,
  CYBLE_CPS_SENSOR_LOCATION,
  CYBLE_CPS_POWER_VECTOR,
  CYBLE_CPS_POWER_CP,
  CYBLE_CPS_CHAR_COUNT
} CYBLE_CPS_CHAR_INDEX_T;
```

Description

Characteristic indexes

Members

Members	Description
CYBLE_CPS_POWER_MEASURE	Cycling Power Measurement Characteristic index
CYBLE_CPS_POWER_FEATURE	Cycling Power Feature Characteristic index
CYBLE_CPS_SENSOR_LOCATION	Sensor Location Characteristic index
CYBLE_CPS_POWER_VECTOR	Cycling Power Vector Characteristic index
CYBLE_CPS_POWER_CP	Cycling Power Control Point Characteristic index
CYBLE_CPS_CHAR_COUNT	Total count of CPS Characteristics

CYBLE_CPSS_T

Prototype

```
typedef struct {
  CYBLE_GATT_DB_ATTR_HANDLE_T serviceHandle;
  CYBLE_CPSS_CHAR_T charInfo[CYBLE_CPS_CHAR_COUNT]; } CYBLE_CPSS_T;
```

Description

Structure with Cycling Power Service attribute handles

Members

Members	Description
CYBLE_GATT_DB_ATTR_HANDLE_T serviceHandle;	Cycling Power Service handle
CYBLE_CPSS_CHAR_T charInfo[CYBLE_CPS_CHAR_COUNT];	Cycling Power Service Characteristic handles

CYBLE_CPSS_CHAR_T

Prototype

```
typedef struct {
  CYBLE_GATT_DB_ATTR_HANDLE_T charHandle;
  CYBLE_GATT_DB_ATTR_HANDLE_T descrHandle[CYBLE_CPS_DESCR_COUNT]; }
CYBLE_CPSS_CHAR_T;
```

Description

Characteristic with Descriptors

Members

Members	Description
CYBLE_GATT_DB_ATTR_HANDLE_T charHandle;	Handle of Characteristic value
CYBLE_GATT_DB_ATTR_HANDLE_T descrHandle[CYBLE_CPS_DESCR_COUNT];	Handle of Descriptor

CYBLE_CPSC_T

Prototype

```
typedef struct {
  CYBLE_CPSC_CHAR_T charInfo[CYBLE_CPS_CHAR_COUNT]; } CYBLE_CPSC_T;
```

Description

Structure with discovered attributes information of Cycling Power Service

Members

Members	Description
CYBLE_CPSC_CHAR_T charInfo[CYBLE_CPS_CHAR_COUNT];	Characteristics handles array

CYBLE_CPSC_CHAR_T

Prototype

```
typedef struct {
  CYBLE_GATT_DB_ATTR_HANDLE_T descrHandle[CYBLE_CPS_DESCR_COUNT];
  CYBLE_GATT_DB_ATTR_HANDLE_T valueHandle;
  CYBLE_GATT_DB_ATTR_HANDLE_T endHandle;
  uint8 properties;
} CYBLE_CPSC_CHAR_T;
```

Description

Characteristic with Descriptors

Members

Members	Description
CYBLE_GATT_DB_ATTR_HANDLE_T descrHandle[CYBLE_CPS_DESCR_COUNT];	Handles of Descriptors
CYBLE_GATT_DB_ATTR_HANDLE_T valueHandle;	Handle of Characteristic value
CYBLE_GATT_DB_ATTR_HANDLE_T endHandle;	End handle of Characteristic
uint8 properties;	Properties for value field

Cycling Speed and Cadence Service (CSCS)

The Cycling Speed and Cadence (CSC) Service exposes speed-related data and/or cadence-related data while using the Cycling Speed and Cadence sensor (Server).

Depending on the chosen GATT role in the GUI, you may use a subset of the supported APIs.

The CSCS API names begin with CyBle_Cscs. In addition to this, the APIs also append the GATT role initial letter in the API name.

CSCS Server and Client Function

These are APIs common to both GATT Client role and GATT Server role. You may use them in either roles.

No letter is appended to the API name: CyBle_Cscs

Functions

Function	Description
CyBle_CscsRegisterAttrCallback	Registers a callback function for Cycling Speed and Cadence Service specific attribute operations.

CyBle_CscsRegisterAttrCallback

Prototype

```
void CyBle_CscsRegisterAttrCallback(CYBLE_CALLBACK_T callbackFunc);
```

Description

Registers a callback function for Cycling Speed and Cadence Service specific attribute operations.

Parameters

Parameters	Description
CYBLE_CALLBACK_T callbackFunc	An application layer event callback function to receive events from the BLE Component. The definition of CYBLE_CALLBACK_T for CSCS is, typedef void (* CYBLE_CALLBACK_T) (uint32 eventCode, void *eventParam) eventCode indicates the event that triggered this callback. eventParam contains the parameters corresponding to the current event

Returns

None.

Side Effects

The *eventParams in the callback function should not be used by the application once the callback function execution is finished. Otherwise this data may become corrupted.

CSCS Server Functions

APIs unique to CSCS designs configured as a GATT Server role.

A letter 's' is appended to the API name: CyBle_Cscss

Functions

Function	Description
CyBle_CscssSetCharacteristicValue	Sets Characteristic value of the Cycling Speed and Cadence Service, which is identified by charIndex, to the local database.
CyBle_CscssGetCharacteristicValue	Gets a Characteristic value of the Cycling Speed and Cadence Service, which is identified by charIndex, from the GATT database.
CyBle_CscssGetCharacteristicDescriptor	Gets a Characteristic Descriptor of a specified Characteristic of the Cycling Speed and Cadence Service, from the GATT database.
CyBle_CscssSendNotification	Sends notification with a Characteristic value, which is specified by charIndex, of the Cycling Speed and Cadence Service to the Client device.

CyBle_CscssSendIndication	Sends indication with a Characteristic value, which is specified by charIndex, of the Cycling Speed and Cadence Service to the Client device.

CyBle_CscssSetCharacteristicValue

Prototype

```
CYBLE_API_RESULT_T CyBle_CscssSetCharacteristicValue(CYBLE_CSCS_CHAR_INDEX_T
charIndex, uint8 attrSize, uint8 * attrValue);
```

Description

Sets Characteristic value of the Cycling Speed and Cadence Service, which is identified by charIndex, to the local database.

Parameters

Parameters	Description
CYBLE_CSCS_CHAR_INDEX_T charIndex	The index of a service Characteristic of type CYBLE_CSCS_CHAR_INDEX_T. Valid values are, CYBLE_CSCS_CSC_FEATURE CYBLE_CSCS_SENSOR_LOCATION.
uint8 attrSize	The size of the Characteristic value attribute.
uint8 * attrValue	The pointer to the Characteristic value data that should be sent to the server device.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The request is handled successfully.
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameters failed.
- CYBLE_ERROR_GATT_DB_INVALID_ATTR_HANDLE - Peer device doesn't have a particular Characteristic.

CyBle_CscssSendNotification

Prototype

```
CYBLE_API_RESULT_T CyBle_CscssSendNotification(CYBLE_CONN_HANDLE_T connHandle,
CYBLE_CSCS_CHAR_INDEX_T charIndex, uint8 attrSize, uint8 * attrValue);
```

Description

Sends notification with a Characteristic value, which is specified by charIndex, of the Cycling Speed and Cadence Service to the Client device.

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	The connection handle.
CYBLE_CSCS_CHAR_INDEX_T charIndex	The index of a service Characteristic of type CYBLE_CSCS_CHAR_INDEX_T. Valid value is CYBLE_CSCS_CSC_MEASUREMENT.
uint8 attrSize	The size of the Characteristic value attribute.
uint8 * attrValue	The pointer to the Characteristic value data that should be sent to the Client device.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The request is handled successfully.
- CYBLE_ERROR_INVALID_PARAMETER - Validation of input parameter is failed.
- CYBLE_ERROR_INVALID_OPERATION - Operation is invalid for this Characteristic.
- CYBLE_ERROR_INVALID_STATE - Connection with the client is not established.
- CYBLE_ERROR_NTF_DISABLED - Notification is not enabled by the client.
- CYBLE_ERROR_MEMORY_ALLOCATION_FAILED -Memory allocation failed.

CyBle_CscssSendIndication

Prototype

```
CYBLE_API_RESULT_T CyBle_CscssSendIndication(CYBLE_CONN_HANDLE_T connHandle,
CYBLE_CSCS_CHAR_INDEX_T charIndex, uint8 attrSize, uint8 * attrValue);
```

Description

Sends indication with a Characteristic value, which is specified by charIndex, of the Cycling Speed and Cadence Service to the Client device.

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	The connection handle.
CYBLE_CSCS_CHAR_INDEX_T charIndex	The index of a service Characteristic of type CYBLE_CSCS_CHAR_INDEX_T.
uint8 attrSize	The size of the Characteristic value attribute.
uint8 * attrValue	The pointer to the Characteristic value data that should be sent to the Client device.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The request is handled successfully.
- CYBLE_ERROR_INVALID_PARAMETER - Validation of input parameter is failed.
- CYBLE_ERROR_INVALID_OPERATION - Operation is invalid for this Characteristic.
- CYBLE_ERROR_INVALID_STATE - Connection with the client is not established.
- CYBLE_ERROR_IND_DISABLED - Indication is not enabled by the client.
- CYBLE_ERROR_MEMORY_ALLOCATION_FAILED -Memory allocation failed.

CyBle_CscssGetCharacteristicValue

Prototype

```
CYBLE_API_RESULT_T CyBle_CscssGetCharacteristicValue(CYBLE_CSCS_CHAR_INDEX_T
charIndex, uint8 attrSize, uint8 * attrValue);
```

Description

Gets a Characteristic value of the Cycling Speed and Cadence Service, which is identified by charIndex, from the GATT database.

Parameters

Parameters	Description
CYBLE_CSCS_CHAR_INDEX_T charIndex	The index of a service Characteristic of type CYBLE_CSCS_CHAR_INDEX_T. Valid value is, CYBLE_CSCS_SC_CONTROL_POINT.
uint8 attrSize	The size of the Characteristic value attribute.

uint8 * attrValue	The pointer to the location where Characteristic value data should be stored.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The request is handled successfully.
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameter failed.
- CYBLE_ERROR_GATT_DB_INVALID_ATTR_HANDLE - Optional Characteristic is absent.

CyBle_CscssGetCharacteristicDescriptor

Prototype

```
CYBLE_API_RESULT_T CyBle_CscssGetCharacteristicDescriptor(CYBLE_CSCS_CHAR_INDEX_T
charIndex, CYBLE_CSCS_DESCR_INDEX_T descrIndex, uint8 attrSize, uint8 * attrValue);
```

Description

Gets a Characteristic Descriptor of a specified Characteristic of the Cycling Speed and Cadence Service, from the GATT database.

Parameters

Parameters	Description
CYBLE_CSCS_CHAR_INDEX_T charIndex	The index of a service Characteristic of type CYBLE_CSCS_CHAR_INDEX_T. Valid values are, CYBLE_CSCS_CSC_MEASUREMENT CYBLE_CSCS_SC_CONTROL_POINT.
CYBLE_CSCS_DESCR_INDEX_T descrIndex	The index of a service Characteristic Descriptor of type CYBLE_CSCS_DESCR_INDEX_T. Valid value is CYBLE_CSCS_CCCD.
uint8 attrSize	The size of the Characteristic Descriptor attribute.
uint8 * attrValue	The pointer to the location where Characteristic Descriptor value data should be stored.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The request is handled successfully.
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameter failed.

- CYBLE_ERROR_GATT_DB_INVALID_ATTR_HANDLE - Peer device doesn't have a particular Characteristic.

CSCS Client Functions

APIs unique to CSCS designs configured as a GATT Client role.

A letter 'c' is appended to the API name: CyBle_Cscsc

Functions

Function	Description
CyBle_CscscGetCharacteristicValue	Sends a request to peer device to get Characteristic value of the Cycling Speed and Cadence Service, which is identified by charIndex.
CyBle_CscscSetCharacteristicValue	Sends a request to peer device to get Characteristic Descriptor of specified Characteristic of the Cycling Speed and Cadence Service.
CyBle_CscscGetCharacteristicDescriptor	Sends a request to peer device to get Characteristic Descriptor of specified Characteristic of the Cycling Speed and Cadence Service.
CyBle_CscscSetCharacteristicDescriptor	Sends a request to peer device to get Characteristic Descriptor of specified Characteristic of the Cycling Speed and Cadence Service.

CyBle_CscscSetCharacteristicValue

Prototype

```
CYBLE_API_RESULT_T CyBle_CscscSetCharacteristicValue(CYBLE_CONN_HANDLE_T connHandle,
CYBLE_CSCS_CHAR_INDEX_T charIndex, uint8 attrSize, uint8 * attrValue);
```

Description

Sends a request to peer device to get Characteristic Descriptor of specified Characteristic of the Cycling Speed and Cadence Service.

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	The connection handle.
CYBLE_CSCS_CHAR_INDEX_T charIndex	The index of a service Characteristic.
uint8 attrSize	size of the Characteristic value attribute.
uint8 * attrValue	Pointer to the Characteristic value data that should be sent to the server device.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The request was sent successfully;
- CYBLE_ERROR_INVALID_STATE - Connection with the client is not established.
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameters failed.
- CYBLE_ERROR_MEMORY_ALLOCATION_FAILED -Memory allocation failed.
- CYBLE_ERROR_INVALID_OPERATION - Operation is invalid for this. Characteristic.
- CYBLE_ERROR_GATT_DB_INVALID_ATTR_HANDLE - Peer device doesn't have a particular Characteristic.

CyBle_CscscSetCharacteristicDescriptor

Prototype

```
CYBLE_API_RESULT_T CyBle_CscscSetCharacteristicDescriptor(CYBLE_CONN_HANDLE_T
connHandle, CYBLE_CSCS_CHAR_INDEX_T charIndex, CYBLE_CSCS_DESCR_INDEX_T descrIndex,
uint8 attrSize, uint8 * attrValue);
```

Description

Sends a request to peer device to get Characteristic Descriptor of specified Characteristic of the Cycling Speed and Cadence Service.

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	The connection handle.
CYBLE_CSCS_CHAR_INDEX_T charIndex	The index of a CSCS Characteristic.
CYBLE_CSCS_DESCR_INDEX_T descrIndex	The index of a CSCS Characteristic Descriptor.
uint8 attrSize	The size of the Characteristic Descriptor attribute.
uint8 * attrValue	The pointer to the Characteristic Descriptor value data that should be sent to the server device.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - the request was sent successfully.

- CYBLE_ERROR_INVALID_STATE - connection with the client is not established.
- CYBLE_ERROR_INVALID_PARAMETER - validation of the input parameters failed.
- CYBLE_ERROR_MEMORY_ALLOCATION_FAILED -Memory allocation failed.
- CYBLE_ERROR_INVALID_OPERATION - Operation is invalid for this Characteristic.
- CYBLE_ERROR_GATT_DB_INVALID_ATTR_HANDLE - Peer device doesn't have a particular Descriptor.

CyBle_CscscGetCharacteristicValue

Prototype

```
CYBLE_API_RESULT_T CyBle_CscscGetCharacteristicValue(CYBLE_CONN_HANDLE_T connHandle,
CYBLE_CSCS_CHAR_INDEX_T charIndex);
```

Description

Sends a request to peer device to get Characteristic value of the Cycling Speed and Cadence Service, which is identified by charIndex.

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	The connection handle.
CYBLE_CSCS_CHAR_INDEX_T charIndex	The index of a service Characteristic.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The request was sent successfully;
- CYBLE_ERROR_INVALID_STATE - Connection with the client is not established.
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameters failed.
- CYBLE_ERROR_MEMORY_ALLOCATION_FAILED -Memory allocation failed.
- CYBLE_ERROR_INVALID_OPERATION - Operation is invalid for this Characteristic.
- CYBLE_ERROR_GATT_DB_INVALID_ATTR_HANDLE - Peer device doesn't have a particular Characteristic.

CyBle_CscscGetCharacteristicDescriptor

Prototype

```
CYBLE_API_RESULT_T CyBle_CscscGetCharacteristicDescriptor(CYBLE_CONN_HANDLE_T
connHandle, CYBLE_CSCS_CHAR_INDEX_T charIndex, CYBLE_CSCS_DESCR_INDEX_T descrIndex);
```

Description

Sends a request to peer device to get Characteristic Descriptor of specified Characteristic of the Cycling Speed and Cadence Service.

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	The connection handle.
CYBLE_CSCS_CHAR_INDEX_T charIndex	The index of a Service Characteristic.
CYBLE_CSCS_DESCR_INDEX_T descrIndex	The index of a Service Characteristic descriptor.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The request was sent successfully.
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameters failed.
- CYBLE_ERROR_MEMORY_ALLOCATION_FAILED -Memory allocation failed.
- CYBLE_ERROR_INVALID_STATE - Connection with the client is not established.
- CYBLE_ERROR_INVALID_OPERATION - Cannot process a request to send PDU due to invalid operation performed by the application.
- CYBLE_ERROR_GATT_DB_INVALID_ATTR_HANDLE - Peer device doesn't have a particular Descriptor.

CSCS Definitions and Data Structures

Contains the CSCS specific definitions and data structures used in the CSCS APIs.

Enumerations

Enumeration	Description
CYBLE_CSCS_CHAR_INDEX_T	Characteristic indexes
CYBLE_CSCS_DESCR_INDEX_T	Characteristic descriptors indexes

Structures

Structure	Description
CYBLE_CSCS_CHAR_VALUE_T	Cycling Speed and Cadence Service Characteristic Value parameter structure
CYBLE_CSCS_DESCR_VALUE_T	Cycling Speed and Cadence Service Characteristic Descriptor Value parameter structure
CYBLE_CSCSC_SRVR_FULL_CHAR_INFO_T	Service full Characteristic information type
CYBLE_CSCSC_T	Structure with discovered attributes information of Cycling Speed and Cadence Service
CYBLE_CSCSS_CHAR_T	Characteristic with Descriptors type
CYBLE_CSCSS_T	Structure with Cycling Speed and Cadence Service attribute handles

CYBLE_CSCSC_T

Prototype

```
typedef struct {
  CYBLE_CSCSC_SRVR_FULL_CHAR_INFO_T Characteristics[CYBLE_CSCS_CHAR_COUNT]; }
CYBLE_CSCSC_T;
```

Description

Structure with discovered attributes information of Cycling Speed and Cadence Service

Members

Members	Description
CYBLE_CSCSC_SRVR_FULL_CHAR_INFO_T Characteristics[CYBLE_CSCS_CHAR_COUNT];	Characteristics handles array

CYBLE_CSCSC_SRVR_FULL_CHAR_INFO_T

Prototype

```
typedef struct {
  CYBLE_SRVR_CHAR_INFO_T charInfo;
  CYBLE_GATT_DB_ATTR_HANDLE_T descriptors[CYBLE_CSCS_DESCR_COUNT];
  CYBLE_GATT_DB_ATTR_HANDLE_T endHandle;
} CYBLE_CSCSC_SRVR_FULL_CHAR_INFO_T;
```

Description

Service full Characteristic information type

Members

Members	Description
CYBLE_SRVR_CHAR_INFO_T charInfo;	Characteristic handle and properties
CYBLE_GATT_DB_ATTR_HANDLE_T Descriptors[CYBLE_CSCS_DESCR_COUNT];	Characteristic Descriptors handles
CYBLE_GATT_DB_ATTR_HANDLE_T endHandle;	End handle of Characteristic

CYBLE_CSCS_DESCR_VALUE_T

Prototype

```
typedef struct {
  CYBLE_CONN_HANDLE_T connHandle;
  CYBLE_CSCS_CHAR_INDEX_T charIndex;
  CYBLE_CSCS_DESCR_INDEX_T descrIndex;
  CYBLE_GATT_VALUE_T * value;
} CYBLE_CSCS_DESCR_VALUE_T;
```

Description

Cycling Speed and Cadence Service Characteristic Descriptor Value parameter structure

Members

Members	Description
CYBLE_CONN_HANDLE_T connHandle;	Connection handle
CYBLE_CSCS_CHAR_INDEX_T charIndex;	Characteristic index of the Service
CYBLE_CSCS_DESCR_INDEX_T descrIndex;	Characteristic Descriptor index
CYBLE_GATT_VALUE_T * value;	Pointer to value of the Service Characteristic Descriptor

CYBLE_CSCS_DESCR_INDEX_T

Prototype

```
typedef enum {
  CYBLE_CSCS_CCCD,
  CYBLE_CSCS_DESCR_COUNT
} CYBLE_CSCS_DESCR_INDEX_T;
```

Description

Characteristic Descriptors indexes

Members

Members	Description
CYBLE_CSCS_CCCD	Client Characteristic Configuration Descriptor index
CYBLE_CSCS_DESCR_COUNT	Total count of Descriptors

CYBLE_CSCS_CHAR_VALUE_T

Prototype

```
typedef struct {
  CYBLE_CONN_HANDLE_T connHandle;
  CYBLE_CSCS_CHAR_INDEX_T charIndex;
  CYBLE_GATT_VALUE_T * value;
} CYBLE_CSCS_CHAR_VALUE_T;
```

Description

Cycling Speed and Cadence Service Characteristic Value parameter structure

Members

Members	Description
CYBLE_CONN_HANDLE_T connHandle;	Peer device handle
CYBLE_CSCS_CHAR_INDEX_T charIndex;	Index of Cycling Speed and Cadence Service Characteristic
CYBLE_GATT_VALUE_T * value;	Characteristic value

CYBLE_CSCS_CHAR_INDEX_T

Prototype

```
typedef enum {
  CYBLE_CSCS_CSC_MEASUREMENT,
  CYBLE_CSCS_CSC_FEATURE,
  CYBLE_CSCS_SENSOR_LOCATION,
  CYBLE_CSCS_SC_CONTROL_POINT,
  CYBLE_CSCS_CHAR_COUNT
} CYBLE_CSCS_CHAR_INDEX_T;
```

Description

Characteristic indexes

Members

Members	Description
CYBLE_CSCS_CSC_MEASUREMENT	CSC Measurement Characteristic index
CYBLE_CSCS_CSC_FEATURE	CSC Feature Characteristic index
CYBLE_CSCS_SENSOR_LOCATION	CSC Sensor Location Characteristic index
CYBLE_CSCS_SC_CONTROL_POINT	CSC SC Control Point Characteristic index
CYBLE_CSCS_CHAR_COUNT	Total count of CSCS Characteristics

CYBLE_CSCSS_T

Prototype

```
typedef struct {
  CYBLE_GATT_DB_ATTR_HANDLE_T serviceHandle;
  CYBLE_CSCSS_CHAR_T charInfo[CYBLE_CSCS_CHAR_COUNT];
} CYBLE_CSCSS_T;
```

Description

Structure with Cycling Speed and Cadence Service attribute handles

Members

Members	Description
CYBLE_GATT_DB_ATTR_HANDLE_T serviceHandle;	Cycling Speed and Cadence Service handle
CYBLE_CSCSS_CHAR_T charInfo[CYBLE_CSCS_CHAR_COUNT];	Array of Cycling Speed and Cadence Service Characteristics and Descriptors handles

CYBLE_CSCSS_CHAR_T

Prototype

```
typedef struct {
  CYBLE_GATT_DB_ATTR_HANDLE_T charHandle;
  CYBLE_GATT_DB_ATTR_HANDLE_T descrHandle[CYBLE_CSCS_DESCR_COUNT]; }
CYBLE_CSCSS_CHAR_T;
```

Description

Characteristic with Descriptors type

Members

Members	Description
CYBLE_GATT_DB_ATTR_HANDLE_T charHandle;	Handle of the Characteristic value
CYBLE_GATT_DB_ATTR_HANDLE_T descrHandle[CYBLE_CSCS_DESCR_COUNT];	Handles of the Descriptors

Device Information Service (DIS)

The Device Information Service exposes manufacturer and/or vendor information about a device.

Depending on the chosen GATT role in the GUI, you may use a subset of the supported APIs.

The DIS API names begin with CyBle_Dis. In addition to this, the APIs also append the GATT role initial letter in the API name.

DIS Server and Client Function

These are APIs common to both GATT Client role and GATT Server role. You may use them in either roles.

No letter is appended to the API name: CyBle_Dis

Functions

Function	Description
CyBle_DisRegisterAttrCallback	Registers a callback function for service specific attribute operations.

CyBle_DisRegisterAttrCallback

Prototype

```
void CyBle_DisRegisterAttrCallback(CYBLE_CALLBACK_T callbackFunc);
```

Description

Registers a callback function for service specific attribute operations.

Parameters

Parameters	Description
CYBLE_CALLBACK_T callbackFunc	An application layer event callback function to receive events from the BLE Component. The definition of CYBLE_CALLBACK_T for Device Information Service is, typedef void (* CYBLE_CALLBACK_T) (uint32 eventCode, void *eventParam) eventCode indicates the event that triggered this callback.

	eventParam contains the parameters corresponding to the current event.

Returns

None

DIS Server Functions

APIs unique to DIS designs configured as a GATT Server role.

A letter 's' is appended to the API name: CyBle_Diss

Functions

Function	Description
CyBle_DissSetCharacteristicValue	Sets a Characteristic value of the service, which is identified by charIndex, to the local database.
CyBle_DissGetCharacteristicValue	Gets a Characteristic value of the service, which is identified by charIndex, from the GATT database.

CyBle_DissSetCharacteristicValue

Prototype

```
CYBLE_API_RESULT_T CyBle_DissSetCharacteristicValue(CYBLE_DIS_CHAR_INDEX_T charIndex,
uint8 attrSize, uint8 * attrValue);
```

Description

Sets a Characteristic value of the service, which is identified by charIndex, to the local database.

Parameters

Parameters	Description
CYBLE_DIS_CHAR_INDEX_T charIndex	The index of a service Characteristic.
uint8 attrSize	The size of the Characteristic value attribute.
uint8 * attrValue	The pointer to the Characteristic value data that should be sent to the server device.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The request handled successfully

- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameter failed

CyBle_DissGetCharacteristicValue

Prototype

```
CYBLE_API_RESULT_T CyBle_DissGetCharacteristicValue(CYBLE_DIS_CHAR_INDEX_T charIndex,
uint8 attrSize, uint8 * attrValue);
```

Description

Gets a Characteristic value of the service, which is identified by charIndex, from the GATT database.

Parameters

Parameters	Description
CYBLE_DIS_CHAR_INDEX_T charIndex	The index of a service Characteristic.
uint8 attrSize	The size of the Characteristic value attribute.
uint8 * attrValue	The pointer to the location where Characteristic value data should be stored.

Returns

Return value is of type CYBLE_API_RESULT_T. Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The request handled successfully
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameter failed

DIS Client Functions

APIs unique to DIS designs configured as a GATT Client role.

A letter 'c' is appended to the API name: CyBle_Disc

Functions

Function	Description
CyBle_DiscGetCharacteristicValue	This function is used to read the Characteristic Value from a server which is identified by charIndex. The Read Response returns the Characteristic value in... more

CyBle_DiscGetCharacteristicValue

Prototype

```
CYBLE_API_RESULT_T CyBle_DiscGetCharacteristicValue(CYBLE_CONN_HANDLE_T connHandle,
CYBLE_DIS_CHAR_INDEX_T charIndex);
```

Description

This function is used to read the Characteristic Value from a server which is identified by charIndex.

The Read Response returns the Characteristic value in the Attribute Value parameter. The Read Response only contains the Characteristic value that is less than or equal to (MTU - 1) octets in length. If the Characteristic value is greater than (MTU - 1) octets in length, a Read Long Characteristic Value procedure may be used if the rest of the Characteristic value is required.

This function call can result in generation of the following events based on the response from the server device.

- CYBLE_EVT_DISC_READ_CHAR_RESPONSE
- CYBLE_EVT_GATTC_ERROR_RSP

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	The connection handle.
CYBLE_DIS_CHAR_INDEX_T charIndex	The index of the service Characteristic.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The read request was sent successfully
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameters failed
- CYBLE_ERROR_MEMORY_ALLOCATION_FAILED -Memory allocation failed
- CYBLE_ERROR_INVALID_OPERATION - Operation is invalid for this Characteristic

DIS Definitions and Data Structures

Contains the DIS specific definitions and data structures used in the DIS APIs.

Enumerations

Enumeration	Description
CYBLE_DIS_CHAR_INDEX_T	DIS Characteristic index

Structures

Structure	Description
CYBLE_DIS_CHAR_VALUE_T	Device Information Service Characteristic Value parameter structure
CYBLE_DISC_T	Structure with discovered attributes information of Device Information Service
CYBLE_DISS_T	Structure with Device Information Service attribute handles

CYBLE_DIS_CHAR_INDEX_T

Prototype

```
typedef enum {
  CYBLE_DIS_MANUFACTURER_NAME,
  CYBLE_DIS_MODEL_NUMBER,
  CYBLE_DIS_SERIAL_NUMBER,
  CYBLE_DIS_HARDWARE_REV,
  CYBLE_DIS_FIRMWARE_REV,
  CYBLE_DIS_SOFTWARE_REV,
  CYBLE_DIS_SYSTEM_ID,
  CYBLE_DIS_REG_CERT_DATA,
  CYBLE_DIS_PNP_ID,
  CYBLE_DIS_CHAR_COUNT
} CYBLE_DIS_CHAR_INDEX_T;
```

Description

DIS Characteristic index

Members

Members	Description
CYBLE_DIS_MANUFACTURER_NAME	Manufacturer Name String Characteristic index
CYBLE_DIS_MODEL_NUMBER	Model Number String Characteristic index
CYBLE_DIS_SERIAL_NUMBER	Serial Number String Characteristic index
CYBLE_DIS_HARDWARE_REV	Hardware Revision String Characteristic index
CYBLE_DIS_FIRMWARE_REV	Firmware Revision String Characteristic index
CYBLE_DIS_SOFTWARE_REV	Software Revision String Characteristic index
CYBLE_DIS_SYSTEM_ID	System ID Characteristic index

CYBLE_DIS_REG_CERT_DATA	IEEE 11073-20601 Characteristic index
CYBLE_DIS_PNP_ID	PnP ID Characteristic index
CYBLE_DIS_CHAR_COUNT	Total count of DIS Characteristics

CYBLE_DIS_CHAR_VALUE_T

Prototype

```
typedef struct {
  CYBLE_CONN_HANDLE_T connHandle;
  CYBLE_DIS_CHAR_INDEX_T charIndex;
  CYBLE_GATT_VALUE_T * value;
} CYBLE_DIS_CHAR_VALUE_T;
```

Description

Device Information Service Characteristic Value parameter structure

Members

Members	Description
CYBLE_CONN_HANDLE_T connHandle;	Peer device handle
CYBLE_DIS_CHAR_INDEX_T charIndex;	Index of service Characteristic
CYBLE_GATT_VALUE_T * value;	Characteristic value

CYBLE_DISC_T

Prototype

```
typedef struct {
  CYBLE_SRVR_CHAR_INFO_T charInfo[CYBLE_DIS_CHAR_COUNT]; } CYBLE_DISC_T;
```

Description

Structure with discovered attributes information of Device Information Service

Members

Members	Description
CYBLE_SRVR_CHAR_INFO_T charInfo[CYBLE_DIS_CHAR_COUNT];	Characteristics handle + properties array

CYBLE_DISS_T

Prototype

```
typedef struct {
  CYBLE_GATT_DB_ATTR_HANDLE_T serviceHandle;
  CYBLE_GATT_DB_ATTR_HANDLE_T charHandle[CYBLE_DIS_CHAR_COUNT]; } CYBLE_DISS_T;
```

Description

Structure with Device Information Service attribute handles

Members

Members	Description
CYBLE_GATT_DB_ATTR_HANDLE_T serviceHandle;	Device Information Service handle
CYBLE_GATT_DB_ATTR_HANDLE_T charHandle[CYBLE_DIS_CHAR_COUNT];	Device Information Service Characteristic handles

Glucose Service (GLS)

The Glucose Service exposes glucose and other data related to a personal glucose sensor for consumer healthcare applications and is not designed for clinical use.

Depending on the chosen GATT role in the GUI, you may use a subset of the supported APIs.

The GLS API names begin with CyBle_Gls. In addition to this, the APIs also append the GATT role initial letter in the API name.

GLS Server and Client Function

These are APIs common to both GATT Client role and GATT Server role. You may use them in either roles.

No letter is appended to the API name: CyBle_Gls

Functions

Function	Description
CyBle_GlsRegisterAttrCallback	Registers a callback function for service specific attribute operations.

CyBle_GlsRegisterAttrCallback

Prototype

```
void CyBle_GlsRegisterAttrCallback(CYBLE_CALLBACK_T callbackFunc);
```

Description

Registers a callback function for service specific attribute operations.

Parameters

Parameters	Description
CYBLE_CALLBACK_T callbackFunc	An application layer event callback function to receive events from the BLE Component. The definition of CYBLE_CALLBACK_T for Glucose Service is, typedef void (* CYBLE_CALLBACK_T) (uint32 eventCode, void *eventParam) eventCode indicates the event that triggered this callback. eventParam contains the parameters corresponding to the current event.

Returns

None

Side Effects

The *eventParams in the callback function should not be used by the application once the callback function execution is finished. Otherwise this data may become corrupted.

GLS Server Functions

APIs unique to GLS designs configured as a GATT Server role.

A letter 's' is appended to the API name: CyBle_Glss

Functions

Function	Description
CyBle_GlssSetCharacteristicValue	Sets a Characteristic value of the service, which is identified by charIndex.
CyBle_GlssGetCharacteristicValue	Gets a Characteristic value of the service, which is identified by charIndex.
CyBle_GlssGetCharacteristicDescriptor	Gets the Characteristic Descriptor of the specified Characteristic.
CyBle_GlssSendNotification	Sends a notification of the specified Characteristic to the client device, as defined by the charIndex value.
CyBle_GlssSendIndication	Sends a indication of the specified Characteristic to the client device, as defined by the charIndex value.

CyBle_GlssSetCharacteristicValue

Prototype

```
CYBLE_API_RESULT_T CyBle_GlssSetCharacteristicValue(CYBLE_GLS_CHAR_INDEX_T charIndex,
uint8 attrSize, uint8 * attrValue);
```

Description

Sets a Characteristic value of the service, which is identified by charIndex.

Parameters

Parameters	Description
CYBLE_GLS_CHAR_INDEX_T charIndex	The index of a service Characteristic.
uint8 attrSize	The size of the Characteristic value attribute.
uint8 * attrValue	Pointer to the Characteristic value data that should be sent to the server device.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The request handled successfully.
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameter failed.
- CYBLE_ERROR_GATT_DB_INVALID_ATTR_HANDLE - Optional Characteristic is absent

CyBle_GlssGetCharacteristicValue

Prototype

```
CYBLE_API_RESULT_T CyBle_GlssGetCharacteristicValue(CYBLE_GLS_CHAR_INDEX_T charIndex,
uint8 attrSize, uint8 * attrValue);
```

Description

Gets a Characteristic value of the service, which is identified by charIndex.

Parameters

Parameters	Description
CYBLE_GLS_CHAR_INDEX_T charIndex	The index of a service Characteristic.
uint8 attrSize	The size of the Characteristic value attribute.
uint8 * attrValue	Pointer to the location where Characteristic value data should be stored.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The request handled successfully
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameter failed
- CYBLE_ERROR_GATT_DB_INVALID_ATTR_HANDLE - Optional Characteristic is absent

CyBle_GlssGetCharacteristicDescriptor

Prototype

```
CYBLE_API_RESULT_T CyBle_GlssGetCharacteristicDescriptor(CYBLE_GLS_CHAR_INDEX_T
charIndex, CYBLE_GLS_DESCR_INDEX_T descrIndex, uint8 attrSize, uint8 * attrValue);
```

Description

Gets the Characteristic Descriptor of the specified Characteristic.

Parameters

Parameters	Description
CYBLE_GLS_CHAR_INDEX_T charIndex	The index of the Characteristic.
CYBLE_GLS_DESCR_INDEX_T descrIndex	The index of the Descriptor.
uint8 attrSize	The size of the Descriptor value attribute.
uint8 * attrValue	Pointer to the location where the Descriptor value data should be stored.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The request handled successfully
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameter failed
- CYBLE_ERROR_GATT_DB_INVALID_ATTR_HANDLE - Optional Descriptor is absent

CyBle_GlssSendNotification

Prototype

```
CYBLE_API_RESULT_T CyBle_GlssSendNotification(CYBLE_CONN_HANDLE_T connHandle,
CYBLE_GLS_CHAR_INDEX_T charIndex, uint8 attrSize, uint8 * attrValue);
```

Description

Sends a notification of the specified Characteristic to the client device, as defined by the charIndex value.

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	The connection handle which consist of the device ID and ATT connection ID.
CYBLE_GLS_CHAR_INDEX_T charIndex	The index of the service Characteristic.
uint8 attrSize	The size of the Characteristic value attribute.
uint8 * attrValue	Pointer to the Characteristic value data that should be sent to Client device.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The request handled successfully
- CYBLE_ERROR_INVALID_ PARAMETER - Validation of the input parameter failed
- CYBLE_ERROR_INVALID_OPERATION - Operation is invalid for this Characteristic
- CYBLE_ERROR_GATT_DB_INVALID_ATTR_HANDLE - Optional Characteristic is absent
- CYBLE_ERROR_INVALID_STATE - Connection with the client is not established
- CYBLE_ERROR_MEMORY_ALLOCATION_FAILED - Memory allocation failed
- CYBLE_ERROR_NTF_DISABLED - Notification is not enabled by the client

CyBle_GlssSendIndication

Prototype

```
CYBLE_API_RESULT_T CyBle_GlssSendIndication(CYBLE_CONN_HANDLE_T connHandle,
CYBLE_GLS_CHAR_INDEX_T charIndex, uint8 attrSize, uint8 * attrValue);
```

Description

Sends a indication of the specified Characteristic to the client device, as defined by the charIndex value.

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	The connection handle which consist of the device ID and ATT connection ID.
CYBLE_GLS_CHAR_INDEX_T charIndex	The index of the service Characteristic.
uint8 attrSize	The size of the Characteristic value attribute.
uint8 * attrValue	Pointer to the Characteristic value data that should be sent to Client device.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The request handled successfully
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameter failed
- CYBLE_ERROR_INVALID_OPERATION - Operation is invalid for this Characteristic
- CYBLE_ERROR_GATT_DB_INVALID_ATTR_HANDLE - Optional Characteristic is absent
- CYBLE_ERROR_INVALID_STATE - Connection with the client is not established
- CYBLE_ERROR_MEMORY_ALLOCATION_FAILED -Memory allocation failed
- CYBLE_ERROR_IND_DISABLED - Indication is not enabled by the client

GLS Client Functions

APIs unique to GLS designs configured as a GATT Client role.

A letter 'c' is appended to the API name: CyBle_Glsc

Functions

Function	Description
CyBle_GlscSetCharacteristicValue	This function is used to write the Characteristic (which is identified by charIndex) value attribute to the server. The Write Response just confirms the operation... more
CyBle_GlscGetCharacteristicValue	This function is used to read the Characteristic Value from a server which is identified by charIndex.
CyBle_GlscSetCharacteristicDescriptor	Sets the Characteristic Descriptor of the specified Characteristic.

CyBle_GlscGetCharacteristicDescriptor	Gets the Characteristic Descriptor of the specified Characteristic.

CyBle_GlscSetCharacteristicValue

Prototype

