BLE IPSP Router and Node 1.0

Features

- BLE Internet Protocol Support Profile (IPSP) in Router and Node role
- Logical Link Control and Adaptation Protocol (L2CAP) data transfer
- Deep Sleep mode support
- LED status indication

Development Kit Configuration

Default CY8CKIT-042 BLE Pioneer Kit configuration.

General Description

This example project demonstrates the Internet Protocol Support Profile operation of the Bluetooth Low Energy (BLE) PSoC Creator Component. The example project requires CY8CKIT-042-BLE Pioneer Kit.

This example demonstrates how to setup an IPv6 communication infrastructure between two devices over a BLE transport using L2CAP channel. Creation and transmission of IPv6 packets over BLE is not part of this example.

The example consists of two projects: IPSP Router (GAP Central) and IPSP Node (GAP Peripheral). Router sends generated packets with different content to Node in the loop and validates them with the afterwards received data packet. Node simply wraps received data coming from Router, back to the Router.

BLE IPSP Router Project Configuration

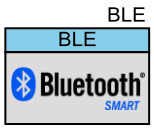
The router example project consists of the following components:

- Bluetooth Low Energy (BLE)
- UART
- Watchdog timer (WDT) that is used as a general timer

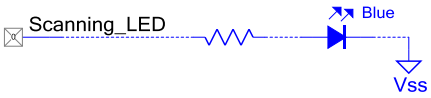
The top design schematic is shown in Figure 1.

Figure 1. Top design schematic

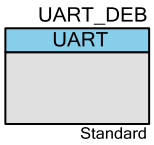
BLE IPSP Router Example Project



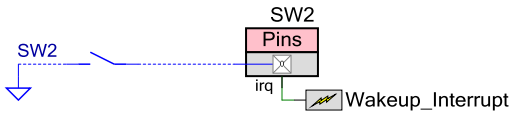
BLE component is configured as IPSP router



The blue LED is used to indicate that the device is scanning.



UART is used for transmitting the debug information.



The button is used to wake the device up from the hibernate mode.

The output pins are used to reflect the line signal output on the LED. The input pin is configured to the resistive pull up mode and is used to wake a device from the hibernate mode.

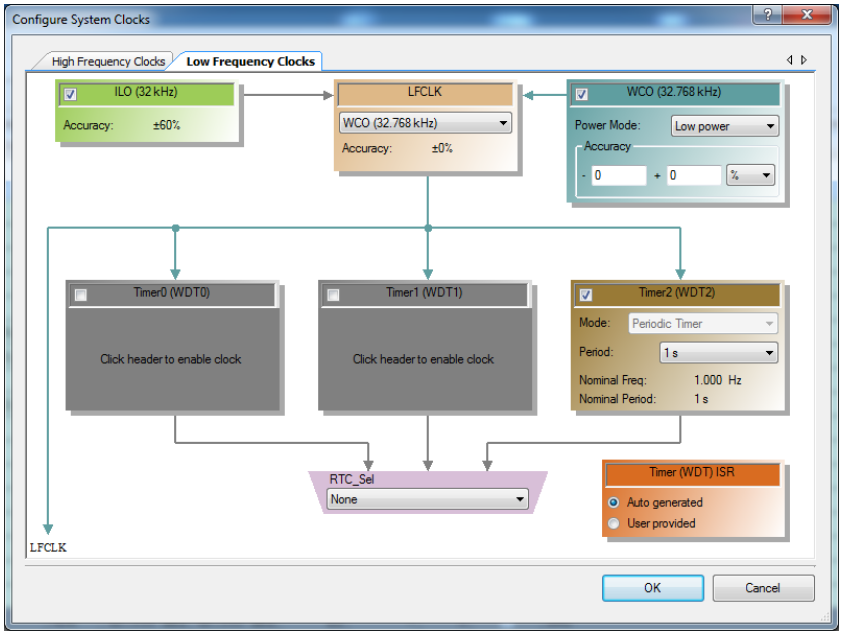
UART

This component is used for printing debug information and scan commands from a terminal.

Watch Dog Timer (WDT)

The WDT timer works over low power Deep Sleep mode, therefore it is used as a general timer for LED indication and simulation purpose. WDT Timer2 is configured in the Low Frequency Clocks tab of the Clocks configuration in the Design Wide Resources (DWR).

Figure 2. WDT Timer2 settings



Bluetooth Low Energy (BLE)

The BLE component is configured as Internet Protocol Support Profile in Router role operation. L2CAP MTU size is set to 1280.