```
CYBLE_API_RESULT_T CyBle_GlscSetCharacteristicValue(CYBLE_CONN_HANDLE_T connHandle,
CYBLE_GLS_CHAR_INDEX_T charIndex, uint8 attrSize, uint8 * attrValue);
```

Description

This function is used to write the Characteristic (which is identified by charIndex) value attribute to the server.

The Write Response just confirms the operation success.

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	The connection handle.
CYBLE_GLS_CHAR_INDEX_T charIndex	The index of a service Characteristic.
uint8 attrSize	The size of the Characteristic value attribute.
uint8 * attrValue	The pointer to the Characteristic value data that should be sent to the server device.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The request was sent successfully
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameters failed
- CYBLE_ERROR_MEMORY_ALLOCATION_FAILED -Memory allocation failed
- CYBLE_ERROR_INVALID_STATE - Connection with the server is not established
- CYBLE_ERROR_GATT_DB_INVALID_ATTR_HANDLE - The peer device doesn't have the particular Characteristic
- CYBLE_ERROR_INVALID_OPERATION - Operation is invalid for this Characteristic

CyBle_GlscGetCharacteristicValue

Prototype

```
CYBLE_API_RESULT_T CyBle_GlscGetCharacteristicValue(CYBLE_CONN_HANDLE_T connHandle,
CYBLE_GLS_CHAR_INDEX_T charIndex);
```

Description

This function is used to read the Characteristic Value from a server which is identified by charIndex.

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	The connection handle.
CYBLE_GLS_CHAR_INDEX_T charIndex	The index of the service Characteristic.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The read request was sent successfully
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameters failed
- CYBLE_ERROR_GATT_DB_INVALID_ATTR_HANDLE - The peer device doesn't have the particular Characteristic
- CYBLE_ERROR_MEMORY_ALLOCATION_FAILED -Memory allocation failed
- CYBLE_ERROR_INVALID_STATE - Connection with the server is not established
- CYBLE_ERROR_INVALID_OPERATION - Operation is invalid for this Characteristic

CyBle_GlscSetCharacteristicDescriptor

Prototype

```
CYBLE_API_RESULT_T CyBle_GlscSetCharacteristicDescriptor(CYBLE_CONN_HANDLE_T
connHandle, CYBLE_GLS_CHAR_INDEX_T charIndex, CYBLE_GLS_DESCR_INDEX_T descrIndex, uint8
attrSize, uint8 * attrValue);
```

Description

Sets the Characteristic Descriptor of the specified Characteristic.

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	The connection handle.
CYBLE_GLS_CHAR_INDEX_T charIndex	The index of a service Characteristic.
CYBLE_GLS_DESCR_INDEX_T descrIndex	The index of a service Characteristic Descriptor.
uint8 attrSize	The size of the Characteristic Descriptor value attribute.
uint8 * attrValue	Pointer to the Characteristic Descriptor value data that should be sent to the server device.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The request was sent successfully
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameters failed
- CYBLE_ERROR_INVALID_STATE - The state is not valid
- CYBLE_ERROR_MEMORY_ALLOCATION_FAILED -Memory allocation failed
- CYBLE_ERROR_GATT_DB_INVALID_ATTR_HANDLE - The peer device doesn't have the particular Characteristic
- CYBLE_ERROR_INVALID_OPERATION - This operation is not permitted on the specified attribute

CyBle_GlscGetCharacteristicDescriptor

Prototype

```
CYBLE_API_RESULT_T CyBle_GlscGetCharacteristicDescriptor(CYBLE_CONN_HANDLE_T
connHandle, CYBLE_GLS_CHAR_INDEX_T charIndex, CYBLE_GLS_DESCR_INDEX_T descrIndex);
```

Description

Gets the Characteristic Descriptor of the specified Characteristic.

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	The connection handle.

CYBLE_GLS_CHAR_INDEX_T charIndex	The index of a service Characteristic.
CYBLE_GLS_DESCR_INDEX_T descrIndex	The index of the service Characteristic Descriptor.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The request was sent successfully
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameters failed
- CYBLE_ERROR_INVALID_STATE - The state is not valid
- CYBLE_ERROR_MEMORY_ALLOCATION_FAILED -Memory allocation failed
- CYBLE_ERROR_GATT_DB_INVALID_ATTR_HANDLE - The peer device doesn't have the particular Descriptor
- CYBLE_ERROR_INVALID_OPERATION - This operation is not permitted on the specified attribute

GLS Definitions and Data Structures

Contains the GLS specific definitions and data structures used in the GLS APIs.

Enumerations

Enumeration	Description
CYBLE_GLS_CHAR_INDEX_T	Service Characteristics indexes
CYBLE_GLS_DESCR_INDEX_T	Service Characteristic Descriptors indexes

Structures

Structure	Description
CYBLE_GLS_CHAR_VALUE_T	Glucose Service Characteristic value parameter structure
CYBLE_GLS_DESCR_VALUE_T	Glucose Service Characteristic Descriptor value parameter structure
CYBLE_GLSC_CHAR_T	Glucose Client Characteristic structure type
CYBLE_GLSC_T	Glucose Service structure type
CYBLE_GLSS_CHAR_T	Glucose Server Characteristic structure type
CYBLE_GLSS_T	Structure with Glucose Service attribute handles

CYBLE_GLS_CHAR_INDEX_T

Prototype

```
typedef enum {
  CYBLE_GLS_GLMT,
  CYBLE_GLS_GLMC,
  CYBLE_GLS_GLFT,
  CYBLE_GLS_RACP,
  CYBLE_GLS_CHAR_COUNT
} CYBLE_GLS_CHAR_INDEX_T;
```

Description

Service Characteristics indexes

Members

Members	Description
CYBLE_GLS_GLMT	Glucose Measurement Characteristic index
CYBLE_GLS_GLMC	Glucose Measurement Context Characteristic index
CYBLE_GLS_GLFT	Glucose Feature Characteristic index
CYBLE_GLS_RACP	Record Access Control Point Characteristic index
CYBLE_GLS_CHAR_COUNT	Total count of GLS Characteristics

CYBLE_GLS_CHAR_VALUE_T

Prototype

```
typedef struct {
  CYBLE_CONN_HANDLE_T connHandle;
  CYBLE_GLS_CHAR_INDEX_T charIndex;
  CYBLE_GATT_VALUE_T * value;
} CYBLE_GLS_CHAR_VALUE_T;
```

Description

Glucose Service Characteristic value parameter structure

Members

Members	Description
CYBLE_CONN_HANDLE_T connHandle;	Peer device handle
CYBLE_GLS_CHAR_INDEX_T charIndex;	Index of service Characteristic
CYBLE_GATT_VALUE_T * value;	Characteristic value

CYBLE_GLS_DESCR_INDEX_T

Prototype

```
typedef enum {
  CYBLE_GLS_CCCD,
  CYBLE_GLS_DESCR_COUNT
} CYBLE_GLS_DESCR_INDEX_T;
```

Description

Service Characteristic Descriptors indexes

Members

Members	Description
CYBLE_GLS_CCCD	Client Characteristic Configuration Descriptor index
CYBLE_GLS_DESCR_COUNT	Total count of GLS Descriptors

CYBLE_GLS_DESCR_VALUE_T

Prototype

```
typedef struct {
  CYBLE_CONN_HANDLE_T connHandle;
  CYBLE_GLS_CHAR_INDEX_T charIndex;
  CYBLE_GLS_DESCR_INDEX_T descrIndex;
  CYBLE_GATT_VALUE_T * value;
} CYBLE_GLS_DESCR_VALUE_T;
```

Description

Glucose Service Characteristic Descriptor value parameter structure

Members

Members	Description
CYBLE_CONN_HANDLE_T connHandle;	Peer device handle
CYBLE_GLS_CHAR_INDEX_T charIndex;	Index of service Characteristic
CYBLE_GLS_DESCR_INDEX_T descrIndex;	Index of service Characteristic Descriptor
CYBLE_GATT_VALUE_T * value;	Descriptor value

CYBLE_GLSC_CHAR_T

Prototype

```
typedef struct {
    uint8 properties;
    CYBLE_GATT_DB_ATTR_HANDLE_T valueHandle;
    CYBLE_GATT_DB_ATTR_HANDLE_T cccdHandle;
    CYBLE_GATT_DB_ATTR_HANDLE_T endHandle;
  } CYBLE_GLSC_CHAR_T;
```

Description

Glucose Client Characteristic structure type

Members

Members	Description
uint8 properties;	Properties for value field
CYBLE_GATT_DB_ATTR_HANDLE_T valueHandle;	Handle of server database attribute value entry
CYBLE_GATT_DB_ATTR_HANDLE_T cccdHandle;	Glucose client char. Descriptor handle
CYBLE_GATT_DB_ATTR_HANDLE_T endHandle;	Characteristic End Handle

CYBLE_GLSC_T

Prototype

```
  typedef struct {
    CYBLE_GLSC_CHAR_T charInfo[CYBLE_GLS_CHAR_COUNT]; } CYBLE_GLSC_T;
```

Description

Glucose Service structure type

Members

Members	Description
CYBLE_GLSC_CHAR_T charInfo[CYBLE_GLS_CHAR_COUNT];	Characteristics handle + properties array

CYBLE_GLSS_CHAR_T

Prototype

```
  typedef struct {
    CYBLE_GATT_DB_ATTR_HANDLE_T charHandle;
    CYBLE_GATT_DB_ATTR_HANDLE_T cccdHandle;
  } CYBLE_GLSS_CHAR_T;
```

Description

Glucose Server Characteristic structure type

Members

Members	Description
CYBLE_GATT_DB_ATTR_HANDLE_T charHandle;	Glucose Service char handle
CYBLE_GATT_DB_ATTR_HANDLE_T cccdHandle;	Glucose Service CCCD handle

CYBLE_GLSS_T

Prototype

```
typedef struct {
  CYBLE_GATT_DB_ATTR_HANDLE_T serviceHandle;
  CYBLE_GLSS_CHAR_T charInfo[CYBLE_GLS_CHAR_COUNT];
} CYBLE_GLSS_T;
```

Description

Structure with Glucose Service attribute handles

Members

Members	Description
CYBLE_GATT_DB_ATTR_HANDLE_T serviceHandle;	Glucose Service handle
CYBLE_GLSS_CHAR_T charInfo[CYBLE_GLS_CHAR_COUNT];	Glucose Service Characteristics info array

HID Service (HIDS)

The HID Service exposes data and associated formatting for HID Devices and HID Hosts. Depending on the chosen GATT role in the GUI, you may use a subset of the supported APIs. The HID API names begin with CyBle_Hid. In addition to this, the APIs also append the GATT role initial letter in the API name.

HIDS Server and Client Function

These are APIs common to both GATT Client role and GATT Server role. You may use them in either roles.

No letter is appended to the API name: CyBle_Hid

Functions

Function	Description
CyBle_HidsRegisterAttrCallback	Registers a callback function for service specific attribute operations.

CyBle_HidsRegisterAttrCallback

Prototype

```
void CyBle_HidsRegisterAttrCallback(CYBLE_CALLBACK_T callbackFunc);
```

Description

Registers a callback function for service specific attribute operations.

Parameters

Parameters	Description
CYBLE_CALLBACK_T callbackFunc	An application layer event callback function to receive events from the BLE Component. The definition of CYBLE_CALLBACK_T for HID Service is, typedef void (* CYBLE_CALLBACK_T) (uint32 eventCode, void *eventParam) eventCode indicates the event that triggered this callback (e.g. CYBLE_EVT_HIDS_NOTIFICATION_ENABLED). eventParam contains the parameters corresponding to the current event. (e.g. pointer to CYBLE_HIDS_CHAR_VALUE_T structure that contains details of the Characteristic for which notification enabled event was triggered).

Returns

None

Side Effects

The *eventParams in the callback function should not be used by the application once the callback function execution is finished. Otherwise this data may become corrupted.

HIDS Server Functions

APIs unique to HID designs configured as a GATT Server role.

A letter 's' is appended to the API name: CyBle_Hids

Functions

Function	Description
CyBle_HidssSetCharacteristicValue	Sets local Characteristic value of the specified HID Service Characteristics.
CyBle_HidssGetCharacteristicValue	Gets local Characteristic value of the specified HID Service Characteristics.
CyBle_HidssGetCharacteristicDescriptor	Gets local Characteristic Descriptor of the specified HID Service Characteristic.
CyBle_HidssSendNotification	Sends specified HID Service Characteristic notification to the Client device.

	CYBLE_EVT_HIDSC_NOTIFICATION event is received by the peer device, on invoking this function.

CyBle_HidssSetCharacteristicValue

Prototype

```
CYBLE_API_RESULT_T CyBle_HidssSetCharacteristicValue(uint8 serviceIndex,
CYBLE_HIDS_CHAR_INDEX_T charIndex, uint8 attrSize, uint8 * attrValue);
```

Description

Sets local Characteristic value of the specified HID Service Characteristics.

Parameters

Parameters	Description
uint8 serviceIndex	The index of the service instance. e.g. If two HID Services are supported in your design, then first service will be identified by serviceIndex of 0 and the second by serviceIndex of 1.
CYBLE_HIDS_CHAR_INDEX_T charIndex	The index of a service Characteristic. • CYBLE_HIDS_PROTOCOL_MODE - Protocol Mode Characteristic • CYBLE_HIDS_REPORT_MAP - Report Map Characteristic • CYBLE_HIDS_INFORMATION - HID Information Characteristic • CYBLE_HIDS_CONTROL_POINT - HID Control Point Characteristic • CYBLE_HIDS_BOOT_KYBRD_IN_REP - Boot Keyboard Input Report Characteristic • CYBLE_HIDS_BOOT_KYBRD_OUT_REP - Boot Keyboard Output Report Characteristic • CYBLE_HIDS_BOOT_MOUSE_IN_REP - Boot Mouse Input Report Characteristic • CYBLE_HIDS_REPORT - Report Characteristic
uint8 attrSize	The size of the Characteristic value attribute.
uint8 * attrValue	The pointer to the Characteristic value data that should be stored in the GATT database.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The request handled successfully.
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameter failed.
- CYBLE_ERROR_GATT_DB_INVALID_ATTR_HANDLE - Optional Characteristic is absent

CyBle_HidssGetCharacteristicValue

Prototype

```
CYBLE_API_RESULT_T CyBle_HidssGetCharacteristicValue(uint8 serviceIndex,
CYBLE_HIDS_CHAR_INDEX_T charIndex, uint8 attrSize, uint8 * attrValue);
```

Description

Gets local Characteristic value of the specified HID Service Characteristics.

Parameters

Parameters	Description
uint8 serviceIndex	The index of the service instance. e.g. If two HID Services are supported in your design, then first service will be identified by serviceIndex of 0 and the second by serviceIndex of 1.
CYBLE_HIDS_CHAR_INDEX_T charIndex	The index of the service Characteristic. • CYBLE_HIDS_PROTOCOL_MODE - Protocol Mode Characteristic • CYBLE_HIDS_REPORT_MAP - Report Map Characteristic • CYBLE_HIDS_INFORMATION - HID Information Characteristic • CYBLE_HIDS_CONTROL_POINT - HID Control Point Characteristic • CYBLE_HIDS_BOOT_KYBRD_IN_REP - Boot Keyboard Input Report Characteristic • CYBLE_HIDS_BOOT_KYBRD_OUT_REP - Boot Keyboard Output Report Characteristic • CYBLE_HIDS_BOOT_MOUSE_IN_REP - Boot Mouse Input Report Characteristic • CYBLE_HIDS_REPORT - Report Characteristic
uint8 attrSize	The size of the Characteristic value attribute.
uint8 * attrValue	The pointer to the location where Characteristic value data should be stored.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The request handled successfully
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameter failed
- CYBLE_ERROR_GATT_DB_INVALID_ATTR_HANDLE - Optional Characteristic is absent

CyBle_HidssGetCharacteristicDescriptor

Prototype

```
CYBLE_API_RESULT_T CyBle_HidssGetCharacteristicDescriptor(uint8 serviceIndex,
CYBLE_HIDS_CHAR_INDEX_T charIndex, CYBLE_HIDS_DESCR_T descrIndex, uint8 attrSize,
uint8 * attrValue);
```

Description

Gets local Characteristic Descriptor of the specified HID Service Characteristic.

Parameters

Parameters	Description
uint8 serviceIndex	The index of the service instance. e.g. If two HID Services are supported in your design, then first service will be identified by serviceIndex of 0 and the second by serviceIndex of 1.
CYBLE_HIDS_CHAR_INDEX_T charIndex	The index of the Characteristic. • CYBLE_HIDS_REPORT_MAP - Report Map Characteristic • CYBLE_HIDS_BOOT_KYBRD_IN_REP - Boot Keyboard Input Report Characteristic • CYBLE_HIDS_BOOT_KYBRD_OUT_REP - Boot Keyboard Output Report Characteristic • CYBLE_HIDS_BOOT_MOUSE_IN_REP - Boot Mouse Input Report Characteristic • CYBLE_HIDS_REPORT - Report Characteristic
CYBLE_HIDS_DESCR_T descrIndex	The index of the Descriptor. • CYBLE_HIDS_REPORT_CCCD - Client Characteristic Configuration Descriptor • CYBLE_HIDS_REPORT_RRD - Report Reference Descriptor • CYBLE_HIDS_REPORT_MAP_ERRD - Report Map External Report Reference Descriptor
uint8 attrSize	The size of the Descriptor value attribute.
uint8 * attrValue	The pointer to the location where Characteristic Descriptor value data should be stored.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The request handled successfully
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameter failed
- CYBLE_ERROR_GATT_DB_INVALID_ATTR_HANDLE - Optional Descriptor is absent

CyBle_HidssSendNotification

Prototype

```
CYBLE_API_RESULT_T CyBle_HidssSendNotification(CYBLE_CONN_HANDLE_T connHandle, uint8
serviceIndex, CYBLE_HIDS_CHAR_INDEX_T charIndex, uint8 attrSize, uint8 * attrValue);
```

Description

Sends specified HID Service Characteristic notification to the Client device.

CYBLE_EVT_HIDSC_NOTIFICATION event is received by the peer device, on invoking this function.

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	BLE peer device connection handle.
uint8 serviceIndex	The index of the HID service instance. e.g. If two HID Services are supported in your design, then first service will be identified by serviceIndex of 0 and the second by serviceIndex of 1.
CYBLE_HIDS_CHAR_INDEX_T charIndex	The index of the service Characteristic.
uint8 attrSize	The size of the Characteristic value attribute.
uint8 * attrValue	Pointer to the Characteristic value data that should be sent to the Client device.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The request handled successfully
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameter failed
- CYBLE_ERROR_INVALID_OPERATION - This operation is not permitted
- CYBLE_ERROR_GATT_DB_INVALID_ATTR_HANDLE - Optional Characteristic is absent
- CYBLE_ERROR_INVALID_STATE - Connection with the client is not established
- CYBLE_ERROR_MEMORY_ALLOCATION_FAILED -Memory allocation failed.
- CYBLE_ERROR_NTF_DISABLED - Notification is not enabled by the client.

HIDS Client Functions

APIs unique to HID designs configured as a GATT Client role.

A letter 'c' is appended to the API name: CyBle_Hidc

Functions

Function	Description
CyBle_HidscGetCharacteristicValue	This function is used to read the Characteristic value from a server which is identified by charIndex. The Read Response returns the Characteristic value in... more
CyBle_HidscSetCharacteristicValue	Sends a request to set Characteristic value of the specified HID Service, which is identified by serviceIndex and reportIndex, on the server device. This function... more
CyBle_HidscSetCharacteristicDescriptor	This function is used to write the Characteristic Descriptor to the server, which is identified by charIndex. This function call can result in generation of... more
CyBle_HidscGetCharacteristicDescriptor	Gets a Characteristic Descriptor of the specified Characteristic of the HID Service from the server device. This function call can result in generation of the... more

CyBle_HidscGetCharacteristicValue

Prototype

```
CYBLE_API_RESULT_T CyBle_HidscGetCharacteristicValue(CYBLE_CONN_HANDLE_T connHandle,
CYBLE_HIDSC_CHAR_READ_T subProcedure, uint8 serviceIndex, CYBLE_HIDS_CHAR_INDEX_T
charIndex);
```

Description

This function is used to read the Characteristic value from a server which is identified by charIndex.

The Read Response returns the Characteristic value in the Attribute Value parameter.

The Read Response only contains the Characteristic value that is less than or equal to (MTU - 1) octets in length. If the Characteristic value is greater than (MTU - 1) octets in length, the Read Long Characteristic Value procedure may be used if the rest of the Characteristic Value is required.

This function call can result in generation of the following events based on the response from the server device:

- CYBLE_EVT_HIDSC_READ_CHAR_RESPONSE
- CYBLE_EVT_GATTC_ERROR_RSP.

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	The connection handle.
CYBLE_HIDSC_CHAR_READ_T subProcedure	The Characteristic value read sub-procedure. CYBLE_HIDSC_READ_CHAR_VALUE CYBLE_HIDSC_READ_LONG_CHAR_VALUE.
uint8 serviceIndex	The index of the service instance.
CYBLE_HIDS_CHAR_INDEX_T charIndex	The index of the service Characteristic.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The read request was sent successfully
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameters failed
- CYBLE_ERROR_GATT_DB_INVALID_ATTR_HANDLE - The peer device doesn't have the particular Characteristic
- CYBLE_ERROR_MEMORY_ALLOCATION_FAILED -Memory allocation failed
- CYBLE_ERROR_INVALID_STATE - Connection with the server is not established
- CYBLE_ERROR_INVALID_OPERATION - Operation is invalid for this Characteristic

CyBle_HidscSetCharacteristicValue

Prototype

```
CYBLE_API_RESULT_T CyBle_HidscSetCharacteristicValue(CYBLE_CONN_HANDLE_T connHandle,
CYBLE_HIDSC_CHAR_WRITE_T subProcedure, uint8 serviceIndex, CYBLE_HIDS_CHAR_INDEX_T
charIndex, uint8 attrSize, uint8 * attrValue);
```

Description

Sends a request to set Characteristic value of the specified HID Service, which is identified by serviceIndex and reportIndex, on the server device. This function call can result in generation of the following events based on the response from the server device:

- CYBLE_EVT_HIDSC_WRITE_CHAR_RESPONSE
- CYBLE_EVT_GATTC_ERROR_RSP

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	The connection handle.
CYBLE_HIDSC_CHAR_WRITE_T subProcedure	Characteristic value write sub-procedure. CYBLE_HIDSC_WRITE_WITHOUT_RESPONSE CYBLE_HIDSC_WRITE_CHAR_VALUE
uint8 serviceIndex	The index of the service instance. e.g. If two HID Services are supported in your design, then first service will be identified by serviceIndex of 0 and the second by serviceIndex of 1.
CYBLE_HIDS_CHAR_INDEX_T charIndex	The index of a service Characteristic.
uint8 attrSize	The size of the Characteristic value attribute.
uint8 * attrValue	The pointer to the Characteristic value data that should be sent to the server device.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The request was sent successfully
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameters failed
- CYBLE_ERROR_MEMORY_ALLOCATION_FAILED -Memory allocation failed
- CYBLE_ERROR_GATT_DB_INVALID_ATTR_HANDLE - The peer device doesn't have the particular Characteristic
- CYBLE_ERROR_INVALID_OPERATION - Operation is invalid for this Characteristic

CyBle_HidscSetCharacteristicDescriptor

Prototype

```
CYBLE_API_RESULT_T CyBle_HidscSetCharacteristicDescriptor(CYBLE_CONN_HANDLE_T
connHandle, uint8 serviceIndex, CYBLE_HIDS_CHAR_INDEX_T charIndex, CYBLE_HIDS_DESCR_T
descrIndex, uint8 attrSize, uint8 * attrValue);
```

Description

This function is used to write the Characteristic Descriptor to the server, which is identified by charIndex. This function call can result in generation of the following events based on the response from the server device:

- CYBLE_EVT_HIDSC_WRITE_DESCR_RESPONSE
- CYBLE_EVT_GATTC_ERROR_RSP

Following event is received by the peer device, on invoking this function:

- CYBLE_EVT_HIDSS_NOTIFICATION_ENABLED
- CYBLE_EVT_HIDSS_NOTIFICATION_DISABLED

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	The BLE peer device connection handle.
uint8 serviceIndex	The index of the service instance. e.g. If two HID Services are supported in your design, then first service will be identified by serviceIndex of 0 and the second by serviceIndex of 1.
CYBLE_HIDS_CHAR_INDEX_T charIndex	The index of the HID service Characteristic.
CYBLE_HIDS_DESCR_T descrIndex	The index of the HID service Characteristic Descriptor.
uint8 attrSize	The size of the Characteristic value attribute.
uint8 * attrValue	The pointer to the Characteristic Descriptor value data that should be stored in the GATT database.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The request was sent successfully
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameters failed
- CYBLE_ERROR_INVALID_STATE - The state is not valid
- CYBLE_ERROR_MEMORY_ALLOCATION_FAILED -Memory allocation failed
- CYBLE_ERROR_GATT_DB_INVALID_ATTR_HANDLE - The peer device doesn't have the particular Characteristic
- CYBLE_ERROR_INVALID_OPERATION - This operation is not permitted on the specified attribute

CyBle_HidscGetCharacteristicDescriptor

Prototype

```
CYBLE_API_RESULT_T CyBle_HidscGetCharacteristicDescriptor(CYBLE_CONN_HANDLE_T
connHandle, uint8 serviceIndex, CYBLE_HIDS_CHAR_INDEX_T charIndex, CYBLE_HIDS_DESCR_T
descrIndex);
```

Description

Gets a Characteristic Descriptor of the specified Characteristic of the HID Service from the server device.

This function call can result in generation of the following events based on the response from the server device.

- CYBLE_EVT_HIDSC_READ_DESCR_RESPONSE
- CYBLE_EVT_GATTC_ERROR_RSP

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	The connection handle.
uint8 serviceIndex	The index of the service instance. e.g. If two HID Services are supported in your design, then first service will be identified by serviceIndex of 0 and the second by serviceIndex of 1.
CYBLE_HIDS_CHAR_INDEX_T charIndex	The index of the service Characteristic.
CYBLE_HIDS_DESCR_T descrIndex	The index of the HID Service Characteristic Descriptor.

Returns

- CYBLE_ERROR_OK - The request was sent successfully
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameters failed
- CYBLE_ERROR_INVALID_STATE - The state is not valid
- CYBLE_ERROR_MEMORY_ALLOCATION_FAILED -Memory allocation failed
- CYBLE_ERROR_GATT_DB_INVALID_ATTR_HANDLE - The peer device doesn't have the particular Descriptor
- CYBLE_ERROR_INVALID_OPERATION - This operation is not permitted on the specified attribute

HIDS Definitions and Data Structures

Contains the HID specific definitions and data structures used in the HID APIs.

Enumerations

Enumeration	Description
CYBLE_HIDS_CHAR_INDEX_T	This is type CYBLE_HIDS_CHAR_INDEX_T.
CYBLE_HIDS_DESCR_T	HID Service Characteristic Descriptors indexes
CYBLE_HIDSC_CHAR_READ_T	Characteristic Value Read Sub-Procedure supported by HID Service
CYBLE_HIDSC_CHAR_WRITE_T	Characteristic Value Write Sub-Procedure supported by HID Service

Structures

Structure	Description
CYBLE_HIDS_CHAR_VALUE_T	HID Service Characteristic value parameter structure
CYBLE_HIDS_DESCR_VALUE_T	HID Service Characteristic Descriptor value parameter structure
CYBLE_HIDSC_REPORT_MAP_T	HID client Report map Characteristic
CYBLE_HIDSC_REPORT_T	HID Client Report Characteristic
CYBLE_HIDSC_T	Structure with discovered attributes information of HID Service
CYBLE_HIDSS_INFORMATION_T	HID Information Characteristic value
CYBLE_HIDSS_REPORT_REF_T	HID server Report Reference Descriptor value - Report ID and Report Type
CYBLE_HIDSS_REPORT_T	HID Server Report Characteristic
CYBLE_HIDSS_T	Structure with HID Service attribute handles

CYBLE_HIDS_CHAR_INDEX_T

Prototype

```
typedef enum {
  CYBLE_HIDS_PROTOCOL_MODE,
  CYBLE_HIDS_INFORMATION,
  CYBLE_HIDS_CONTROL_POINT,
  CYBLE_HIDS_REPORT_MAP,
  CYBLE_HIDS_BOOT_KYBRD_IN_REP,
  CYBLE_HIDS_BOOT_KYBRD_OUT_REP,
  CYBLE_HIDS_BOOT_MOUSE_IN_REP,
  CYBLE_HIDS_REPORT,
  CYBLE_HIDS_CHAR_COUNT
} CYBLE_HIDS_CHAR_INDEX_T;
```

Description

This is type CYBLE_HIDS_CHAR_INDEX_T.

Members

Members	Description
CYBLE_HIDS_PROTOCOL_MODE	Protocol Mode Characteristic index
CYBLE_HIDS_INFORMATION	HID Information Characteristic index
CYBLE_HIDS_CONTROL_POINT	HID Control Point Characteristic index
CYBLE_HIDS_REPORT_MAP	Report Map Characteristic index
CYBLE_HIDS_BOOT_KYBRD_IN_REP	Boot Keyboard Input Report Characteristic index
CYBLE_HIDS_BOOT_KYBRD_OUT_REP	Boot Keyboard Output Report Characteristic index
CYBLE_HIDS_BOOT_MOUSE_IN_REP	Boot Mouse Input Report Characteristic index
CYBLE_HIDS_REPORT	Report Characteristic index
CYBLE_HIDS_CHAR_COUNT	Total count of Characteristics

CYBLE_HIDS_CHAR_VALUE_T

Prototype

```
typedef struct {
  CYBLE_CONN_HANDLE_T connHandle;
  uint8 serviceIndex;
  CYBLE_HIDS_CHAR_INDEX_T charIndex;
  CYBLE_GATT_VALUE_T * value;
} CYBLE_HIDS_CHAR_VALUE_T;
```

Description

HID Service Characteristic value parameter structure

Members

Members	Description
CYBLE_CONN_HANDLE_T connHandle;	Peer device handle
uint8 serviceIndex;	Index of HID Service
CYBLE_HIDS_CHAR_INDEX_T charIndex;	Index of HID Service Characteristic
CYBLE_GATT_VALUE_T * value;	Pointer to Characteristic value

CYBLE_HIDS_DESCR_T

Prototype

```
typedef enum {
  CYBLE_HIDS_REPORT_CCCD,
  CYBLE_HIDS_REPORT_RRD,
  CYBLE_HIDS_REPORT_MAP_ERRD,
  CYBLE_HIDS_DESCR_COUNT
} CYBLE_HIDS_DESCR_T;
```

Description

HID Service Characteristic Descriptors indexes

Members

Members	Description
CYBLE_HIDS_REPORT_CCCD	Client Characteristic Configuration Descriptor index
CYBLE_HIDS_REPORT_RRD	Report Reference Descriptor index
CYBLE_HIDS_REPORT_MAP_ERRD	Report Map External Report Reference Descriptor index
CYBLE_HIDS_DESCR_COUNT	Total count of Descriptors

CYBLE_HIDS_DESCR_VALUE_T

Prototype

```
typedef struct {
  CYBLE_CONN_HANDLE_T connHandle;
  uint8 serviceIndex;
  CYBLE_HIDS_CHAR_INDEX_T charIndex;
  CYBLE_HIDS_DESCR_T descrIndex;
  CYBLE_GATT_VALUE_T * value;
} CYBLE_HIDS_DESCR_VALUE_T;
```

Description

HID Service Characteristic Descriptor value parameter structure

Members

Members	Description
CYBLE_CONN_HANDLE_T connHandle;	Peer device handle
uint8 serviceIndex;	Index of HID Service
CYBLE_HIDS_CHAR_INDEX_T charIndex;	Index of HID Service Characteristic
CYBLE_HIDS_DESCR_T descrIndex;	Service Characteristic Descriptor index

CYBLE_GATT_VALUE_T * value;	Pointer to value of Service Characteristic Descriptor value

CYBLE_HIDSC_CHAR_READ_T

Prototype

```
typedef enum {
  CYBLE_HIDSC_READ_CHAR_VALUE,
  CYBLE_HIDSC_READ_LONG_CHAR_VALUE
} CYBLE_HIDSC_CHAR_READ_T;
```

Description

Characteristic Value Read Sub-Procedure supported by HID Service

Members

Members	Description
CYBLE_HIDSC_READ_CHAR_VALUE	Read Characteristic Value
CYBLE_HIDSC_READ_LONG_CHAR_VALUE	Read Long Characteristic Values

CYBLE_HIDSC_CHAR_WRITE_T

Prototype

```
typedef enum {
  CYBLE_HIDSC_WRITE_WITHOUT_RESPONSE,
  CYBLE_HIDSC_WRITE_CHAR_VALUE
} CYBLE_HIDSC_CHAR_WRITE_T;
```

Description

Characteristic Value Write Sub-Procedure supported by HID Service

Members

Members	Description
CYBLE_HIDSC_WRITE_WITHOUT_RESPONSE	Write Without Response
CYBLE_HIDSC_WRITE_CHAR_VALUE	Write Characteristic Value

CYBLE_HIDSC_REPORT_MAP_T

Prototype

```
typedef struct {
  CYBLE_GATT_DB_ATTR_HANDLE_T errdHandle;
  CYBLE_GATT_DB_ATTR_HANDLE_T valueHandle;
```

```
    CYBLE_GATT_DB_ATTR_HANDLE_T endHandle;
    uint8 properties;
  } CYBLE_HIDSC_REPORT_MAP_T;
```

Description

HID client Report map Characteristic

Members

Members	Description
CYBLE_GATT_DB_ATTR_HANDLE_T errdHandle;	Handle of Report Map External Report Reference Descriptor
CYBLE_GATT_DB_ATTR_HANDLE_T valueHandle;	Handle of Report Characteristic value
CYBLE_GATT_DB_ATTR_HANDLE_T endHandle;	End handle of Characteristic
uint8 properties;	Properties for value field

CYBLE_HIDSC_REPORT_T

Prototype

```
  typedef struct {
    CYBLE_GATT_DB_ATTR_HANDLE_T cccdHandle;
    CYBLE_GATT_DB_ATTR_HANDLE_T rrdHandle;
    CYBLE_GATT_DB_ATTR_HANDLE_T valueHandle;
    CYBLE_GATT_DB_ATTR_HANDLE_T endHandle;
    uint8 properties;
  } CYBLE_HIDSC_REPORT_T;
```

Description

HID Client Report Characteristic

Members

Members	Description
CYBLE_GATT_DB_ATTR_HANDLE_T cccdHandle;	Handle of Client Characteristic Configuration Descriptor
CYBLE_GATT_DB_ATTR_HANDLE_T rrdHandle;	Handle of Report Reference Descriptor
CYBLE_GATT_DB_ATTR_HANDLE_T valueHandle;	Handle of Report Characteristic value
CYBLE_GATT_DB_ATTR_HANDLE_T	End handle of Characteristic

endHandle;	
uint8 properties;	Properties for value field

CYBLE_HIDSC_T

Prototype

```
typedef struct {
  CYBLE_CONN_HANDLE_T connHandle;
  CYBLE_SRVR_CHAR_INFO_T protocolMode;
  CYBLE_HIDSC_REPORT_T bootReport[CYBLE_HIDS_BOOT_REPORT_COUNT];
  CYBLE_HIDSC_REPORT_MAP_T reportMap;
  CYBLE_SRVR_CHAR_INFO_T information;
  CYBLE_SRVR_CHAR_INFO_T controlPoint;
  CYBLE_HIDSC_REPORT_T report[CYBLE_HIDSC_REPORT_COUNT];
  uint8 reportCount;
  CYBLE_GATT_DB_ATTR_HANDLE_T includeHandle;
} CYBLE_HIDSC_T;
```

Description

Structure with discovered attributes information of HID Service

Members

Members	Description
CYBLE_CONN_HANDLE_T connHandle;	Peer device handle
CYBLE_SRVR_CHAR_INFO_T protocolMode;	Protocol Mode Characteristic handle and properties
CYBLE_HIDSC_REPORT_T bootReport[CYBLE_HIDS_BOOT_REPORT_COUNT];	Boot Report Characteristic info
CYBLE_HIDSC_REPORT_MAP_T reportMap;	Report Map Characteristic handle and Descriptors
CYBLE_SRVR_CHAR_INFO_T information;	Information Characteristic handle and properties
CYBLE_SRVR_CHAR_INFO_T controlPoint;	Control Point Characteristic handle and properties
CYBLE_HIDSC_REPORT_T report[CYBLE_HIDSC_REPORT_COUNT];	Report Characteristic info
uint8 reportCount;	Number of report Characteristics
CYBLE_GATT_DB_ATTR_HANDLE_T includeHandle;	Included declaration handle

CYBLE_HIDSS_INFORMATION_T

Prototype

```
typedef struct {
  uint16 bcdHID;
  uint8 bCountryCode;
  uint8 flags;
} CYBLE_HIDSS_INFORMATION_T;
```

Description

HID Information Characteristic value

Members

Members	Description
uint16 bcdHID;	Version number of HIDSe USB HID Specification implemented by HID Device
uint8 bCountryCode;	Identifies which country hardware is localized for
uint8 flags;	Bit 0: RemoteWake - Indicates whether HID Device is capable of sending wake-signal to HID Host. Bit 1: NormallyConnectable - Indicates whether HID Device will be advertising when bonded but not connected.

CYBLE_HIDSS_REPORT_REF_T

Prototype

```
typedef struct {
  uint8 reportId;
  uint8 reportType;
} CYBLE_HIDSS_REPORT_REF_T;
```

Description

HID server Report Reference Descriptor value - Report ID and Report Type

Members

Members	Description
uint8 reportId;	Non-zero value if there are more than one instance of the same Report Type
uint8 reportType;	Type of Report Characteristic

CYBLE_HIDSS_REPORT_T

Prototype

```
typedef struct {
  CYBLE_GATT_DB_ATTR_HANDLE_T reportHandle;
  CYBLE_GATT_DB_ATTR_HANDLE_T cccdHandle;
  CYBLE_GATT_DB_ATTR_HANDLE_T rrdHandle;
} CYBLE_HIDSS_REPORT_T;
```

Description

HID Server Report Characteristic

Members

Members	Description
CYBLE_GATT_DB_ATTR_HANDLE_T reportHandle;	Handle of Report Characteristic value
CYBLE_GATT_DB_ATTR_HANDLE_T cccdHandle;	Handle of Client Characteristic Configuration Descriptor
CYBLE_GATT_DB_ATTR_HANDLE_T rrdHandle;	Handle of Report Reference Descriptor

CYBLE_HIDSS_T

Prototype

```
typedef struct {
  CYBLE_GATT_DB_ATTR_HANDLE_T serviceHandle;
  CYBLE_GATT_DB_ATTR_HANDLE_T protocolModeHandle;
  uint8 reportCount;
  const CYBLE_HIDSS_REPORT_T * reportArray;
  CYBLE_HIDSS_REPORT_T bootReportArray[CYBLE_HIDS_BOOT_REPORT_COUNT];
  CYBLE_GATT_DB_ATTR_HANDLE_T reportMapHandle;
  CYBLE_GATT_DB_ATTR_HANDLE_T reportMapErrdHandle;
  CYBLE_GATT_DB_ATTR_HANDLE_T informationHandle;
  CYBLE_GATT_DB_ATTR_HANDLE_T controlPointHandle;
} CYBLE_HIDSS_T;
```

Description

Structure with HID Service attribute handles

Members

Members	Description
CYBLE_GATT_DB_ATTR_HANDLE_T serviceHandle;	Handle of HID service
CYBLE_GATT_DB_ATTR_HANDLE_T protocolModeHandle;	Handle of Protocol Mode Characteristic

uint8 reportCount;	Number of report Characteristics
const CYBLE_HIDSS_REPORT_T * reportArray;	Info about report Characteristics
CYBLE_HIDSS_REPORT_T bootReportArray[CYBLE_HIDS_BOOT_REPORT_COUNT];	Info about Boot Report Characteristics
CYBLE_GATT_DB_ATTR_HANDLE_T reportMapHandle;	Handle of Report Map Characteristic
CYBLE_GATT_DB_ATTR_HANDLE_T reportMapErrdHandle;	Handle of Report Map External Report Reference descr.
CYBLE_GATT_DB_ATTR_HANDLE_T informationHandle;	Handle of HID Information Characteristic
CYBLE_GATT_DB_ATTR_HANDLE_T controlPointHandle;	Handle of HID Control Point Characteristic

Heart Rate Service (HRS)

The Heart Rate Service exposes heart rate and other data related to a heart rate sensor intended for fitness applications.

Depending on the chosen GATT role in the GUI, you may use a subset of the supported APIs.

The HRS API names begin with CyBle_Hrs. In addition to this, the APIs also append the GATT role initial letter in the API name.

HRS Server and Client Function

These are APIs common to both GATT Client role and GATT Server role. You may use them in either roles.

No letter is appended to the API name: CyBle_Hrs

Functions

Function	Description
CyBle_HrsRegisterAttrCallback	Registers a callback function for service specific attribute operations.

CyBle_HrsRegisterAttrCallback

Prototype

```
void CyBle_HrsRegisterAttrCallback(CYBLE_CALLBACK_T callbackFunc);
```

Description

Registers a callback function for service specific attribute operations.

Parameters

Parameters	Description
CYBLE_CALLBACK_T callbackFunc	An application layer event callback function to receive events from the BLE Component. The definition of CYBLE_CALLBACK_T for HRS Service is, typedef void (* CYBLE_CALLBACK_T) (uint32 eventCode, void *eventParam) eventCode indicates the event that triggered this callback (e.g. CYBLE_EVT_HRSS_NOTIFICATION_ENABLED). eventParam contains the parameters corresponding to the current event. (e.g. pointer to CYBLE_HRS_CHAR_VALUE_T structure that contains details of the Characteristic for which notification enabled event was triggered).

Returns

None

HRS Server Functions

APIs unique to HRS designs configured as a GATT Server role.

A letter 's' is appended to the API name: CyBle_Hrss

Functions

Function	Description
CyBle_HrssSetCharacteristicValue	Sets local Characteristic value of the specified Heart Rate Service Characteristic.
CyBle_HrssGetCharacteristicValue	Gets the local Characteristic value of specified Heart Rate Service Characteristic.
CyBle_HrssGetCharacteristicDescriptor	Gets the local Characteristic Descriptor of the specified Heart Rate Service Characteristic.
CyBle_HrssSendNotification	Sends notification of a specified Heart Rate Service Characteristic value to the Client device. No response is expected. The CYBLE_EVT_HRSC_NOTIFICATION event is received by the... more

CyBle_HrssSetCharacteristicValue

Prototype

```
CYBLE_API_RESULT_T CyBle_HrssSetCharacteristicValue(CYBLE_HRS_CHAR_INDEX_T charIndex,
uint8 attrSize, uint8 * attrValue);
```

Description

Sets local Characteristic value of the specified Heart Rate Service Characteristic.

Parameters

Parameters	Description
CYBLE_HRS_CHAR_INDEX_T charIndex	The index of a service Characteristic.
uint8 attrSize	The size of the Characteristic value attribute. The Heart Rate Measurement Characteristic has a 20 byte length (by default). The Body Sensor Location and Control Point Characteristic both have 1 byte length.
uint8 * attrValue	The pointer to the Characteristic value data that should be stored in the GATT database.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The request handled successfully.
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameter failed.
- CYBLE_ERROR_GATT_DB_INVALID_ATTR_HANDLE - Optional Characteristic is absent

CyBle_HrssGetCharacteristicValue

Prototype

```
CYBLE_API_RESULT_T CyBle_HrssGetCharacteristicValue(CYBLE_HRS_CHAR_INDEX_T charIndex,
uint8 attrSize, uint8 * attrValue);
```

Description

Gets the local Characteristic value of specified Heart Rate Service Characteristic.

Parameters

Parameters	Description
CYBLE_HRS_CHAR_INDEX_T charIndex	The index of a service Characteristic.
uint8 attrSize	The size of the Characteristic value attribute. The Heart Rate Measurement Characteristic has a 20 byte length (by default). The Body Sensor Location and Control Point Characteristic both have 1 byte length.
uint8 * attrValue	The pointer to the location where Characteristic value data should be stored.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The request handled successfully
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameter failed
- CYBLE_ERROR_GATT_DB_INVALID_ATTR_HANDLE - Optional Characteristic is absent

CyBle_HrssGetCharacteristicDescriptor

Prototype

```
CYBLE_API_RESULT_T CyBle_HrssGetCharacteristicDescriptor(CYBLE_HRS_CHAR_INDEX_T
charIndex, CYBLE_HRS_DESCR_INDEX_T descrIndex, uint8 attrSize, uint8 * attrValue);
```

Description

Gets the local Characteristic Descriptor of the specified Heart Rate Service Characteristic.

Parameters

Parameters	Description
CYBLE_HRS_CHAR_INDEX_T charIndex	The index of the Characteristic.
CYBLE_HRS_DESCR_INDEX_T descrIndex	The index of the Descriptor.
uint8 attrSize	The size of the Descriptor value attribute. The Heart Rate Measurement Characteristic client configuration Descriptor has 2 bytes length.
uint8 * attrValue	The pointer to the location where Characteristic Descriptor value data should be stored.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The request handled successfully
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameter failed
- CYBLE_ERROR_GATT_DB_INVALID_ATTR_HANDLE - Optional Descriptor is absent

CyBle_HrssSendNotification

Prototype

```
CYBLE_API_RESULT_T CyBle_HrssSendNotification(CYBLE_CONN_HANDLE_T connHandle,
CYBLE_HRS_CHAR_INDEX_T charIndex, uint8 attrSize, uint8 * attrValue);
```

Description

Sends notification of a specified Heart Rate Service Characteristic value to the Client device. No response is expected.

The CYBLE_EVT_HRSC_NOTIFICATION event is received by the peer device, on invoking this function.

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	The connection handle which consist of the device ID and ATT connection ID.
CYBLE_HRS_CHAR_INDEX_T charIndex	The index of a service Characteristic.
uint8 attrSize	The size of the Characteristic value attribute. The Heart Rate Measurement Characteristic has a 20 byte length (by default). The Body Sensor Location and Control Point Characteristic both have 1 byte length.
uint8 * attrValue	The pointer to the Characteristic value data that should be sent to the client device.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The request handled successfully
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameter failed
- CYBLE_ERROR_INVALID_OPERATION - This operation is not permitted
- CYBLE_ERROR_INVALID_STATE - Connection with the client is not established
- CYBLE_ERROR_MEMORY_ALLOCATION_FAILED -Memory allocation failed.
- CYBLE_ERROR_NTF_DISABLED - Notification is not enabled by the client.

HRS Client Functions

APIs unique to HRS designs configured as a GATT Client role.

A letter 'c' is appended to the API name: CyBle_Hrsc

Functions

Function	Description
CyBle_HrscSetCharacteristicValue	This function is used to write the Characteristic value attribute (identified by charIndex) to the server. The Write Response just confirms the operation

	success. This... more
CyBle_HrscGetCharacteristicValue	This function is used to read the Characteristic Value from a server which is identified by charIndex. The Read Response returns the Characteristic Value in... more
CyBle_HrscSetCharacteristicDescriptor	This function is used to write the Characteristic Value to the server, which is identified by charIndex. This function call can result in generation of... more
CyBle_HrscGetCharacteristicDescriptor	Gets a Characteristic Descriptor of a specified Characteristic of the service. This function call can result in generation of the following events based on the... more

CyBle_HrscSetCharacteristicValue

Prototype

```
CYBLE_API_RESULT_T CyBle_HrscSetCharacteristicValue(CYBLE_CONN_HANDLE_T connHandle,
CYBLE_HRS_CHAR_INDEX_T charIndex, uint8 attrSize, uint8 * attrValue);
```

Description

This function is used to write the Characteristic value attribute (identified by charIndex) to the server. The Write Response just confirms the operation success.

This function call can result in generation of the following events based on the response from the server device:

- CYBLE_EVT_HRSC_WRITE_CHAR_RESPONSE
- CYBLE_EVT_GATTC_ERROR_RSP

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	The connection handle.
CYBLE_HRS_CHAR_INDEX_T charIndex	The index of a service Characteristic.
uint8 attrSize	The size of the Characteristic value attribute.
uint8 * attrValue	The pointer to the Characteristic value data that should be sent to the server device.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The request was sent successfully

- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameters failed
- CYBLE_ERROR_MEMORY_ALLOCATION_FAILED -Memory allocation failed
- CYBLE_ERROR_INVALID_STATE - Connection with the server is not established
- CYBLE_ERROR_GATT_DB_INVALID_ATTR_HANDLE - The peer device doesn't have the
- particular Characteristic
- CYBLE_ERROR_INVALID_OPERATION - Operation is invalid for this Characteristic

CyBle_HrscGetCharacteristicValue

Prototype

```
CYBLE_API_RESULT_T CyBle_HrscGetCharacteristicValue(CYBLE_CONN_HANDLE_T connHandle,
CYBLE_HRS_CHAR_INDEX_T charIndex);
```

Description

This function is used to read the Characteristic Value from a server which is identified by charIndex.

The Read Response returns the Characteristic Value in the Attribute Value parameter.

The Read Response only contains the Characteristic Value that is less than or equal to (MTU - 1) octets in length. If the Characteristic Value is greater than (MTU - 1) octets in length, the Read Long Characteristic Value procedure may be used if the rest of the Characteristic Value is required.

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	The connection handle.
CYBLE_HRS_CHAR_INDEX_T charIndex	The index of the service Characteristic.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The read request was sent successfully
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameters failed
- CYBLE_ERROR_GATT_DB_INVALID_ATTR_HANDLE - The peer device doesn't have the particular Characteristic

- CYBLE_ERROR_MEMORY_ALLOCATION_FAILED -Memory allocation failed
- CYBLE_ERROR_INVALID_STATE - Connection with the server is not established
- CYBLE_ERROR_INVALID_OPERATION - Operation is invalid for this Characteristic

CyBle_HrscSetCharacteristicDescriptor

Prototype

```
CYBLE_API_RESULT_T CyBle_HrscSetCharacteristicDescriptor(CYBLE_CONN_HANDLE_T
connHandle, CYBLE_HRS_CHAR_INDEX_T charIndex, CYBLE_HRS_DESCR_INDEX_T descrIndex, uint8
attrSize, uint8 * attrValue);
```

Description

This function is used to write the Characteristic Value to the server, which is identified by charIndex.

This function call can result in generation of the following events based on the response from the server device:

- CYBLE_EVT_HRSC_WRITE_DESCR_RESPONSE
- CYBLE_EVT_GATTC_ERROR_RSP

One of the following events is received by the peer device, on invoking this function:

- CYBLE_EVT_HRSS_NOTIFICATION_ENABLED
- CYBLE_EVT_HRSS_NOTIFICATION_DISABLED

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	The connection handle.
CYBLE_HRS_CHAR_INDEX_T charIndex	The index of the service Characteristic.
CYBLE_HRS_DESCR_INDEX_T descrIndex	The index of the service Characteristic Descriptor.
uint8 attrSize	The size of the Characteristic Descriptor value attribute.
uint8 * attrValue	The pointer to the Characteristic Descriptor value data that should be sent to the server device.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The request was sent successfully
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameters failed
- CYBLE_ERROR_INVALID_STATE - The state is not valid
- CYBLE_ERROR_MEMORY_ALLOCATION_FAILED -Memory allocation failed
- CYBLE_ERROR_GATT_DB_INVALID_ATTR_HANDLE - The peer device doesn't have the particular Characteristic
- CYBLE_ERROR_INVALID_OPERATION - This operation is not permitted on the specified attribute

CyBle_HrscGetCharacteristicDescriptor

Prototype

```
CYBLE_API_RESULT_T CyBle_HrscGetCharacteristicDescriptor(CYBLE_CONN_HANDLE_T
connHandle, CYBLE_HRS_CHAR_INDEX_T charIndex, CYBLE_HRS_DESCR_INDEX_T descrIndex);
```

Description

Gets a Characteristic Descriptor of a specified Characteristic of the service.

This function call can result in generation of the following events based on the response from the server device:

- CYBLE_EVT_HRSC_READ_DESCR_RESPONSE
- CYBLE_EVT_GATTC_ERROR_RSP

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	The connection handle.
CYBLE_HRS_CHAR_INDEX_T charIndex	The index of the service Characteristic.
CYBLE_HRS_DESCR_INDEX_T descrIndex	The index of the service Characteristic Descriptor.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The request was sent successfully
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameters failed
- CYBLE_ERROR_INVALID_STATE - The state is not valid

- CYBLE_ERROR_MEMORY_ALLOCATION_FAILED -Memory allocation failed
- CYBLE_ERROR_GATT_DB_INVALID_ATTR_HANDLE - The peer device doesn't have the
- particular Descriptor
- CYBLE_ERROR_INVALID_OPERATION - This operation is not permitted on the specified attribute

HRS Definitions and Data Structures

Contains the HRS specific definitions and data structures used in the HRS APIs.

Enumerations

Enumeration	Description
CYBLE_HRS_CHAR_INDEX_T	HRS Characteristics indexes
CYBLE_HRS_DESCR_INDEX_T	HRS Characteristic Descriptors indexes

Structures

Structure	Description
CYBLE_HRS_CHAR_VALUE_T	HRS Characteristic value parameter structure
CYBLE_HRS_DESCR_VALUE_T	HRS Characteristic Descriptor value parameter structure
CYBLE_HRSC_T	Structure with discovered attributes information of Heart Rate Service
CYBLE_HRSS_T	Structure with Heart Rate Service attribute handles

CYBLE_HRS_CHAR_INDEX_T

Prototype

```
typedef enum {
  CYBLE_HRS_HRM,
  CYBLE_HRS_BSL,
  CYBLE_HRS_CPT,
  CYBLE_HRS_CHAR_COUNT
} CYBLE_HRS_CHAR_INDEX_T;
```

Description

HRS Characteristics indexes

Members

Members	Description
CYBLE_HRS_HRM	Heart Rate Measurement Characteristic index
CYBLE_HRS_BSL	Body Sensor Location Characteristic index
CYBLE_HRS_CPT	Control Point Characteristic index
CYBLE_HRS_CHAR_COUNT	Total count of HRS Characteristics

CYBLE_HRS_CHAR_VALUE_T

Prototype

```
typedef struct {
  CYBLE_CONN_HANDLE_T connHandle;
  CYBLE_HRS_CHAR_INDEX_T charIndex;
  CYBLE_GATT_VALUE_T * value;
} CYBLE_HRS_CHAR_VALUE_T;
```

Description

HRS Characteristic value parameter structure

Members

Members	Description
CYBLE_CONN_HANDLE_T connHandle;	Peer device handle
CYBLE_HRS_CHAR_INDEX_T charIndex;	Index of service Characteristic
CYBLE_GATT_VALUE_T * value;	Characteristic value

CYBLE_HRS_DESCR_INDEX_T

Prototype

```
typedef enum {
  CYBLE_HRS_HRM_CCCD,
  CYBLE_HRS_DESCR_COUNT
} CYBLE_HRS_DESCR_INDEX_T;
```

Description

HRS Characteristic Descriptors indexes

Members

Members	Description
CYBLE_HRS_HRM_CCCD	Heart Rate Measurement client char. config. Descriptor index
CYBLE_HRS_DESCR_COUNT	Total count of HRS HRM Descriptors

CYBLE_HRS_DESCR_VALUE_T

Prototype

```
typedef struct {
  CYBLE_CONN_HANDLE_T connHandle;
  CYBLE_HRS_CHAR_INDEX_T charIndex;
  CYBLE_HRS_DESCR_INDEX_T descrIndex;
  CYBLE_GATT_VALUE_T * value;
} CYBLE_HRS_DESCR_VALUE_T;
```

Description

HRS Characteristic Descriptor value parameter structure

Members

Members	Description
CYBLE_CONN_HANDLE_T connHandle;	Peer device handle
CYBLE_HRS_CHAR_INDEX_T charIndex;	Index of service Characteristic
CYBLE_HRS_DESCR_INDEX_T descrIndex;	Index of service Characteristic Descriptor
CYBLE_GATT_VALUE_T * value;	Descriptor value

CYBLE_HRSC_T

Prototype

```
typedef struct {
  CYBLE_SRVR_CHAR_INFO_T charInfo[CYBLE_HRS_CHAR_COUNT];
  CYBLE_GATT_DB_ATTR_HANDLE_T hrmCccdHandle;
} CYBLE_HRSC_T;
```

Description

Structure with discovered attributes information of Heart Rate Service

Members

Members	Description
CYBLE_SRVR_CHAR_INFO_T charInfo[CYBLE_HRS_CHAR_COUNT];	Heart Rate Service Characteristics handles and properties array
CYBLE_GATT_DB_ATTR_HANDLE_T hrmCccdHandle;	Heart Rate Measurement client char. config. Descriptor Handle

CYBLE_HRSS_T

Prototype

```
typedef struct {
  CYBLE_GATT_DB_ATTR_HANDLE_T serviceHandle;
  CYBLE_GATT_DB_ATTR_HANDLE_T charHandle[CYBLE_HRS_CHAR_COUNT];
  CYBLE_GATT_DB_ATTR_HANDLE_T hrmCccdHandle;
} CYBLE_HRSS_T;
```

Description

Structure with Heart Rate Service attribute handles

Members

Members	Description
CYBLE_GATT_DB_ATTR_HANDLE_T serviceHandle;	Heart Rate Service handle
CYBLE_GATT_DB_ATTR_HANDLE_T charHandle[CYBLE_HRS_CHAR_COUNT];	Heart Rate Service Characteristics handles and properties array
CYBLE_GATT_DB_ATTR_HANDLE_T hrmCccdHandle;	Heart Rate Measurement client char. config. Descriptor Handle

Health Thermometer Service (HTS)

The Health Thermometer Service exposes temperature and other data related to a thermometer used for healthcare applications.

Depending on the chosen GATT role in the GUI, you may use a subset of the supported APIs.

The HTS API names begin with CyBle_Hts. In addition to this, the APIs also append the GATT role initial letter in the API name.

HTS Server and Client Function

These are APIs common to both GATT Client role and GATT Server role. You may use them in either roles.

No letter is appended to the API name: CyBle_Hts

Functions

Function	Description
CyBle_HtsRegisterAttrCallback	Registers a callback function for service specific attribute operations.

CyBle_HtsRegisterAttrCallback

Prototype

```
void CyBle_HtsRegisterAttrCallback(CYBLE_CALLBACK_T callbackFunc);
```

Description

Registers a callback function for service specific attribute operations.

Parameters

Parameters	Description
CYBLE_CALLBACK_T callbackFunc	An application layer event callback function to receive events from the BLE Component. The definition of CYBLE_CALLBACK_T for HTS Service is, typedef void (* CYBLE_CALLBACK_T) (uint32 eventCode, void *eventParam) eventCode indicates the event that triggered this callback (e.g. CYBLE_EVT_HTSS_NOTIFICATION_ENABLED). eventParam contains the parameters corresponding to the current event. (e.g. pointer to CYBLE_HTS_CHAR_VALUE_T structure that contains details of the Characteristic for which notification enabled event was triggered).

Returns

None

HTS Server Functions

APIs unique to HTS designs configured as a GATT Server role.

A letter 's' is appended to the API name: CyBle_Htss

Functions

Function	Description
CyBle_HtssSetCharacteristicValue	Sets the Characteristic value of the service in the local database.
CyBle_HtssGetCharacteristicValue	Gets the Characteristic value of the service, which is a value identified by charIndex.
CyBle_HtssSetCharacteristicDescriptor	Sets the Characteristic Descriptor of the specified Characteristic.

CyBle_HtssGetCharacteristicDescriptor	Gets the Characteristic Descriptor of the specified Characteristic.
CyBle_HtssSendIndication	Sends indication with a Characteristic value of the Health Thermometer Service, which is a value specified by charIndex, to the Client device.
CyBle_HtssSendNotification	Sends notification with a Characteristic value of the Health Thermometer Service, which is a value specified by charIndex, to the Client device.

CyBle_HtssSetCharacteristicValue

Prototype

```
CYBLE_API_RESULT_T CyBle_HtssSetCharacteristicValue(CYBLE_HTS_CHAR_INDEX_T charIndex,
uint8 attrSize, uint8 * attrValue);
```

Description

Sets the Characteristic value of the service in the local database.

Parameters

Parameters	Description
CYBLE_HTS_CHAR_INDEX_T charIndex	The index of the service Characteristic.
uint8 attrSize	The size (in Bytes) of the Characteristic value attribute.
uint8 * attrValue	The pointer to the Characteristic value data that should be sent to the server device.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The request handled successfully
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameter failed

CyBle_HtssGetCharacteristicValue

Prototype

```
CYBLE_API_RESULT_T CyBle_HtssGetCharacteristicValue(CYBLE_HTS_CHAR_INDEX_T charIndex,
uint8 attrSize, uint8 * attrValue);
```

Description

Gets the Characteristic value of the service, which is a value identified by charIndex.

Parameters

Parameters	Description
CYBLE_HTS_CHAR_INDEX_T charIndex	The index of the service Characteristic.
uint8 attrSize	The size of the Characteristic value attribute.
uint8 * attrValue	The pointer to the location where Characteristic value data should be stored.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The request handled successfully
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameter failed

CyBle_HtssSetCharacteristicDescriptor

Prototype

```
CYBLE_API_RESULT_T CyBle_HtssSetCharacteristicDescriptor(CYBLE_HTS_CHAR_INDEX_T
charIndex, CYBLE_HTS_DESCR_INDEX_T descrIndex, uint8 attrSize, uint8 * attrValue);
```

Description

Sets the Characteristic Descriptor of the specified Characteristic.

Parameters

Parameters	Description
CYBLE_HTS_CHAR_INDEX_T charIndex	The index of the service Characteristic.
CYBLE_HTS_DESCR_INDEX_T descrIndex	The index of the service Characteristic Descriptor.
uint8 attrSize	The size of the Characteristic Descriptor attribute.
uint8 * attrValue	The pointer to the Descriptor value data that should be stored in the GATT database.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The request handled successfully
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameter failed

CyBle_HtssGetCharacteristicDescriptor

Prototype

```
CYBLE_API_RESULT_T CyBle_HtssGetCharacteristicDescriptor(CYBLE_HTS_CHAR_INDEX_T
charIndex, CYBLE_HTS_DESCR_INDEX_T descrIndex, uint8 attrSize, uint8 * attrValue);
```

Description

Gets the Characteristic Descriptor of the specified Characteristic.

Parameters

Parameters	Description
CYBLE_HTS_CHAR_INDEX_T charIndex	The index of the service Characteristic.
CYBLE_HTS_DESCR_INDEX_T descrIndex	The index of the service Characteristic Descriptor.
uint8 attrSize	The size of the Characteristic Descriptor attribute.
uint8 * attrValue	The pointer to the location where Characteristic Descriptor value data should be stored.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The request handled successfully
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameter failed

CyBle_HtssSendIndication

Prototype

```
CYBLE_API_RESULT_T CyBle_HtssSendIndication(CYBLE_CONN_HANDLE_T connHandle,
CYBLE_HTS_CHAR_INDEX_T charIndex, uint8 attrSize, uint8 * attrValue);
```

Description

Sends indication with a Characteristic value of the Health Thermometer Service, which is a value specified by charIndex, to the Client device.

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	The connection handle.

CYBLE_HTS_CHAR_INDEX_T charIndex	The index of the service Characteristic.
uint8 attrSize	The size of the Characteristic value attribute.
uint8 * attrValue	The pointer to the Characteristic value data that should be sent to the Client device.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The request handled successfully
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameter failed
- CYBLE_ERROR_INVALID_OPERATION - This operation is not permitted
- CYBLE_ERROR_INVALID_STATE - Connection with the client is not established
- CYBLE_ERROR_MEMORY_ALLOCATION_FAILED -Memory allocation failed
- CYBLE_ERROR_IND_DISABLED - Indication is not enabled by the client

CyBle_HtssSendNotification

Prototype

```
CYBLE_API_RESULT_T CyBle_HtssSendNotification(CYBLE_CONN_HANDLE_T connHandle,
CYBLE_HTS_CHAR_INDEX_T charIndex, uint8 attrSize, uint8 * attrValue);
```

Description

Sends notification with a Characteristic value of the Health Thermometer Service, which is a value specified by charIndex, to the Client device.

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	The connection handle.
CYBLE_HTS_CHAR_INDEX_T charIndex	The index of the service Characteristic.
uint8 attrSize	The size of the Characteristic value attribute.
uint8 * attrValue	The pointer to the Characteristic value data that should be sent to the Client device.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The request handled successfully
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameter failed
- CYBLE_ERROR_INVALID_OPERATION - This operation is not permitted
- CYBLE_ERROR_INVALID_STATE - Connection with the client is not established
- CYBLE_ERROR_MEMORY_ALLOCATION_FAILED -Memory allocation failed.
- CYBLE_ERROR_NTF_DISABLED - Notification is not enabled by the client.

HTS Client Functions

APIs unique to HTS designs configured as a GATT Client role.

A letter 'c' is appended to the API name: CyBle_Htsc

Functions

Function	Description
CyBle_HtscSetCharacteristicValue	Sends a request to set a Characteristic value of the service, which is a value identified by charIndex,to the server device.
CyBle_HtscGetCharacteristicValue	This function is used to read a Characteristic value, which is a value identified by charIndex, from the server.
CyBle_HtscSetCharacteristicDescriptor	This function is used to write the Characteristic Descriptor to the server, which is identified by charIndex.
CyBle_HtscGetCharacteristicDescriptor	Gets the Characteristic Descriptor of the specified Characteristic of the service.

CyBle_HtscSetCharacteristicValue

Prototype

```
CYBLE_API_RESULT_T CyBle_HtscSetCharacteristicValue(CYBLE_CONN_HANDLE_T connHandle,
CYBLE_HTS_CHAR_INDEX_T charIndex, uint8 attrSize, uint8 * attrValue);
```

Description

Sends a request to set a Characteristic value of the service, which is a value identified by charIndex,to the server device.

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	The connection handle.
CYBLE_HTS_CHAR_INDEX_T charIndex	The index of the service Characteristic.
uint8 attrSize	The size of the Characteristic value attribute.
uint8 * attrValue	The pointer to the Characteristic value data that should be sent to the server device.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The request was sent successfully
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameters failed
- CYBLE_ERROR_MEMORY_ALLOCATION_FAILED -Memory allocation failed
- CYBLE_ERROR_INVALID_STATE - Connection with the server is not established
- CYBLE_ERROR_GATT_DB_INVALID_ATTR_HANDLE - The peer device doesn't have the particular Characteristic
- CYBLE_ERROR_INVALID_OPERATION - Operation is invalid for this Characteristic

CyBle_HtscGetCharacteristicValue

Prototype

```
CYBLE_API_RESULT_T CyBle_HtscGetCharacteristicValue(CYBLE_CONN_HANDLE_T connHandle,
CYBLE_HTS_CHAR_INDEX_T charIndex);
```

Description

This function is used to read a Characteristic value, which is a value identified by charIndex, from the server.

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	The connection handle.
CYBLE_HTS_CHAR_INDEX_T charIndex	The index of the service Characteristic.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The read request was sent successfully
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameters failed
- CYBLE_ERROR_GATT_DB_INVALID_ATTR_HANDLE - The peer device doesn't have the particular Characteristic
- CYBLE_ERROR_MEMORY_ALLOCATION_FAILED -Memory allocation failed
- CYBLE_ERROR_INVALID_STATE - Connection with the server is not established
- CYBLE_ERROR_INVALID_OPERATION - Operation is invalid for this Characteristic

CyBle_HtscSetCharacteristicDescriptor

Prototype

```
CYBLE_API_RESULT_T CyBle_HtscSetCharacteristicDescriptor(CYBLE_CONN_HANDLE_T
connHandle, CYBLE_HTS_CHAR_INDEX_T charIndex, CYBLE_HTS_DESCR_INDEX_T descrIndex, uint8
attrSize, uint8 * attrValue);
```

Description

This function is used to write the Characteristic Descriptor to the server, which is identified by charIndex and descrIndex.

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	The connection handle.
CYBLE_HTS_CHAR_INDEX_T charIndex	The index of the service Characteristic.
CYBLE_HTS_DESCR_INDEX_T descrIndex	The index of the service Characteristic Descriptor.
uint8 attrSize	The size of the Characteristic value attribute.
uint8 * attrValue	The pointer to the Characteristic Descriptor value data that should be sent to the server device.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The request was sent successfully

- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameters failed
- CYBLE_ERROR_INVALID_STATE - The state is not valid
- CYBLE_ERROR_MEMORY_ALLOCATION_FAILED -Memory allocation failed
- CYBLE_ERROR_INVALID_OPERATION - This operation is not permitted on the specified attribute

CyBle_HtscGetCharacteristicDescriptor

Prototype

```
CYBLE_API_RESULT_T CyBle_HtscGetCharacteristicDescriptor(CYBLE_CONN_HANDLE_T
connHandle, CYBLE_HTS_CHAR_INDEX_T charIndex, CYBLE_HTS_DESCR_INDEX_T descrIndex);
```

Description

Gets the Characteristic Descriptor of the specified Characteristic of the service.

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	The connection handle.
CYBLE_HTS_CHAR_INDEX_T charIndex	The index of the service Characteristic.
CYBLE_HTS_DESCR_INDEX_T descrIndex	The index of the service Characteristic Descriptor.

Returns

- CYBLE_ERROR_OK - The request was sent successfully
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameters failed
- CYBLE_ERROR_INVALID_STATE - The state is not valid
- CYBLE_ERROR_MEMORY_ALLOCATION_FAILED -Memory allocation failed
- CYBLE_ERROR_INVALID_OPERATION - This operation is not permitted on the specified attribute

HTS Definitions and Data Structures

Contains the HTS specific definitions and data structures used in the HTS APIs.

Enumerations

Enumeration	Description
CYBLE_HTS_CHAR_INDEX_T	HTS Characteristic indexes
CYBLE_HTS_DESCR_INDEX_T	HTS Characteristic Descriptors indexes
CYBLE_HTS_TEMP_TYPE_T	Temperature Type measurement indicates where the temperature was measured

Structures

Structure	Description
CYBLE_HTS_CHAR_VALUE_T	HTS Characteristic value parameter structure
CYBLE_HTS_DESCR_VALUE_T	HTS Characteristic Descriptor value parameter structure
CYBLE_HTSC_CHAR_T	HTS Characteristic with Descriptors
CYBLE_HTSC_T	Structure with discovered attributes information of Health Thermometer Service
CYBLE_HTSS_CHAR_T	HTS Characteristic with Descriptors
CYBLE_HTSS_T	Structure with Health Thermometer Service attribute handles
CYBLE_HTS_FLOAT32	The IEEE-11073 FLOAT-Type is defined as a 32-bit value with a 24-bit mantissa and an 8-bit exponent.

CYBLE_HTS_CHAR_INDEX_T

Prototype

```
typedef enum {
  CYBLE_HTS_TEMP_MEASURE,
  CYBLE_HTS_TEMP_TYPE,
  CYBLE_HTS_INTERM_TEMP,
  CYBLE_HTS_MEASURE_INTERVAL,
  CYBLE_HTS_CHAR_COUNT
} CYBLE_HTS_CHAR_INDEX_T;
```

Description

HTS Characteristic indexes

Members

Members	Description
CYBLE_HTS_TEMP_MEASURE	Temperature Measurement Characteristic index
CYBLE_HTS_TEMP_TYPE	Temperature Type Characteristic index

CYBLE_HTS_INTERM_TEMP	Intermediate Temperature Characteristic index
CYBLE_HTS_MEASURE_INTERVAL	Measurement Interval Characteristic index
CYBLE_HTS_CHAR_COUNT	Total count of HTS Characteristics

CYBLE_HTS_CHAR_VALUE_T

Prototype

```
typedef struct {
  CYBLE_CONN_HANDLE_T connHandle;
  CYBLE_HTS_CHAR_INDEX_T charIndex;
  CYBLE_GATT_VALUE_T * value;
} CYBLE_HTS_CHAR_VALUE_T;
```

Description

HTS Characteristic value parameter structure

Members

Members	Description
CYBLE_CONN_HANDLE_T connHandle;	Peer device handle
CYBLE_HTS_CHAR_INDEX_T charIndex;	Index of service Characteristic
CYBLE_GATT_VALUE_T * value;	Characteristic value

CYBLE_HTS_DESCR_INDEX_T

Prototype

```
typedef enum {
  CYBLE_HTS_CCCD,
  CYBLE_HTS_VRD,
  CYBLE_HTS_DESCR_COUNT
} CYBLE_HTS_DESCR_INDEX_T;
```

Description

HTS Characteristic Descriptors indexes

Members

Members	Description
CYBLE_HTS_CCCD	Client Characteristic Configuration Descriptor index
CYBLE_HTS_VRD	Valid Range Descriptor index

CYBLE_HTS_DESCR_COUNT	Total count of Descriptors

CYBLE_HTS_DESCR_VALUE_T

Prototype

```
typedef struct {
  CYBLE_CONN_HANDLE_T connHandle;
  CYBLE_HTS_CHAR_INDEX_T charIndex;
  CYBLE_HTS_DESCR_INDEX_T descrIndex;
  CYBLE_GATT_VALUE_T * value;
} CYBLE_HTS_DESCR_VALUE_T;
```

Description

HTS Characteristic Descriptor value parameter structure

Members

Members	Description
CYBLE_CONN_HANDLE_T connHandle;	Peer device handle
CYBLE_HTS_CHAR_INDEX_T charIndex;	Index of service Characteristic
CYBLE_HTS_DESCR_INDEX_T descrIndex;	Index of Descriptor
CYBLE_GATT_VALUE_T * value;	Characteristic value

CYBLE_HTS_TEMP_TYPE_T

Prototype

```
typedef enum {
  CYBLE_HTS_TEMP_TYPE_ARMPIT = 0x01u,
  CYBLE_HTS_TEMP_TYPE_BODY,
  CYBLE_HTS_TEMP_TYPE_EAR,
  CYBLE_HTS_TEMP_TYPE_FINGER,
  CYBLE_HTS_TEMP_TYPE_GI_TRACT,
  CYBLE_HTS_TEMP_TYPE_MOUTH,
  CYBLE_HTS_TEMP_TYPE_RECTUM,
  CYBLE_HTS_TEMP_TYPE_TOE,
  CYBLE_HTS_TEMP_TYPE_TYMPANUM
} CYBLE_HTS_TEMP_TYPE_T;
```

Description

Temperature Type measurement indicates where the temperature was measured

Members

Members	Description
CYBLE_HTS_TEMP_TYPE_ARMPIT = 0x01u	Armpit
CYBLE_HTS_TEMP_TYPE_BODY	Body (general)
CYBLE_HTS_TEMP_TYPE_EAR	Ear (usually ear lobe)
CYBLE_HTS_TEMP_TYPE_FINGER	Finger
CYBLE_HTS_TEMP_TYPE_GI_TRACT	Gastro-intestinal Tract
CYBLE_HTS_TEMP_TYPE_MOUTH	Mouth
CYBLE_HTS_TEMP_TYPE_RECTUM	Rectum
CYBLE_HTS_TEMP_TYPE_TOE	Toe
CYBLE_HTS_TEMP_TYPE_TYMPANUM	Tympanum (ear drum)

CYBLE_HTSC_CHAR_T

Prototype

```
typedef struct {
  CYBLE_GATT_DB_ATTR_HANDLE_T descrHandle[CYBLE_HTS_DESCR_COUNT];
  CYBLE_GATT_DB_ATTR_HANDLE_T valueHandle;
  CYBLE_GATT_DB_ATTR_HANDLE_T endHandle;
  uint8 properties;
} CYBLE_HTSC_CHAR_T;
```

Description

HTS Characteristic with Descriptors

Members

Members	Description
CYBLE_GATT_DB_ATTR_HANDLE_T descrHandle[CYBLE_HTS_DESCR_COUNT];	Handle of Descriptor
CYBLE_GATT_DB_ATTR_HANDLE_T valueHandle;	Handle of Report Characteristic value
CYBLE_GATT_DB_ATTR_HANDLE_T endHandle;	End handle of Characteristic
uint8 properties;	Properties for value field

CYBLE_HTSC_T

Prototype

```
typedef struct {
  CYBLE_HTSC_CHAR_T charInfo[CYBLE_HTS_CHAR_COUNT]; } CYBLE_HTSC_T;
```

Description

Structure with discovered attributes information of Health Thermometer Service

Members

Members	Description
CYBLE_HTSC_CHAR_T charInfo[CYBLE_HTS_CHAR_COUNT];	Characteristics handles array

CYBLE_HTSS_CHAR_T

Prototype

```
typedef struct {
  CYBLE_GATT_DB_ATTR_HANDLE_T charHandle;
  CYBLE_GATT_DB_ATTR_HANDLE_T descrHandle[CYBLE_HTS_DESCR_COUNT]; }
CYBLE_HTSS_CHAR_T;
```

Description

HTS Characteristic with Descriptors

Members

Members	Description
CYBLE_GATT_DB_ATTR_HANDLE_T charHandle;	Handle of Characteristic value
CYBLE_GATT_DB_ATTR_HANDLE_T descrHandle[CYBLE_HTS_DESCR_COUNT];	Handle of Descriptor

CYBLE_HTSS_T

Prototype