Figure 3. GATT settings

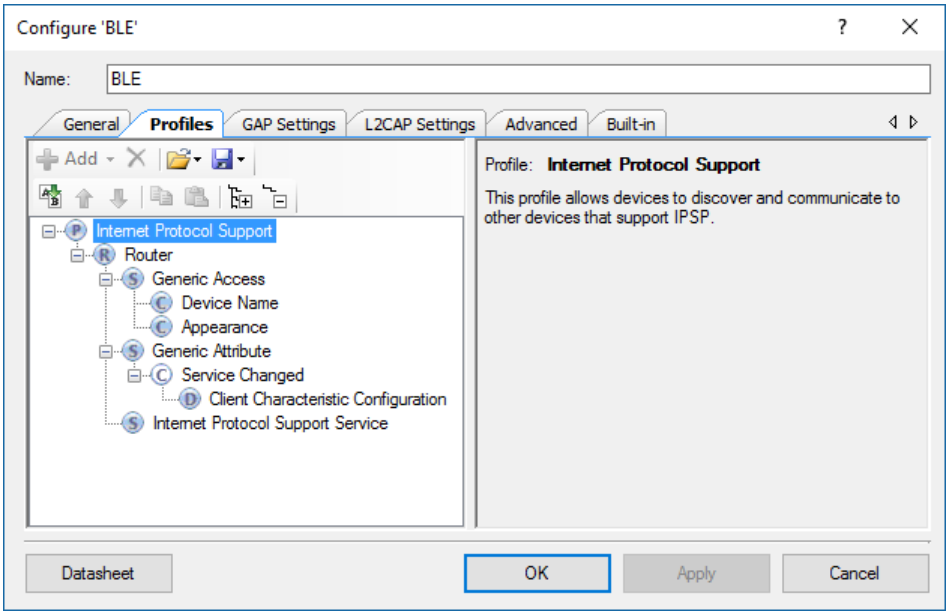


Figure 4. GAP settings

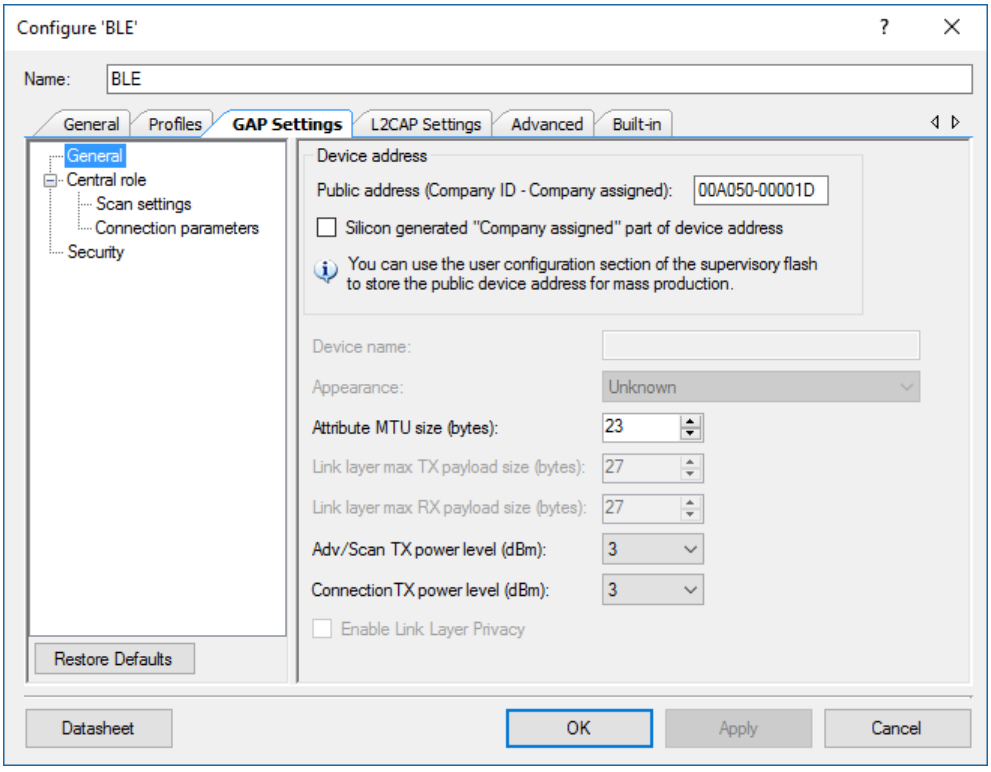


Figure 5. GAP settings -> Scan settings

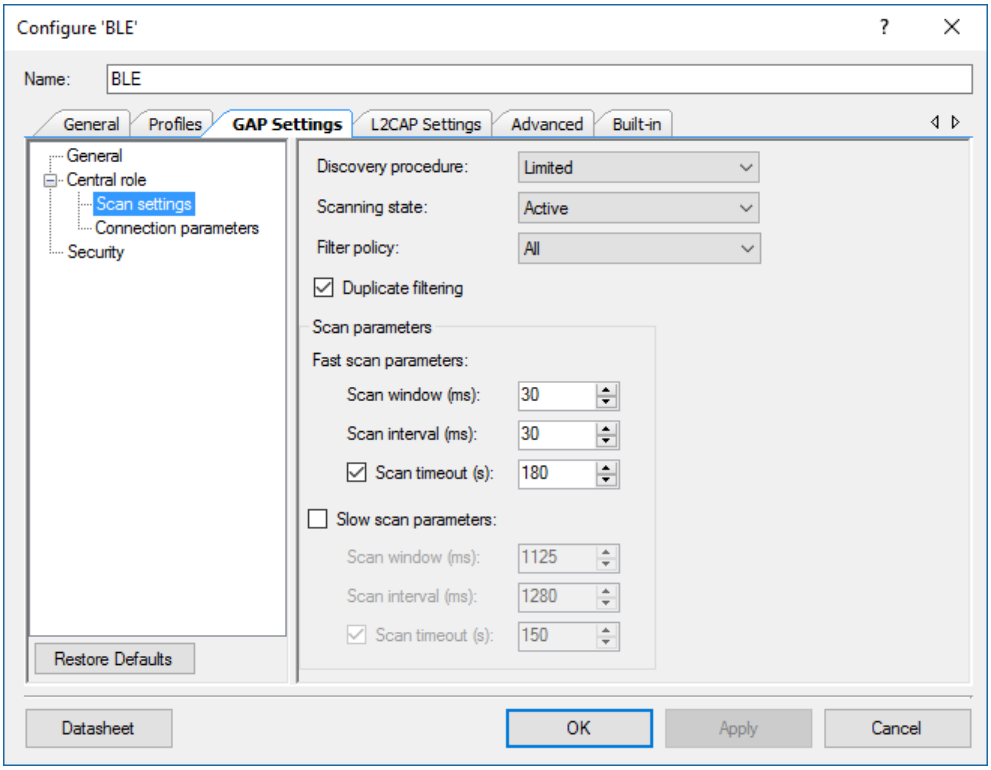


Figure 6. Security settings

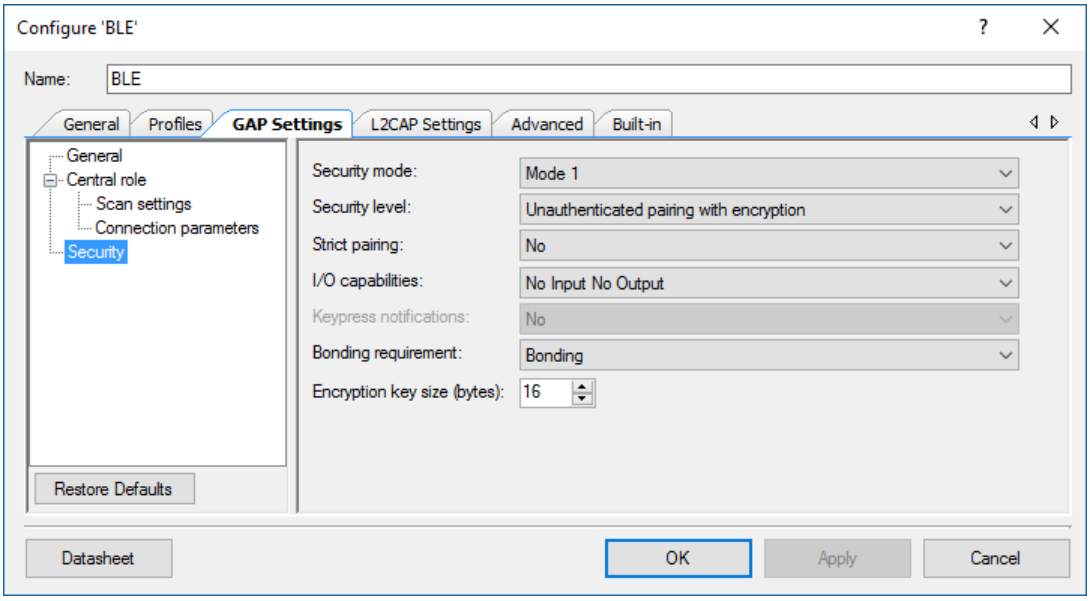
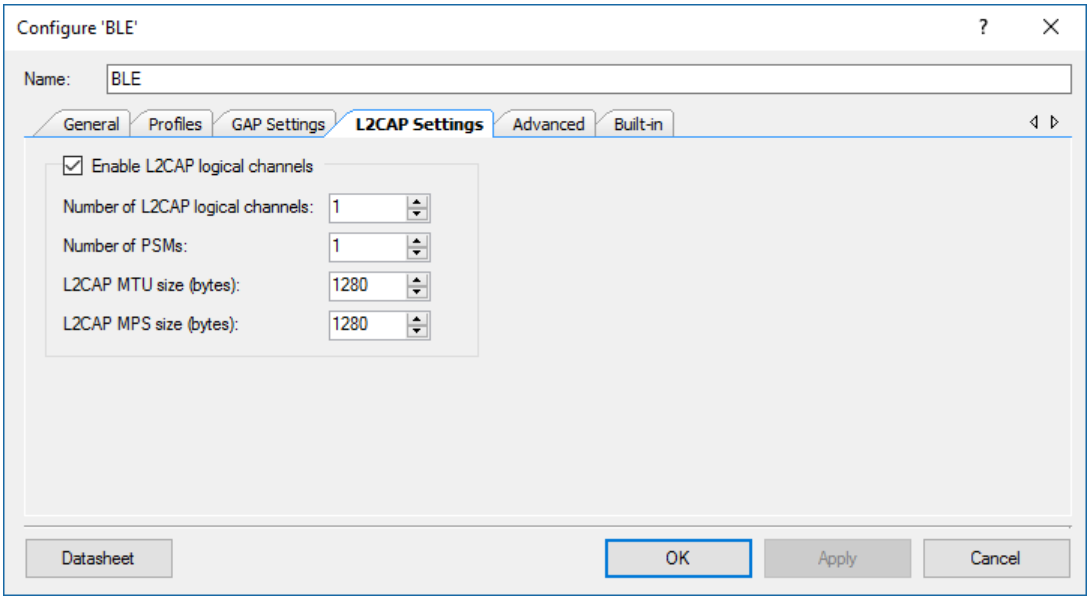


Figure 7. L2CAP Settings



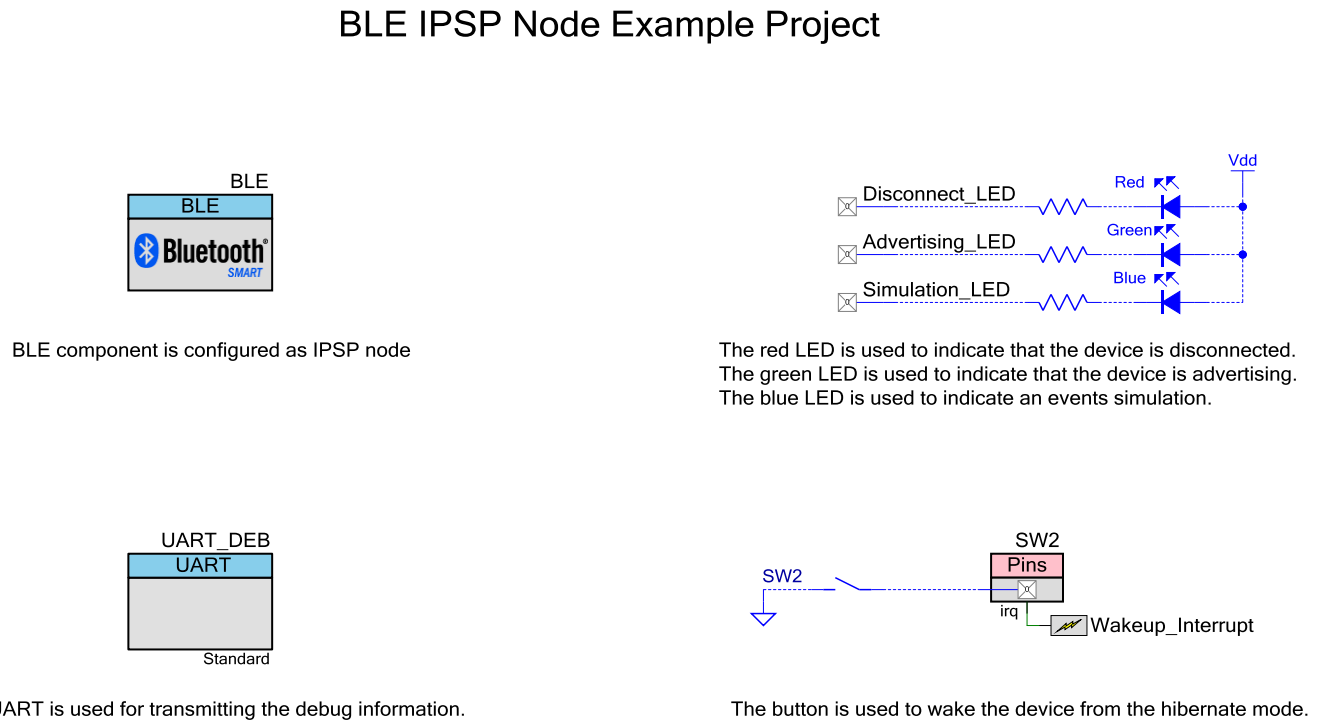
BLE IPSP Node Project Configuration

The node example project consists of the following components:

- Bluetooth Low Energy (BLE)
- UART
- Watchdog timer (WDT) that is used as general timer

The top design schematic is shown in Figure **8**.

Figure 8. Top design schematic



The output pins are used to reflect the line signal output on the LED. The input pin is configured to the resistive pull up mode and is used to wake a device from the hibernate mode.

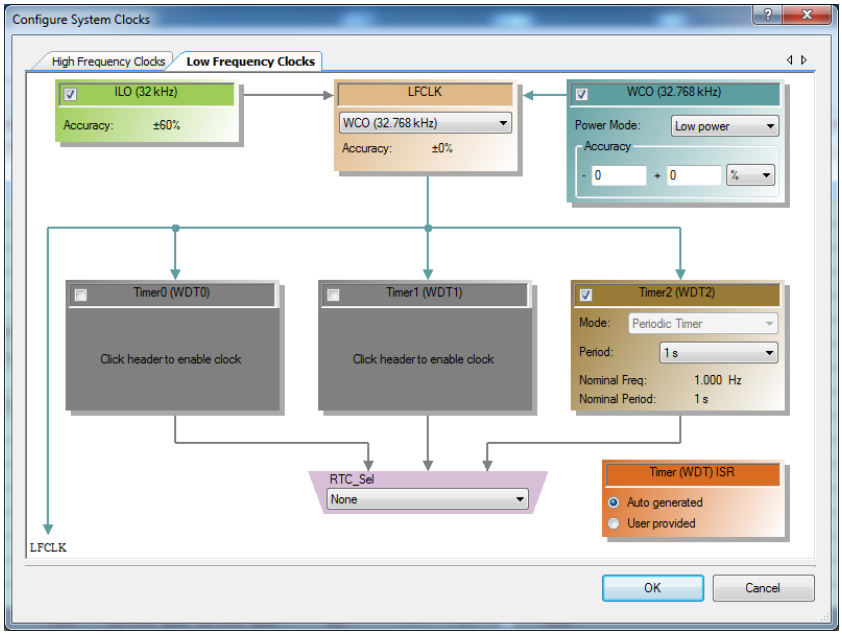
UART

This component is used for printing debug information.