```
typedef struct {
  CYBLE_GATT_DB_ATTR_HANDLE_T serviceHandle;
  CYBLE_HTSS_CHAR_T charInfo[CYBLE_HTS_CHAR_COUNT];
} CYBLE_HTSS_T;
```

Description

Structure with Health Thermometer Service attribute handles

Members

Members	Description
CYBLE_GATT_DB_ATTR_HANDLE_T serviceHandle;	Health Thermometer Service handle
CYBLE_HTSS_CHAR_T charInfo[CYBLE_HTS_CHAR_COUNT];	Health Thermometer Service Characteristic handles

CYBLE_HTS_FLOAT32

Prototype

```
typedef struct {
  int8 exponent;
  int32 mantissa;
} CYBLE_HTS_FLOAT32;
```

Description

The IEEE-11073 FLOAT-Type is defined as a 32-bit value with a 24-bit mantissa and an 8-bit exponent.

Members

Members	Description
int8 exponent;	Base 10 exponent
int32 mantissa;	Mantissa, should be using only 24 bits

Immediate Alert Service (IAS)

The Immediate Alert Service uses the Alert Level Characteristic to cause an alert when it is written with a value other than "No Alert".

Depending on the chosen GATT role in the GUI, you may use a subset of the supported APIs.

The IAS API names begin with CyBle_Ias. In addition to this, the APIs also append the GATT role initial letter in the API name.

IAS Server and Client Function

These are APIs common to both GATT Client role and GATT Server role. You may use them in either roles.

No letter is appended to the API name: CyBle_Ias

Functions

Function	Description
CyBle_IasRegisterAttrCallback	Registers callback function for service specific attribute operations.

CyBle_IasRegisterAttrCallback

Prototype

```
void CyBle_IasRegisterAttrCallback(CYBLE_CALLBACK_T callbackFunc);
```

Description

Registers callback function for service specific attribute operations.

Parameters

Parameters	Description
CYBLE_CALLBACK_T callbackFunc	An application layer event callback function to receive events from the BLE Component. The definition of CYBLE_CALLBACK_T for IAS Service is, typedef void (* CYBLE_CALLBACK_T) (uint32 eventCode, void *eventParam) eventCode indicates the event that triggered this callback (e.g. CYBLE_EVT_IASS_NOTIFICATION_ENABLED). eventParam contains the parameters corresponding to the current event. (e.g. pointer to CYBLE_IAS_CHAR_VALUE_T structure that contains details of the Characteristic for which notification enabled event was triggered).

Returns

None

Notes

IAS only has events for the GATT server. There are no events for the GATT client since the client sends data without waiting for response. Therefore there is no need to register a callback through CyBle_IasRegisterAttrCallback for an IAS GATT client.

Side Effects

The *eventParams in the callback function should not be used by the application once the callback function execution is finished. Otherwise this data may become corrupted.

IAS Server Functions

APIs unique to IAS designs configured as a GATT Server role.

A letter 's' is appended to the API name: CyBle_Iass

Functions

Function	Description
CyBle_IassGetCharacteristicValue	Gets the Alert Level Characteristic value of the service, which is identified by charIndex.

CyBle_IassGetCharacteristicValue

Prototype

```
CYBLE_API_RESULT_T CyBle_IassGetCharacteristicValue(CYBLE_IAS_CHAR_INDEX_T charIndex,
uint8 attrSize, uint8 * attrValue);
```

Description

Gets the Alert Level Characteristic value of the service, which is identified by charIndex.

Parameters

Parameters	Description
CYBLE_IAS_CHAR_INDEX_T charIndex	The index of the Alert Level Characteristic.
uint8 attrSize	The size of the Alert Level Characteristic value attribute.
uint8 * attrValue	The pointer to the location where the Alert Level Characteristic value data should be stored.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The Characteristic value was read successfully
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameters failed

IAS Client Functions

APIs unique to IAS designs configured as a GATT Client role.

A letter 'c' is appended to the API name: CyBle_Iasc

Functions

Function	Description
CyBle_IascSetCharacteristicValue	Sets a Alert Level Characteristic value of the service, which is identified by charIndex.

CyBle_IascSetCharacteristicValue

Prototype

```
CYBLE_API_RESULT_T CyBle_IascSetCharacteristicValue(CYBLE_CONN_HANDLE_T connHandle,
CYBLE_IAS_CHAR_INDEX_T charIndex, uint8 attrSize, uint8 * attrValue);
```

Description

Sets a Alert Level Characteristic value of the service, which is identified by charIndex.

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	The connection handle.
CYBLE_IAS_CHAR_INDEX_T charIndex	The index of the Alert Level service Characteristic.
uint8 attrSize	The size of the Alert Level Characteristic value attribute.
uint8 * attrValue	The pointer to the Alert Level Characteristic value data that should be stored in the GATT database.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The request was sent successfully
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameters failed
- CYBLE_ERROR_MEMORY_ALLOCATION_FAILED -Memory allocation failed
- CYBLE_ERROR_INVALID_STATE - Connection with the server is not established
- CYBLE_ERROR_INVALID_OPERATION - Operation is invalid for this Characteristic

IAS Definitions and Data Structures

Contains the IAS specific definitions and data structures used in the IAS APIs.

Enumerations

Enumeration	Description
CYBLE_IAS_CHAR_INDEX_T	Immediate Alert Service Characteristic indexes

Structures

Structure	Description
CYBLE_IAS_CHAR_VALUE_T	Immediate Alert Service Characteristic Value parameters structure
CYBLE_IASC_T	Structure with discovered attributes information of Immediate Alert Service
CYBLE_IASS_T	Structure with Immediate Alert Service attribute handles

CYBLE_IAS_CHAR_INDEX_T

Prototype

```
typedef enum {
  CYBLE_IAS_ALERT_LEVEL,
  CYBLE_IAS_CHAR_COUNT
} CYBLE_IAS_CHAR_INDEX_T;
```

Description

Immediate Alert Service Characteristic indexes

Members

Members	Description
CYBLE_IAS_ALERT_LEVEL	Alert Level Characteristic index
CYBLE_IAS_CHAR_COUNT	Total count of Characteristics

CYBLE_IAS_CHAR_VALUE_T

Prototype

```
typedef struct {
  CYBLE_CONN_HANDLE_T connHandle;
  CYBLE_IAS_CHAR_INDEX_T charIndex;
  CYBLE_GATT_VALUE_T * value;
} CYBLE_IAS_CHAR_VALUE_T;
```

Description

Immediate Alert Service Characteristic Value parameters structure

Members

Members	Description
CYBLE_CONN_HANDLE_T connHandle;	Connection handle
CYBLE_IAS_CHAR_INDEX_T charIndex;	Characteristic index of Immediate Alert Service

CYBLE_GATT_VALUE_T * value;	Pointer to value of Immediate Alert Service Characteristic

CYBLE_IASC_T

Prototype

```
typedef struct {
  CYBLE_SRVR_CHAR_INFO_T alertLevelChar;
} CYBLE_IASC_T;
```

Description

Structure with discovered attributes information of Immediate Alert Service

Members

Members	Description
CYBLE_SRVR_CHAR_INFO_T alertLevelChar;	Handle of Alert Level Characteristic of Immediate Alert Service

CYBLE_IASS_T

Prototype

```
typedef struct {
  CYBLE_GATT_DB_ATTR_HANDLE_T serviceHandle;
  CYBLE_GATT_DB_ATTR_HANDLE_T alertLevelCharHandle;
} CYBLE_IASS_T;
```

Description

Structure with Immediate Alert Service attribute handles

Members

Members	Description
CYBLE_GATT_DB_ATTR_HANDLE_T serviceHandle;	Immediate Alert Service handle
CYBLE_GATT_DB_ATTR_HANDLE_T alertLevelCharHandle;	Handle of Alert Level Characteristic

Link Loss Service (LLS)

The Link Loss Service uses the Alert Level Characteristic to cause an alert in the device when the link is lost.

Depending on the chosen GATT role in the GUI, you may use a subset of the supported APIs.

The LLS API names begin with CyBle_Lls. In addition to this, the APIs also append the GATT role initial letter in the API name.

LLS Server and Client Function

These are APIs common to both GATT Client role and GATT Server role. You may use them in either roles.

No letter is appended to the API name: CyBle_Lls

Functions

Function	Description
CyBle_LlsRegisterAttrCallback	Registers a callback function for service specific attribute operations.

CyBle_LlsRegisterAttrCallback

Prototype

```
void CyBle_LlsRegisterAttrCallback(CYBLE_CALLBACK_T callbackFunc);
```

Description

Registers a callback function for service specific attribute operations.

Parameters

Parameters	Description
CYBLE_CALLBACK_T callbackFunc	An application layer event callback function to receive events from the BLE Component. The definition of CYBLE_CALLBACK_T for Link Loss Service is, typedef void (* CYBLE_CALLBACK_T) (uint32 eventCode, void *eventParam) eventCode indicates the event that triggered this callback (e.g. CYBLE_EVT_LLSS_NOTIFICATION_ENABLED). eventParam contains the parameters corresponding to the current event. (e.g. pointer to CYBLE_LLS_CHAR_VALUE_T structure that contains details of the Characteristic for which notification enabled event was triggered).

Returns

None

Side Effects

The *eventParams in the callback function should not be used by the application once the callback function execution is finished. Otherwise this data may become corrupted.

LLS Server Functions

APIs unique to LLS designs configured as a GATT Server role.

A letter 's' is appended to the API name: CyBle_Llss

Functions

Function	Description
CyBle_LlssGetCharacteristicValue	Gets an Alert Level Characteristic value of the service, which is identified by charIndex.

CyBle_LlssGetCharacteristicValue

Prototype

```
CYBLE_API_RESULT_T CyBle_LlssGetCharacteristicValue(CYBLE_LLS_CHAR_INDEX_T charIndex,
uint8 attrSize, uint8 * attrValue);
```

Description

Gets an Alert Level Characteristic value of the service, which is identified by charIndex.

Parameters

Parameters	Description
CYBLE_LLS_CHAR_INDEX_TcharIndex	The index of an Alert Level Characteristic.
uint8 attrSize	The size of the Alert Level Characteristic value attribute.
uint8 * attrValue	The pointer to the location where an Alert Level Characteristic value data should be stored.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The Characteristic value was read successfully
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameters failed

LLS Client Functions

APIs unique to LLS designs configured as a GATT Client role.

A letter 'c' is appended to the API name: CyBle_Llsc

Functions

Function	Description
CyBle_LlscSetCharacteristicValue	Sets the Alert Level Characteristic value of the Link Loss Service, which is identified by charIndex. This function call can result in generation of the... more
CyBle_LlscGetCharacteristicValue	Sends a request to get Characteristic value of the Link Loss Service, which is identified by charIndex. This function call can result in generation of... more

CyBle_LlscSetCharacteristicValue

Prototype

```
CYBLE_API_RESULT_T CyBle_LlscSetCharacteristicValue(CYBLE_CONN_HANDLE_T connHandle,
CYBLE_LLS_CHAR_INDEX_T charIndex, uint8 attrSize, uint8 * attrValue);
```

Description

Sets the Alert Level Characteristic value of the Link Loss Service, which is identified by charIndex.

This function call can result in generation of the following events based on the response from the server device.

- CYBLE_EVT_LLSC_WRITE_CHAR_RESPONSE
- CYBLE_EVT_GATTC_ERROR_RSP

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	The connection handle.
CYBLE_LLS_CHAR_INDEX_T charIndex	The index of the Alert Level service Characteristic.
uint8 attrSize	The size of the Alert Level Characteristic value attribute.
uint8 * attrValue	The pointer to the Alert Level Characteristic value data that should be sent to the server device.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The request was sent successfully
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameters failed

CyBle_LlscGetCharacteristicValue

Prototype

```
CYBLE_API_RESULT_T CyBle_LlscGetCharacteristicValue(CYBLE_CONN_HANDLE_T connHandle,
CYBLE_LLS_CHAR_INDEX_T charIndex);
```

Description

Sends a request to get Characteristic value of the Link Loss Service, which is identified by charIndex.

This function call can result in generation of the following events based on the response from the server device:

- CYBLE_EVT_LLSC_READ_CHAR_RESPONSE
- CYBLE_EVT_GATTC_ERROR_RSP

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	The connection handle.
CYBLE_LLS_CHAR_INDEX_T charIndex	The index of the Link Loss Service Characteristic.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The request was sent successfully
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameters failed
- CYBLE_ERROR_MEMORY_ALLOCATION_FAILED -Memory allocation failed.
- CYBLE_ERROR_INVALID_STATE - Connection with the server is not established
- CYBLE_ERROR_INVALID_OPERATION - Operation is invalid for this Characteristic

LLS Definitions and Data Structures

Contains the LLS specific definitions and data structures used in the LLS APIs.

Enumerations

Enumeration	Description
CYBLE_LLS_CHAR_INDEX_T	Link Loss Service Characteristic indexes

Structures

Structure	Description
CYBLE_LLS_CHAR_VALUE_T	Link Loss Service Characteristic Value parameter structure
CYBLE_LLSC_T	Structure with discovered attributes information of Link Loss Service

Structure	Description
CYBLE_LLSS_T	Structure with Link Loss Service attribute handles

CYBLE_LLS_CHAR_INDEX_T

Prototype

```
typedef enum {
  CYBLE_LLS_ALERT_LEVEL,
  CYBLE_LLS_CHAR_COUNT
} CYBLE_LLS_CHAR_INDEX_T;
```

Description

Link Loss Service Characteristic indexes

Members

Members	Description
CYBLE_LLS_ALERT_LEVEL	Alert Level Characteristic index
CYBLE_LLS_CHAR_COUNT	Total count of Characteristics

CYBLE_LLS_CHAR_VALUE_T

Prototype

```
typedef struct {
  CYBLE_CONN_HANDLE_T connHandle;
  CYBLE_LLS_CHAR_INDEX_T charIndex;
  CYBLE_GATT_VALUE_T * value;
} CYBLE_LLS_CHAR_VALUE_T;
```

Description

Link Loss Service Characteristic Value parameter structure

Members

Members	Description
CYBLE_CONN_HANDLE_T connHandle;	Connection handle
CYBLE_LLS_CHAR_INDEX_T charIndex;	Characteristic index of Link Loss Service
CYBLE_GATT_VALUE_T * value;	Pointer to value of Link Loss Service Characteristic

CYBLE_LLSC_T

Prototype

```
typedef struct {
  CYBLE_SRVR_CHAR_INFO_T alertLevelChar;
} CYBLE_LLSC_T;
```

Description

Structure with discovered attributes information of Link Loss Service

Members

Members	Description
CYBLE_SRVR_CHAR_INFO_T alertLevelChar;	Handle of Alert Level Characteristic of Link Loss Service

CYBLE_LLSS_T

Prototype

```
typedef struct {
  CYBLE_GATT_DB_ATTR_HANDLE_T serviceHandle;
  CYBLE_GATT_DB_ATTR_HANDLE_T alertLevelCharHandle;
} CYBLE_LLSS_T;
```

Description

Structure with Link Loss Service attribute handles

Members

Members	Description
CYBLE_GATT_DB_ATTR_HANDLE_T serviceHandle;	Link Loss Service handle
CYBLE_GATT_DB_ATTR_HANDLE_T alertLevelCharHandle;	Handle of Alert Level Characteristic

Location and Navigation Service (LNS)

The Location and Navigation Service exposes location and navigation-related data from a Location and Navigation sensor (Server) intended for outdoor activity applications.

Depending on the chosen GATT role in the GUI, you may use a subset of the supported APIs.

The LNS API names begin with CyBle_Lns. In addition to this, the APIs also append the GATT role initial letter in the API name.

LNS Server and Client Function

These are APIs common to both GATT Client role and GATT Server role. You may use them in either roles.

No letter is appended to the API name: CyBle_Lns

Functions

Function	Description
CyBle_LnsRegisterAttrCallback	Registers a callback function for service specific attribute operations.

CyBle_LnsRegisterAttrCallback

Prototype

```
void CyBle_LnsRegisterAttrCallback(CYBLE_CALLBACK_T callbackFunc);
```

Description

Registers a callback function for service specific attribute operations.

Parameters

Parameters	Description
CYBLE_CALLBACK_T callbackFunc	An application layer event callback function to receive events from the BLE Component. The definition of CYBLE_CALLBACK_T for LNS is, typedef void (* CYBLE_CALLBACK_T) (uint32 eventCode, void *eventParam) eventCode indicates the event that triggered this callback. eventParam contains the parameters corresponding to the current event.

Returns

None

Side Effects

The *eventParams in the callback function should not be used by the application once the callback function execution is finished. Otherwise this data may become corrupted.

LNS Server Functions

APIs unique to LNS designs configured as a GATT Server role.

A letter 's' is appended to the API name: CyBle_Lnss

Functions

Function	Description
CyBle_LnssGetCharacteristicDescriptor	Gets a Characteristic Descriptor of the specified Characteristic.
CyBle_LnssGetCharacteristicValue	Gets the value of the Characteristic, as identified by charIndex.
CyBle_LnssSendIndication	Sends an indication of the specified Characteristic value, as identified by the charIndex.
CyBle_LnssSendNotification	Sends a notification of the specified Characteristic value, as identified by the charIndex.
CyBle_LnssSetCharacteristicValue	Sets the value of the Characteristic, as identified by charIndex.

CyBle_LnssGetCharacteristicDescriptor

Prototype

```
CYBLE_API_RESULT_T CyBle_LnssGetCharacteristicDescriptor(CYBLE_LNS_CHAR_INDEX_T
charIndex, CYBLE_LNS_DESCR_INDEX_T descrIndex, uint8 attrSize, uint8 * attrValue);
```

Description

Gets a Characteristic Descriptor of the specified Characteristic.

Parameters

Parameters	Description
CYBLE_LNS_CHAR_INDEX_T charIndex	The index of the Characteristic.
CYBLE_LNS_DESCR_INDEX_T descrIndex	The index of the Descriptor.
uint8 attrSize	The size of the Descriptor value attribute.
uint8 * attrValue	The pointer to the location where Characteristic Descriptor value data should be stored.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - Characteristic Descriptor value was read successfully
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameter failed.
- CYBLE_ERROR_GATT_DB_INVALID_ATTR_HANDLE - Characteristic is absent.

CyBle_LnssGetCharacteristicValue

Prototype

```
CYBLE_API_RESULT_T CyBle_LnssGetCharacteristicValue(CYBLE_LNS_CHAR_INDEX_T
charIndex, uint8 attrSize, uint8 * attrValue);
```

Description

Gets the value of the Characteristic, as identified by charIndex.

Parameters

Parameters	Description
CYBLE_LNS_CHAR_INDEX_T charIndex	The index of the service Characteristic.
uint8 attrSize	The size of the Characteristic value attribute.
uint8 * attrValue	The pointer to the location where Characteristic value data should be stored.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - Characteristic value was read successfully
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the CyBle_GattsWriteAttributeValue input parameter failed.
- CYBLE_ERROR_INVALID_OPERATION - Operation is invalid for this Characteristic.
- CYBLE_ERROR_GATT_DB_INVALID_ATTR_HANDLE - Characteristic is absent.

CyBle_LnssSendIndication

Prototype

```
CYBLE_API_RESULT_T CyBle_LnssSendIndication(CYBLE_CONN_HANDLE_T connHandle,
CYBLE_LNS_CHAR_INDEX_T charIndex, uint8 attrSize, uint8 * attrValue);
```

Description

Sends an indication of the specified Characteristic value, as identified by the charIndex.

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T	The connection handle which consist of the device ID and ATT

connHandle	connection ID.
CYBLE_LNS_CHAR_INDEX_T charIndex	The index of the service Characteristic.
uint8 attrSize	The size of the Characteristic value attribute.
uint8 * attrValue	The pointer to the Characteristic value data that should be sent to the client device.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The request handled successfully
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameter failed.
- CYBLE_ERROR_INVALID_OPERATION - Operation is invalid for this Characteristic
- CYBLE_ERROR_GATT_DB_INVALID_ATTR_HANDLE – Optional Characteristic is absent
- CYBLE_ERROR_INVALID_STATE - Connection with the client is not established
- CYBLE_ERROR_MEMORY_ALLOCATION_FAILED - Memory allocation failed
- CYBLE_ERROR_NTF_DISABLED - Notification is not enabled by the client
- CYBLE_ERROR_IND_DISABLED - Indication is disabled for this Characteristic

CyBle_LnssSendNotification

Prototype

```
CYBLE_API_RESULT_T CyBle_LnssSendNotification(CYBLE_CONN_HANDLE_T connHandle,
CYBLE_LNS_CHAR_INDEX_T charIndex, uint8 attrSize, uint8 * attrValue);
```

Description

Sends a notification of the specified Characteristic value, as identified by the charIndex.

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	The connection handle which consist of the device ID and ATT connection ID.
CYBLE_LNS_CHAR_INDEX_T charIndex	The index of the service Characteristic.

uint8 attrSize	The size of the Characteristic value attribute.
uint8 * attrValue	The pointer to the Characteristic value data that should be sent to the client device.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The request handled successfully
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameter failed
- CYBLE_ERROR_INVALID_OPERATION - Operation is invalid for this Characteristic
- CYBLE_ERROR_GATT_DB_INVALID_ATTR_HANDLE - Optional Characteristic is absent
- CYBLE_ERROR_INVALID_STATE - Connection with the client is not established
- CYBLE_ERROR_MEMORY_ALLOCATION_FAILED -Memory allocation failed
- CYBLE_ERROR_NTF_DISABLED - Notification is not enabled by the client

CyBle_LnssSetCharacteristicValue

Prototype

```
CYBLE_API_RESULT_T CyBle_LnssSetCharacteristicValue(CYBLE_LNS_CHAR_INDEX_T charIndex,
uint8 attrSize, uint8 * attrValue);
```

Description

Sets the value of the Characteristic, as identified by charIndex.

Parameters

Parameters	Description
CYBLE_LNS_CHAR_INDEX_T charIndex	The index of the service Characteristic.
uint8 attrSize	The size of the Characteristic value attribute.
uint8 * attrValue	The pointer to the Characteristic value data that should be sent to the server device.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The request handled successfully.
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameter failed.
- CYBLE_ERROR_GATT_DB_INVALID_ATTR_HANDLE - Optional Characteristic is absent

LNS Client Functions

APIs unique to LNS designs configured as a GATT Client role.

A letter 'c' is appended to the API name: CyBle_Lnsc

Functions

Function	Description
CyBle_LnscSetCharacteristicValue	This function is used to write the Characteristic (which is identified by charIndex) value attribute in the server. The Write Response just confirms the operation... more
CyBle_LnscGetCharacteristicValue	This function is used to read the Characteristic Value from a server, as identified by its charIndex The Read Response returns the Characteristic Value in... more
CyBle_LnscSetCharacteristicDescriptor	This function is used to write the Characteristic Value to the server, as identified by its charIndex.
CyBle_LnscGetCharacteristicDescriptor	Gets the Characteristic Descriptor of the specified Characteristic.

CyBle_LnscSetCharacteristicValue

Prototype

```
CYBLE_API_RESULT_T CyBle_LnscSetCharacteristicValue(CYBLE_CONN_HANDLE_T connHandle,
CYBLE_LNS_CHAR_INDEX_T charIndex, uint8 attrSize, uint8 * attrValue);
```

Description

This function is used to write the Characteristic (which is identified by charIndex) value attribute in the server.

The Write Response just confirms the operation success.

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	The connection handle.
CYBLE_LNS_CHAR_INDEX_T	The index of the service Characteristic.

charIndex	
uint8 attrSize	The size of the Characteristic value attribute.
uint8 * attrValue	The pointer to the Characteristic value data that should be sent to the server device.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The request was sent successfully
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameters failed
- CYBLE_ERROR_MEMORY_ALLOCATION_FAILED -Memory allocation failed
- CYBLE_ERROR_INVALID_STATE - Connection with the server is not established
- CYBLE_ERROR_GATT_DB_INVALID_ATTR_HANDLE - The peer device doesn't have the particular Characteristic
- CYBLE_ERROR_INVALID_OPERATION - Operation is invalid for this Characteristic

CyBle_LnscGetCharacteristicValue

Prototype

```
CYBLE_API_RESULT_T CyBle_LnscGetCharacteristicValue(CYBLE_CONN_HANDLE_T connHandle,
CYBLE_LNS_CHAR_INDEX_T charIndex);
```

Description

This function is used to read the Characteristic Value from a server, as identified by its charIndex

The Read Response returns the Characteristic Value in the Attribute Value parameter.

The Read Response only contains the Characteristic Value that is less than or equal to (MTU - 1) octets in length. If the Characteristic Value is greater than (MTU - 1) octets in length, the Read Long Characteristic Value procedure may be used if the rest of the Characteristic Value is required.

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	The connection handle.
CYBLE_LNS_CHAR_INDEX_T charIndex	The index of the service Characteristic.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The read request was sent successfully
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameters failed
- CYBLE_ERROR_GATT_DB_INVALID_ATTR_HANDLE - The peer device doesn't have the particular Characteristic
- CYBLE_ERROR_MEMORY_ALLOCATION_FAILED -Memory allocation failed
- CYBLE_ERROR_INVALID_STATE - Connection with the server is not established
- CYBLE_ERROR_INVALID_OPERATION - Operation is invalid for this Characteristic

CyBle_LnscSetCharacteristicDescriptor

Prototype

```
CYBLE_API_RESULT_T CyBle_LnscSetCharacteristicDescriptor(CYBLE_CONN_HANDLE_T
connHandle, CYBLE_LNS_CHAR_INDEX_T charIndex, CYBLE_LNS_DESCR_INDEX_T descrIndex, uint8
attrSize, uint8 * attrValue);
```

Description

This function is used to write the Characteristic Value to the server, as identified by its charIndex.

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	The connection handle.
CYBLE_LNS_CHAR_INDEX_T charIndex	The index of the service Characteristic.
CYBLE_LNS_DESCR_INDEX_T descrIndex	The index of the service Characteristic Descriptor.
uint8 attrSize	The size of the Characteristic Descriptor value attribute.
uint8 * attrValue	The pointer to the Characteristic Descriptor value data that should be sent to the server device.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The request was sent successfully

- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameters failed
- CYBLE_ERROR_INVALID_STATE - The state is not valid
- CYBLE_ERROR_MEMORY_ALLOCATION_FAILED -Memory allocation failed
- CYBLE_ERROR_GATT_DB_INVALID_ATTR_HANDLE - The peer device doesn't have the particular Characteristic
- CYBLE_ERROR_INVALID_OPERATION - This operation is not permitted on the specified attribute

CyBle_LnscGetCharacteristicDescriptor

Prototype

```
CYBLE_API_RESULT_T CyBle_LnscGetCharacteristicDescriptor(CYBLE_CONN_HANDLE_T
connHandle, CYBLE_LNS_CHAR_INDEX_T charIndex, CYBLE_LNS_DESCR_INDEX_T descrIndex);
```

Description

Gets the Characteristic Descriptor of the specified Characteristic.

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	The connection handle.
CYBLE_LNS_CHAR_INDEX_T charIndex	The index of the service Characteristic.
CYBLE_LNS_DESCR_INDEX_T descrIndex	The index of the service Characteristic Descriptor.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The request was sent successfully
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameters failed
- CYBLE_ERROR_INVALID_STATE - The state is not valid
- CYBLE_ERROR_MEMORY_ALLOCATION_FAILED -Memory allocation failed
- CYBLE_ERROR_GATT_DB_INVALID_ATTR_HANDLE - The peer device doesn't have the particular Descriptor
- CYBLE_ERROR_INVALID_OPERATION - This operation is not permitted on the specified attribute

LNS Definitions and Data Structures

Contains the LNS specific definitions and data structures used in the LNS APIs.

Enumerations

Enumeration	Description
CYBLE_LNS_CHAR_INDEX_T	LNS Service Characteristics indexes
CYBLE_LNS_DESCR_INDEX_T	LNS Service Characteristic Descriptors indexes

Structures

Structure	Description
CYBLE_LNS_CHAR_VALUE_T	LNS Characteristic Value parameter structure
CYBLE_LNS_DESCR_VALUE_T	LNS Characteristic Descriptor Value parameter structure
CYBLE_LNSC_CHAR_T	Location and Navigation Client Characteristic structure type
CYBLE_LNSC_T	Structure with discovered attributes information of Location and Navigation Service
CYBLE_LNSS_CHAR_T	Location and Navigation Server Characteristic structure type
CYBLE_LNSS_T	Structure with Location and Navigation Service attribute handles

CYBLE_LNS_CHAR_INDEX_T

Prototype

```
typedef enum {
  CYBLE_LNS_FT,
  CYBLE_LNS_LS,
  CYBLE_LNS_PQ,
  CYBLE_LNS_CP,
  CYBLE_LNS_NV,
  CYBLE_LNS_CHAR_COUNT
} CYBLE_LNS_CHAR_INDEX_T;
```

Description

LNS Service Characteristics indexes

Members

Members	Description
CYBLE_LNS_FT	Location and Navigation Feature Characteristic index
CYBLE_LNS_LS	Location and Speed Characteristic index

CYBLE_LNS_PQ	Position Quality Characteristic index
CYBLE_LNS_CP	Location and Navigation Control Point Characteristic index
CYBLE_LNS_NV	Navigation Characteristic index
CYBLE_LNS_CHAR_COUNT	Total count of LNS Characteristics

CYBLE_LNS_CHAR_VALUE_T

Prototype

```
typedef struct {
  CYBLE_CONN_HANDLE_T connHandle;
  CYBLE_LNS_CHAR_INDEX_T charIndex;
  CYBLE_GATT_VALUE_T * value;
} CYBLE_LNS_CHAR_VALUE_T;
```

Description

LNS Characteristic Value parameter structure

Members

Members	Description
CYBLE_CONN_HANDLE_T connHandle;	Peer device handle
CYBLE_LNS_CHAR_INDEX_T charIndex;	Index of service Characteristic
CYBLE_GATT_VALUE_T * value;	Characteristic value

CYBLE_LNS_DESCR_INDEX_T

Prototype

```
typedef enum {
  CYBLE_LNS_CCCD,
  CYBLE_LNS_DESCR_COUNT
} CYBLE_LNS_DESCR_INDEX_T;
```

Description

LNS Service Characteristic Descriptors indexes

Members

Members	Description
CYBLE_LNS_CCCD	Client Characteristic Configuration Descriptor index
CYBLE_LNS_DESCR_COUNT	Total count of LNS Descriptors

CYBLE_LNS_DESCR_VALUE_T

Prototype

```
typedef struct {
  CYBLE_CONN_HANDLE_T connHandle;
  CYBLE_LNS_CHAR_INDEX_T charIndex;
  CYBLE_LNS_DESCR_INDEX_T descrIndex;
  CYBLE_GATT_VALUE_T * value;
} CYBLE_LNS_DESCR_VALUE_T;
```

Description

LNS Characteristic Descriptor Value parameter structure

Members

Members	Description
CYBLE_CONN_HANDLE_T connHandle;	Peer device handle
CYBLE_LNS_CHAR_INDEX_T charIndex;	Index of service Characteristic
CYBLE_LNS_DESCR_INDEX_T descrIndex;	Index of service Characteristic Descriptor
CYBLE_GATT_VALUE_T * value;	Descriptor value

CYBLE_LNSC_CHAR_T

Prototype

```
typedef struct {
  uint8 properties;
  CYBLE_GATT_DB_ATTR_HANDLE_T valueHandle;
  CYBLE_GATT_DB_ATTR_HANDLE_T descrHandle[CYBLE_LNS_DESCR_COUNT];
  CYBLE_GATT_DB_ATTR_HANDLE_T endHandle;
} CYBLE_LNSC_CHAR_T;
```

Description

Location and Navigation Client Characteristic structure type

Members

Members	Description
uint8 properties;	Properties for value field
CYBLE_GATT_DB_ATTR_HANDLE_T valueHandle;	Handle of server database attribute value entry
CYBLE_GATT_DB_ATTR_HANDLE_T descrHandle[CYBLE_LNS_DESCR_COUNT];	Location and Navigation client char. Descriptor handle

CYBLE_GATT_DB_ATTR_HANDLE_T endHandle;	Characteristic End Handle

CYBLE_LNSC_T

Prototype

```
typedef struct {
  CYBLE_LNSC_CHAR_T charInfo[CYBLE_LNS_CHAR_COUNT];
} CYBLE_LNSC_T;
```

Description

Structure with discovered attributes information of Location and Navigation Service

Members

Members	Description
CYBLE_LNSC_CHAR_T charInfo[CYBLE_LNS_CHAR_COUNT];	Characteristics handle + properties array

CYBLE_LNSS_CHAR_T

Prototype

```
typedef struct {
  CYBLE_GATT_DB_ATTR_HANDLE_T charHandle;
  CYBLE_GATT_DB_ATTR_HANDLE_T descrHandle[CYBLE_LNS_DESCR_COUNT]; }
CYBLE_LNSS_CHAR_T;
```

Description

Location and Navigation Server Characteristic structure type

Members

Members	Description
CYBLE_GATT_DB_ATTR_HANDLE_T charHandle;	Handle of Characteristic value
CYBLE_GATT_DB_ATTR_HANDLE_T descrHandle[CYBLE_LNS_DESCR_COUNT];	Handle of Descriptor

CYBLE_LNSS_T

Prototype

```
typedef struct {
  CYBLE_GATT_DB_ATTR_HANDLE_T serviceHandle;
  CYBLE_LNSS_CHAR_T charInfo[CYBLE_LNS_CHAR_COUNT];
} CYBLE_LNSS_T;
```

Description

Structure with Location and Navigation Service attribute handles

Members

Members	Description
CYBLE_GATT_DB_ATTR_HANDLE_T serviceHandle;	Location and Navigation Service handle
CYBLE_LNSS_CHAR_T charInfo[CYBLE_LNS_CHAR_COUNT];	Location and Navigation Service Characteristics info array

Next DST Change Service (NDCS)

This Service enables a BLE device that has knowledge about the next occurrence of a DST change to expose this information to another Bluetooth device. The Service uses the "Time with DST" Characteristic and the functions exposed in this Service are used to interact with that Characteristic.

Depending on the chosen GATT role in the GUI, you may use a subset of the supported APIs.

The NDSC API names begin with CyBle_Ndsc. In addition to this, the APIs also append the GATT role initial letter in the API name.

NDCS Server and Client Function

These are APIs common to both GATT Client role and GATT Server role. You may use them in either roles.

No letter is appended to the API name: CyBle_Ndsc

Functions

Function	Description
CyBle_NdcsRegisterAttrCallback	Registers a callback function for Next DST Change Service specific attribute operations.

CyBle_NdcsRegisterAttrCallback

Prototype

```
void CyBle_NdcsRegisterAttrCallback(CYBLE_CALLBACK_T callbackFunc);
```

Description

Registers a callback function for Next DST Change Service specific attribute operations.

Parameters

Parameters	Description
CYBLE_CALLBACK_T callbackFunc	An application layer event callback function to receive events from the BLE Component. The definition of CYBLE_CALLBACK_T for NDCS is, typedef void (* CYBLE_CALLBACK_T) (uint32 eventCode, void *eventParam) eventCode indicates the event that triggered this callback. eventParam contains the parameters corresponding to the current event.

Returns

None.

NDCS Server Functions

APIs unique to NDSC designs configured as a GATT Server role.

A letter 's' is appended to the API name: CyBle_Ndscs

Functions

Function	Description
CyBle_NdcssGetCharacteristicValue	Gets a Characteristic value of the Next DST Change Service, which is identified by charIndex.
CyBle_NdcssSetCharacteristicValue	Sets Characteristic value of the Next DST Change Service, which is identified by charIndex in the local database.

CyBle_NdcssGetCharacteristicValue

Prototype

```
CYBLE_API_RESULT_T CyBle_NdcssGetCharacteristicValue(CYBLE_NDCS_CHAR_INDEX_T
charIndex, uint8 attrSize, uint8 * attrValue);
```

Description

Gets a Characteristic value of the Next DST Change Service, which is identified by charIndex.

Parameters

Parameters	Description
CYBLE_NDCS_CHAR_INDEX_T charIndex	the index of a service Characteristic of type CYBLE_NDCS_CHAR_INDEX_T.
uint8 attrSize	the size of the Characteristic value attribute.
uint8 * attrValue	the pointer to the location where Characteristic value data should be stored.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - the request is handled successfully;
- CYBLE_ERROR_INVALID_PARAMETER - validation of the input parameter failed.

CyBle_NdcssSetCharacteristicValue

Prototype

```
CYBLE_API_RESULT_T CyBle_NdcssSetCharacteristicValue(CYBLE_NDCS_CHAR_INDEX_T
charIndex, uint8 attrSize, uint8 * attrValue);
```

Description

Sets Characteristic value of the Next DST Change Service, which is identified by charIndex in the local database.

Parameters

Parameters	Description
CYBLE_NDCS_CHAR_INDEX_T charIndex	the index of a service Characteristic of type CYBLE_NDCS_CHAR_INDEX_T.
uint8 attrSize	the size of the Characteristic value attribute.
uint8 * attrValue	the pointer to the Characteristic value data that should be sent to the server device.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - the request is handled successfully;
- CYBLE_ERROR_INVALID_PARAMETER - validation of the input parameters failed.

NDCS Client Functions

APIs unique to NDSC designs configured as a GATT Client role.

A letter 'c' is appended to the API name: CyBle_Ndscc

Functions

Function	Description
CyBle_NdcscGetCharacteristicValue	Sends a request to peer device to set Characteristic value of the Next DST Change Service, which is identified by charIndex.

CyBle_NdcscGetCharacteristicValue

Prototype

```
CYBLE_API_RESULT_T CyBle_NdcscGetCharacteristicValue(CYBLE_CONN_HANDLE_T connHandle,
CYBLE_NDCS_CHAR_INDEX_T charIndex);
```

Description

Sends a request to peer device to set Characteristic value of the Next DST Change Service, which is identified by charIndex.

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	the connection handle.
CYBLE_NDCS_CHAR_INDEX_T charIndex	the index of a service Characteristic.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - the request was sent successfully.
- CYBLE_ERROR_INVALID_STATE - connection with the client is not established.
- CYBLE_ERROR_INVALID_PARAMETER - validation of the input parameters failed.
- CYBLE_ERROR_MEMORY_ALLOCATION_FAILED -Memory allocation failed.
- CYBLE_ERROR_INVALID_OPERATION - Operation is invalid for this Characteristic.

NDCS Definitions and Data Structures

Contains the NDSC specific definitions and data structures used in the NDSC APIs.

Enumerations

Enumeration	Description
CYBLE_NDCS_CHAR_INDEX_T	Characteristic indexes

Structures

Structure	Description
CYBLE_NDCS_CHAR_VALUE_T	Next DST Change Service Characteristic Value parameter structure
CYBLE_NDCSC_T	Structure with discovered attributes information of Next DST Change Service

CYBLE_NDCSS_T	Structure with Device Information Service atribute handles

CYBLE_NDCS_CHAR_INDEX_T

Prototype

```
typedef enum {
  CYBLE_NDCS_TIME_WITH_DST,
  CYBLE_NDCS_CHAR_COUNT
} CYBLE_NDCS_CHAR_INDEX_T;
```

Description

Characteristic indexes

Members

Members	Description
CYBLE_NDCS_TIME_WITH_DST	Time with DST Characteristic index
CYBLE_NDCS_CHAR_COUNT	Total count of NDCS Characteristics

CYBLE_NDCS_CHAR_VALUE_T

Prototype

```
typedef struct {
  CYBLE_CONN_HANDLE_T connHandle;
  CYBLE_NDCS_CHAR_INDEX_T charIndex;
  CYBLE_GATT_VALUE_T * value;
} CYBLE_NDCS_CHAR_VALUE_T;
```

Description

Next DST Change Service Characteristic Value parameter structure

Members

Members	Description
CYBLE_CONN_HANDLE_T connHandle;	Peer device handle
CYBLE_NDCS_CHAR_INDEX_T charIndex;	Index of Next DST Change Service Characteristic
CYBLE_GATT_VALUE_T * value;	Characteristic value

CYBLE_NDCSC_T

Prototype

```
typedef struct {
  CYBLE_SRVR_CHAR_INFO_T charInfo[CYBLE_NDCS_CHAR_COUNT]; } CYBLE_NDCSC_T;
```

Description

Structure with discovered attributes information of Next DST Change Service

Members

Members	Description
CYBLE_SRVR_CHAR_INFO_T charInfo[CYBLE_NDCS_CHAR_COUNT];	Characteristic handle + properties

CYBLE_NDCSS_T

Prototype

```
typedef struct {
  CYBLE_GATT_DB_ATTR_HANDLE_T serviceHandle;
  CYBLE_GATT_DB_ATTR_HANDLE_T timeWithDst;
} CYBLE_NDCSS_T;
```

Description

Structure with Device Information Service atribute handles

Members

Members	Description
CYBLE_GATT_DB_ATTR_HANDLE_T serviceHandle;	Handle of the Next DST Change Service
CYBLE_GATT_DB_ATTR_HANDLE_T timeWithDst;	Handle of the Time with DST Characteristic

Phone Alert Status Service (PASS)

The Phone Alert Status Service uses the Alert Status Characteristic and Ringer Setting Characteristic to expose the phone alert status and uses the Ringer Control Point Characteristic to control the phone's ringer into mute or enable.

Depending on the chosen GATT role in the GUI, you may use a subset of the supported APIs.

The PASS API names begin with CyBle_Pass. In addition to this, the APIs also append the GATT role initial letter in the API name.

PASS Server and Client Function

These are APIs common to both GATT Client role and GATT Server role. You may use them in either roles.

No letter is appended to the API name: CyBle_Pass

Functions

Function	Description
CyBle_PassRegisterAttrCallback	Registers a callback function for service specific attribute operations.

CyBle_PassRegisterAttrCallback

Prototype

```
void CyBle_PassRegisterAttrCallback(CYBLE_CALLBACK_T callbackFunc);
```

Description

Registers a callback function for service specific attribute operations.

Parameters

Parameters	Description
CYBLE_CALLBACK_T callbackFunc	An application layer event callback function to receive events from the BLE Component. The definition of CYBLE_CALLBACK_T for PASS is, typedef void (* CYBLE_CALLBACK_T) (uint32 eventCode, void *eventParam) eventCode indicates the event that triggered this callback. eventParam contains the parameters corresponding to the current event.

Returns

None.

PASS Server Functions

APIs unique to PASS designs configured as a GATT Server role.

A letter 's' is appended to the API name: CyBle_Passs

Functions

Function	Description
CyBle_PasssGetCharacteristicValue	Gets the value of a Characteristic which is identified by charIndex.
CyBle_PasssSetCharacteristicValue	Sets the value of a Characteristic which is identified by charIndex.

CyBle_PasssGetCharacteristicDescriptor	Gets a Characteristic Descriptor of a specified Characteristic of the service.
CyBle_PasssSendNotification	Sends a notification of the specified by the charIndex Characteristic value.

CyBle_PasssSetCharacteristicValue

Prototype

```
CYBLE_API_RESULT_T CyBle_PasssSetCharacteristicValue(CYBLE_PASS_CHAR_INDEX_T
charIndex, uint8 attrSize, uint8 * attrValue);
```

Description

Sets the value of a Characteristic which is identified by charIndex.

Parameters

Parameters	Description
CYBLE_PASS_CHAR_INDEX_T charIndex	the index of a service Characteristic.
uint8 attrSize	the size of the Characteristic value attribute.
uint8 * attrValue	the pointer to the Characteristic value data that should be sent to the server device.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The request handled successfully.
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameter failed.
- CYBLE_ERROR_GATT_DB_INVALID_ATTR_HANDLE - Optional Characteristic is absent

CyBle_PasssSendNotification

Prototype

```
CYBLE_API_RESULT_T CyBle_PasssSendNotification(CYBLE_CONN_HANDLE_T connHandle,
CYBLE_PASS_CHAR_INDEX_T charIndex, uint8 attrSize, uint8 * attrValue);
```

Description

Sends a notification of the specified by the charIndex Characteristic value.

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	the connection handle which consists of the device ID and ATT connection ID.
CYBLE_PASS_CHAR_INDEX_T charIndex	the index of a service Characteristic.
uint8 attrSize	the size of the Characteristic value attribute.
uint8 * attrValue	the pointer to the Characteristic value data that should be sent to the client device.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The request handled successfully
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameter failed
- CYBLE_ERROR_INVALID_OPERATION - This operation is not permitted
- CYBLE_ERROR_GATT_DB_INVALID_ATTR_HANDLE - Optional Characteristic is absent
- CYBLE_ERROR_INVALID_STATE - Connection with the client is not established
- CYBLE_ERROR_MEMORY_ALLOCATION_FAILED -Memory allocation failed
- CYBLE_ERROR_NTF_DISABLED - Notification is not enabled by the client

CyBle_PasssGetCharacteristicValue

Prototype

```
CYBLE_API_RESULT_T CyBle_PasssGetCharacteristicValue(CYBLE_PASS_CHAR_INDEX_T
charIndex, uint8 attrSize, uint8 * attrValue);
```

Description

Gets the value of a Characteristic which is identified by charIndex.

Parameters

Parameters	Description
CYBLE_PASS_CHAR_INDEX_T charIndex	the index of a service Characteristic.

uint8 attrSize	the size of the Characteristic value attribute.
uint8 * attrValue	the pointer to the location where Characteristic value data should be stored.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The request handled successfully
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameter failed
- CYBLE_ERROR_GATT_DB_INVALID_ATTR_HANDLE - Optional Descriptor is absent

CyBle_PasssGetCharacteristicDescriptor

Prototype

```
CYBLE_API_RESULT_T CyBle_PasssGetCharacteristicDescriptor(CYBLE_PASS_CHAR_INDEX_T
charIndex, CYBLE_PASS_DESCR_INDEX_T descrIndex, uint8 attrSize, uint8 * attrValue);
```

Description

Gets a Characteristic Descriptor of a specified Characteristic of the service.

Parameters

Parameters	Description
CYBLE_PASS_CHAR_INDEX_T charIndex	the index of the Characteristic.
CYBLE_PASS_DESCR_INDEX_T descrIndex	the index of the Descriptor.
uint8 attrSize	the size of the Descriptor value attribute.
uint8 * attrValue	the pointer to the Descriptor value data that should be stored to the GATT database.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The request handled successfully
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameter failed
- CYBLE_ERROR_GATT_DB_INVALID_ATTR_HANDLE - Optional Descriptor is absent

PASS Client Functions

APIs unique to PASS designs configured as a GATT Client role.

A letter 'c' is appended to the API name: CyBle_Passc

Functions

Function	Description
CyBle_PasscGetCharacteristicValue	This function is used to read the Characteristic Value from a server which is identified by the charIndex. The Read Response returns the Characteristic Value in... more
CyBle_PasscSetCharacteristicValue	This function is used to write the Characteristic (which is identified by charIndex) value attribute to the server. The Write Response just confirms the operation... more
CyBle_PasscGetCharacteristicDescriptor	Gets a Characteristic Descriptor of a specified Characteristic of the service.
CyBle_PasscSetCharacteristicDescriptor	This function is used to write the Characteristic Value to the server which is identified by the charIndex

CyBle_PasscSetCharacteristicValue

Prototype

```
CYBLE_API_RESULT_T CyBle_PasscSetCharacteristicValue(CYBLE_CONN_HANDLE_T connHandle,
CYBLE_PASS_CHAR_INDEX_T charIndex, uint8 attrSize, uint8 * attrValue);
```

Description

This function is used to write the Characteristic (which is identified by charIndex) value attribute to the server.

The Write Response just confirms the operation success.

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	the connection handle.
CYBLE_PASS_CHAR_INDEX_T charIndex	the index of a service Characteristic.
uint8 attrSize	the size of the Characteristic value attribute.
uint8 * attrValue	the pointer to the Characteristic value data that should be sent to the server device.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The request was sent successfully
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameters failed
- CYBLE_ERROR_MEMORY_ALLOCATION_FAILED -Memory allocation failed
- CYBLE_ERROR_INVALID_STATE - Connection with the server is not established
- CYBLE_ERROR_GATT_DB_INVALID_ATTR_HANDLE - The peer device doesn't have the
- particular Characteristic
- CYBLE_ERROR_INVALID_OPERATION - Operation is invalid for this Characteristic

CyBle_PasscSetCharacteristicDescriptor

Prototype

```
CYBLE_API_RESULT_T CyBle_PasscSetCharacteristicDescriptor(CYBLE_CONN_HANDLE_T
connHandle, CYBLE_PASS_CHAR_INDEX_T charIndex, CYBLE_PASS_DESCR_INDEX_T descrIndex,
uint8 attrSize, uint8 * attrValue);
```

Description

This function is used to write the Characteristic Value to the server which is identified by the charIndex

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	the connection handle.
CYBLE_PASS_CHAR_INDEX_T charIndex	the index of a service Characteristic.
CYBLE_PASS_DESCR_INDEX_T descrIndex	the index of a service Characteristic Descriptor.
uint8 attrSize	the size of the Characteristic Descriptor value attribute.
uint8 * attrValue	the pointer to the Characteristic Descriptor value data that should be sent to the server device.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The request was sent successfully
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameters failed
- CYBLE_ERROR_INVALID_STATE - The state is not valid
- CYBLE_ERROR_MEMORY_ALLOCATION_FAILED -Memory allocation failed
- CYBLE_ERROR_GATT_DB_INVALID_ATTR_HANDLE - The peer device doesn't have the
- particular Characteristic
- CYBLE_ERROR_INVALID_OPERATION - This operation is not permitted on the specified attribute

CyBle_PasscGetCharacteristicValue

Prototype

```
CYBLE_API_RESULT_T CyBle_PasscGetCharacteristicValue(CYBLE_CONN_HANDLE_T connHandle,
CYBLE_PASS_CHAR_INDEX_T charIndex);
```

Description

This function is used to read the Characteristic Value from a server which is identified by the charIndex

The Read Response returns the Characteristic Value in the Attribute Value parameter.

The Read Response only contains the Characteristic Value that is less than or equal to (MTU - 1) octets in length. If the Characteristic Value is greater than (MTU - 1) octets in length, the Read Long Characteristic Value procedure may be used if the rest of the Characteristic Value is required.

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	the connection handle.
CYBLE_PASS_CHAR_INDEX_T charIndex	the index of a service Characteristic.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The read request was sent successfully
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameters failed

- CYBLE_ERROR_GATT_DB_INVALID_ATTR_HANDLE - The peer device doesn't have the particular Characteristic
- CYBLE_ERROR_MEMORY_ALLOCATION_FAILED -Memory allocation failed
- CYBLE_ERROR_INVALID_STATE - Connection with the server is not established
- CYBLE_ERROR_INVALID_OPERATION - Operation is invalid for this Characteristic

CyBle_PasscGetCharacteristicDescriptor

Prototype

```
CYBLE_API_RESULT_T CyBle_PasscGetCharacteristicDescriptor(CYBLE_CONN_HANDLE_T
connHandle, CYBLE_PASS_CHAR_INDEX_T charIndex, CYBLE_PASS_DESCR_INDEX_T descrIndex);
```

Description

Gets a Characteristic Descriptor of a specified Characteristic of the service.

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	the connection handle.
CYBLE_PASS_CHAR_INDEX_T charIndex	the index of a service Characteristic.
CYBLE_PASS_DESCR_INDEX_T descrIndex	the index of a service Characteristic Descriptor.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The request was sent successfully
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameters failed
- CYBLE_ERROR_INVALID_STATE - The state is not valid
- CYBLE_ERROR_MEMORY_ALLOCATION_FAILED -Memory allocation failed
- CYBLE_ERROR_GATT_DB_INVALID_ATTR_HANDLE - The peer device doesn't have the particular Descriptor
- CYBLE_ERROR_INVALID_OPERATION - This operation is not permitted on the specified attribute

PASS Definitions and Data Structures

Contains the PASS specific definitions and data structures used in the PASS APIs.

Enumerations

Enumeration	Description
CYBLE_PASS_CHAR_INDEX_T	Service Characteristics indexes
CYBLE_PASS_CP_T	Ringer Control Point values
CYBLE_PASS_DESCR_INDEX_T	Service Characteristic Descriptors indexes
CYBLE_PASS_RS_T	Ringer Setting values

Structures

Structure	Description
CYBLE_PASS_CHAR_VALUE_T	Phone Alert Status Service Characteristic value parameter structure
CYBLE_PASS_DESCR_VALUE_T	Phone Alert Status Service Characteristic Descriptor value parameter structure
CYBLE_PASSC_CHAR_T	Phone Alert Status Client Server's Characteristic structure type
CYBLE_PASSC_T	Structure with discovered attributes information of Phone Alert Status Service
CYBLE_PASSS_CHAR_T	Structure with Phone Alert Status Service Characteristics and Descriptors attribute handles
CYBLE_PASSS_T	Structure with Phone Alert Status Service attribute handles

CYBLE_PASSS_T

Prototype

```
typedef struct {
  CYBLE_GATT_DB_ATTR_HANDLE_T serviceHandle;
  CYBLE_PASSS_CHAR_T charInfo[CYBLE_PASS_CHAR_COUNT];
} CYBLE_PASSS_T;
```

Description

Structure with Phone Alert Status Service attribute handles

Members

Members	Description
CYBLE_GATT_DB_ATTR_HANDLE_T serviceHandle;	Phone Alert Status Service handle
CYBLE_PASSS_CHAR_T charInfo[CYBLE_PASS_CHAR_COUNT];	Phone Alert Status Service Characteristics info array

CYBLE_PASSS_CHAR_T

Prototype

```
typedef struct {
  CYBLE_GATT_DB_ATTR_HANDLE_T charHandle;
  CYBLE_GATT_DB_ATTR_HANDLE_T descrHandle[CYBLE_PASS_DESCR_COUNT]; }
CYBLE_PASSS_CHAR_T;
```

Description

Structure with Phone Alert Status Service Characteristics and Descriptors attribute handles.

Members

Members	Description
CYBLE_GATT_DB_ATTR_HANDLE_T charHandle;	Handle of Characteristic value
CYBLE_GATT_DB_ATTR_HANDLE_T descrHandle[CYBLE_PASS_DESCR_COUNT];	Handle of Descriptor

CYBLE_PASSC_T

Prototype

```
typedef struct {
  CYBLE_PASSC_CHAR_T charInfo[CYBLE_PASS_CHAR_COUNT];
} CYBLE_PASSC_T;
```

Description

Structure with discovered attributes information of Phone Alert Status Service

Members

Members	Description
CYBLE_PASSC_CHAR_T charInfo[CYBLE_PASS_CHAR_COUNT];	Characteristics handle + properties array

CYBLE_PASSC_CHAR_T

Prototype

```
typedef struct {
  uint8 properties;
  CYBLE_GATT_DB_ATTR_HANDLE_T valueHandle;
  CYBLE_GATT_DB_ATTR_HANDLE_T descrHandle[CYBLE_PASS_DESCR_COUNT];
  CYBLE_GATT_DB_ATTR_HANDLE_T endHandle;
} CYBLE_PASSC_CHAR_T;
```

Description

Phone Alert Status Client Server's Characteristic structure type

Members

Members	Description
uint8 properties;	Properties for value field
CYBLE_GATT_DB_ATTR_HANDLE_T valueHandle;	Handle of server database attribute value entry
CYBLE_GATT_DB_ATTR_HANDLE_T descrHandle[CYBLE_PASS_DESCR_COUNT];	Phone Alert Status Client Characteristics Descriptors handles
CYBLE_GATT_DB_ATTR_HANDLE_T endHandle;	Characteristic End Handle

CYBLE_PASS_RS_T

Prototype

```
typedef enum {
  CYBLE_PASS_RS_SILENT,
  CYBLE_PASS_RS_NORMAL
} CYBLE_PASS_RS_T;
```

Description

Ringer Setting values

Members

Members	Description
CYBLE_PASS_RS_SILENT	Ringer Silent
CYBLE_PASS_RS_NORMAL	Ringer Normal

CYBLE_PASS_DESCR_VALUE_T

Prototype

```
typedef struct {
  CYBLE_CONN_HANDLE_T connHandle;
  CYBLE_PASS_CHAR_INDEX_T charIndex;
  CYBLE_PASS_DESCR_INDEX_T descrIndex;
  CYBLE_GATT_VALUE_T * value;
} CYBLE_PASS_DESCR_VALUE_T;
```

Description

Phone Alert Status Service Characteristic Descriptor value parameter structure.

Members

Members	Description
CYBLE_CONN_HANDLE_T connHandle;	Peer device handle
CYBLE_PASS_CHAR_INDEX_T charIndex;	Index of service Characteristic
CYBLE_PASS_DESCR_INDEX_T descrIndex;	Index of service Characteristic Descriptor
CYBLE_GATT_VALUE_T * value;	Descriptor value

CYBLE_PASS_DESCR_INDEX_T

Prototype

```
typedef enum {
  CYBLE_PASS_CCCD,
  CYBLE_PASS_DESCR_COUNT
} CYBLE_PASS_DESCR_INDEX_T;
```

Description

Service Characteristic Descriptors indexes

Members

Members	Description
CYBLE_PASS_CCCD	Client Characteristic Configuration Descriptor index
CYBLE_PASS_DESCR_COUNT	Total count of PASS Descriptors

CYBLE_PASS_CP_T

Prototype

```
typedef enum {
  CYBLE_PASS_CP_SILENT = 1,
  CYBLE_PASS_CP_MUTE,
  CYBLE_PASS_CP_CANCEL
} CYBLE_PASS_CP_T;
```

Description

Ringer Control Point values

Members

Members	Description
CYBLE_PASS_CP_SILENT = 1	Silent Mode

CYBLE_PASS_CP_MUTE	Mute Once
CYBLE_PASS_CP_CANCEL	Cancel Silent Mode

CYBLE_PASS_CHAR_VALUE_T

Prototype

```
typedef struct {
  CYBLE_CONN_HANDLE_T connHandle;
  CYBLE_PASS_CHAR_INDEX_T charIndex;
  CYBLE_GATT_VALUE_T * value;
} CYBLE_PASS_CHAR_VALUE_T;
```

Description

Phone Alert Status Service Characteristic value parameter structure.

Members

Members	Description
CYBLE_CONN_HANDLE_T connHandle;	Peer device handle
CYBLE_PASS_CHAR_INDEX_T charIndex;	Index of service Characteristic
CYBLE_GATT_VALUE_T * value;	Characteristic value

CYBLE_PASS_CHAR_INDEX_T

Prototype

```
typedef enum {
  CYBLE_PASS_AS,
  CYBLE_PASS_RS,
  CYBLE_PASS_CP,
  CYBLE_PASS_CHAR_COUNT
} CYBLE_PASS_CHAR_INDEX_T;
```

Description

Service Characteristics indexes

Members

Members	Description
CYBLE_PASS_AS	Alert Status Characteristic index
CYBLE_PASS_RS	Ringer Setting Characteristic index
CYBLE_PASS_CP	Ringer Control Point Characteristic index

CYBLE_PASS_CHAR_COUNT	Total count of PASS Characteristics

Running Speed and Cadence Service (RSCS)

The Running Speed and Cadence (RSC) Service exposes speed, cadence and other data related to fitness applications such as the stride length and the total distance the user has travelled while using the Running Speed and Cadence sensor (Server).

Depending on the chosen GATT role in the GUI, you may use a subset of the supported APIs.

The RSCS API names begin with CyBle_Rscs. In addition to this, the APIs also append the GATT role initial letter in the API name.

RSCS Server and Client Function

These are APIs common to both GATT Client role and GATT Server role. You may use them in either roles.

No letter is appended to the API name: CyBle_Rscs

Functions

Function	Description
CyBle_RscsRegisterAttrCallback	Registers a callback function for Running Speed and Cadence Service specific attribute operations.

CyBle_RscsRegisterAttrCallback

Prototype

```
void CyBle_RscsRegisterAttrCallback(CYBLE_CALLBACK_T callbackFunc);
```

Description

Registers a callback function for Running Speed and Cadence Service specific attribute operations.

Parameters

Parameters	Description
CYBLE_CALLBACK_T callbackFunc	An application layer event callback function to receive events from the BLE Component. The definition of CYBLE_CALLBACK_T for RSCS is, typedef void (* CYBLE_CALLBACK_T) (uint32 eventCode, void *eventParam) eventCode indicates the event that triggered this callback. eventParam contains the parameters corresponding to the current event.

Returns

None

RSCS Server Functions

APIs unique to RSCS designs configured as a GATT Server role.

A letter 's' is appended to the API name: CyBle_Rscss

Functions

Function	Description
CyBle_RscssSetCharacteristicValue	Sets the Characteristic value of the Running Speed and Cadence Service in the local GATT database. The Characteristic is identified by charIndex.
CyBle_RscssGetCharacteristicValue	Gets the Characteristic value of the Running Speed and Cadence Service from the GATT database. The Characteristic is identified by charIndex.
CyBle_RscssGetCharacteristicDescriptor	Gets the Characteristic Descriptor of a specified Characteristic of the Running Speed and Cadence Service from the GATT database.
CyBle_RscssSendNotification	Sends a notification with the Characteristic value to the Client device. This is specified by charIndex of the Running Speed and Cadence Service.
CyBle_RscssSendIndication	Sends an indication with a Characteristic value to the Client device. This is specified by charIndex of the Running Speed and Cadence Service.

CyBle_RscssSetCharacteristicValue

Prototype

```
CYBLE_API_RESULT_T CyBle_RscssSetCharacteristicValue(CYBLE_RSCS_CHAR_INDEX_T
charIndex, uint8 attrSize, uint8 * attrValue);
```

Description

Sets the Characteristic value of the Running Speed and Cadence Service in the local GATT database. The Characteristic is identified by charIndex.

Parameters

Parameters	Description
CYBLE_RSCS_CHAR_INDEX_T charIndex	The index of a service Characteristic. Valid values are, CYBLE_RSCS_RSC_FEATURE CYBLE_RSCS_SENSOR_LOCATION.
uint8 attrSize	The size of the Characteristic value attribute.
uint8 * attrValue	The pointer to the Characteristic value data that should be sent to the server device.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The request handled successfully
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameters failed
- CYBLE_ERROR_GATT_DB_INVALID_ATTR_HANDLE - Optional Characteristic is absent

CyBle_RscssGetCharacteristicValue

Prototype

```
CYBLE_API_RESULT_T CyBle_RscssGetCharacteristicValue(CYBLE_RSCS_CHAR_INDEX_T
charIndex, uint8 attrSize, uint8 * attrValue);
```

Description

Gets the Characteristic value of the Running Speed and Cadence Service from the GATT database. The Characteristic is identified by charIndex.

Parameters

Parameters	Description
CYBLE_RSCS_CHAR_INDEX_T charIndex	The index of a service Characteristic. Valid value is, CYBLE_RSCS_SC_CONTROL_POINT.
uint8 attrSize	The size of the Characteristic value attribute.
uint8 * attrValue	The pointer to the location where Characteristic value data should be stored.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The request handled successfully
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameter failed
- CYBLE_ERROR_GATT_DB_INVALID_ATTR_HANDLE - Peer device doesn't have a particular
- Characteristic

CyBle_RscssGetCharacteristicDescriptor

Prototype

```
CYBLE_API_RESULT_T CyBle_RscssGetCharacteristicDescriptor(CYBLE_RSCS_CHAR_INDEX_T
charIndex, CYBLE_RSCS_DESCR_INDEX_T descrIndex, uint8 attrSize, uint8 * attrValue);
```

Description

Gets the Characteristic Descriptor of a specified Characteristic of the Running Speed and Cadence Service from the GATT database.

Parameters

Parameters	Description
CYBLE_RSCS_CHAR_INDEX_T charIndex	The index of a service Characteristic. Valid values are, CYBLE_RSCS_RSC_MEASUREMENT CYBLE_RSCS_SC_CONTROL_POINT
CYBLE_RSCS_DESCR_INDEX_T descrIndex	The index of a service Characteristic Descriptor. Valid value is, CYBLE_RSCS_CCCD
uint8 attrSize	The size of the Characteristic Descriptor attribute.
uint8 * attrValue	The pointer to the location where Characteristic Descriptor value data should be stored.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The request handled successfully
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameter failed
- CYBLE_ERROR_GATT_DB_INVALID_ATTR_HANDLE - Peer device doesn't have a particular Descriptor

CyBle_RscssSendNotification

Prototype

```
CYBLE_API_RESULT_T CyBle_RscssSendNotification(CYBLE_CONN_HANDLE_T connHandle,
CYBLE_RSCS_CHAR_INDEX_T charIndex, uint8 attrSize, uint8 * attrValue);
```

Description

Sends a notification with the Characteristic value to the Client device. This is specified by charIndex of the Running Speed and Cadence Service.

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	The connection handle.
CYBLE_RSCS_CHAR_INDEX_T charIndex	The index of a service Characteristic. Valid value is, CYBLE_RSCS_RSC_MEASUREMENT
uint8 attrSize	The size of the Characteristic value attribute.
uint8 * attrValue	The pointer to the Characteristic value data that should be sent to the client device.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The request handled successfully
- CYBLE_ERROR_INVALID_PARAMETER - Validation of input parameter is failed
- CYBLE_ERROR_INVALID_OPERATION - Operation is invalid for this Characteristic
- CYBLE_ERROR_INVALID_STATE - Connection with the client is not established
- CYBLE_ERROR_NTF_DISABLED - Notification is not enabled by the client
- CYBLE_ERROR_MEMORY_ALLOCATION_FAILED -Memory allocation failed

CyBle_RscssSendIndication

Prototype

```
CYBLE_API_RESULT_T CyBle_RscssSendIndication(CYBLE_CONN_HANDLE_T connHandle,
CYBLE_RSCS_CHAR_INDEX_T charIndex, uint8 attrSize, uint8 * attrValue);
```

Description

Sends an indication with a Characteristic value to the Client device. This is specified by charIndex of the Running Speed and Cadence Service.

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	The connection handle.
CYBLE_RSCS_CHAR_INDEX_T charIndex	The index of a service Characteristic.
uint8 attrSize	The size of the Characteristic value attribute.

uint8 * attrValue	The pointer to the Characteristic value data that should be sent to the client device.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The request handled successfully
- CYBLE_ERROR_INVALID_PARAMETER - Validation of input parameter is failed
- CYBLE_ERROR_INVALID_OPERATION - Operation is invalid for this Characteristic
- CYBLE_ERROR_INVALID_STATE - Connection with the client is not established
- CYBLE_ERROR_IND_DISABLED - Indication is not enabled by the client
- CYBLE_ERROR_MEMORY_ALLOCATION_FAILED -Memory allocation failed
- CYBLE_ERROR_GATT_DB_INVALID_ATTR_HANDLE - Peer device doesn't have a particular Characteristic

RSCS Client Functions

APIs unique to RSCS designs configured as a GATT Client role.

A letter 'c' is appended to the API name: CyBle_Rscsc

Functions

Function	Description
CyBle_RscscSetCharacteristicValue	Sends a request to the peer device to get the Characteristic Descriptor of the specified Characteristic of the Running Speed and Cadence Service.
CyBle_RscscGetCharacteristicValue	Sends a request to the peer device to set the Characteristic value of the Running Speed and Cadence Service.
CyBle_RscscSetCharacteristicDescriptor	Sends a request to the peer device to get the Characteristic Descriptor of the specified Characteristic of the Running Speed and Cadence Service.
CyBle_RscscGetCharacteristicDescriptor	Sends a request to the peer device to get Characteristic Descriptor of the specified Characteristic of the Running Speed and Cadence Service.

CyBle_RscscSetCharacteristicValue

Prototype

```
CYBLE_API_RESULT_T CyBle_RscscSetCharacteristicValue(CYBLE_CONN_HANDLE_T connHandle,
CYBLE_RSCS_CHAR_INDEX_T charIndex, uint8 attrSize, uint8 * attrValue);
```

Description

Sends a request to the peer device to get the Characteristic Descriptor of the specified Characteristic of the Running Speed and Cadence Service.

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	The connection handle.
CYBLE_RSCS_CHAR_INDEX_T charIndex	The index of a service Characteristic.
uint8 attrSize	Size of the Characteristic value attribute.
uint8 * attrValue	Pointer to the Characteristic value data that should be sent to the server device.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The request was sent successfully
- CYBLE_ERROR_INVALID_STATE - Connection with the client is not established
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameters failed
- CYBLE_ERROR_MEMORY_ALLOCATION_FAILED -Memory allocation failed
- CYBLE_ERROR_INVALID_OPERATION - Operation is invalid for this Characteristic.
- CYBLE_ERROR_GATT_DB_INVALID_ATTR_HANDLE - Peer device doesn't have a particular Characteristic

CyBle_RscscGetCharacteristicValue

Prototype

```
CYBLE_API_RESULT_T CyBle_RscscGetCharacteristicValue(CYBLE_CONN_HANDLE_T connHandle,
CYBLE_RSCS_CHAR_INDEX_T charIndex);
```

Description

Sends a request to the peer device to set the Characteristic value of the Running Speed and Cadence Service.

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	The connection handle.
CYBLE_RSCS_CHAR_INDEX_T charIndex	The index of the service Characteristic.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The request was sent successfully
- CYBLE_ERROR_INVALID_STATE - Connection with the client is not established
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameters failed
- CYBLE_ERROR_MEMORY_ALLOCATION_FAILED -Memory allocation failed
- CYBLE_ERROR_INVALID_OPERATION - Operation is invalid for this Characteristic.
- CYBLE_ERROR_GATT_DB_INVALID_ATTR_HANDLE - Peer device doesn't have a particular Characteristic

CyBle_RscscSetCharacteristicDescriptor

Prototype

```
CYBLE_API_RESULT_T CyBle_RscscSetCharacteristicDescriptor(CYBLE_CONN_HANDLE_T
connHandle, CYBLE_RSCS_CHAR_INDEX_T charIndex, CYBLE_RSCS_DESCR_INDEX_T descrIndex,
uint8 attrSize, uint8 * attrValue);
```

Description

Sends a request to the peer device to get the Characteristic Descriptor of the specified Characteristic of the Running Speed and Cadence Service.

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	The connection handle.
CYBLE_RSCS_CHAR_INDEX_T charIndex	The index of a RSCS Characteristic.
CYBLE_RSCS_DESCR_INDEX_T descrIndex	The index of a RSCS Characteristic Descriptor.
uint8 attrSize	The size of the Characteristic Descriptor attribute.

uint8 * attrValue	The pointer to the Characteristic Descriptor value data that should be sent to the server device.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - the request was sent successfully
- CYBLE_ERROR_INVALID_STATE - connection with the client is not established
- CYBLE_ERROR_INVALID_PARAMETER - validation of the input parameters failed
- CYBLE_ERROR_MEMORY_ALLOCATION_FAILED -Memory allocation failed.
- CYBLE_ERROR_INVALID_OPERATION - Operation is invalid for this Characteristic
- CYBLE_ERROR_GATT_DB_INVALID_ATTR_HANDLE - Peer device doesn't have a particular Descriptor

CyBle_RscscGetCharacteristicDescriptor

Prototype

```
CYBLE_API_RESULT_T CyBle_RscscGetCharacteristicDescriptor(CYBLE_CONN_HANDLE_T
connHandle, CYBLE_RSCS_CHAR_INDEX_T charIndex, uint8 descrIndex);
```

Description

Sends a request to the peer device to get Characteristic Descriptor of the specified Characteristic of the Running Speed and Cadence Service.

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	The connection handle.
CYBLE_RSCS_CHAR_INDEX_T charIndex	The index of a Service Characteristic.
uint8 descrIndex	The index of a Service Characteristic Descriptor.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The request was sent successfully
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameters failed

- CYBLE_ERROR_MEMORY_ALLOCATION_FAILED -Memory allocation failed
- CYBLE_ERROR_INVALID_STATE - Connection with the client is not established
- CYBLE_ERROR_INVALID_OPERATION - Cannot process a request to send PDU due to invalid operation performed by the application
- CYBLE_ERROR_GATT_DB_INVALID_ATTR_HANDLE - Peer device doesn't have a particular Descriptor

RSCS Definitions and Data Structures

RSCS Definitions and Data Structures

Contains the RSCS specific definitions and data structures used in the RSCS APIs.

Enumerations

Enumeration	Description
CYBLE_RSCS_CHAR_INDEX_T	RSCS Characteristic indexes
CYBLE_RSCS_DESCR_INDEX_T	RSCS Characteristic Descriptors indexes

Structures

Structure	Description
CYBLE_RSCS_CHAR_VALUE_T	Running Speed and Cadence Service Characteristic Value parameter structure
CYBLE_RSCS_DESCR_VALUE_T	Running Speed and Cadence Service Characteristic Descriptor Value parameter structure
CYBLE_RSCSC_T	Structure with discovered attributes information of Running Speed and Cadence Service
CYBLE_RSCSS_CHAR_T	RSCS Characteristic with Descriptors
CYBLE_RSCSS_T	Structure with Running Speed and Cadence Service attribute handles
CYBLE_SRVR_FULL_CHAR_INFO_T	Service Full Characteristic information type
CYBLE_RSCSC_SRVR_FULL_CHAR_INFO_T	RSCS Service Full Characteristic information type

CYBLE_RSCS_CHAR_INDEX_T

Prototype

```
typedef enum {
  CYBLE_RSCS_RSC_MEASUREMENT,
    CYBLE_RSCS_RSC_FEATURE,
    CYBLE_RSCS_SENSOR_LOCATION,
    CYBLE_RSCS_SC_CONTROL_POINT,
    CYBLE_RSCS_CHAR_COUNT
  } CYBLE_RSCS_CHAR_INDEX_T;
```

Description

RSCS Characteristic indexes

Members

Members	Description
CYBLE_RSCS_RSC_MEASUREMENT	RSC Measurement Characteristic index
CYBLE_RSCS_RSC_FEATURE	RSC Feature Characteristic index
CYBLE_RSCS_SENSOR_LOCATION	Sensor Location Characteristic index
CYBLE_RSCS_SC_CONTROL_POINT	SC Control Point Characteristic index
CYBLE_RSCS_CHAR_COUNT	Total count of RSCS Characteristics

CYBLE_RSCS_CHAR_VALUE_T

Prototype

```
  typedef struct {
    CYBLE_CONN_HANDLE_T connHandle;
    CYBLE_RSCS_CHAR_INDEX_T charIndex;
    CYBLE_GATT_VALUE_T * value;
  } CYBLE_RSCS_CHAR_VALUE_T;
```

Description

Running Speed and Cadence Service Characteristic Value parameter structure

Members

Members	Description
CYBLE_CONN_HANDLE_T connHandle;	Peer device handle
CYBLE_RSCS_CHAR_INDEX_T charIndex;	Index of Running Speed and Cadence Service Characteristic
CYBLE_GATT_VALUE_T * value;	Characteristic value

CYBLE_RSCS_DESCR_INDEX_T

Prototype

```
  typedef enum {
    CYBLE_RSCS_CCCD,
    CYBLE_RSCS_DESCR_COUNT
} CYBLE_RSCS_DESCR_INDEX_T;
```

Description

RSCS Characteristic Descriptors indexes

Members

Members	Description
CYBLE_RSCS_CCCD	Client Characteristic Configuration Descriptor index
CYBLE_RSCS_DESCR_COUNT	Total count of Descriptors

CYBLE_RSCS_DESCR_VALUE_T

Prototype

```
typedef struct {
  CYBLE_CONN_HANDLE_T connHandle;
  CYBLE_RSCS_CHAR_INDEX_T charIndex;
  CYBLE_RSCS_DESCR_INDEX_T descrIndex;
  CYBLE_GATT_VALUE_T * value;
} CYBLE_RSCS_DESCR_VALUE_T;
```

Description

Running Speed and Cadence Service Characteristic Descriptor Value parameter structure

Members

Members	Description
CYBLE_CONN_HANDLE_T connHandle;	Connection handle
CYBLE_RSCS_CHAR_INDEX_T charIndex;	Characteristic index of the Service
CYBLE_RSCS_DESCR_INDEX_T descrIndex;	Characteristic index Descriptor the Service
CYBLE_GATT_VALUE_T * value;	Pointer to value of the Service Characteristic Descriptor

CYBLE_RSCSC_T

Prototype

```
typedef struct {
  CYBLE_RSCSC_SRVR_FULL_CHAR_INFO_T Characteristics[CYBLE_RSCS_CHAR_COUNT]; }
CYBLE_RSCSC_T;
```

Description

Structure with discovered attributes information of Running Speed and Cadence Service

Members

Members	Description
CYBLE_RSCSC_SRVR_FULL_CHAR_INFO_T Characteristics[CYBLE_RSCS_CHAR_COUNT];	Characteristics handles array

CYBLE_RSCSS_CHAR_T

Prototype

```
typedef struct {
  CYBLE_GATT_DB_ATTR_HANDLE_T charHandle;
  CYBLE_GATT_DB_ATTR_HANDLE_T descrHandle[CYBLE_RSCS_DESCR_COUNT]; }
CYBLE_RSCSS_CHAR_T;
```

Description

RSCS Characteristic with Descriptors

Members

Members	Description
CYBLE_GATT_DB_ATTR_HANDLE_T charHandle;	Handle of the Characteristic value
CYBLE_GATT_DB_ATTR_HANDLE_T descrHandle[CYBLE_RSCS_DESCR_COUNT];	Handle of the Descriptor

CYBLE_RSCSS_T

Prototype

```
typedef struct {
  CYBLE_GATT_DB_ATTR_HANDLE_T serviceHandle;
  CYBLE_RSCSS_CHAR_T charInfo[CYBLE_RSCS_CHAR_COUNT];
} CYBLE_RSCSS_T;
```

Description

Structure with Running Speed and Cadence Service attribute handles

Members

Members	Description
CYBLE_GATT_DB_ATTR_HANDLE_T serviceHandle;	Running Speed and Cadence Service handle

CYBLE_RSCSS_CHAR_T charInfo[CYBLE_RSCS_CHAR_COUNT];	Array of Running Speed and Cadence Service Characteristics + Descriptors handles

CYBLE_SRVR_FULL_CHAR_INFO_T

Prototype