Watch Dog Timer (WDT)

The WDT timer works over low power Deep Sleep mode, therefore it is used as a general timer for LED indication and simulation purpose. WDT Timer2 is configured in the Low Frequency Clocks tab of the Clocks configuration in the Design Wide Resources (DWR).

Figure 9. WDT Timer2 settings



Bluetooth Low Energy (BLE)

The BLE component is configured as Internet Protocol Support Profile in Node role operation. Node Advertisement packet is configured to connectable undirected advertising type with complete local name and IPSS service UUID. L2CAP MTU size is set to 1280.

Figure 10. GATT settings

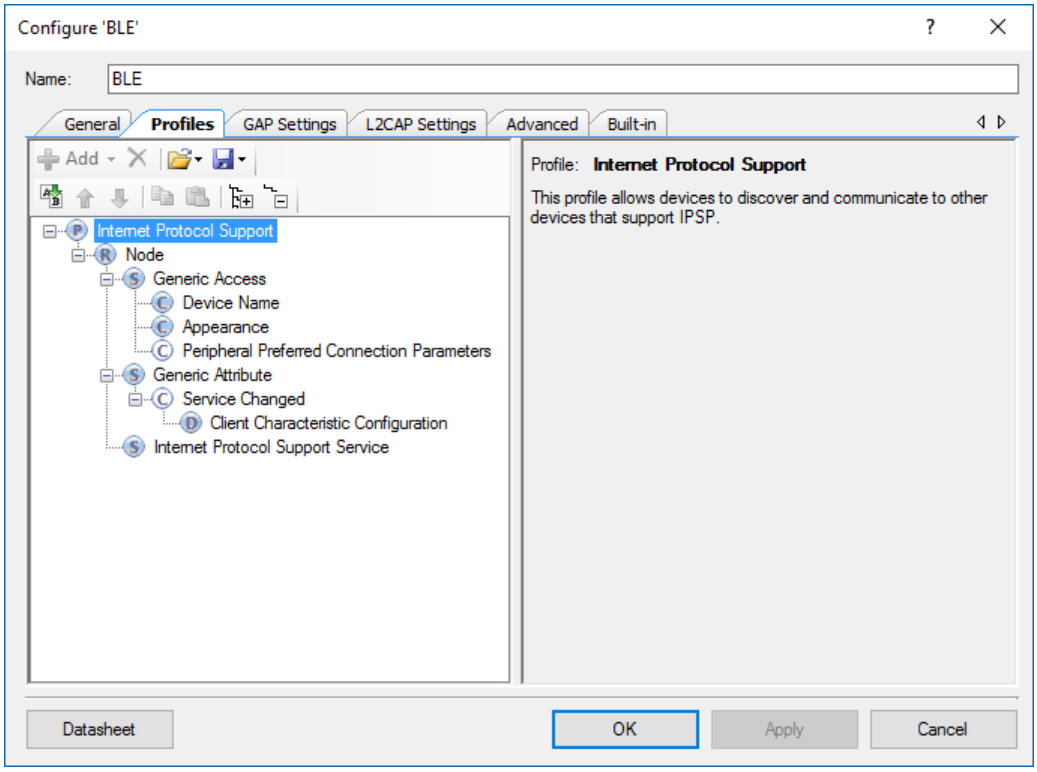


Figure 11. GAP settings

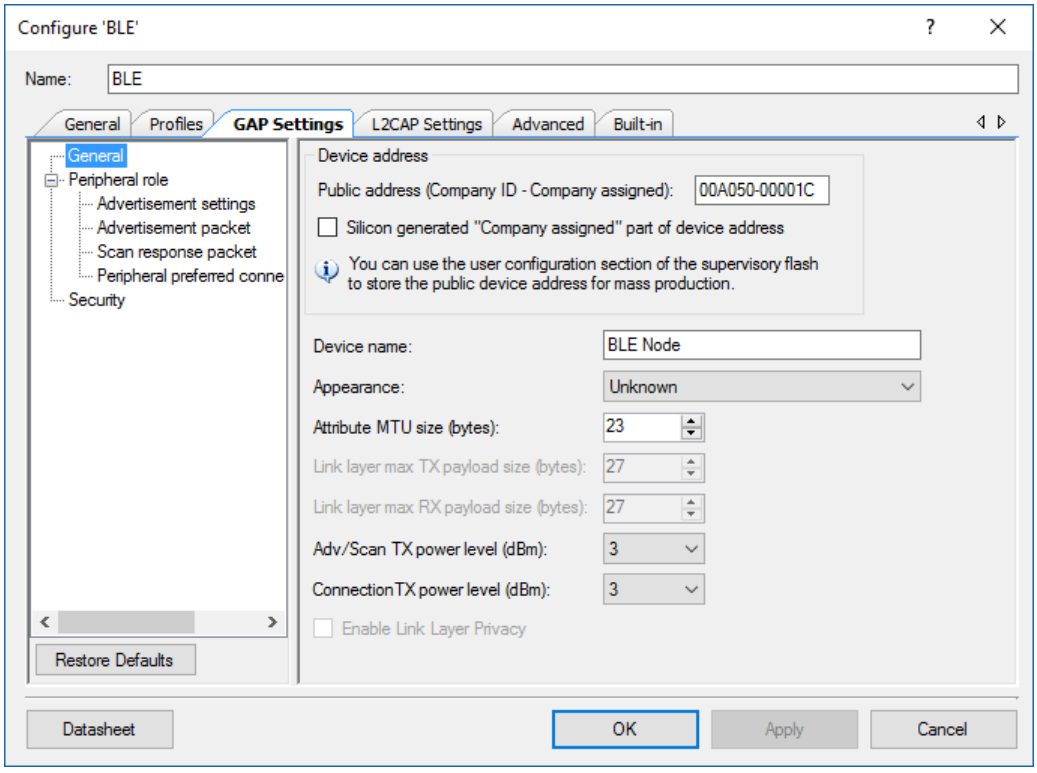


Figure 12. GAP settings -> Advertisement settings

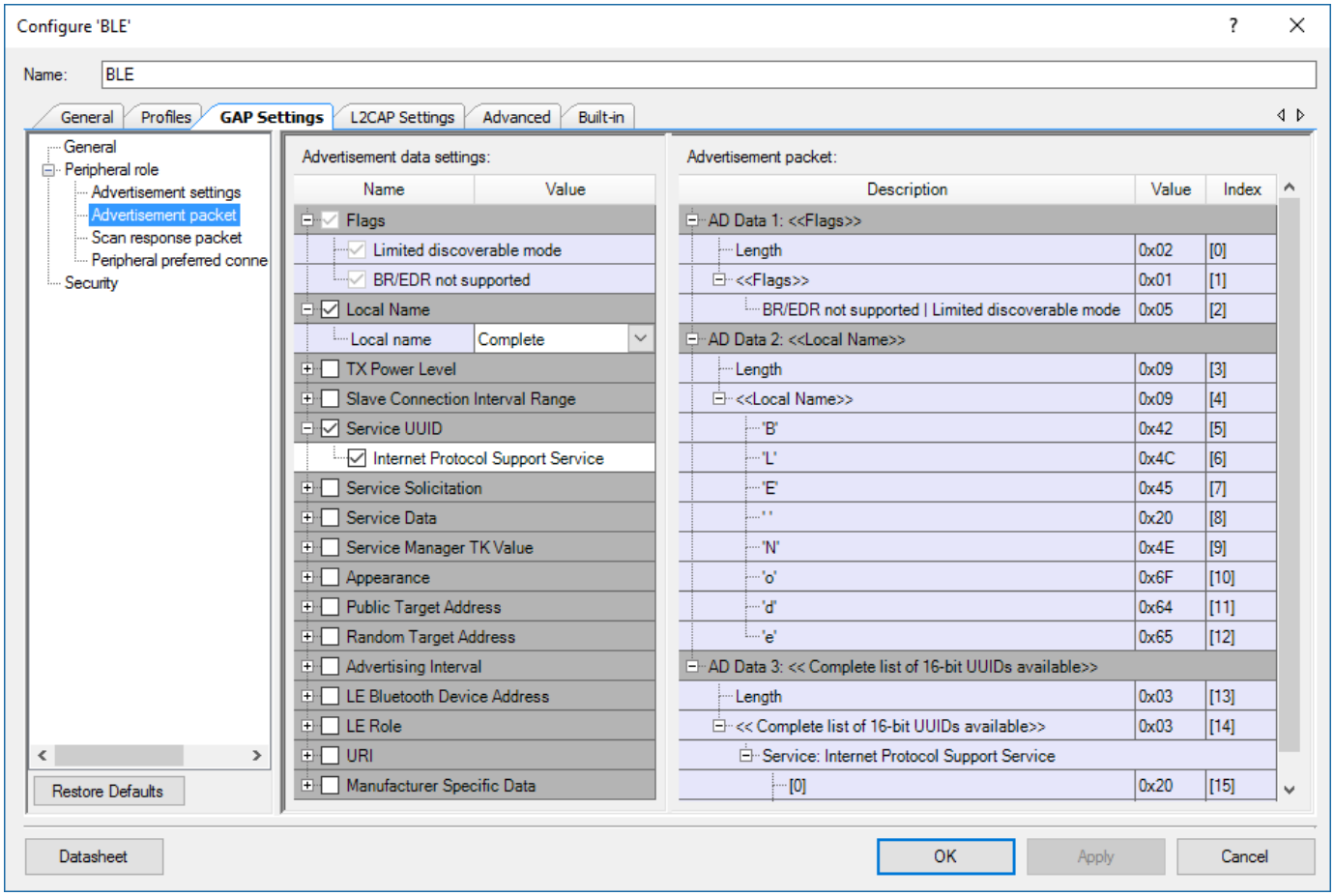
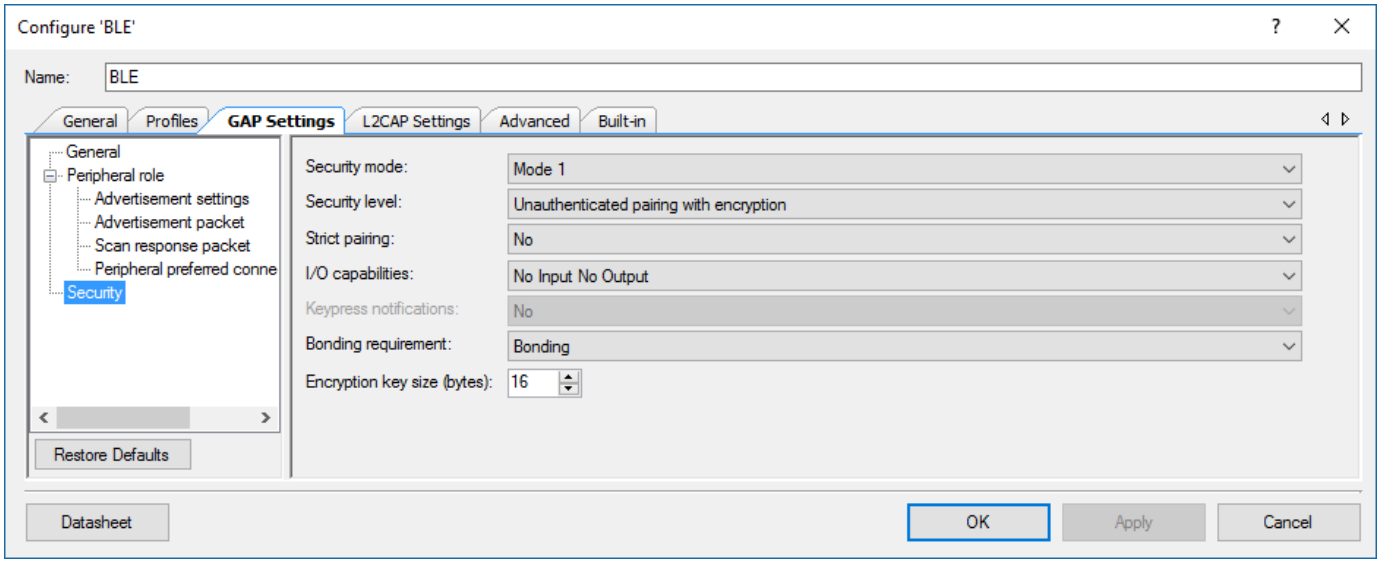


Figure 13. Security settings



BLE IPSP Router Project Description

The project demonstrates the BLE component functionality configured as a Router.
For operation the example project uses a callback function AppCallBack() for receiving the generic and L2CAP events from the BLE Stack.

To start the example project operation, build it and program into development kit BLE Dongle. Right after the startup, the BLE, UART, and ISR components are initialized. After the initialization the BLE component begins its operation that can be seen on the user LED which starts blinking with a blue color. This indicates that the device has started scanning. After 180 seconds timeout, if no peripheral device has been connected, the Router stops scanning. User LED is turned off indicating the disconnection state and the system enters into the hibernate mode. Press the user button on BLE Dongle (SW2) to wake up the system and start re-scanning.

Advertising packets, received during scanning procedure from peripheral devices, are parsed and filtered. Only packets with IPSS service UUID are handled and showed in the debug terminal with the device sequence number as a candidate to connect.

The example project uses UART component for displaying debug information and also for sending commands through the Terminal emulator app. Commands are the procedures which a user can perform. The list of the commands is shown below:

Command	Description
'z'+'#'	Select specific peer device, where '#' – sequence number from advertising packet (0-7). Default value: 0.
'c'	Send connection request to selected peer device.
'd'	Send disconnect request to peer device.
'v'	Cancel connection request.
's'	Start discovery procedure.
'1'	Send Data packet to Node though IPSP channel

The above list is prompted to Terminal emulator when 'h' is entered in the app.

To connect to the Node device, select a specific device by pressing 'z' + device number listed along with the advertising report in the terminal and send a connection request to the device ('c') when the device is advertising.

IPSP protocol multiplexer for L2CAP is registered and the initial Receive Credit Low Mark for Based Flow Control mode is set after CYBLE_EVT_STACK_ON event.
When GAP connection is established, after CYBLE_EVT_GATT_CONNECT_IND event, Router automatically initiates an L2CAP LE credit based connection with a PSM set to LE_PSM_IPSP.
Use '1' command to generate and send first Data packet to Node though IPSP channel. Sent data will be compared with the received data in response packed after CYBLE_EVT_L2CAP_CBFC_DATA_READ event. When no failure is observed, a new packet is generated and sent to the Node automatically. Otherwise the transfer is stopped and "Wraparound failed" message will indicate a failure.

The Router updates the LE credits dynamically, when the credit count goes below the low mark (CYBLE_EVT_L2CAP_CBFC_RX_CREDIT_IND event), to allow continuous transfer of data between the Node and the Router.

LE IPSP Node Project Description

The project demonstrates the BLE component functionality configured as a Node.
For operation the example project uses a callback function - AppCallBack() for receiving the generic and L2CAP events from the BLE Stack. CyBle_GappStartAdvertisement() API is called after CYBLE_EVT_STACK_ON event to start advertising with the packet shown in Figure **12**.

To start the example project operation, build it and program into baseboard of development kit. Right after the startup, the BLE, UART, and ISR components are initialized. After the initialization the BLE component begins its operation that can be seen on the RGB LED which starts blinking with a green color. This indicates that the device has started advertising. After 180 seconds timeout, if no central device has been connected, the Node stops advertising, a red LED is turned on indicating the disconnection state and the system enters into the hibernate mode. Press the mechanical button on CY8CKIT-042 BLE (SW2) to wake up the system and start re-advertising.

When a Client has connected successfully, both red and green LEDs are turned off. Blinking blue LED indicates data packet transaction.

IPSP protocol multiplexer for L2CAP is registered and the initial Receive Credit Low Mark for Based Flow Control mode is set after CYBLE_EVT_STACK_ON event.
The Node automatically accept L2CAP LE credit based connection request with a PSM set to LE_PSM_IPSP after CYBLE_EVT_L2CAP_CBFC_CONN_IND event.
Received data after CYBLE_EVT_L2CAP_CBFC_DATA_READ event are automatically wrapped back to the Router.
The Node updates the LE credits dynamically, when the credit count goes below the low mark (CYBLE_EVT_L2CAP_CBFC_RX_CREDIT_IND event), to allow continuous transfer of data between the Node and the Router.

Expected Results

The BLE IPSP Router project is intended to work in pair with BLE IPSP Node. Two projects send log messages through UART. After starting, the Router project logs Advertising and Scan response reports from Node, for example:

Advertisement report: eventType = 0, peerAddrType - 0, peerBdAddr - #0: 00a05000001C, rssi - -58 dBm

where #0 is a sequence number of the Node device. Use this number after ‘z’ command to select required Node if multiple Node devices are available. Press ‘c’ to connect to Node. Press ‘1’ to start wraparound test. Blue LED on Node indicates data transfer process. In case of wraparound

data validation failure Blue LED will stop blinking and “Wraparound failed” message will appear in Router UART log.

The example log is shown below:

```
BLE IPSP Router Example Project
Stack Version: 2.1.0.11
Bluetooth On, StartScan with addr: 00a05000001d
CYBLE_EVT_GAPC_SCAN_START_STOP, state: 3
Advertisement report: eventType = 0, peerAddrType - 0, peerBdAddr - 0: 00a05000001c, rssi - -29 dBm
CYBLE_EVT_GAPC_SCAN_START_STOP, state: 5
GAPC_END_SCANNING
EVT_GATT_CONNECT_IND: 0, 4
L2CAP channel connection request sent.
EVT_GAP_DEVICE_CONNECTED: connIntv = 7 ms
CYBLE_EVT_L2CAP_CBFC_CONN_CNF: bdHandle=4, lCid=64, responce=0, connParam: mtu=1280, mps=1280, credit=1000
-> CyBle_L2capChannelDataWrite #0
<- EVT_L2CAP_CBFC_DATA_READ: lCid=64, result=0, len=1278
-> CyBle_L2capChannelDataWrite #1
<- EVT_L2CAP_CBFC_DATA_READ: lCid=64, result=0, len=1278
-> CyBle_L2capChannelDataWrite #2
<- EVT_L2CAP_CBFC_DATA_READ: lCid=64, result=0, len=1278
-> CyBle_L2capChannelDataWrite #3
<- EVT_L2CAP_CBFC_DATA_READ: lCid=64, result=0, len=1278
-> CyBle_L2capChannelDataWrite #4
<- EVT_L2CAP_CBFC_DATA_READ: lCid=64, result=0, len=1278
-> CyBle_L2capChannelDataWrite #5
<- EVT_L2CAP_CBFC_DATA_READ: lCid=64, result=0, len=1278
-> CyBle_L2capChannelDataWrite #6
<- EVT_L2CAP_CBFC_DATA_READ: lCid=64, result=0, len=1278
-> CyBle_L2capChannelDataWrite #7
<- EVT_L2CAP_CBFC_DATA_READ: lCid=64, result=0, len=1278
-> CyBle_L2capChannelDataWrite #8
```