```
typedef struct {
  CYBLE_SRVR_CHAR_INFO_T charInfo;
  CYBLE_GATT_DB_ATTR_HANDLE_T endHandle;
  CYBLE_GATT_DB_ATTR_HANDLE_T descriptors[CYBLE_ANS_DESCR_COUNT]; }
CYBLE_SRVR_FULL_CHAR_INFO_T;
```

Description

Service Full Characteristic information type

Members

Members	Description
CYBLE_SRVR_CHAR_INFO_T charInfo;	Characteristic handle + properties
CYBLE_GATT_DB_ATTR_HANDLE_T endHandle;	End handle of Characteristic
CYBLE_GATT_DB_ATTR_HANDLE_T Descriptors[CYBLE_ANS_DESCR_COUNT];	Characteristic Descriptors handles

CYBLE_RSCSC_SRVR_FULL_CHAR_INFO_T

Prototype

```
typedef struct {
  CYBLE_SRVR_CHAR_INFO_T charInfo;
  CYBLE_GATT_DB_ATTR_HANDLE_T descriptors[CYBLE_RSCS_DESCR_COUNT];
  CYBLE_GATT_DB_ATTR_HANDLE_T endHandle;
} CYBLE_RSCSC_SRVR_FULL_CHAR_INFO_T;
```

Description

RSCS Service Full Characteristic information type

Members

Members	Description
CYBLE_SRVR_CHAR_INFO_T charInfo;	Characteristic handle + properties
CYBLE_GATT_DB_ATTR_HANDLE_T	Characteristic Descriptors handles

Descriptors[CYBLE_RSCS_DESCR_COUNT];	handle
CYBLE_GATT_DB_ATTR_HANDLE_T endHandle;	End handle of Characteristic

Reference Time Update Service (RTUS)

This Service enables a Bluetooth device that can update the system time using the reference time such as a GPS receiver to expose a control point and expose the accuracy (drift) of the local system time compared to the reference time source.

Depending on the chosen GATT role in the GUI, you may use a subset of the supported APIs.

The RTUS API names begin with CyBle_Rtus. In addition to this, the APIs also append the GATT role initial letter in the API name.

RTUS Server and Client Function

These are APIs common to both GATT Client role and GATT Server role. You may use them in either roles.

No letter is appended to the API name: CyBle_Rtus

Functions

Function	Description
CyBle_RtusRegisterAttrCallback	Registers a callback function for Reference Time Update Service specific attribute operations.

CyBle_RtusRegisterAttrCallback

Prototype

```
void CyBle_RtusRegisterAttrCallback(CYBLE_CALLBACK_T callbackFunc);
```

Description

Registers a callback function for Reference Time Update Service specific attribute operations.

Parameters

Parameters	Description
CYBLE_CALLBACK_T callbackFunc	An application layer event callback function to receive events from the BLE Component. The definition of CYBLE_CALLBACK_T for RTUS is, typedef void (* CYBLE_CALLBACK_T) (uint32 eventCode, void *eventParam) eventCode indicates the event that triggered this callback. eventParam contains the parameters corresponding to the current event.

Returns

None.

RTUS Server Functions

APIs unique to RTUS designs configured as a GATT Server role.

A letter 's' is appended to the API name: CyBle_Rtuss

Functions

Function	Description
CyBle_RtussGetCharacteristicValue	Gets a Characteristic value of the Reference Time Update Service, which is identified by charIndex.
CyBle_RtussSetCharacteristicValue	Sets Characteristic value of the Reference Time Update Service, which is identified by charIndex in the local database.

CyBle_RtussGetCharacteristicValue

Prototype

```
CYBLE_API_RESULT_T CyBle_RtussGetCharacteristicValue(CYBLE_RTUS_CHAR_INDEX_T
charIndex, uint8 attrSize, uint8 * attrValue);
```

Description

Gets a Characteristic value of the Reference Time Update Service, which is identified by charIndex.

Parameters

Parameters	Description
CYBLE_RTUS_CHAR_INDEX_T charIndex	the index of a service Characteristic of type CYBLE_RTUS_CHAR_INDEX_T. Valid value is CYBLE_RTUS_SC_CONTROL_POINT.
uint8 attrSize	the size of the Characteristic value attribute.
uint8 * attrValue	the pointer to the location where Characteristic value data should be stored.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - the request is handled successfully;
- CYBLE_ERROR_INVALID_PARAMETER - validation of the input parameter failed.

CyBle_RtussSetCharacteristicValue

Prototype

```
CYBLE_API_RESULT_T CyBle_RtussSetCharacteristicValue(CYBLE_RTUS_CHAR_INDEX_T
charIndex, uint8 attrSize, uint8 * attrValue);
```

Description

Sets Characteristic value of the Reference Time Update Service, which is identified by charIndex in the local database.

Parameters

Parameters	Description
CYBLE_RTUS_CHAR_INDEX_T charIndex	the index of a service Characteristic of type CYBLE_RTUS_CHAR_INDEX_T.
uint8 attrSize	the size of the Characteristic value attribute.
uint8 * attrValue	the pointer to the Characteristic value data that should be sent to the server device.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - the request is handled successfully
- CYBLE_ERROR_INVALID_PARAMETER - validation of the input parameters failed

RTUS Client Functions

APIs unique to RTUS designs configured as a GATT Client role.

A letter 'c' is appended to the API name: CyBle_Rtusc

Functions

Function	Description
CyBle_RtuscSetCharacteristicValue	Sends a request to a peer device to get Characteristic Descriptor of specified Characteristic of the Reference Time Update Service.
CyBle_RtuscGetCharacteristicValue	Sends a request to a peer device to set Characteristic value of the Reference Time Update Service, which is identified by charIndex.

CyBle_RtuscSetCharacteristicValue

Prototype

```
CYBLE_API_RESULT_T CyBle_RtuscSetCharacteristicValue(CYBLE_CONN_HANDLE_T connHandle,
CYBLE_RTUS_CHAR_INDEX_T charIndex, uint8 attrSize, uint8 * attrValue);
```

Description

Sends a request to a peer device to get Characteristic Descriptor of specified Characteristic of the Reference Time Update Service.

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	the connection handle.
CYBLE_RTUS_CHAR_INDEX_T charIndex	the index of a service Characteristic.
uint8 attrSize	size of the Characteristic value attribute.
uint8 * attrValue	pointer to the Characteristic value data that should be sent to the server device.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - the request was sent successfully;
- CYBLE_ERROR_INVALID_STATE - connection with the client is not established;
- CYBLE_ERROR_INVALID_PARAMETER - validation of the input parameters failed.
- CYBLE_ERROR_MEMORY_ALLOCATION_FAILED -Memory allocation failed.
- CYBLE_ERROR_INVALID_OPERATION - Operation is invalid for this Characteristic.

CyBle_RtuscGetCharacteristicValue

Prototype

```
CYBLE_API_RESULT_T CyBle_RtuscGetCharacteristicValue(CYBLE_CONN_HANDLE_T connHandle,
CYBLE_RTUS_CHAR_INDEX_T charIndex);
```

Description

Sends a request to a peer device to set Characteristic value of the Reference Time Update Service, which is identified by charIndex.

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	the connection handle.
CYBLE_RTUS_CHAR_INDEX_T charIndex	the index of a service Characteristic.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - the request was sent successfully;
- CYBLE_ERROR_INVALID_STATE - connection with the client is not established.
- CYBLE_ERROR_INVALID_PARAMETER - validation of the input parameters failed.

RTUS Definitions and Data Structures

Contains the RTUS specific definitions and data structures used in the RTUS APIs.

Enumerations

Enumeration	Description
CYBLE_RTUS_CHAR_INDEX_T	Characteristic indexes

Structures

Structure	Description
CYBLE_RTUS_CHAR_VALUE_T	Reference Time Update Service Characteristic Value parameter structure
CYBLE_RTUSC_T	Structure with discovered attributes information of Reference Time Update Service
CYBLE_RTUSS_T	Structure with Reference Time Update Service atribute handles

CYBLE_RTUS_CHAR_INDEX_T

Prototype

```
typedef enum {
  CYBLE_RTUS_TIME_UPDATE_CONTROL_POINT,
  CYBLE_RTUS_TIME_UPDATE_STATE,
  CYBLE_RTUS_CHAR_COUNT
} CYBLE_RTUS_CHAR_INDEX_T;
```

Description

Characteristic indexes

Members

Members	Description
CYBLE_RTUS_TIME_UPDATE_CONTROL_POINT	Time Update Control Point Characteristic index
CYBLE_RTUS_TIME_UPDATE_STATE	Time Update State Characteristic index
CYBLE_RTUS_CHAR_COUNT	Total count of RTUS Characteristics

CYBLE_RTUS_CHAR_VALUE_T

Prototype

```
typedef struct {
  CYBLE_CONN_HANDLE_T connHandle;
  CYBLE_RTUS_CHAR_INDEX_T charIndex;
  CYBLE_GATT_VALUE_T * value;
} CYBLE_RTUS_CHAR_VALUE_T;
```

Description

Reference Time Update Service Characteristic Value parameter structure

Members

Members	Description
CYBLE_CONN_HANDLE_T connHandle;	Peer device handle
CYBLE_RTUS_CHAR_INDEX_T charIndex;	Index of Reference Time Update Service Characteristic
CYBLE_GATT_VALUE_T * value;	Characteristic value

CYBLE_RTUSC_T

Prototype

```
typedef struct {
  CYBLE_SRVR_CHAR_INFO_T charInfo[CYBLE_RTUS_CHAR_COUNT]; } CYBLE_RTUSC_T;
```

Description

Structure with discovered attributes information of Reference Time Update Service

Members

Members	Description
CYBLE_SRVR_CHAR_INFO_T charInfo[CYBLE_RTUS_CHAR_COUNT];	Characteristic handle + properties

CYBLE_RTUSS_T

Prototype

```
typedef struct {
  CYBLE_GATT_DB_ATTR_HANDLE_T serviceHandle;
  CYBLE_GATT_DB_ATTR_HANDLE_T timeUpdateCpHandle;
  CYBLE_GATT_DB_ATTR_HANDLE_T timeUpdateStateHandle;
} CYBLE_RTUSS_T;
```

Description

Structure with Reference Time Update Service atribute handles

Members

Members	Description
CYBLE_GATT_DB_ATTR_HANDLE_T serviceHandle;	Handle of the Reference Time Update Service
CYBLE_GATT_DB_ATTR_HANDLE_T timeUpdateCpHandle;	Handle of the Time Update Control Point Characteristic
CYBLE_GATT_DB_ATTR_HANDLE_T timeUpdateStateHandle;	Handle of the Time Update State Characteristic

Scan Parameters Service (ScPS)

The Scan Parameters Service enables a Server device to expose a Characteristic for the GATT Client to write its scan interval and scan window on the Server device, and enables a Server to request a refresh of the GATT Client scan interval and scan window.

Depending on the chosen GATT role in the GUI, you may use a subset of the supported APIs.

The ScPS API names begin with CyBle_Scps. In addition to this, the APIs also append the GATT role initial letter in the API name.

ScPS Server and Client Function

These are APIs common to both GATT Client role and GATT Server role. You may use them in either roles.

No letter is appended to the API name: CyBle_Scps

Functions

Function	Description
CyBle_ScpsRegisterAttrCallback	Registers a callback function for service specific attribute operations.

CyBle_ScpsRegisterAttrCallback

Prototype

```
void CyBle_ScpsRegisterAttrCallback(CYBLE_CALLBACK_T callbackFunc);
```

Description

Registers a callback function for service specific attribute operations.

Parameters

Parameters	Description
CYBLE_CALLBACK_T callbackFunc	An application layer event callback function to receive events from the BLE Component. The definition of CYBLE_CALLBACK_T for ScPS is, typedef void (* CYBLE_CALLBACK_T) (uint32 eventCode, void *eventParam) • eventCode indicates the event that triggered this callback. • eventParam contains the parameters corresponding to the current event.

Returns

None

ScPS Server Functions

APIs unique to ScPS designs configured as a GATT Server role.

A letter 's' is appended to the API name: CyBle_Scpss

Functions

Function	Description
CyBle_ScpssSetCharacteristicValue	Sets a Characteristic value of the Scan Parameters service, which is identified by charIndex.
CyBle_ScpssGetCharacteristicValue	Gets a Characteristic value of the Scan Parameters service, which is identified by charIndex.
CyBle_ScpssGetCharacteristicDescriptor	Gets a Characteristic Descriptor of the specified Characteristic of the Scan Parameters service.
CyBle_ScpssSendNotification	This function notifies the client that the server requires the Scan Interval Window Characteristic to be written with the latest values upon notification. The CYBLE_EVT_SCPSC_NOTIFICATION... more

CyBle_ScpssSetCharacteristicValue

Prototype

```
CYBLE_API_RESULT_T CyBle_ScpssSetCharacteristicValue(CYBLE_SCPS_CHAR_INDEX_T
charIndex, uint8 attrSize, uint8 * attrValue);
```

Description

Sets a Characteristic value of the Scan Parameters service, which is identified by charIndex.

Parameters

Parameters	Description
CYBLE_SCPS_CHAR_INDEX_T charIndex	The index of the service Characteristic. • CYBLE_SCPS_SCAN_INT_WIN - The Scan Interval Window Characteristic index • CYBLE_SCPS_SCAN_REFRESH - The Scan Refresh Characteristic index
uint8 attrSize	The size of the Characteristic value attribute.
uint8 * attrValue	The pointer to the Characteristic value data that should be sent to the server device.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The request handled successfully
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameter failed
- CYBLE_ERROR_GATT_DB_INVALID_ATTR_HANDLE - An optional Characteristic is absent

CyBle_ScpssGetCharacteristicValue

Prototype

```
CYBLE_API_RESULT_T CyBle_ScpssGetCharacteristicValue(CYBLE_SCPS_CHAR_INDEX_T
charIndex, uint8 attrSize, uint8 * attrValue);
```

Description

Gets a Characteristic value of the Scan Parameters service, which is identified by charIndex.

Parameters

Parameters	Description
CYBLE_SCPS_CHAR_INDEX_T charIndex	The index of the service Characteristic. • CYBLE_SCPS_SCAN_INT_WIN - The Scan Interval Window Characteristic index • CYBLE_SCPS_SCAN_REFRESH - The Scan Refresh Characteristic index
uint8 attrSize	The size of the Characteristic value attribute.
uint8 * attrValue	The pointer to the location where Characteristic value data should be stored.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The request handled successfully
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameter failed
- CYBLE_ERROR_GATT_DB_INVALID_ATTR_HANDLE - Optional Characteristic is absent

CyBle_ScpssGetCharacteristicDescriptor

Prototype

```
CYBLE_API_RESULT_T CyBle_ScpssGetCharacteristicDescriptor(CYBLE_SCPS_CHAR_INDEX_T
charIndex, CYBLE_SCPS_DESCR_INDEX_T descrIndex, uint8 attrSize, uint8 * attrValue);
```

Description

Gets a Characteristic Descriptor of the specified Characteristic of the Scan Parameters service.

Parameters

Parameters	Description
CYBLE_SCPS_CHAR_INDEX_T charIndex	The index of the Characteristic. • CYBLE_SCPS_SCAN_REFRESH - The Scan Refresh Characteristic index
CYBLE_SCPS_DESCR_INDEX_T descrIndex	The index of the Descriptor. • CYBLE_SCPS_SCAN_REFRESH_CCCD - The Client Characteristic Configuration Descriptor index of the Scan Refresh Characteristic
uint8 attrSize	The size of the Characteristic value attribute.
uint8 * attrValue	The pointer to the location where the Characteristic Descriptor value data

	should be stored.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The request handled successfully
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameter failed
- CYBLE_ERROR_GATT_DB_INVALID_ATTR_HANDLE - Optional Descriptor is absent

CyBle_ScpssSendNotification

Prototype

```
CYBLE_API_RESULT_T CyBle_ScpssSendNotification(CYBLE_CONN_HANDLE_T connHandle,
CYBLE_SCPS_CHAR_INDEX_T charIndex, uint8 attrSize, uint8 * attrValue);
```

Description

This function notifies the client that the server requires the Scan Interval Window Characteristic to be written with the latest values upon notification.

The CYBLE_EVT_SCPSC_NOTIFICATION event is received by the peer device, on invoking this function.

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	The connection handle
CYBLE_SCPS_CHAR_INDEX_T charIndex	The index of the Characteristic. • CYBLE_SCPS_SCAN_REFRESH - The Scan Refresh Characteristic index
uint8 attrSize	The size of the Characteristic value attribute.
uint8 * attrValue	The pointer to the Characteristic value data that should be sent to the Client device.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The request handled successfully
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameter failed

- CYBLE_ERROR_INVALID_OPERATION - This operation is not permitted
- CYBLE_ERROR_INVALID_STATE - Connection with the client is not established
- CYBLE_ERROR_MEMORY_ALLOCATION_FAILED -Memory allocation failed.
- CYBLE_ERROR_NTF_DISABLED - Notification is not enabled by the client.

ScPS Client Functions

APIs unique to ScPS designs configured as a GATT Client role.

A letter 'c' is appended to the API name: CyBle_Scpsc

Functions

Function	Description
CyBle_ScpscSetCharacteristicValue	Sets a Characteristic value of the Scan Parameters Service, which is identified by charIndex. This function call can result in generation of the following events... more
CyBle_ScpscSetCharacteristicDescriptor	Sets Characteristic Descriptor of specified Characteristic of the Scan Parameters Service. This function call can result in generation of the following events based on the... more
CyBle_ScpscGetCharacteristicDescriptor	Gets Characteristic Descriptor of specified Characteristic of the Scan Parameters Service. This function call can result in generation of the following events based on the... more

CyBle_ScpscSetCharacteristicValue

Prototype

```
CYBLE_API_RESULT_T CyBle_ScpscSetCharacteristicValue(CYBLE_CONN_HANDLE_T connHandle,
CYBLE_SCPS_CHAR_INDEX_T charIndex, uint8 attrSize, uint8 * attrValue);
```

Description

Sets a Characteristic value of the Scan Parameters Service, which is identified by charIndex.

This function call can result in generation of the following events based on the response from the server device:

- CYBLE_EVT_GATTC_WRITE_RSP
- CYBLE_EVT_GATTC_ERROR_RSP

The CYBLE_EVT_SCPSS_SCAN_INT_WIN_CHAR_WRITE event is received by the peer device on invoking this function.

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	The connection handle.
CYBLE_SCPS_CHAR_INDEX_T charIndex	The index of the service Characteristic.
uint8 attrSize	The size of the Characteristic value attribute.
uint8 * attrValue	The pointer to the Characteristic value data that should be sent to the server device.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The request was sent successfully
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameters failed
- CYBLE_ERROR_MEMORY_ALLOCATION_FAILED -Memory allocation failed
- CYBLE_ERROR_GATT_DB_INVALID_ATTR_HANDLE - The peer device doesn't have the particular Characteristic
- CYBLE_ERROR_INVALID_OPERATION - Operation is invalid for this Characteristic

CyBle_ScpscSetCharacteristicDescriptor

Prototype

```
CYBLE_API_RESULT_T CyBle_ScpscSetCharacteristicDescriptor(CYBLE_CONN_HANDLE_T
connHandle, CYBLE_SCPS_CHAR_INDEX_T charIndex, CYBLE_SCPS_DESCR_INDEX_T descrIndex,
uint8 attrSize, uint8 * attrValue);
```

Description

Sets Characteristic Descriptor of specified Characteristic of the Scan Parameters Service.

This function call can result in generation of the following events based on the response from the server device:

- CYBLE_EVT_SCPSC_WRITE_DESCR_RESPONSE
- CYBLE_EVT_GATTC_ERROR_RSP

Following events can be received by the peer device on invoking this function:

- CYBLE_EVT_SCPSS_NOTIFICATION_ENABLED
- CYBLE_EVT_SCPSS_NOTIFICATION_DISABLED

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	The connection handle.
CYBLE_SCPS_CHAR_INDEX_T charIndex	The index of the service Characteristic.
CYBLE_SCPS_DESCR_INDEX_T descrIndex	The index of the service Characteristic Descriptor.
uint8 attrSize	The size of the Descriptor value attribute.
uint8 * attrValue	The pointer to the Characteristic Descriptor value data that should be sent to the server device.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The request was sent successfully
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameters failed
- CYBLE_ERROR_INVALID_STATE - The state is not valid
- CYBLE_ERROR_MEMORY_ALLOCATION_FAILED -Memory allocation failed
- CYBLE_ERROR_GATT_DB_INVALID_ATTR_HANDLE - The peer device doesn't have the particular Characteristic
- CYBLE_ERROR_INVALID_OPERATION - This operation is not permitted on the specified attribute

CyBle_ScpscGetCharacteristicDescriptor

Prototype

```
CYBLE_API_RESULT_T CyBle_ScpscGetCharacteristicDescriptor(CYBLE_CONN_HANDLE_T
connHandle, CYBLE_SCPS_CHAR_INDEX_T charIndex, CYBLE_SCPS_DESCR_INDEX_T descrIndex);
```

Description

Gets Characteristic Descriptor of specified Characteristic of the Scan Parameters Service.

This function call can result in generation of the following events based on the response from the server device:

- CYBLE_EVT_SCPSC_READ_DESCR_RESPONSE
- CYBLE_EVT_GATTC_ERROR_RSP

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	The connection handle.
CYBLE_SCPS_CHAR_INDEX_T charIndex	The index of a Service Characteristic.
CYBLE_SCPS_DESCR_INDEX_T descrIndex	The index of a Service Characteristic Descriptor.

Returns

- CYBLE_ERROR_OK - The request was sent successfully
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameters failed
- CYBLE_ERROR_INVALID_STATE - The state is not valid
- CYBLE_ERROR_MEMORY_ALLOCATION_FAILED -Memory allocation failed
- CYBLE_ERROR_GATT_DB_INVALID_ATTR_HANDLE - The peer device doesn't have the particular Descriptor
- CYBLE_ERROR_INVALID_OPERATION - This operation is not permitted on the specified attribute

ScPS Definitions and Data Structures

Contains the ScPS specific definitions and data structures used in the ScPS APIs.

Enumerations

Enumeration	Description
CYBLE_SCPS_CHAR_INDEX_T	ScPS Characteristic indexes
CYBLE_SCPS_DESCR_INDEX_T	ScPS Characteristic Descriptors indexes

Structures

Structure	Description
CYBLE_SCPS_CHAR_VALUE_T	Scan Parameters Service Characteristic Value parameter structure
CYBLE_SCPS_DESCR_VALUE_T	Scan Parameters Service Characteristic Descriptor Value parameter structure
CYBLE_SCPSC_T	Structure with discovered attributes information of Scan Parameters Service
CYBLE_SCPSS_T	Structure with Scan Parameters Service attribute handles

CYBLE_SCPS_CHAR_INDEX_T

Prototype

```
typedef enum {
  CYBLE_SCPS_SCAN_INT_WIN,
  CYBLE_SCPS_SCAN_REFRESH,
  CYBLE_SCPS_CHAR_COUNT
} CYBLE_SCPS_CHAR_INDEX_T;
```

Description

ScPS Characteristic indexes

Members

Members	Description
CYBLE_SCPS_SCAN_INT_WIN	Scan Interval Window Characteristic index
CYBLE_SCPS_SCAN_REFRESH	Scan Refresh Characteristic index
CYBLE_SCPS_CHAR_COUNT	Total count of Characteristics

CYBLE_SCPS_CHAR_VALUE_T

Prototype

```
typedef struct {
  CYBLE_CONN_HANDLE_T connHandle;
  CYBLE_SCPS_CHAR_INDEX_T charIndex;
  CYBLE_GATT_VALUE_T * value;
} CYBLE_SCPS_CHAR_VALUE_T;
```

Description

Scan Parameters Service Characteristic Value parameter structure

Members

Members	Description
CYBLE_CONN_HANDLE_T connHandle;	Peer device handle
CYBLE_SCPS_CHAR_INDEX_T charIndex;	Index of service Characteristic
CYBLE_GATT_VALUE_T * value;	Characteristic value

CYBLE_SCPS_DESCR_INDEX_T

Prototype

```
typedef enum {
  CYBLE_SCPS_SCAN_REFRESH_CCCD,
  CYBLE_SCPS_DESCR_COUNT
} CYBLE_SCPS_DESCR_INDEX_T;
```

Description

ScPS Characteristic Descriptors indexes

Members

Members	Description
CYBLE_SCPS_SCAN_REFRESH_CCCD	Client Characteristic Configuration Descriptor index
CYBLE_SCPS_DESCR_COUNT	Total count of Descriptors

CYBLE_SCPS_DESCR_VALUE_T

Prototype

```
typedef struct {
  CYBLE_CONN_HANDLE_T connHandle;
  CYBLE_SCPS_CHAR_INDEX_T charIndex;
  CYBLE_SCPS_DESCR_INDEX_T descrIndex;
  CYBLE_GATT_VALUE_T * value;
} CYBLE_SCPS_DESCR_VALUE_T;
```

Description

Scan Parameters Service Characteristic Descriptor Value parameter structure

Members

Members	Description
CYBLE_CONN_HANDLE_T connHandle;	Peer device handle
CYBLE_SCPS_CHAR_INDEX_T charIndex;	Index of service Characteristic
CYBLE_SCPS_DESCR_INDEX_T descrIndex;	Index of service Characteristic Descriptor
CYBLE_GATT_VALUE_T * value;	Descriptor value

CYBLE_SCPSC_T

Prototype

```
typedef struct {
  CYBLE_CONN_HANDLE_T connHandle;
  CYBLE_SRVR_CHAR_INFO_T intervalWindowChar;
  CYBLE_SRVR_CHAR_INFO_T refreshChar;
  CYBLE_GATT_DB_ATTR_HANDLE_T refreshCccdHandle;
} CYBLE_SCPSC_T;
```

Description

Structure with discovered attributes information of Scan Parameters Service

Members

Members	Description
CYBLE_CONN_HANDLE_T connHandle;	Peer device handle
CYBLE_SRVR_CHAR_INFO_T intervalWindowChar;	Handle + properties of Scan Interval Window Characteristic
CYBLE_SRVR_CHAR_INFO_T refreshChar;	Handle + properties of Scan Refresh Characteristic
CYBLE_GATT_DB_ATTR_HANDLE_T refreshCccdHandle;	Handle of Client Characteristic Configuration Descriptor

CYBLE_SCPSS_T

Prototype

```
typedef struct {
  CYBLE_GATT_DB_ATTR_HANDLE_T serviceHandle;
  CYBLE_GATT_DB_ATTR_HANDLE_T intervalWindowCharHandle;
  CYBLE_GATT_DB_ATTR_HANDLE_T refreshCharHandle;
  CYBLE_GATT_DB_ATTR_HANDLE_T refreshCccdHandle;
} CYBLE_SCPSS_T;
```

Description

Structure with Scan Parameters Service attribute handles

Members

Members	Description
CYBLE_GATT_DB_ATTR_HANDLE_T serviceHandle;	Scan Parameter Service handle
CYBLE_GATT_DB_ATTR_HANDLE_T intervalWindowCharHandle;	Handle of Scan Interval Window Characteristic
CYBLE_GATT_DB_ATTR_HANDLE_T refreshCharHandle;	Handle of Scan Refresh Characteristic
CYBLE_GATT_DB_ATTR_HANDLE_T refreshCccdHandle;	Handle of Client Characteristic Configuration Descriptor

TX Power Service (TPS)

The Tx Power Service uses the Tx Power Level Characteristic to expose the current transmit power level of a device when in a connection.

Depending on the chosen GATT role in the GUI, you may use a subset of the supported APIs.

The TPS API names begin with CyBle_Tps. In addition to this, the APIs also append the GATT role initial letter in the API name.

TPS Server and Client Function

These are APIs common to both GATT Client role and GATT Server role. You may use them in either roles.

No letter is appended to the API name: CyBle_Tps

Functions

Function	Description
CyBle_TpsRegisterAttrCallback	Registers a callback function for service specific attribute operations.

CyBle_TpsRegisterAttrCallback

Prototype

```
void CyBle_TpsRegisterAttrCallback(CYBLE_CALLBACK_T callbackFunc);
```

Description

Registers a callback function for service specific attribute operations.

Parameters

Parameters	Description
CYBLE_CALLBACK_T callbackFunc	An application layer event callback function to receive events from the BLE Component. The definition of CYBLE_CALLBACK_T for TPS is, typedef void (* CYBLE_CALLBACK_T) (uint32 eventCode, void *eventParam) eventCode indicates the event that triggered this callback. eventParam contains the parameters corresponding to the current event.

Returns

None

Side Effects

The *eventParams in the callback function should not be used by the application once the callback function execution is finished. Otherwise this data may become corrupted.

TPS Server Functions

APIs unique to TPS designs configured as a GATT Server role.

A letter 's' is appended to the API name: CyBle_Tpss

Functions

Function	Description
CyBle_TpssSetCharacteristicValue	Sets Characteristic value of the Tx Power Service, which is identified by charIndex.

CyBle_TpssGetCharacteristicValue	Gets Characteristic value of the Tx Power Service, which is identified by charIndex.
CyBle_TpssGetCharacteristicDescriptor	Gets Characteristic Descriptor of specified Characteristic of the Tx Power Service.
CyBle_TpssSendNotification	Sends a notification with the Characteristic value, as specified by charIndex, to the Client device. The CYBLE_EVT_TPSC_NOTIFICATION event is received by the peer device on... more

CyBle_TpssSetCharacteristicValue

Prototype

```
CYBLE_API_RESULT_T CyBle_TpssSetCharacteristicValue(CYBLE_TPS_CHAR_INDEX_T charIndex,
uint8 attrSize, int8 * attrValue);
```

Description

Sets Characteristic value of the Tx Power Service, which is identified by charIndex.

Parameters

Parameters	Description
CYBLE_TPS_CHAR_INDEX_T charIndex	The index of the service Characteristic.
uint8 attrSize	The size of the Characteristic value attribute.
int8 * attrValue	The pointer to the Characteristic value data that should be sent to the server device.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The Characteristic value was read successfully.
- CYBLE_ERROR_INVALID_PARAMETER - Validation of input parameters failed.

CyBle_TpssGetCharacteristicValue

Prototype

```
CYBLE_API_RESULT_T CyBle_TpssGetCharacteristicValue(CYBLE_TPS_CHAR_INDEX_T charIndex,
uint8 attrSize, int8 * attrValue);
```

Description

Gets Characteristic value of the Tx Power Service, which is identified by charIndex.

Parameters

Parameters	Description
CYBLE_TPS_CHAR_INDEX_T charIndex	The index of the Tx Power Characteristic.
uint8 attrSize	The size of the Tx Power Characteristic value attribute.
int8 * attrValue	The pointer to the location where Tx Power Characteristic value data should be stored.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - Characteristic value was read successfully
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameters failed

CyBle_TpssGetCharacteristicDescriptor

Prototype

```
CYBLE_API_RESULT_T CyBle_TpssGetCharacteristicDescriptor(CYBLE_TPS_CHAR_INDEX_T
charIndex, CYBLE_TPS_CHAR_DESCRIPTORS_T descrIndex, uint8 attrSize, uint8 *
attrValue);
```

Description

Gets Characteristic Descriptor of specified Characteristic of the Tx Power Service.

Parameters

Parameters	Description
CYBLE_TPS_CHAR_INDEX_T charIndex	The index of the Characteristic.
CYBLE_TPS_CHAR_DESCRIPTORS_T descrIndex	The index of the Descriptor.
uint8 attrSize	The size of the Characteristic value attribute.
uint8 * attrValue	The pointer to the location where Characteristic Descriptor value data should be stored.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - Characteristic Descriptor value was read successfully
- CYBLE_ERROR_INVALID_PARAMETER - Validation of input parameters failed
- CYBLE_ERROR_GATT_DB_INVALID_ATTR_HANDLE - Optional Descriptor is absent

CyBle_TpssSendNotification

Prototype

```
CYBLE_API_RESULT_T CyBle_TpssSendNotification(CYBLE_CONN_HANDLE_T connHandle,
CYBLE_TPS_CHAR_INDEX_T charIndex, uint8 attrSize, int8 * attrValue);
```

Description

Sends a notification with the Characteristic value, as specified by charIndex, to the Client device.

The CYBLE_EVT_TPSC_NOTIFICATION event is received by the peer device on invoking this function.

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	The connection handle.
CYBLE_TPS_CHAR_INDEX_T charIndex	The index of the service Characteristic.
uint8 attrSize	The size of the Characteristic value attribute.
int8 * attrValue	The pointer to the Characteristic value data that should be sent to the Client device.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The request handled successfully.
- CYBLE_ERROR_INVALID_PARAMETER - Validation of input parameter failed.
- CYBLE_ERROR_INVALID_OPERATION - Operation is invalid for this Characteristic.
- CYBLE_ERROR_INVALID_STATE - Connection with client is not established.
- CYBLE_ERROR_NTF_DISABLED - Notification is not enabled by the client.
- CYBLE_ERROR_MEMORY_ALLOCATION_FAILED -Memory allocation failed.

TPS Client Functions

APIs unique to TPS designs configured as a GATT Client role.

A letter 'c' is appended to the API name: CyBle_Tpsc

Functions

Functions	Description
CyBle_TpscGetCharacteristicValue	Gets the Characteristic value of the TX Power service, which is identified by charIndex. This function call can result in generation of the following events... more
CyBle_TpscSetCharacteristicDescriptor	Sets a Characteristic Descriptor value of the Tx Power Service. This function call can result in generation of the following events based on the response... more
CyBle_TpscGetCharacteristicDescriptor	Gets a Characteristic Descriptor of the Tx Power Service. This function call can result in generation of the following events based on the response from... more

CyBle_TpscGetCharacteristicValue

Prototype

```
CYBLE_API_RESULT_T CyBle_TpscGetCharacteristicValue(CYBLE_CONN_HANDLE_T connHandle,
CYBLE_TPS_CHAR_INDEX_T charIndex);
```

Description

Gets the Characteristic value of the TX Power service, which is identified by charIndex.

This function call can result in generation of the following events based on the response from the server device:

- CYBLE_EVT_TPSC_READ_CHAR_RESPONSE
- CYBLE_EVT_GATTC_ERROR_RSP

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	The connection handle.
CYBLE_TPS_CHAR_INDEX_T charIndex	The index of the Characteristic.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - Request was sent successfully
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameters failed
- CYBLE_ERROR_INVALID_STATE - Connection with the server is not established
- CYBLE_ERROR_MEMORY_ALLOCATION_FAILED -Memory allocation failed
- CYBLE_ERROR_INVALID_OPERATION - Operation is invalid for this Characteristic

CyBle_TpscSetCharacteristicDescriptor

Prototype

```
CYBLE_API_RESULT_T CyBle_TpscSetCharacteristicDescriptor(CYBLE_CONN_HANDLE_T
connHandle, CYBLE_TPS_CHAR_INDEX_T charIndex, CYBLE_TPS_CHAR_DESCRIPTORS_T descrIndex,
uint8 attrSize, uint8 * attrValue);
```

Description

Sets a Characteristic Descriptor value of the Tx Power Service.

This function call can result in generation of the following events based on the response from the server device:

- CYBLE_EVT_TPSC_WRITE_DESCR_RESPONSE
- CYBLE_EVT_GATTC_ERROR_RSP

Following events can be received by the peer device, on invoking this function:

- CYBLE_EVT_TPSS_NOTIFICATION_ENABLED
- CYBLE_EVT_TPSS_NOTIFICATION_DISABLED

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	The connection handle.
CYBLE_TPS_CHAR_INDEX_T charIndex	The index of the Characteristic
CYBLE_TPS_CHAR_DESCRIPTORS_T descrIndex	The index of the TX Power Service Characteristic Descriptor.
uint8 attrSize	The size of the Characteristic Descriptor attribute.
uint8 * attrValue	The pointer to the Characteristic Descriptor value data that should be sent to the server device.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - The request was sent successfully
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameters failed
- CYBLE_ERROR_INVALID_STATE - Connection with the server is not established
- CYBLE_ERROR_INVALID_OPERATION - This operation is not permitted on the specified attribute.
- CYBLE_ERROR_MEMORY_ALLOCATION_FAILED -Memory allocation failed

CyBle_TpscGetCharacteristicDescriptor

Prototype

```
CYBLE_API_RESULT_T CyBle_TpscGetCharacteristicDescriptor(CYBLE_CONN_HANDLE_T
connHandle, CYBLE_TPS_CHAR_INDEX_T charIndex, CYBLE_TPS_CHAR_DESCRIPTORS_T descrIndex);
```

Description

Gets a Characteristic Descriptor of the Tx Power Service.

This function call can result in generation of the following events based on the response from the server device:

- CYBLE_EVT_TPSC_READ_DESCR_RESPONSE
- CYBLE_EVT_GATTC_ERROR_RSP

Parameters

Parameters	Description
CYBLE_CONN_HANDLE_T connHandle	The connection handle.
CYBLE_TPS_CHAR_INDEX_T charIndex	The index of the Characteristic.
CYBLE_TPS_CHAR_DESCRIPTORS_T descrIndex	The index of the Characteristic Descriptor.

Returns

Return value is of type CYBLE_API_RESULT_T.

- CYBLE_ERROR_OK - Request was sent successfully.
- CYBLE_ERROR_INVALID_PARAMETER - Validation of the input parameters failed.
- CYBLE_ERROR_INVALID_STATE - The Component is in invalid state for current operation.
- CYBLE_ERROR_MEMORY_ALLOCATION_FAILED -Memory allocation failed.
- CYBLE_ERROR_INVALID_OPERATION - Cannot process request to send PDU due to invalid operation performed by the application.

TPS Definitions and Data Structures

Contains the TPS specific definitions and data structures used in the TPS APIs.

Enumerations

Enumeration	Description
CYBLE_TPS_CHAR_DESCRIPTORS_T	TPS Characteristic Descriptors indexes

CYBLE_TPS_CHAR_INDEX_T	TPS Characteristic indexes

Structures

Structure	Description
CYBLE_TPS_CHAR_VALUE_T	Tx Power Service Characteristic Value parameter structure
CYBLE_TPS_DESCR_VALUE_T	Tx Power Service Characteristic Descriptor Value parameter structure
CYBLE_TPSC_T	Structure with discovered attributes information of Tx Power Service
CYBLE_TPSS_T	Structure with Tx Power Service attribute handles

CYBLE_TPS_CHAR_DESCRIPTORS_T

Prototype

```
typedef enum {
  CYBLE_TPS_CCCD,
  CYBLE_TPS_DESCR_COUNT
} CYBLE_TPS_CHAR_DESCRIPTORS_T;
```

Description

TPS Characteristic Descriptors indexes

Members

Members	Description
CYBLE_TPS_CCCD	Tx Power Level Client Characteristic configuration Descriptor index
CYBLE_TPS_DESCR_COUNT	Total count of Tx Power Service Characteristic Descriptors

CYBLE_TPS_CHAR_INDEX_T

Prototype