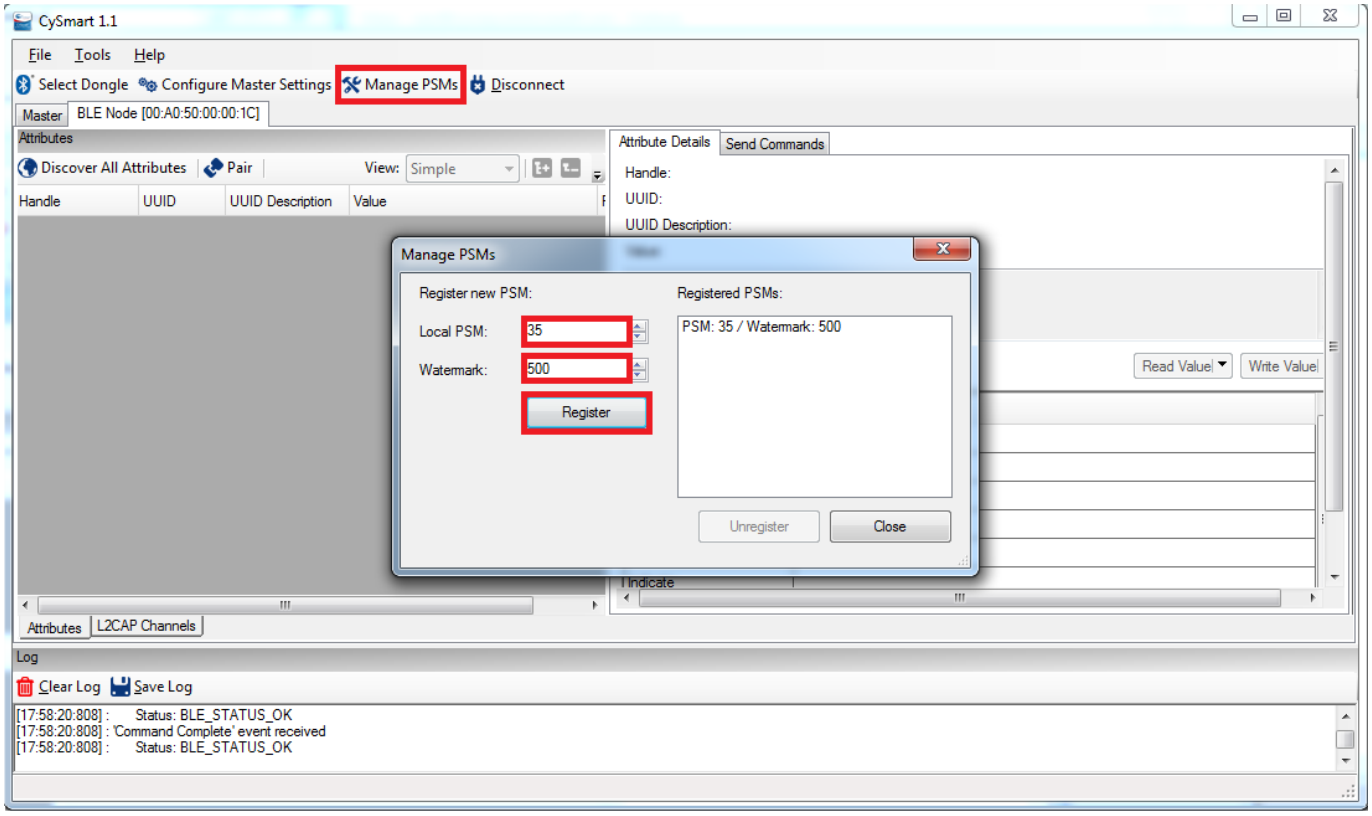
```
BLE Node Example Project
Stack Version: 2.1.0.11
Bluetooth On, StartAdvertisement with addr: 00a05000001c
CYBLE_EVT_GAPP_ADVERTISEMENT_START_STOP, state: 3
EVT_GATT_CONNECT_IND: 0, 4
EVT_GAP_DEVICE_CONNECTED: connIntv = 7 ms
EVT_L2CAP_CBFC_CONN_IND: bdHandle=4, lCid=64, psm=35,connParam mtu=1280, mps=1280, credit=1000 SUCCESSFUL
<- EVT_L2CAP_CBFC_DATA_READ: lCid=64, result=0, len=1278
-> CyBle_L2capChannelDataWrite API result: 0
<- EVT_L2CAP_CBFC_DATA_READ: lCid=64, result=0, len=1278
-> CyBle_L2capChannelDataWrite API result: 0
<- EVT_L2CAP_CBFC_DATA_READ: lCid=64, result=0, len=1278
-> CyBle_L2capChannelDataWrite API result: 0
<- EVT_L2CAP_CBFC_DATA_READ: lCid=64, result=0, len=1278
-> CyBle_L2capChannelDataWrite API result: 0
<- EVT_L2CAP_CBFC_DATA_READ: lCid=64, result=0, len=1278
-> CyBle_L2capChannelDataWrite API result: 0
<- EVT_L2CAP_CBFC_DATA_READ: lCid=64, result=0, len=1278
-> CyBle_L2capChannelDataWrite API result: 0
<- EVT_L2CAP_CBFC_DATA_READ: lCid=64, result=0, len=1278
-> CyBle_L2capChannelDataWrite API result: 0
<- EVT_L2CAP_CBFC_DATA_READ: lCid=64, result=0, len=1278
-> CyBle_L2capChannelDataWrite API result: 0
<- EVT_L2CAP_CBFC_DATA_READ: lCid=64, result=0, len=1278
-> CyBle_L2capChannelDataWrite API result: 0
<- EVT_L2CAP_CBFC_DATA_READ: lCid=64, result=0, len=1278
-> CyBle_L2capChannelDataWrite API result: 0
<- EVT_L2CAP_CBFC_DATA_READ: lCid=64, result=0, len=1278
-> CyBle_L2capChannelDataWrite API result: 0
```

You can use CySmart app on a Windows PC BLE-compatible device as Client for connection to Node.

To use CySmart Windows application as Client:

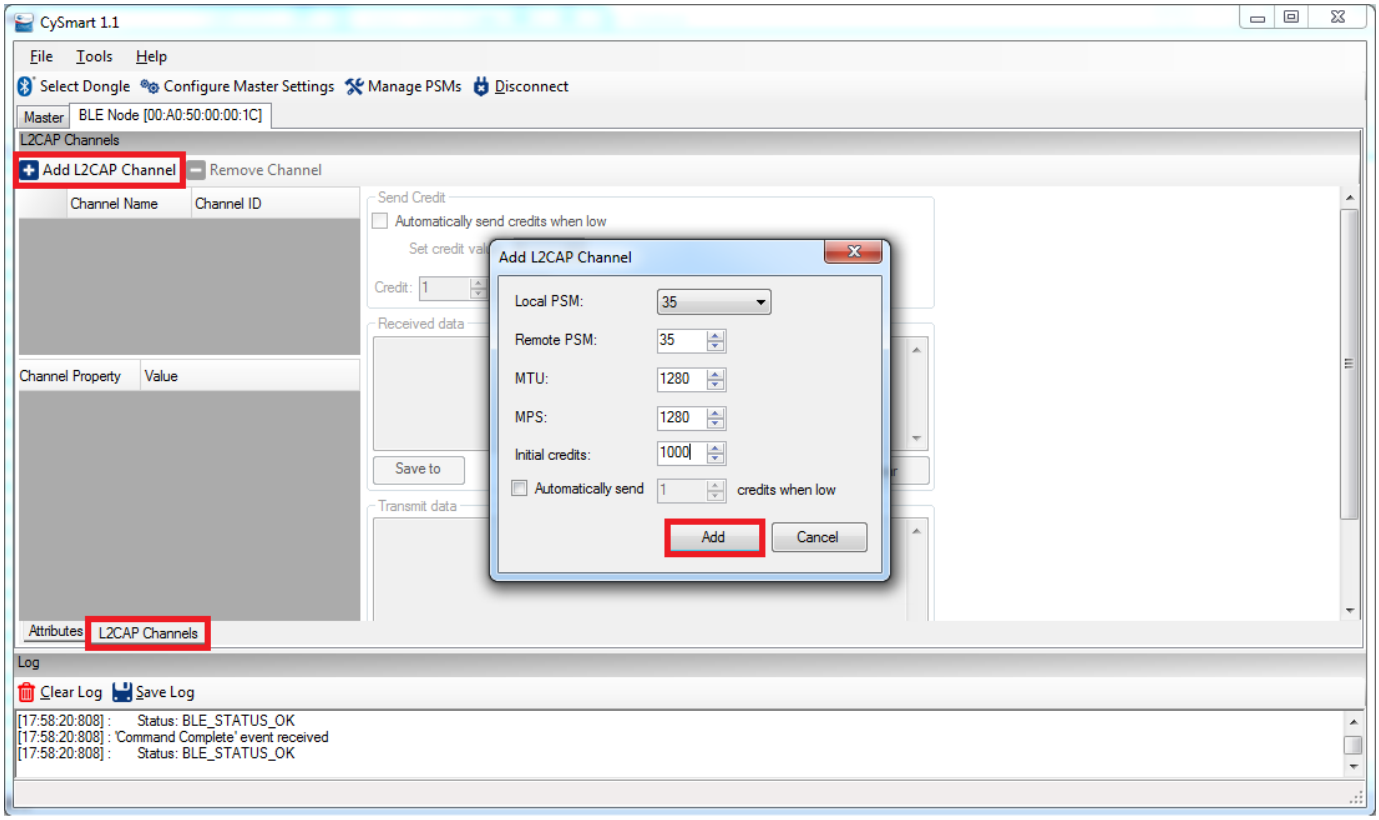
- Connect the CySmart BLE dongle to a USB port on the PC.
- Launch CySmart app and select connected dongle in dialog window.
- Reset development kit to start advertising by pressing SW1 button.
- Click **Start Scan** button to discover available devices.
- Select **BLE Node** in the list of available devices and connect to it.
- Click **Manage PSMs** button. In the "Manage PSMs" window enter **Local PSM:** 35(LE_PSM_IPSP), **Watermark:** 500 and press **Register** button, then close the window.

Figure 14. CySmart Manage PSM window



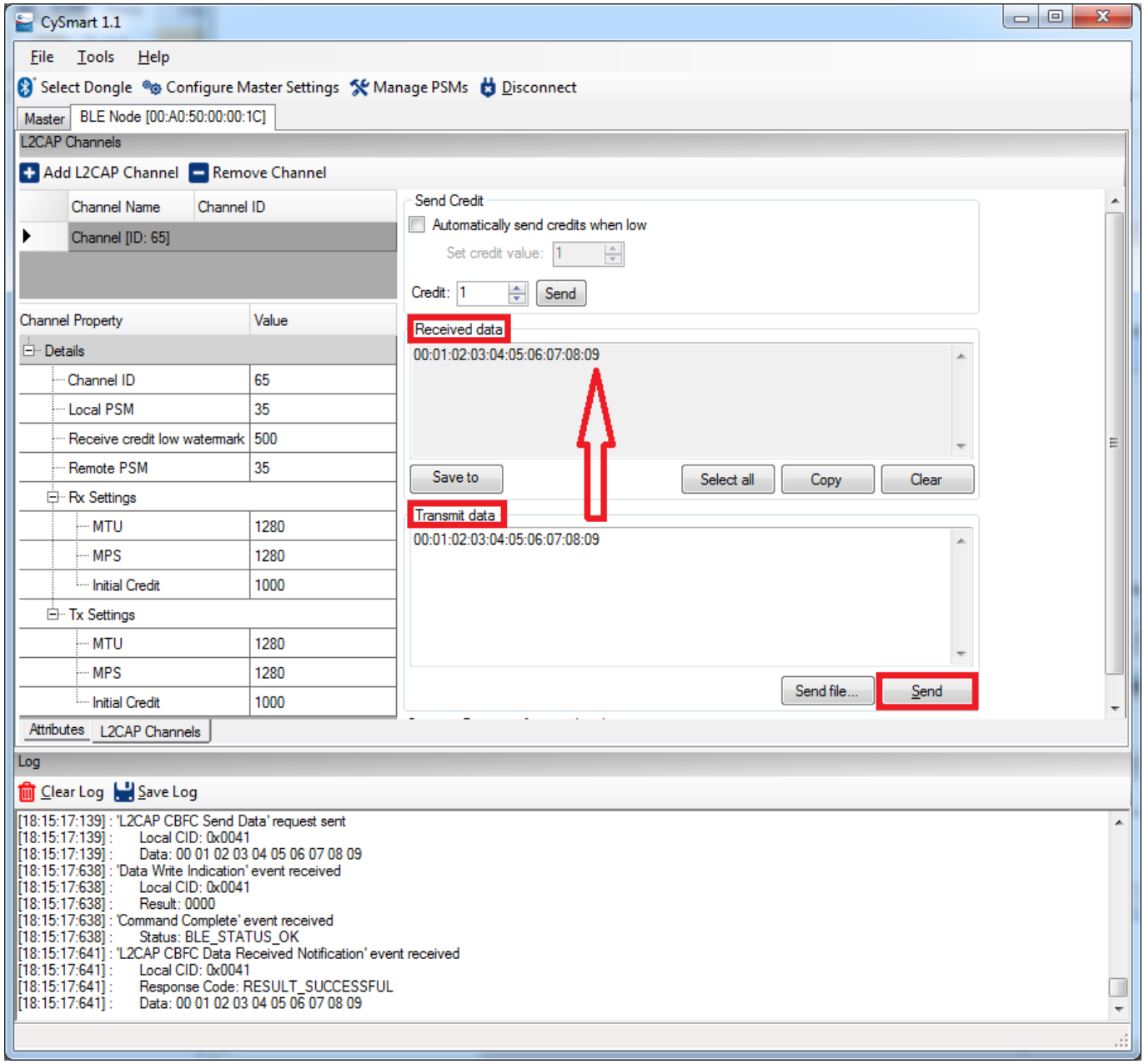
- Select **L2CAP Channels** tab and press **Add L2CAP Channel** to create L2CAP channel. In “Add L2CAP Channel” window select **Local PSM**: 35, enter **Remote PSM**: 35, **MTU**: 1280, **MPS**: 1280, **Initial credits**: 1000 and press **Add** button.

Figure 15. CySmart Add L2CAP Channel window



- Now L2CAP channel is ready to transmit and receive data. Enter some data into **Transmit data** area and press **Send** button. The same data will appear in **Receive data** area

Figure 16. CySmart L2CAP channels window.



If you have problems with usage of CySmart app, refer to CySmart User Guide.

CySmart mobile app does not have IPSP profile support.