```
typedef enum {
  CYBLE_TPS_TX_POWER_LEVEL,
  CYBLE_TPS_CHAR_COUNT
} CYBLE_TPS_CHAR_INDEX_T;
```

Description

TPS Characteristic indexes

Members

Members	Description
CYBLE_TPS_TX_POWER_LEVEL	Tx Power Level Characteristic index
CYBLE_TPS_CHAR_COUNT	Total count of Characteristics

CYBLE_TPS_CHAR_VALUE_T

Prototype

```
typedef struct {
  CYBLE_CONN_HANDLE_T connHandle;
  CYBLE_TPS_CHAR_INDEX_T charIndex;
  CYBLE_GATT_VALUE_T * value;
} CYBLE_TPS_CHAR_VALUE_T;
```

Description

Tx Power Service Characteristic Value parameter structure

Members

Members	Description
CYBLE_CONN_HANDLE_T connHandle;	Connection handle
CYBLE_TPS_CHAR_INDEX_T charIndex;	Characteristic index of Tx Power Service
CYBLE_GATT_VALUE_T * value;	Pointer to value of Tx Power Service Characteristic

CYBLE_TPS_DESCR_VALUE_T

Prototype

```
typedef struct {
  CYBLE_CONN_HANDLE_T connHandle;
  CYBLE_TPS_CHAR_INDEX_T charIndex;
  CYBLE_TPS_CHAR_DESCRIPTORS_T descrIndex;
  CYBLE_GATT_VALUE_T * value;
} CYBLE_TPS_DESCR_VALUE_T;
```

Description

Tx Power Service Characteristic Descriptor Value parameter structure

Members

Members	Description
CYBLE_CONN_HANDLE_T connHandle;	Connection handle
CYBLE_TPS_CHAR_INDEX_T charIndex;	Characteristic index of Tx Power Service
CYBLE_TPS_CHAR_DESCRIPTORS_T descrIndex;	Characteristic index Descriptor of Tx Power Service
CYBLE_GATT_VALUE_T * value;	Pointer to value of Tx Power Service Characteristic

CYBLE_TPSC_T

Prototype

```
typedef struct {
  CYBLE_SRVR_CHAR_INFO_T txPowerLevelChar;
  CYBLE_GATT_DB_ATTR_HANDLE_T txPowerLevelCccdHandle;
} CYBLE_TPSC_T;
```

Description

Structure with discovered attributes information of Tx Power Service

Members

Members	Description
CYBLE_SRVR_CHAR_INFO_T txPowerLevelChar;	Tx Power Level Characteristic handle
CYBLE_GATT_DB_ATTR_HANDLE_T txPowerLevelCccdHandle;	Tx Power Level Client Characteristic Configuration Descriptor handle

CYBLE_TPSS_T

Prototype

```
typedef struct {
  CYBLE_GATT_DB_ATTR_HANDLE_T serviceHandle;
  CYBLE_GATT_DB_ATTR_HANDLE_T txPowerLevelCharHandle;
  CYBLE_GATT_DB_ATTR_HANDLE_T txPowerLevelCccdHandle;
} CYBLE_TPSS_T;
```

Description

Structure with Tx Power Service attribute handles

Members

Members	Description
CYBLE_GATT_DB_ATTR_HANDLE_T serviceHandle;	Tx Power Service handle
CYBLE_GATT_DB_ATTR_HANDLE_T txPowerLevelCharHandle;	Tx Power Level Characteristic handle
CYBLE_GATT_DB_ATTR_HANDLE_T txPowerLevelCccdHandle;	Tx Power Level Client Characteristic Configuration Descriptor handle

Custom Service

This section contains the CYBLE_CUSTOMS_INFO_T and CYBLE_CUSTOMS_T structs used for Custom Serivces.

Structures

Structure	Description
CYBLE_CUSTOMS_INFO_T	Below are the indexes and handles of the defined Custom Services and their Characteristics
CYBLE_CUSTOMS_T	Structure with Custom Service attribute handles.

CYBLE_CUSTOMS_INFO_T

Prototype

```
typedef struct {
  CYBLE_GATT_DB_ATTR_HANDLE_T customServiceCharHandle;
  CYBLE_GATT_DB_ATTR_HANDLE_T
customServiceCharDescriptors[CYBLE_CUSTOM_SERVICE_CHAR_DESCRIPTORS_COUNT]; }
CYBLE_CUSTOMS_INFO_T;
```

Description

Below are the indexes and handles of the defined Custom Services and their Characteristics

Members

Members	Description
CYBLE_GATT_DB_ATTR_HANDLE_T customServiceCharHandle;	Custom Characteristic handle
CYBLE_GATT_DB_ATTR_HANDLE_T customServiceCharDescriptors[CYBLE_CUSTOM_SERVICE_CHAR_DESCRIPTORS_COUNT];	Custom Characteristic Descriptors handles

CYBLE_CUSTOMS_T

Prototype

```
typedef struct {
  CYBLE_GATT_DB_ATTR_HANDLE_T customServiceHandle;
  CYBLE_CUSTOMS_INFO_T customServiceInfo[CYBLE_CUSTOM_SERVICE_CHAR_COUNT]; }
CYBLE_CUSTOMS_T;
```

Description

Structure with Custom Service attribute handles.

Members

Members	Description
CYBLE_GATT_DB_ATTR_HANDLE_T customServiceHandle;	Handle of a Custom Service
CYBLE_CUSTOMS_INFO_T customServiceInfo[CYBLE_CUSTOM_SERVICE_CHAR_COUNT];	Information about Custom Characteristics

BLE Service-Specific Events

The BLE stack generates service-specific events to notify the application that a service specific status change needs attention. For general stack events, refer to BLE Common Events.

CYBLE_EVT_T

Prototype

```
typedef enum {
  CYBLE_EVT_GATTS_INDICATION_ENABLED,
  CYBLE_EVT_GATTS_INDICATION_DISABLED,
  CYBLE_EVT_GATTC_INDICATION,
  CYBLE_EVT_GATTC_SRVC_DISCOVERY_FAILED,
  CYBLE_EVT_GATTC_INCL_DISCOVERY_FAILED,
  CYBLE_EVT_GATTC_CHAR_DISCOVERY_FAILED,
  CYBLE_EVT_GATTC_DESCR_DISCOVERY_FAILED,
  CYBLE_EVT_GATTC_SRVC_DUPLICATION,
  CYBLE_EVT_GATTC_CHAR_DUPLICATION,
  CYBLE_EVT_GATTC_DESCR_DUPLICATION,
  CYBLE_EVT_GATTC_SRVC_DISCOVERY_COMPLETE,
  CYBLE_EVT_GATTC_INCL_DISCOVERY_COMPLETE,
  CYBLE_EVT_GATTC_CHAR_DISCOVERY_COMPLETE,
  CYBLE_EVT_GATTC_DISCOVERY_COMPLETE,
  CYBLE_EVT_ANSS_NOTIFICATION_ENABLED,
  CYBLE_EVT_ANSS_NOTIFICATION_DISABLED,
  CYBLE_EVT_ANSS_CHAR_WRITE,
  CYBLE_EVT_ANSC_NOTIFICATION,
  CYBLE_EVT_ANSC_READ_CHAR_RESPONSE,
  CYBLE_EVT_ANSC_WRITE_CHAR_RESPONSE,
  CYBLE_EVT_ANSC_READ_DESCR_RESPONSE,
  CYBLE_EVT_ANSC_WRITE_DESCR_RESPONSE,
  CYBLE_EVT_BASS_NOTIFICATION_ENABLED,
  CYBLE_EVT_BASS_NOTIFICATION_DISABLED,
  CYBLE_EVT_BASC_NOTIFICATION,
  CYBLE_EVT_BASC_READ_CHAR_RESPONSE,
  CYBLE_EVT_BASC_READ_DESCR_RESPONSE,
  CYBLE_EVT_BASC_WRITE_DESCR_RESPONSE,
  CYBLE_EVT_BLSS_INDICATION_ENABLED,
  CYBLE_EVT_BLSS_INDICATION_DISABLED,
  CYBLE_EVT_BLSS_INDICATION_CONFIRMED,
  CYBLE_EVT_BLSS_NOTIFICATION_ENABLED,
  CYBLE_EVT_BLSS_NOTIFICATION_DISABLED,
  CYBLE_EVT_BLSC_INDICATION,
  CYBLE_EVT_BLSC_NOTIFICATION,
  CYBLE_EVT_BLSC_READ_CHAR_RESPONSE,
  CYBLE_EVT_BLSC_READ_DESCR_RESPONSE,
  CYBLE_EVT_BLSC_WRITE_DESCR_RESPONSE,
  CYBLE_EVT_CPSS_NOTIFICATION_ENABLED,
  CYBLE_EVT_CPSS_NOTIFICATION_DISABLED,
  CYBLE_EVT_CPSS_INDICATION_ENABLED,
  CYBLE_EVT_CPSS_INDICATION_DISABLED,
  CYBLE_EVT_CPSS_INDICATION_CONFIRMED,
  CYBLE_EVT_CPSS_BROADCAST_ENABLED,
  CYBLE_EVT_CPSS_BROADCAST_DISABLED,
    CYBLE_EVT_CPSS_CHAR_WRITE,
    CYBLE_EVT_CPSC_NOTIFICATION,
    CYBLE_EVT_CPSC_INDICATION,
    CYBLE_EVT_CPSC_READ_CHAR_RESPONSE,
    CYBLE_EVT_CPSC_WRITE_CHAR_RESPONSE,
    CYBLE_EVT_CPSC_READ_DESCR_RESPONSE,
    CYBLE_EVT_CPSC_WRITE_DESCR_RESPONSE,
    CYBLE_EVT_CPSC_SCAN_PROGRESS_RESULT,
    CYBLE_EVT_CSCSS_NOTIFICATION_ENABLED,
    CYBLE_EVT_CSCSS_NOTIFICATION_DISABLED,
    CYBLE_EVT_CSCSS_INDICATION_ENABLED,
    CYBLE_EVT_CSCSS_INDICATION_DISABLED,
    CYBLE_EVT_CSCSS_INDICATION_CONFIRMATION,
    CYBLE_EVT_CSCSS_CHAR_WRITE,
    CYBLE_EVT_CSCSC_NOTIFICATION,
    CYBLE_EVT_CSCSC_INDICATION,
    CYBLE_EVT_CSCSC_READ_CHAR_RESPONSE,
    CYBLE_EVT_CSCSC_WRITE_CHAR_RESPONSE,
    CYBLE_EVT_CSCSC_READ_DESCR_RESPONSE,
    CYBLE_EVT_CSCSC_WRITE_DESCR_RESPONSE,
    CYBLE_EVT_CTSS_NOTIFICATION_ENABLED,
    CYBLE_EVT_CTSS_NOTIFICATION_DISABLED,
    CYBLE_EVT_CTSC_NOTIFICATION,
    CYBLE_EVT_CTSC_READ_CHAR_RESPONSE,
    CYBLE_EVT_CTSC_READ_DESCR_RESPONSE,
    CYBLE_EVT_CTSC_WRITE_DESCR_RESPONSE,
    CYBLE_EVT_DISC_READ_CHAR_RESPONSE,
    CYBLE_EVT_GLSS_INDICATION_ENABLED,
    CYBLE_EVT_GLSS_INDICATION_DISABLED,
    CYBLE_EVT_GLSS_INDICATION_CONFIRMED,
    CYBLE_EVT_GLSS_NOTIFICATION_ENABLED,
    CYBLE_EVT_GLSS_NOTIFICATION_DISABLED,
    CYBLE_EVT_GLSS_WRITE_CHAR,
    CYBLE_EVT_GLSC_INDICATION,
    CYBLE_EVT_GLSC_NOTIFICATION,
    CYBLE_EVT_GLSC_READ_CHAR_RESPONSE,
    CYBLE_EVT_GLSC_WRITE_CHAR_RESPONSE,
    CYBLE_EVT_GLSC_READ_DESCR_RESPONSE,
    CYBLE_EVT_GLSC_WRITE_DESCR_RESPONSE,
    CYBLE_EVT_HIDSS_NOTIFICATION_ENABLED,
    CYBLE_EVT_HIDSS_NOTIFICATION_DISABLED,
    CYBLE_EVT_HIDSS_BOOT_MODE_ENTER,
    CYBLE_EVT_HIDSS_REPORT_MODE_ENTER,
    CYBLE_EVT_HIDSS_SUSPEND,
    CYBLE_EVT_HIDSS_EXIT_SUSPEND,
    CYBLE_EVT_HIDSS_REPORT_CHAR_WRITE,
    CYBLE_EVT_HIDSC_NOTIFICATION,
    CYBLE_EVT_HIDSC_READ_CHAR_RESPONSE,
    CYBLE_EVT_HIDSC_WRITE_CHAR_RESPONSE,
    CYBLE_EVT_HIDSC_READ_DESCR_RESPONSE,
    CYBLE_EVT_HIDSC_WRITE_DESCR_RESPONSE,
    CYBLE_EVT_HRSS_ENERGY_EXPENDED_RESET,
    CYBLE_EVT_HRSS_NOTIFICATION_ENABLED,
    CYBLE_EVT_HRSS_NOTIFICATION_DISABLED,
    CYBLE_EVT_HRSC_NOTIFICATION,
    CYBLE_EVT_HRSC_READ_CHAR_RESPONSE,
    CYBLE_EVT_HRSC_WRITE_CHAR_RESPONSE,
    CYBLE_EVT_HRSC_READ_DESCR_RESPONSE,
    CYBLE_EVT_HRSC_WRITE_DESCR_RESPONSE,
    CYBLE_EVT_HTSS_NOTIFICATION_ENABLED,
    CYBLE_EVT_HTSS_NOTIFICATION_DISABLED,
    CYBLE_EVT_HTSS_INDICATION_ENABLED,
    CYBLE_EVT_HTSS_INDICATION_DISABLED,
    CYBLE_EVT_HTSS_INDICATION_CONFIRMED,
    CYBLE_EVT_HTSS_CHAR_WRITE,
    CYBLE_EVT_HTSC_NOTIFICATION,
    CYBLE_EVT_HTSC_INDICATION,
    CYBLE_EVT_HTSC_READ_CHAR_RESPONSE,
    CYBLE_EVT_HTSC_WRITE_CHAR_RESPONSE,
    CYBLE_EVT_HTSC_READ_DESCR_RESPONSE,
    CYBLE_EVT_HTSC_WRITE_DESCR_RESPONSE,
    CYBLE_EVT_IASS_WRITE_CHAR_CMD,
    CYBLE_EVT_LLSS_WRITE_CHAR_REQ,
    CYBLE_EVT_LLSC_READ_CHAR_RESPONSE,
    CYBLE_EVT_LLSC_WRITE_CHAR_RESPONSE,
    CYBLE_EVT_LNSS_INDICATION_ENABLED,
    CYBLE_EVT_LNSS_INDICATION_DISABLED,
    CYBLE_EVT_LNSS_INDICATION_CONFIRMED,
    CYBLE_EVT_LNSS_NOTIFICATION_ENABLED,
    CYBLE_EVT_LNSS_NOTIFICATION_DISABLED,
    CYBLE_EVT_LNSS_WRITE_CHAR,
    CYBLE_EVT_LNSC_INDICATION,
    CYBLE_EVT_LNSC_NOTIFICATION,
    CYBLE_EVT_LNSC_READ_CHAR_RESPONSE,
    CYBLE_EVT_LNSC_WRITE_CHAR_RESPONSE,
    CYBLE_EVT_LNSC_READ_DESCR_RESPONSE,
    CYBLE_EVT_LNSC_WRITE_DESCR_RESPONSE,
    CYBLE_EVT_NDCSC_READ_CHAR_RESPONSE,
    CYBLE_EVT_PASSS_NOTIFICATION_ENABLED,
    CYBLE_EVT_PASSS_NOTIFICATION_DISABLED,
    CYBLE_EVT_PASSS_WRITE_CHAR,
    CYBLE_EVT_PASSC_NOTIFICATION,
    CYBLE_EVT_PASSC_READ_CHAR_RESPONSE,
    CYBLE_EVT_PASSC_WRITE_CHAR_RESPONSE,
    CYBLE_EVT_PASSC_READ_DESCR_RESPONSE,
    CYBLE_EVT_PASSC_WRITE_DESCR_RESPONSE,
    CYBLE_EVT_RSCSS_NOTIFICATION_ENABLED,
    CYBLE_EVT_RSCSS_NOTIFICATION_DISABLED,
    CYBLE_EVT_RSCSS_INDICATION_ENABLED,
    CYBLE_EVT_RSCSS_INDICATION_DISABLED,
    CYBLE_EVT_RSCSS_INDICATION_CONFIRMATION,
    CYBLE_EVT_RSCSS_CHAR_WRITE,
    CYBLE_EVT_RSCSC_NOTIFICATION,
    CYBLE_EVT_RSCSC_INDICATION,
    CYBLE_EVT_RSCSC_READ_CHAR_RESPONSE,
    CYBLE_EVT_RSCSC_WRITE_CHAR_RESPONSE,
    CYBLE_EVT_RSCSC_READ_DESCR_RESPONSE,
    CYBLE_EVT_RSCSC_WRITE_DESCR_RESPONSE,
    CYBLE_EVT_RTUSS_WRITE_CHAR_CMD,
    CYBLE_EVT_RTUSC_READ_CHAR_RESPONSE,
     CYBLE_EVT_SCPSS_NOTIFICATION_ENABLED,
     CYBLE_EVT_SCPSS_NOTIFICATION_DISABLED,
     CYBLE_EVT_SCPSS_SCAN_INT_WIN_CHAR_WRITE,
     CYBLE_EVT_SCPSC_NOTIFICATION,
     CYBLE_EVT_SCPSC_READ_DESCR_RESPONSE,
     CYBLE_EVT_SCPSC_WRITE_DESCR_RESPONSE,
     CYBLE_EVT_TPSS_NOTIFICATION_ENABLED,
     CYBLE_EVT_TPSS_NOTIFICATION_DISABLED,
     CYBLE_EVT_TPSC_NOTIFICATION,
     CYBLE_EVT_TPSC_READ_CHAR_RESPONSE,
     CYBLE_EVT_TPSC_READ_DESCR_RESPONSE,
     CYBLE_EVT_TPSC_WRITE_DESCR_RESPONSE,
     CYBLE_DEBUG_EVT_BLESS_INT = 0xE000u
   } CYBLE_EVT_T;
```

Description

Service specific events

Members

Members	Description
CYBLE_EVT_GATTS_INDICATION_ENABLED	GATT Server - Notifications for GATT Service's "Service Changed" Characteristic were enabled. The parameter of this event is a structure of CYBLE_GATTS_WRITE_REQ_PARAM_T type.
CYBLE_EVT_GATTS_INDICATION_DISABLED	GATT Server - Notifications for GATT Service's "Service Changed" Characteristic were disabled. The parameter of this event is a structure of CYBLE_GATTS_WRITE_REQ_PARAM_T type.
CYBLE_EVT_GATTC_INDICATION	GATT Client - GATT Service's "Service Changed" Characteristic Indications were received. The parameter of this event is a structure of CYBLE_ANS_CHAR_VALUE_T type.
CYBLE_EVT_GATTC_SRVC_DISCOVERY_FAILED	GATT Client - Service discovery procedure failed. This event may be generated on calling CyBle_GattcDiscoverAllPrimaryServices(). No parameters passed for this event.
CYBLE_EVT_GATTC_INCL_DISCOVERY_FAILED	GATT Client - Discovery of included services failed. This event may be generated on calling CyBle_GattcFindIncludedServices(). No parameters passed for this event.
CYBLE_EVT_GATTC_CHAR_DISCOVERY_FAILED	GATT Client - Discovery of service's Characteristics failed. This event may be generated on calling CyBle_GattcDiscoverAllCharacteristics() or CyBle_GattcReadUsingCharacteristicUuid(). No parameters passed for this event.

Members	Description
CYBLE_EVT_GATTC_DESCR_DISCOVERY_FAILED	GATT Client - Discovery of service's Characteristics failed. This event may be generated on calling CyBle_GattcDiscoverAllCharacteristicDescriptors(). No parameters passed for this event.
CYBLE_EVT_GATTC_SRVC_DUPLICATION	GATT Client - Duplicate service record was found during server device discovery. The parameter of this event is a structure of uint16 (UUID16) type.
CYBLE_EVT_GATTC_CHAR_DUPLICATION	GATT Client - Duplicate service's Characteristic record was found during server device discovery. The parameter of this event is a structure of uint16 (UUID16) type.
CYBLE_EVT_GATTC_DESCR_DUPLICATION	GATT Client - Duplicate service's Characteristic Descriptor record was found during server device discovery. The parameter of this event is a structure of uint16 (UUID16) type.
CYBLE_EVT_GATTC_SRVC_DISCOVERY_COMPLETE	GATT Client - Service discovery procedure completed successfully. This event may be generated on calling CyBle_GattcDiscoverAllPrimaryServices(). No parameters passed for this event.
CYBLE_EVT_GATTC_INCL_DISCOVERY_COMPLETE	GATT Client - Included services discovery is completed successfully. This event may be generated on calling CyBle_GattcFindIncludedServices(). No parameters passed for this event.
CYBLE_EVT_GATTC_CHAR_DISCOVERY_COMPLETE	GATT Client - Discovery of service's Characteristics discovery is completed successfully. This event may be generated on calling CyBle_GattcDiscoverAllCharacteristics() or CyBle_GattcReadUsingCharacteristicUuid(). No parameters passed for this event.
CYBLE_EVT_GATTC_DISCOVERY_COMPLETE	GATT Client - Discovery of remote device completed successfully. No parameters passed for this event.
CYBLE_EVT_ANSS_NOTIFICATION_ENABLED	ANS Server - Notifications for Alert Notification Service Characteristic were enabled. The parameter of this event is a structure of CYBLE_ANS_CHAR_VALUE_T type.
CYBLE_EVT_ANSS_NOTIFICATION_DISABLED	ANS Server - Notifications for Alert Notification Service Characteristic were disabled. The parameter of this event is a structure of CYBLE_ANS_CHAR_VALUE_T type.
CYBLE_EVT_ANSS_CHAR_WRITE	ANS Server - Write Request for Alert Notification Service Characteristic was received. The parameter of this event is a structure of CYBLE_ANS_CHAR_VALUE_T type.
CYBLE_EVT_ANSC_NOTIFICATION	ANS Client - Alert Notification Characteristic Service Notification was received. The parameter of this event is a structure of CYBLE_ANS_CHAR_VALUE_T type.

Members	Description
CYBLE_EVT_ANSC_READ_CHAR_RESPONSE	ANS Client - Read Response for Alert Notification Service Characteristic Value. The parameter of this event is a structure of CYBLE_ANS_CHAR_VALUE_T type.
CYBLE_EVT_ANSC_WRITE_CHAR_RESPONSE	ANS Client - Write Response for Write Request for Alert Notification Service Characteristic Value. The parameter of this event is a structure of CYBLE_ANS_CHAR_VALUE_T type.
CYBLE_EVT_ANSC_READ_DESCR_RESPONSE	ANS Client - Read Response for Read Request for Alert Notification Service Characteristic Descriptor Read Request. The parameter of this event is a structure of CYBLE_ANS_DESCR_VALUE_T type.
CYBLE_EVT_ANSC_WRITE_DESCR_RESPONSE	ANS Client - Write Response for Write Request for Alert Notification Service Client Characteristic Configuration Descriptor Value. The parameter of this event is a structure of CYBLE_ANS_DESCR_VALUE_T type.
CYBLE_EVT_BASS_NOTIFICATION_ENABLED	BAS Server - Notifications for Battery Level Characteristic were enabled. The parameter of this event is a structure of CYBLE_BAS_CHAR_VALUE_T type.
CYBLE_EVT_BASS_NOTIFICATION_DISABLED	BAS Server - Notifications for Battery Level Characteristic were disabled. The parameter of this event is a structure of CYBLE_BAS_CHAR_VALUE_T type.
CYBLE_EVT_BASC_NOTIFICATION	BAS Client - Battery Level Characteristic Notification was received. The parameter of this event is a structure of CYBLE_BAS_CHAR_VALUE_T type.
CYBLE_EVT_BASC_READ_CHAR_RESPONSE	BAS Client - Read Response for Battery Level Characteristic Value. The parameter of this event is a structure of CYBLE_BAS_CHAR_VALUE_T type.
CYBLE_EVT_BASC_READ_DESCR_RESPONSE	BAS Client - Read Response for Battery Level Characteristic Descriptor Read Request. The parameter of this event is a structure of CYBLE_BAS_DESCR_VALUE_T type.
CYBLE_EVT_BASC_WRITE_DESCR_RESPONSE	BAS Client - Write Response for Battery Level Client Characteristic Configuration Descriptor Value. The parameter of this event is a structure of CYBLE_BAS_DESCR_VALUE_T type.
CYBLE_EVT_BLSS_INDICATION_ENABLED	BLS Server - Indication for Blood Pressure Service Characteristic was enabled. The parameter of this event is a structure of CYBLE_BLS_CHAR_VALUE_T type
CYBLE_EVT_BLSS_INDICATION_DISABLED	BLS Server - Indication for Blood Pressure Service Characteristic was disabled. The parameter of this event is a structure of CYBLE_BLS_CHAR_VALUE_T type
CYBLE_EVT_BLSS_INDICATION_CONFIRMED	BLS Server - Blood Pressure Service Characteristic Indication was confirmed. The parameter of this event is a structure of CYBLE_BLS_CHAR_VALUE_T type

Members	Description
CYBLE_EVT_BLSS_NOTIFICATION_ENABLED	BLS Server - Notifications for Blood Pressure Service Characteristic were enabled. The parameter of this event is a structure of CYBLE_BLS_CHAR_VALUE_T type.
CYBLE_EVT_BLSS_NOTIFICATION_DISABLED	BLS Server - Notifications for Blood Pressure Service Characteristic were disabled. The parameter of this event is a structure of CYBLE_BLS_CHAR_VALUE_T type
CYBLE_EVT_BLSC_INDICATION	BLS Client - Blood Pressure Service Characteristic Indication was received. The parameter of this event is a structure of CYBLE_BLS_CHAR_VALUE_T type
CYBLE_EVT_BLSC_NOTIFICATION	BLS Client - Blood Pressure Service Characteristic Notification was received. The parameter of this event is a structure of CYBLE_BLS_CHAR_VALUE_T type
CYBLE_EVT_BLSC_READ_CHAR_RESPONSE	BLS Client - Read Response for Read Request of Blood Pressure Service Characteristic value. The parameter of this event is a structure of CYBLE_BLS_CHAR_VALUE_T type
CYBLE_EVT_BLSC_READ_DESCR_RESPONSE	BLS Client - Read Response for Read Request of Blood Pressure Service Characteristic Descriptor Read request. The parameter of this event is a structure of CYBLE_BLS_DESCR_VALUE_T type
CYBLE_EVT_BLSC_WRITE_DESCR_RESPONSE	BLS Client - Write Response for Write Request of Blood Pressure Service Characteristic Configuration Descriptor value. The parameter of this event is a structure of CYBLE_BLS_DESCR_VALUE_T type
CYBLE_EVT_CPSS_NOTIFICATION_ENABLED	CPS Server - Notifications for Cycling Power Service Characteristic were enabled. The parameter of this event is a structure of CYBLE_CPS_CHAR_VALUE_T type.
CYBLE_EVT_CPSS_NOTIFICATION_DISABLED	CPS Server - Notifications for Cycling Power Service Characteristic were disabled. The parameter of this event is a structure of CYBLE_CPS_CHAR_VALUE_T type
CYBLE_EVT_CPSS_INDICATION_ENABLED	CPS Server - Indication for Cycling Power Service Characteristic was enabled. The parameter of this event is a structure of CYBLE_CPS_CHAR_VALUE_T type
CYBLE_EVT_CPSS_INDICATION_DISABLED	CPS Server - Indication for Cycling Power Service Characteristic was disabled. The parameter of this event is a structure of CYBLE_CPS_CHAR_VALUE_T type
CYBLE_EVT_CPSS_INDICATION_CONFIRMED	CPS Server - Cycling Power Service Characteristic Indication was confirmed. The parameter of this event is a structure of CYBLE_CPS_CHAR_VALUE_T type
CYBLE_EVT_CPSS_BROADCAST_ENABLED	CPS Server - Broadcast for Cycling Power Service Characteristic was enabled. The parameter of this event is a structure of CYBLE_CPS_CHAR_VALUE_T type

Members	Description
CYBLE_EVT_CPSS_BROADCAST_DISABLED	CPS Server - Broadcast for Cycling Power Service Characteristic was disabled. The parameter of this event is a structure of CYBLE_CPS_CHAR_VALUE_T type
CYBLE_EVT_CPSS_CHAR_WRITE	CPS Server - Write Request for Cycling Power Service Characteristic was received. The parameter of this event is a structure of CYBLE_CPS_CHAR_VALUE_T type.
CYBLE_EVT_CPSC_NOTIFICATION	CPS Client - Cycling Power Service Characteristic Notification was received. The parameter of this event is a structure of CYBLE_CPS_CHAR_VALUE_T type
CYBLE_EVT_CPSC_INDICATION	CPS Client - Cycling Power Service Characteristic Indication was received. The parameter of this event is a structure of CYBLE_CPS_CHAR_VALUE_T type
CYBLE_EVT_CPSC_READ_CHAR_RESPONSE	CPS Client - Read Response for Read Request of Cycling Power Service Characteristic value. The parameter of this event is a structure of CYBLE_CPS_CHAR_VALUE_T type
CYBLE_EVT_CPSC_WRITE_CHAR_RESPONSE	CPS Client - Write Response for Write Request of Cycling Power Service Characteristic value. The parameter of this event is a structure of CYBLE_CPS_CHAR_VALUE_T type
CYBLE_EVT_CPSC_READ_DESCR_RESPONSE	CPS Client - Read Response for Read Request of Cycling Power Service Characteristic Descriptor Read request. The parameter of this event is a structure of CYBLE_CPS_DESCR_VALUE_T type
CYBLE_EVT_CPSC_WRITE_DESCR_RESPONSE	CPS Client - Write Response for Write Request of Cycling Power Service Characteristic Configuration Descriptor value. The parameter of this event is a structure of CYBLE_CPS_DESCR_VALUE_T type
CYBLE_EVT_CPSC_SCAN_PROGRESS_RESULT	CPS Client - This event is triggered every time a device receive non-connectable undirected advertising event. The parameter of this event is a structure of CYBLE_CPS_CHAR_VALUE_T type
CYBLE_EVT_CSCSS_NOTIFICATION_ENABLED	CSCS Server - Notifications for Cycling Speed and Cadence Service Characteristic were enabled. The parameter of this event is a structure of CYBLE_CSCS_CHAR_VALUE_T type.
CYBLE_EVT_CSCSS_NOTIFICATION_DISABLED	CSCS Server - Notifications for Cycling Speed and Cadence Service Characteristic were disabled. The parameter of this event is a structure of CYBLE_CSCS_CHAR_VALUE_T type
CYBLE_EVT_CSCSS_INDICATION_ENABLED	CSCS Server - Indication for Cycling Speed and Cadence Service Characteristic was enabled. The parameter of this event is a structure of CYBLE_CSCS_CHAR_VALUE_T type

Members	Description
CYBLE_EVT_CSCSS_INDICATION_DISABLED	CSCS Server - Indication for Cycling Speed and Cadence Service Characteristic was disabled. The parameter of this event is a structure of CYBLE_CSCS_CHAR_VALUE_T type
CYBLE_EVT_CSCSS_INDICATION_CONFIRMATION	CSCS Server - Cycling Speed and Cadence Service Characteristic Indication was confirmed. The parameter of this event is a structure of CYBLE_CSCS_CHAR_VALUE_T type
CYBLE_EVT_CSCSS_CHAR_WRITE	CSCS Server - Write Request for Cycling Speed and Cadence Service Characteristic was received. The parameter of this event is a structure of CYBLE_CSCS_CHAR_VALUE_T type.
CYBLE_EVT_CSCSC_NOTIFICATION	CSCS Client - Cycling Speed and Cadence Service Characteristic Notification was received. The parameter of this event is a structure of CYBLE_CSCS_CHAR_VALUE_T type
CYBLE_EVT_CSCSC_INDICATION	CSCS Client - Cycling Speed and Cadence Service Characteristic Indication was received. The parameter of this event is a structure of CYBLE_CSCS_CHAR_VALUE_T type
CYBLE_EVT_CSCSC_READ_CHAR_RESPONSE	CSCS Client - Read Response for Read Request of Cycling Speed and Cadence Service Characteristic value. The parameter of this event is a structure of CYBLE_CSCS_CHAR_VALUE_T type
CYBLE_EVT_CSCSC_WRITE_CHAR_RESPONSE	CSCS Client - Write Response for Write Request of Cycling Speed and Cadence Service Characteristic value. The parameter of this event is a structure of CYBLE_CSCS_CHAR_VALUE_T type
CYBLE_EVT_CSCSC_READ_DESCR_RESPONSE	CSCS Client - Read Response for Read Request of Cycling Speed and Cadence Service Characteristic Descriptor Read request. The parameter of this event is a structure of CYBLE_CSCS_DESCR_VALUE_T type
CYBLE_EVT_CSCSC_WRITE_DESCR_RESPONSE	CSCS Client - Write Response for Write Request of Cycling Speed and Cadence Service Characteristic Configuration Descriptor value. The parameter of this event is a structure of CYBLE_CSCS_DESCR_VALUE_T type
CYBLE_EVT_CTSS_NOTIFICATION_ENABLED	CTS Server - Notification for Current Time Characteristic was enabled. The parameter of this event is a structure of CYBLE_CTS_CHAR_VALUE_T type
CYBLE_EVT_CTSS_NOTIFICATION_DISABLED	CTS Server - Notification for Current Time Characteristic was disabled. The parameter of this event is a structure of CYBLE_CTS_CHAR_VALUE_T type

Members	Description
CYBLE_EVT_CTSC_NOTIFICATION	CTS Client - Current Time Characteristic Notification was received. The parameter of this event is a structure of CYBLE_CTS_CHAR_VALUE_T type
CYBLE_EVT_CTSC_READ_CHAR_RESPONSE	CTS Client - Read Response for Current Time Characteristic Value Read Request. The parameter of this event is a structure of CYBLE_CTS_CHAR_VALUE_T type
CYBLE_EVT_CTSC_READ_DESCR_RESPONSE	CTS Client - Read Response for Current Time Client Characteristic Configuration Descriptor Value Read Request. The parameter of this event is a structure of CYBLE_CTS_DESCR_VALUE_T type
CYBLE_EVT_CTSC_WRITE_DESCR_RESPONSE	CTS Client - Write Response for Current Time Characteristic Configuration Descriptor Value. The parameter of this event is a structure of CYBLE_CTS_DESCR_VALUE_T type
CYBLE_EVT_DISC_READ_CHAR_RESPONSE	DIS Client - Read Response for a Read Request for a Device Service Characteristic. The parameter of this event is a structure of CYBLE_DIS_CHAR_VALUE_T type
CYBLE_EVT_GLSS_INDICATION_ENABLED	GLS Server - Indication for Glucose Service Characteristic was enabled. The parameter of this event is a structure of CYBLE_GLS_CHAR_VALUE_T type
CYBLE_EVT_GLSS_INDICATION_DISABLED	GLS Server - Indication for Glucose Service Characteristic was disabled. The parameter of this event is a structure of CYBLE_GLS_CHAR_VALUE_T type
CYBLE_EVT_GLSS_INDICATION_CONFIRMED	GLS Server - Glucose Service Characteristic Indication was confirmed. The parameter of this event is a structure of CYBLE_GLS_CHAR_VALUE_T type
CYBLE_EVT_GLSS_NOTIFICATION_ENABLED	GLS Server - Notifications for Glucose Service Characteristic were enabled. The parameter of this event is a structure of CYBLE_GLS_CHAR_VALUE_T type.
CYBLE_EVT_GLSS_NOTIFICATION_DISABLED	GLS Server - Notifications for Glucose Service Characteristic were disabled. The parameter of this event is a structure of CYBLE_GLS_CHAR_VALUE_T type
CYBLE_EVT_GLSS_WRITE_CHAR	GLS Server - Write Request for Glucose Service was received. The parameter of this event is a structure of CYBLE_GLS_CHAR_VALUE_T type.
CYBLE_EVT_GLSC_INDICATION	GLS Client - Glucose Service Characteristic Indication was received. The parameter of this event is a structure of CYBLE_GLS_CHAR_VALUE_T type
CYBLE_EVT_GLSC_NOTIFICATION	GLS Client - Glucose Service Characteristic Notification was received. The parameter of this event is a structure of CYBLE_GLS_CHAR_VALUE_T type

Members	Description
CYBLE_EVT_GLSC_READ_CHAR_RESPONSE	GLS Client - Read Response for Read Request of Glucose Service Characteristic value. The parameter of this event is a structure of CYBLE_GLS_CHAR_VALUE_T type
CYBLE_EVT_GLSC_WRITE_CHAR_RESPONSE	GLS Client - Write Response for Write Request of Glucose Service Characteristic value. The parameter of this event is a structure of CYBLE_GLS_CHAR_VALUE_T type
CYBLE_EVT_GLSC_READ_DESCR_RESPONSE	GLS Client - Read Response for Read Request of Glucose Service Characteristic Descriptor Read request. The parameter of this event is a structure of CYBLE_GLS_DESCR_VALUE_T type
CYBLE_EVT_GLSC_WRITE_DESCR_RESPONSE	GLS Client - Write Response for Write Request of Glucose Service Characteristic Configuration Descriptor value. The parameter of this event is a structure of CYBLE_GLS_DESCR_VALUE_T type
CYBLE_EVT_HIDSS_NOTIFICATION_ENABLED	HIDS Server - Notifications for HID service were enabled. The parameter of this event is a structure of CYBLE_HIDS_CHAR_VALUE_T type
CYBLE_EVT_HIDSS_NOTIFICATION_DISABLED	HIDS Server - Notifications for HID service were disabled. The parameter of this event is a structure of CYBLE_HIDS_CHAR_VALUE_T type
CYBLE_EVT_HIDSS_BOOT_MODE_ENTER	HIDS Server - Enter boot mode request. The parameter of this event is a structure of CYBLE_HIDS_CHAR_VALUE_T type
CYBLE_EVT_HIDSS_REPORT_MODE_ENTER	HIDS Server - Enter report mode request. The parameter of this event is a structure of CYBLE_HIDS_CHAR_VALUE_T type
CYBLE_EVT_HIDSS_SUSPEND	HIDS Server - Enter suspend mode request. The parameter of this event is a structure of CYBLE_HIDS_CHAR_VALUE_T type
CYBLE_EVT_HIDSS_EXIT_SUSPEND	HIDS Server - Exit suspend mode request. The parameter of this event is a structure of CYBLE_HIDS_CHAR_VALUE_T type
CYBLE_EVT_HIDSS_REPORT_CHAR_WRITE	HIDS Server - Write Report Characteristic request. The parameter of this event is a structure of CYBLE_HIDSS_REPORT_VALUE_T type
CYBLE_EVT_HIDSC_NOTIFICATION	HIDS Client - HID Service Characteristic Notification was received. The parameter of this event is a structure of CYBLE_HIDS_CHAR_VALUE_T type
CYBLE_EVT_HIDSC_READ_CHAR_RESPONSE	HIDS Client - Read Response for Read Request of HID Service Characteristic value. The parameter of this event is a structure of CYBLE_HIDS_DESCR_VALUE_T type

Members	Description
CYBLE_EVT_HIDSC_WRITE_CHAR_RESPONSE	HIDS Client - Write Response for Write Request of HID Service Characteristic value. The parameter of this event is a structure of CYBLE_HIDS_CHAR_VALUE_T type
CYBLE_EVT_HIDSC_READ_DESCR_RESPONSE	HIDS Client - Read Response for Read Request of HID Service Characteristic Descriptor Read request. The parameter of this event is a structure of CYBLE_HIDS_CHAR_VALUE_T type
CYBLE_EVT_HIDSC_WRITE_DESCR_RESPONSE	HIDS Client - Write Response for Write Request of HID Service Characteristic Configuration Descriptor value. The parameter of this event is a structure of CYBLE_HIDS_CHAR_VALUE_T type
CYBLE_EVT_HRSS_ENERGY_EXPENDED_RESET	HRS Server - Reset Energy Expended. The parameter of this event is a structure of CYBLE_HRS_CHAR_VALUE_T type
CYBLE_EVT_HRSS_NOTIFICATION_ENABLED	HRS Server - Notification for Heart Rate Measurement Characteristic was enabled. The parameter of this event is a structure of CYBLE_HRS_CHAR_VALUE_T type
CYBLE_EVT_HRSS_NOTIFICATION_DISABLED	HRS Server - Notification for Heart Rate Measurement Characteristic was disabled. The parameter of this event is a structure of CYBLE_HRS_CHAR_VALUE_T type
CYBLE_EVT_HRSC_NOTIFICATION	HRS Client - Heart Rate Measurement Characteristic Notification was received. The parameter of this event is a structure of CYBLE_HRS_CHAR_VALUE_T type
CYBLE_EVT_HRSC_READ_CHAR_RESPONSE	HRS Client - Read Response for Read Request of HRS Service Characteristic value. The parameter of this event is a structure of CYBLE_HRS_CHAR_VALUE_T type
CYBLE_EVT_HRSC_WRITE_CHAR_RESPONSE	HRS Client - Write Response for Write Request of HRS Service Characteristic value. The parameter of this event is a structure of CYBLE_HRS_CHAR_VALUE_T type
CYBLE_EVT_HRSC_READ_DESCR_RESPONSE	HRS Client - Read Response for Read Request of HRS Service Characteristic Descriptor Read request. The parameter of this event is a structure of CYBLE_HRS_CHAR_VALUE_T type
CYBLE_EVT_HRSC_WRITE_DESCR_RESPONSE	HRS Client - Write Response for Write Request of HRS Service Characteristic Configuration Descriptor value. The parameter of this event is a structure of CYBLE_HRS_CHAR_VALUE_T type
CYBLE_EVT_HTSS_NOTIFICATION_ENABLED	HTS Server - Notifications for Health Thermometer Service Characteristic were enabled. The parameter of this event is a structure of CYBLE_HTS_CHAR_VALUE_T type.

Members	Description
CYBLE_EVT_HTSS_NOTIFICATION_DISABLED	HTS Server - Notifications for Health Thermometer Service Characteristic were disabled. The parameter of this event is a structure of CYBLE_HTS_CHAR_VALUE_T type
CYBLE_EVT_HTSS_INDICATION_ENABLED	HTS Server - Indication for Health Thermometer Service Characteristic was enabled. The parameter of this event is a structure of CYBLE_HTS_CHAR_VALUE_T type
CYBLE_EVT_HTSS_INDICATION_DISABLED	HTS Server - Indication for Health Thermometer Service Characteristic was disabled. The parameter of this event is a structure of CYBLE_HTS_CHAR_VALUE_T type
CYBLE_EVT_HTSS_INDICATION_CONFIRMED	HTS Server - Health Thermometer Service Characteristic Indication was confirmed. The parameter of this event is a structure of CYBLE_HTS_CHAR_VALUE_T type
CYBLE_EVT_HTSS_CHAR_WRITE	HTS Server - Write Request for Health Thermometer Service Characteristic was received. The parameter of this event is a structure of CYBLE_HTS_CHAR_VALUE_T type.
CYBLE_EVT_HTSC_NOTIFICATION	HTS Client - Health Thermometer Service Characteristic Notification was received. The parameter of this event is a structure of CYBLE_HTS_CHAR_VALUE_T type
CYBLE_EVT_HTSC_INDICATION	HTS Client - Health Thermometer Service Characteristic Indication was received. The parameter of this event is a structure of CYBLE_HTS_CHAR_VALUE_T type
CYBLE_EVT_HTSC_READ_CHAR_RESPONSE	HTS Client - Read Response for Read Request of Health Thermometer Service Characteristic value. The parameter of this event is a structure of CYBLE_HTS_CHAR_VALUE_T type
CYBLE_EVT_HTSC_WRITE_CHAR_RESPONSE	HTS Client - Write Response for Write Request of Health Thermometer Service Characteristic value. The parameter of this event is a structure of CYBLE_HTS_CHAR_VALUE_T type
CYBLE_EVT_HTSC_READ_DESCR_RESPONSE	HTS Client - Read Response for Read Request of Health Thermometer Service Characteristic Descriptor Read request. The parameter of this event is a structure of CYBLE_HTS_DESCR_VALUE_T type
CYBLE_EVT_HTSC_WRITE_DESCR_RESPONSE	HTS Client - Write Response for Write Request of Health Thermometer Service Characteristic Configuration Descriptor value. The parameter of this event is a structure of CYBLE_HTS_DESCR_VALUE_T type

Members	Description
CYBLE_EVT_IASS_WRITE_CHAR_CMD	IAS Server - Write command request for Alert Level Characteristic. The parameter of this event is a structure of CYBLE_IAS_CHAR_VALUE_T type
CYBLE_EVT_LLSS_WRITE_CHAR_REQ	LLS Server - Write request for Alert Level Characteristic. The parameter of this event is a structure of CYBLE_LLS_CHAR_VALUE_T type
CYBLE_EVT_LLSC_READ_CHAR_RESPONSE	LLS Client - Read response for Alert Level Characteristic. The parameter of this event is a structure of CYBLE_LLS_CHAR_VALUE_T type
CYBLE_EVT_LLSC_WRITE_CHAR_RESPONSE	LLS Client - Write response for write request of Alert Level Characteristic. The parameter of this event is a structure of CYBLE_LLS_CHAR_VALUE_T type
CYBLE_EVT_LNSS_INDICATION_ENABLED	LNS Server - Indication for Location and Navigation Service Characteristic was enabled. The parameter of this event is a structure of CYBLE_LNS_CHAR_VALUE_T type
CYBLE_EVT_LNSS_INDICATION_DISABLED	LNS Server - Indication for Location and Navigation Service Characteristic was disabled. The parameter of this event is a structure of CYBLE_LNS_CHAR_VALUE_T type
CYBLE_EVT_LNSS_INDICATION_CONFIRMED	LNS Server - Location and Navigation Service Characteristic Indication was confirmed. The parameter of this event is a structure of CYBLE_LNS_CHAR_VALUE_T type
CYBLE_EVT_LNSS_NOTIFICATION_ENABLED	LNS Server - Notifications for Location and Navigation Service Characteristic were enabled. The parameter of this event is a structure of CYBLE_LNS_CHAR_VALUE_T type.
CYBLE_EVT_LNSS_NOTIFICATION_DISABLED	LNS Server - Notifications for Location and Navigation Service Characteristic were disabled. The parameter of this event is a structure of CYBLE_LNS_CHAR_VALUE_T type
CYBLE_EVT_LNSS_WRITE_CHAR	LNS Server - Write Request for Location and Navigation Service Characteristic was received. The parameter of this event is a structure of CYBLE_LNS_CHAR_VALUE_T type.
CYBLE_EVT_LNSC_INDICATION	LNS Client - Location and Navigation Service Characteristic Indication was received. The parameter of this event is a structure of CYBLE_LNS_CHAR_VALUE_T type
CYBLE_EVT_LNSC_NOTIFICATION	LNS Client - Location and Navigation Service Characteristic Notification was received. The parameter of this event is a structure of CYBLE_LNS_CHAR_VALUE_T type

Members	Description
CYBLE_EVT_LNSC_READ_CHAR_RESPONSE	LNS Client - Read Response for Read Request of Location and Navigation Service Characteristic value. The parameter of this event is a structure of CYBLE_LNS_CHAR_VALUE_T type
CYBLE_EVT_LNSC_WRITE_CHAR_RESPONSE	LNS Client - Write Response for Write Request of Location and Navigation Service Characteristic value. The parameter of this event is a structure of CYBLE_LNS_CHAR_VALUE_T type
CYBLE_EVT_LNSC_READ_DESCR_RESPONSE	LNS Client - Read Response for Read Request of Location and Navigation Service Characteristic Descriptor Read request. The parameter of this event is a structure of CYBLE_LNS_DESCR_VALUE_T type
CYBLE_EVT_LNSC_WRITE_DESCR_RESPONSE	LNS Client - Write Response for Write Request of Location and Navigation Service Characteristic Configuration Descriptor value. The parameter of this event is a structure of CYBLE_LNS_DESCR_VALUE_T type
CYBLE_EVT_PASSS_NOTIFICATION_ENABLED	PASS Server - Notifications for Phone Alert Status Service Characteristic were enabled. The parameter of this event is a structure of CYBLE_PASS_CHAR_VALUE_T type.
CYBLE_EVT_PASSS_NOTIFICATION_DISABLED	PASS Server - Notifications for Phone Alert Status Service Characteristic were disabled. The parameter of this event is a structure of CYBLE_PASS_CHAR_VALUE_T type
CYBLE_EVT_PASSS_WRITE_CHAR	PASS Server - Write Request for Phone Alert Status Service Characteristic was received. The parameter of this event is a structure of CYBLE_PASS_CHAR_VALUE_T type.
CYBLE_EVT_PASSC_NOTIFICATION	PASS Client - Phone Alert Status Service Characteristic Notification was received. The parameter of this event is a structure of CYBLE_PASS_CHAR_VALUE_T type
CYBLE_EVT_PASSC_READ_CHAR_RESPONSE	PASS Client - Read Response for Read Request of Phone Alert Status Service Characteristic value. The parameter of this event is a structure of CYBLE_PASS_CHAR_VALUE_T type
CYBLE_EVT_PASSC_WRITE_CHAR_RESPONSE	PASS Client - Write Response for Write Request of Phone Alert Status Service Characteristic value. The parameter of this event is a structure of CYBLE_PASS_CHAR_VALUE_T type
CYBLE_EVT_PASSC_READ_DESCR_RESPONSE	PASS Client - Read Response for Read Request of Phone Alert Status Service Characteristic Descriptor Read request. The parameter of this event is a structure of CYBLE_PASS_DESCR_VALUE_T type

Members	Description
CYBLE_EVT_PASSC_WRITE_DESCR_RESPONSE	PASS Client - Write Response for Write Request of Phone Alert Status Service Characteristic Configuration Descriptor value. The parameter of this event is a structure of CYBLE_PASS_DESCR_VALUE_T type
CYBLE_EVT_RSCSS_NOTIFICATION_ENABLED	RSCS Server - Notifications for Running Speed and Cadence Service Characteristic were enabled. The parameter of this event is a structure of CYBLE_RSCS_CHAR_VALUE_T type.
CYBLE_EVT_RSCSS_NOTIFICATION_DISABLED	RSCS Server - Notifications for Running Speed and Cadence Service Characteristic were disabled. The parameter of this event is a structure of CYBLE_RSCS_CHAR_VALUE_T type
CYBLE_EVT_RSCSS_INDICATION_ENABLED	RSCS Server - Indication for Running Speed and Cadence Service Characteristic was enabled. The parameter of this event is a structure of CYBLE_RSCS_CHAR_VALUE_T type
CYBLE_EVT_RSCSS_INDICATION_DISABLED	RSCS Server - Indication for Running Speed and Cadence Service Characteristic was disabled. The parameter of this event is a structure of CYBLE_RSCS_CHAR_VALUE_T type
CYBLE_EVT_RSCSS_INDICATION_CONFIRMATION	RSCS Server - Running Speed and Cadence Service Characteristic Indication was confirmed. The parameter of this event is a structure of CYBLE_RSCS_CHAR_VALUE_T type
CYBLE_EVT_RSCSS_CHAR_WRITE	RSCS Server - Write Request for Running Speed and Cadence Service Characteristic was received. The parameter of this event is a structure of CYBLE_RSCS_CHAR_VALUE_T type.
CYBLE_EVT_RSCSC_NOTIFICATION	RSCS Client - Running Speed and Cadence Service Characteristic Notification was received. The parameter of this event is a structure of CYBLE_RSCS_CHAR_VALUE_T type
CYBLE_EVT_RSCSC_INDICATION	RSCS Client - Running Speed and Cadence Service Characteristic Indication was received. The parameter of this event is a structure of CYBLE_RSCS_CHAR_VALUE_T type
CYBLE_EVT_RSCSC_READ_CHAR_RESPONSE	RSCS Client - Read Response for Read Request of Running Speed and Cadence Service Characteristic value. The parameter of this event is a structure of CYBLE_RSCS_CHAR_VALUE_T type
CYBLE_EVT_RSCSC_WRITE_CHAR_RESPONSE	RSCS Client - Write Response for Write Request of Running Speed and Cadence Service Characteristic value. The parameter of this event is a structure of CYBLE_RSCS_CHAR_VALUE_T type

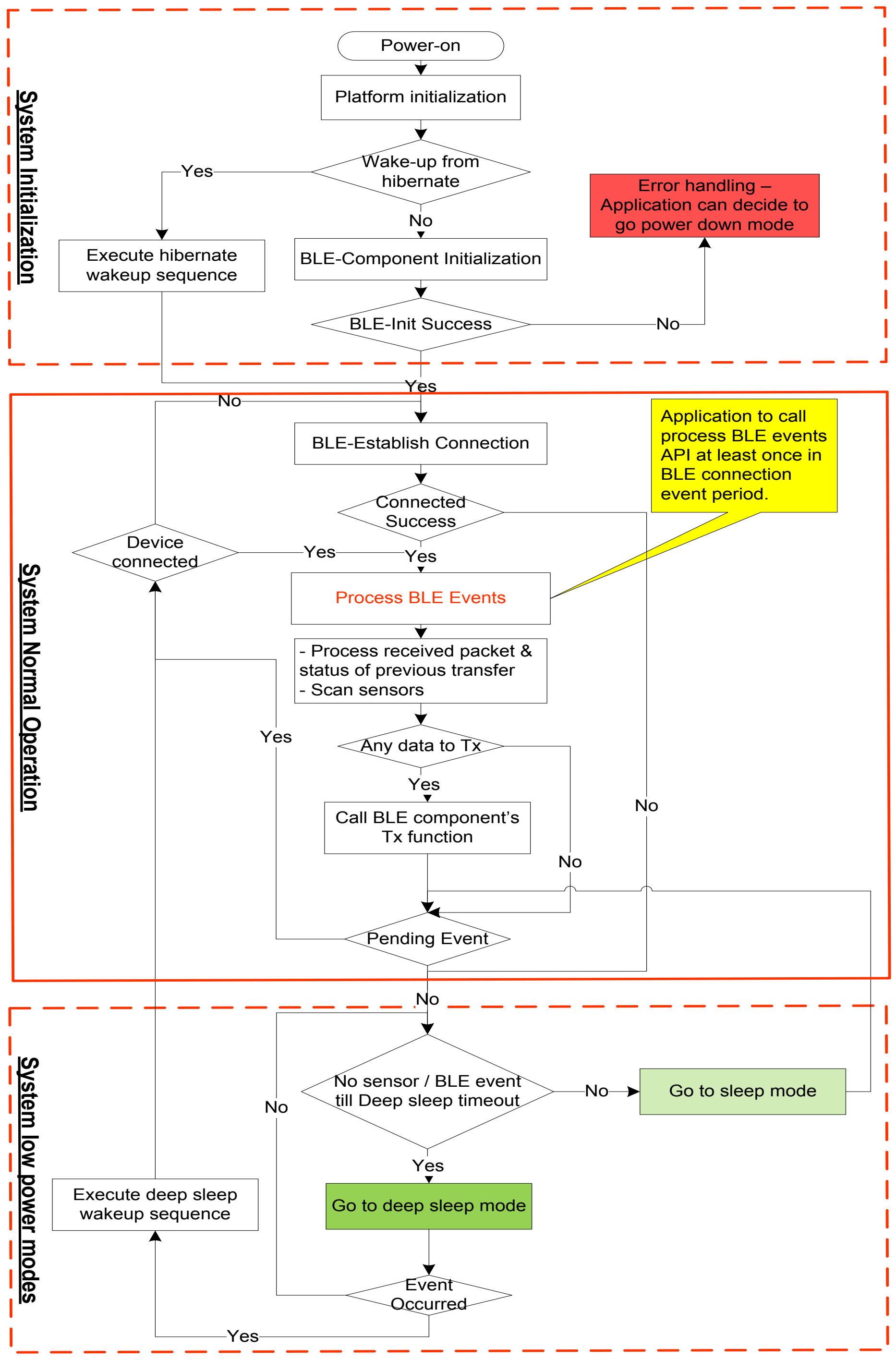
Members	Description
CYBLE_EVT_RSCSC_READ_DESCR_RESPONSE	RSCS Client - Read Response for Read Request of Running Speed and Cadence Service Characteristic Descriptor Read request. The parameter of this event is a structure of CYBLE_RSCS_DESCR_VALUE_T type
CYBLE_EVT_RSCSC_WRITE_DESCR_RESPONSE	RSCS Client - Write Response for Write Request of Running Speed and Cadence Service Characteristic Configuration Descriptor value. The parameter of this event is a structure of CYBLE_RSCS_DESCR_VALUE_T type
CYBLE_EVT_RTUSS_WRITE_CHAR_CMD	RTUS Server - Write command request for Reference Time Update Characteristic value. The parameter of this event is a structure of CYBLE_RTUS_CHAR_VALUE_T type
CYBLE_EVT_RTUSC_READ_CHAR_RESPONSE	RTUS Client - Read Response for Read Request of Reference Time Update Service Characteristic value. The parameter of this event is a structure of CYBLE_RTUS_CHAR_VALUE_T type
CYBLE_EVT_SCPSS_NOTIFICATION_ENABLED	ScPS Server - Notifications for Scan Refresh Characteristic were enabled. The parameter of this event is a structure of CYBLE_SCPS_CHAR_VALUE_T type
CYBLE_EVT_SCPSS_NOTIFICATION_DISABLED	ScPS Server - Notifications for Scan Refresh Characteristic were disabled. The parameter of this event is a structure of CYBLE_SCPS_CHAR_VALUE_T type
CYBLE_EVT_SCPSS_SCAN_INT_WIN_CHAR_WRITE	ScPS Client - Read Response for Scan Interval Window Characteristic Value of Scan Parameters Service. The parameter of this event is a structure of CYBLE_SCPS_CHAR_VALUE_T type
CYBLE_EVT_SCPSC_NOTIFICATION	ScPS Client - Scan Refresh Characteristic Notification was received. The parameter of this event is a structure of CYBLE_SCPS_CHAR_VALUE_T type
CYBLE_EVT_SCPSC_READ_DESCR_RESPONSE	ScPS Client - Read Response for Scan Refresh Characteristic Descriptor Read Request. The parameter of this event is a structure of CYBLE_SCPS_DESCR_VALUE_T type
CYBLE_EVT_SCPSC_WRITE_DESCR_RESPONSE	ScPS Client - Write Response for Scan Refresh Client Characteristic Configuration Descriptor Value. The parameter of this event is a structure of CYBLE_SCPS_DESCR_VALUE_T type
CYBLE_EVT_TPSS_NOTIFICATION_ENABLED	TPS Server - Notification for Tx Power Level Characteristic was enabled. The parameter of this event is a structure of CYBLE_TPS_CHAR_VALUE_T type
CYBLE_EVT_TPSS_NOTIFICATION_DISABLED	TPS Server - Notification for Tx Power Level Characteristic was disabled. The parameter of this event is a structure of CYBLE_TPS_CHAR_VALUE_T type

Members	Description
CYBLE_EVT_TPSC_NOTIFICATION	TPS Client - Tx Power Level Characteristic Notification. The parameter of this event is a structure of CYBLE_TPS_CHAR_VALUE_T type
CYBLE_EVT_TPSC_READ_CHAR_RESPONSE	TPS Client - Read Response for Tx Power Level Characteristic Value Read Request. The parameter of this event is a structure of CYBLE_TPS_CHAR_VALUE_T type
CYBLE_EVT_TPSC_READ_DESCR_RESPONSE	TPS Client - Read Response for Tx Power Level Client Characteristic Configuration Descriptor Value Read Request. The parameter of this event is a structure of CYBLE_TPS_DESCR_VALUE_T type
CYBLE_EVT_TPSC_WRITE_DESCR_RESPONSE	TPS Client - Write Response for Tx Power Level Characteristic Descriptor Value Write Request. The parameter of this event is a structure of CYBLE_TPS_DESCR_VALUE_T type

Functional Description

Operation Flow

A typical application code consists of three separate stages: Initialization, Normal operation, and Low power operation.


System Initialization
Power-on
Platform initialization
Wake-up from hibernate
Yes
No
Execute hibernate wakeup sequence
BLE-Component Initialization
Error handling – Application can decide to go power down mode
BLE-Init Success
No
Yes
System Normal Operation
No
BLE-Establish Connection
Application to call process BLE events API at least once in BLE connection event period.
Connected Success
Device connected
Yes
Yes
Process BLE Events
- Process received packet & status of previous transfer
- Scan sensors
Yes
Any data to Tx
Yes
Call BLE component's Tx function
No
No
Pending Event
No
System low power modes
No sensor / BLE event till Deep sleep timeout
No
Go to sleep mode
No
Yes
Execute deep sleep wakeup sequence
Go to deep sleep mode
Event Occurred
Yes

Once the Component is initialized, it enters normal operation and periodically enters various degrees of low power operation to conserve power. Hence initialization should only happen at system power-up, and the Component should operate between normal mode and low power mode afterwards.

System Initialization

The initialization stage happens at system power-up or when waking from system hibernation. This stage sets up the platform and the Component parameters. The application code should also start the Component and set up the callback functions for the event callbacks that will happen in the other modes of operation.

System Normal Operation

Upon successful initialization of the BLE Component or hibernate wakeup sequence, the Component enters normal mode. Normal operation first establishes a BLE connection if it is not already connected. It should then process all pending BLE events by checking the stack status. This is accomplished by calling CyBle_ProcessEvents(). When all events have been processed, it can transmit any data that need to be communicated and enters low power operation unless there is another pending event. In such a case, it should execute the normal operation flow again. Processing of BLE events should be performed at least once in a BLE connection event period. The BLE connection event is configured by application using the customizer.

System Low power Operation

When there are no pending interrupts in Normal operation, the Component should be placed in low power mode. It should first enter sleep mode. After a certain application defined timeout, you may place the Component in Deep Sleep Mode. If an event happens at any time in low power mode, it should re-enter normal operation.

Note The MCU and BLE Sub-System (BLESS) have separate power modes and are able to go to different power modes independent of each other. The check marks in the the following table show the possible combination of power modes of MCU and BLESS.

BLESS Power Modes	PSoC 4200-BL, PRoC 4200-BL MCUs Power Modes				
	Active	Sleep	Deep Sleep	Hibernate	Off
Active (idle/Tx/Rx)	✓	✓			
Sleep	✓	✓			
Deep Sleep (ECO off)	✓	✓	✓		
Off				✓	✓

Callback Functions

The BLE Component requires that you define a callback function for handling BLE stack events. This is passed as a parameter to the CyBle_Start() API. The callback function is of type CYBLE_CALLBACK_T, as defined by:

```
void (* CYBLE_CALLBACK_T)(uint32 eventCode, void *eventParam);
```

- eventCode: The stack event code
- eventParam : Stack event parameters

The callback function should then evaluate the eventCode (and eventParam for certain events) and provide stack event-specific actions. Hence the events are used to build your application specific state machine for general events such as advertisement, scan, connection and timeout. Refer to the BLE Common Events section for the BLE stack events.

Similarly, you will need to provide a callback function for each Service that you wish to use. This function is also of type CYBLE_CALLBACK_T and is passed as a parameter to the Service-specific callback registration function. The callback function is used to evaluate the Service-specific events and to take appropriate action as defined by your application. Then a Service specific state machine can be built using these events. Refer to the BLE Service-Specific Events section for the BLE Service-specific events.

Device Bonding

The BLE Component will store the link key of a connection after pairing with the remote device. If a connection is lost and re-established, the devices will use the previously stored key for the connection.

The BLE stack will update the bonding data in RAM while the devices are connected. If the bonding data is to be retained during shutdown, the application can use CyBle_StoreBondingData() API to write the bonding data from RAM to the dedicated Flash location, as defined by the Component. Refer to the BLE_HID_Keyboard example project for usage details.

Notes

- The Flash write modifies the IMO of the chip to 48 MHz temporarily during the write cycle. Therefore, you should only perform the bonding data Flash storage while the BLE devices are disconnected, because the change in IMO will disrupt the communication. Likewise, you should either temporarily halt all peripherals running off of the IMO or compensate for the brief frequency change during the Flash write cycle.
- If your design is configured to run at 48 MHz, then the IMO does not change and does not affect other peripherals. However, the Flash write is a blocking call and may disrupt the BLE communication. Therefore, it is advisable to perform the Flash write while the devices are disconnected.

LFCLK configuration

The LFCLK configuration as set in the **Clocks** tab of the Design-Wide Resources (*<project>.cydwr*) file affects the BLE Component's ability to operate in Deep Sleep Mode. If the WCO is chosen, then the Component Deep Sleep Mode is available for use. However, if the ILO is chosen, then the Component cannot enter Deep Sleep.

Note The LFCLK is used in the BLE Component only during Deep Sleep Mode and hence the ILO inaccuracy does not affect the BLE communication.

Unsupported Features

The BLE Component stack does not support the following optional Bluetooth v4.1 protocol features, as listed in Vol 6, Part B, section 4.6 of the specification:

- Connection Parameters Request Procedure (Vol 6, Part B, section 4.6.2)
- Extended Reject Indication (Vol 6, Part B, section 4.6.3)
- Slave-initiated Features Exchange (Vol 6, Part B, section 4.6.4)

The BLE Component does not support automatic Custom Service discovery in a GATT Client implementation.

Resources

The BLE Component uses one BLESS block, two external crystals, interrupt(s), and an optional SCB Block:

Configuration	Resource Type				
	BLESS	SCB [1]	Interrupt	ECO	WCO [2]
Profile Mode	1	-	1	1	1
HCI Mode	1	1	2	1	1

[1] The BLE Component instantiates an SCB Component when configured in HCI Mode. Refer to the SCB Component datasheet for its resource usage.
[2] WCO is optional. It is used if Component deep sleep is required. If WCO is not used, then ILO is used as the LFCLK source.

DC and AC Electrical Characteristics

Specifications are valid for –40 °C ≤ T_A ≤ 85 °C and T_J ≤ 100 °C, except where noted.
Specifications are valid for 1.71 V to 5.5 V, except where noted.

Parameter	Description	Min	Typ	Max	Units	Details/Conditions
RF Receiver Specification						
RXS, IDLE	RX sensitivity with idle transmitter	–	–89	–	dBm	
	RX sensitivity with idle transmitter excluding Balun loss	–	–91	–	dBm	Guaranteed by design simulation
RXS, DIRTY	RX sensitivity with dirty transmitter	–	–87	–70	dBm	RF-PHY Specification (RCV-LE/CA/01/C)
RXS, HIGHGAIN	RX sensitivity in high-gain mode with idle transmitter	–	–91	–	dBm	
PRXMAX	Maximum input power	–10	–1	–	dBm	RF-PHY Specification (RCV-LE/CA/06/C)
CI1	Cochannel interference, Wanted signal at –67 dBm and Interferer at FRX	–	9	21	dB	RF-PHY Specification (RCV-LE/CA/03/C)
CI2	Adjacent channel interference Wanted signal at –67 dBm and Interferer at FRX ±1 MHz	–	3	15	dB	RF-PHY Specification (RCV-LE/CA/03/C)
CI3	Adjacent channel interference Wanted signal at –67 dBm and Interferer at FRX ±2 MHz	–	–29	–	dB	RF-PHY Specification (RCV-LE/CA/03/C)
CI4	Adjacent channel interference Wanted signal at –67 dBm and Interferer at ≥FRX ±3 MHz	–	–39	–	dB	RF-PHY Specification (RCV-LE/CA/03/C)
CI5	Adjacent channel interference Wanted Signal at –67 dBm and Interferer at Image frequency (F_{IMAGE})	–	–20	–	dB	RF-PHY Specification (RCV-LE/CA/03/C)
CI3	Adjacent channel interference Wanted signal at –67 dBm and Interferer at Image frequency (F_{IMAGE} ± 1 MHz)	–	–30	–	dB	RF-PHY Specification (RCV-LE/CA/03/C)
OBB1	Out-of-band blocking, Wanted signal at –67 dBm and Interferer at F = 30–2000 MHz	–30	–27	–	dBm	RF-PHY Specification (RCV-LE/CA/04/C)
OBB2	Out-of-band blocking, Wanted signal at –67 dBm and Interferer at F = 2003–2399 MHz	–35	–27	–	dBm	RF-PHY Specification (RCV-LE/CA/04/C)
OBB3	Out-of-band blocking, Wanted signal at –67 dBm and Interferer at F = 2484–2997 MHz	–35	–27	–	dBm	RF-PHY Specification (RCV-LE/CA/04/C)

Parameter	Description	Min	Typ	Max	Units	Details/Conditions
OBB4	Out-of-band blocking, Wanted signal a –67 dBm and Interferer at F = 3000–12750 MHz	–30	–27	–	dBm	RF-PHY Specification (RCV-LE/CA/04/C)
IMD	Intermodulation performance Wanted signal at –64 dBm and 1-Mbps BLE, third, fourth, and fifth offset channel	–50	–	–	dBm	RF-PHY Specification (RCV-LE/CA/05/C)
RXSE1	Receiver spurious emission 30 MHz to 1.0 GHz	–	–	–57	dBm	100-kHz measurement bandwidth ETSI EN300 328 V1.8.1
RXSE2	Receiver spurious emission 1.0 GHz to 12.75 GHz	–	–	–47	dBm	1-MHz measurement bandwidth ETSI EN300 328 V1.8.1
RF Transmitter Specifications						
TXP, ACC	RF power accuracy	–	–	±4	dB	
TXP, RANGE	RF power control range	–	20	–	dB	
TXP, 0dBm	Output power, 0-dB Gain setting (PA7)	–4	0	3	dBm	
TXP, MAX	Output power, maximum power setting (PA10)	–1	3	6	dBm	
TXP, MIN	Output power, minimum power setting (PA1)	–	–18	–	dBm	
F2AVG	Average frequency deviation for 10101010 pattern	185	–	–	kHz	RF-PHY Specification (TRM-LE/CA/05/C)
F1AVG	Average frequency deviation for 11110000 pattern	225	250	275	kHz	RF-PHY Specification (TRM-LE/CA/05/C)
EO	Eye opening = ∆F2AVG/∆F1AVG	0.8	–	–		RF-PHY Specification (TRM-LE/CA/05/C)
FTX, ACC	Frequency accuracy	–150	–	150	kHz	RF-PHY Specification (TRM-LE/CA/06/C)
FTX, MAXDR	Maximum frequency drift	–50	–	50	kHz	RF-PHY Specification (TRM-LE/CA/06/C)
FTX, INITDR	Initial frequency drift	–20	–	20	kHz	RF-PHY Specification (TRM-LE/CA/06/C)
FTX, DR	Maximum drift rate	–20	–	20	kHz/ 50 µs	RF-PHY Specification (TRM-LE/CA/06/C)
IBSE1	In-band spurious emission at 2-MHz offset	–	–	–20	dBm	RF-PHY Specification (TRM-LE/CA/03/C)
IBSE2	In-band spurious emission at ≥3-MHz offset	–	–	-30	dBm	RF-PHY Specification (TRM-LE/CA/03/C)
TXSE1	Transmitter spurious emissions (average), <1.0 GHz	–	–	-55.5	dBm	FCC-15.247

Parameter	Description	Min	Typ	Max	Units	Details/Conditions
TXSE2	Transmitter spurious emissions (average), >1.0 GHz	–	–	-41.5	dBm	FCC-15.247
RF Current Specifications						
IRX	Receive current in normal mode	–	18.7	–	mA	
IRX_RF	Radio receive current in normal mode	–	16.4	–	mA	Measured at V_{DDR}
IRX, HIGHGAIN	Receive current in high-gain mode	–	21.5	–	mA	
ITX, 3dBm	TX current at 3-dBm setting (PA10)	–	20	–	mA	
ITX, 0dBm	TX current at 0-dBm setting (PA7)	–	16.5	–	mA	
ITX_RF, 0dBm	Radio TX current at 0 dBm setting (PA7)	–	15.6	–	mA	Measured at V_{DDR}
ITX_RF, 0dBm	Radio TX current at 0 dBm excluding Balun loss	–	14.2	–	mA	Guaranteed by design simulation
ITX,-3dBm	TX current at –3-dBm setting (PA4)	–	15.5	–	mA	
ITX,-6dBm	TX current at –6-dBm setting (PA3)	–	14.5	–	mA	
ITX,-12dBm	TX current at –12-dBm setting (PA2)	–	13.2	–	mA	
ITX,-18dBm	TX current at –18-dBm setting (PA1)	–	12.5	–	mA	
Iavg_1sec, 0dBm	Average current at 1-second BLE connection interval	–	18.9	–	µA	TXP: 0 dBm; ±20-ppm master and slave clock accuracy.
Iavg_4sec, 0dBm	Average current at 4-second BLE connection interval	–	6.25	–	µA	TXP: 0 dBm; ±20-ppm master and slave clock accuracy.
General RF Specifications						
FREQ	RF operating frequency	2400	–	2482	MHz	
CHBW	Channel spacing	–	2	–	MHz	
DR	On-air data rate	–	1000	–	kbps	
IDLE2TX	BLE.IDLE to BLE. TX transition time	–	120	140	µs	
IDLE2RX	BLE.IDLE to BLE. RX transition time	–	75	120	µs	
RSSI Specifications						
RSSI, ACC	RSSI accuracy	–	±5	–	dB	
RSSI, RES	RSSI resolution	–	1	–	dB	
RSSI, PER	RSSI sample period	–	6	–	µs	

The following table summarizes the different measurements of the time taken by the BLE firmware stack to perform / initiate different BLE operations. The measurements have been performed with IMO set to 12 MHz and connection interval set to 7.5ms.

Operation	Duration (µs)
'CyBle_ProcessEvents' execution time (Best case)	4
Worst case BLE ISR Execution time	83
Start Scan execution time	1860
Passive Scan receive advertisement duration	168
Active Scan receive {Advertisement + Scan Response} duration	320
Read request processing time on GATT Server (MTU = 512 Bytes)	23600
Write request processing time on GATT Server (MTU = 512 Bytes)	16800
Connection time on GAP Central	2690
Connection time on GAP Peripheral	1300
Start advertisement execution time	2960
'CyBle_EnterLPM' execution time	342
Notification processing time on GATT Server (MTU = 23 Bytes)	900
Write command processing time on GATT Server (MTU = 23 Bytes)	930

Component Changes

This section lists the major changes in the Component from the previous version.

Version	Description of Changes	Reason for Changes / Impact
1.0.b	Support of the following profiles was added to the component: • Phone Alert Status Profile (PASP) • Location and Navigation Profile (LNP) • Cycling Speed and Cadence Profile (CSCP) • Cycling Power Profile (CPP)	New feature-support added.
	The `CYBLE_L2CAP_COMMAND_REJ_REASON_T` event was renamed to `CYBLE_EVT_L2CAP_COMMAND_REJ`.	The event was renamed to be consistent with other event name formats.
	The `CYBLE_EVT_GAP_RESOLVE_PVT_ADDR_VERIFY_CNF` event was removed.	The event became obsolete.

Version	Description of Changes	Reason for Changes / Impact

	The following members of the `CYBLE_API_RESULT_T` structure were deprecated:
	`CYBLE_ERROR_GATT_DB_INVALID_OFFSET,`
	`CYBLE_ERROR_GATT_DB_NULL_PARAMETER_NOT_ALLOWED,`
	`CYBLE_ERROR_GATT_DB_UNSUPPORTED_GROUP_TYPE,`
	`CYBLE_ERROR_GATT_DB_INSUFFICIENT_BUFFER_LEN,`
	`CYBLE_ERROR_GATT_DB_MORE_MATCHING_RESULT_FOUND,`
	`CYBLE_ERROR_GATT_DB_NO_MATCHING_RESULT,`
	`CYBLE_ERROR_GATT_DB_HANDLE_NOT_FOUND,`
	`CYBLE_ERROR_GATT_DB_HANDLE_NOT_IN_RANGE,`
	`CYBLE_ERROR_GATT_DB_HANDLE_IN_GROUP_RANGE,`
	`CYBLE_ERROR_GATT_DB_INVALID_OPERATION,`
	`CYBLE_ERROR_GATT_DB_UUID_NOT_IN_BT_SPACE,`
	`CYBLE_ERROR_GATT_DB_INVALID_ATTR_HANDLE,`
	`CYBLE_ERROR_GATT_DB_INSUFFICIENT_SECURITY,`
	`CYBLE_ERROR_GATT_DB_INSUFFICIENT_ENC_KEY_SIZE,`
	`CYBLE_ERROR_GATT_DB_INVALID_INSTANCE,`
	`CYBLE_ERROR_GATT_DB_INCORRECT_UUID_FRMT,`
	`CYBLE_ERROR_GATT_DB_UUID_FRMT_UNSUPPORTED,`
	`CYBLE_ERROR_GATT_DB_TYPE_MISMATCH,`
	`CYBLE_ERROR_GATT_DB_INSUFFICIENT_ENCRYPTION,`
	`CYBLE_ERROR_L2CAP_NOT_ENOUGH_CREDITS`
		The elements weren't used as return values in any of the API functions.

	Removed WDT from the BLE Component.	In the preliminary release of the BLE Component, the protocol procedure timeout functionality was implemented using the WDT. For the production release, the Component was optimized to use the BLESS Link Layer timer.
	Edits to the datasheet.	Update Configure dialog screen captures. Added the APIs into the datasheet. Added Unsupported Features section. Added characterization data. Addressed all Errata from the preliminary version of the datasheet and removed the section.
1.0.a	Edits to the datasheet.	Added sections to describe WDT counter and interrupt. Clarified descriptions for several APIs and GUIs. Added Errata section. Moved API documentation to separate CHM file. Updated Functional Description section.
1.0	Initial document for new Component.	