# User Manual

# DA14585/586 SDK 6 Porting Guide

## UM-B-082


### Abstract

*This is a guide explaining the porting steps of an application to the latest SDK 6 release.*

# Contents



 

## References

[1] UM-B-079, DA14585/586 Software Platform Reference, User Manual, Dialog Semiconductor.
[2] UM-B-080, DA14585/586 Software Developer's Guide, User manual, Dialog Semiconductor.
[3] DA14580 Data sheet, Dialog Semiconductor.
[4] DA14585 Data sheet, Dialog Semiconductor.
[5] DA14586 Data sheet, Dialog Semiconductor.
[6] Bluetooth Specification Version 5.0, Bluetooth SIG.
[7] ATTDB Interface Specification (RW-BLE-ATTDB-IS), Riviera Waves

 

# 1 Introduction

## 1.1 Scope

This document describes the changes and the steps needed for porting an application that has been developed either under DA14580/581/583 SDK 5.0.4 release, or in a previous DA14585/586 6.0.x SDK release, to the latest DA14585/586 SDK 6 release.

## 1.2 Summary

The latest release SDK 6 for DA14585/586 has changes and improvements that affect the backwards compatibility for existing applications developed in a previous DA14585/586 6.0.x SDK release.

Moreover given also the differences between the DA14580/581/583 and the DA14585/586 devices, it illustrates the changes needed in order to port an application developed in the SDK 5.0.4 for DA14580/581/583.

The following section 2 focuses on 5.0.4 to 6.0.6 SDK porting, while section 3 on the 6.0.4 to 6.0.6 SDK porting essentials.

# 2 Porting from SDK 5.0.4 to 6.0.4

The SDK 5.0.4 (and earlier versions) is explicitly used with the DA14580/581/583 devices while the SDK 6 release is to be used explicitly with the DA14585/586 devices.

The SDK areas where there have been changes that affect the compatibility are due to that:

- The DA14585/586 devices incorporate a different BLE stack compared to the BLE stack used in the DA14580/581/583. The changes in the BLE stack are reflected to the API which has some modifications compared to the SDK 5.0.4.
- The DA14585/586 have a different memory layout compared to the 580/581/583. As a result scatter file has changed.
- The DA14585/586 sleep modes have changed. As a result, the Sleep API has been modified.
- The new SDK supports Data Length extension, which affects the MTU size.

Due to the changes mentioned above and the changes to the APIs caused by them, an application developed with SDK 5.0.4 will not compile and link under SDK 6 release by just importing the application. Even if it compiles it will not produce code able to run as designed.

For more detailed information about the new APIs and the demo projects please refer to [1] and [2].

The steps to port an application developed in SDK 5.0.4 to SDK 6, are explained in detail in the following sections.

## 2.1 Changes in SDK 6

The changes are in Sleep behaviour, due to support the Data Length Extension feature, OTP programming, custom profile, custom database service with 128 bit UUIDs support, attribute permissions, service and characteristic definition, Non-Volatile Data Storage (NVDS), periodic wakeup period and at the maximum valid timeout for Kernel Timer.

### 2.1.1 Sleep Modes

This section describes the software architecture of the sleep modes of DA14585/586. The various modes of operation of the chip are:

- No sleep
- Extended sleep without OTP copy.
- Extended sleep with OTP copy.
- Deep sleep

When the **Extended sleep without OTP copy** is used, the application code shall be stored in a memory source other than OTP memory. After wake-up from sleep, no OTP mirror and no boot from SPI, I2C, UART interface shall happen. The RAM blocks which include code will be retained. The RAM blocks which include retention data will be retained. RAM block 4 is always retained (BLE state).This mode resembles to the extended sleep mode of 580.

When the **Extended sleep with OTP copy** is used, the application code shall be stored in OTP memory. After wake-up from sleep, the OTP mirror will take place. The RAM blocks which include code will not be retained. The RAM blocks which include retention data will be retained. RAM block 4 is always retained (BLE state). This mode resembles the deep sleep mode of 580.

As in the 580 SDK, the user can preselect to use either the Extended sleep mode without OTP copy or the Extended sleep mode with OTP copy, by providing the respective sleep configuration in the user_config.h file.

 

```
/*****************************************
 * Default sleep mode. Possible values are:
 *
 * - ARCH_SLEEP_OFF
 * - ARCH_EXT_SLEEP_ON
 * - ARCH_EXT_SLEEP_OTP_COPY_ON
 *
 *****************************************
 */
```

One of the above sleep modes will be used when the system logic decides that the 585 chip is ready to sleep.

In **Deep sleep**, nothing is retained (all RAM blocks off). The system wakes-up from external interrupt or a POR source and the boot code will be invoked. This mode can be used as the shipping or hibernation mode of the 585 chip. A BLE connection cannot be maintained.

The system enters the deep sleep mode, only if the user explicitly calls the API function:

```
void arch_set_deep_sleep(bool ext_wakeup)
```

### 2.1.2 Support for DLE (Data Packet Length Extension)

The DA14585/586 support the Data Length Extension feature. The Maximum Transmission Unit (MTU) should be computed according to the maximum allowed size. This means that, during a connection, if the packet length changes, the max packet length should be taken into consideration for the computation of the MTU. For the SUOTA examples which demonstrate the DLE feature, the max payload has 251 octets length while the L2CAP header has 4 octets length. This means that the MTU size will be 251-4 = 247 octets.

### 2.1.3 OTP Programming

DA14585/586 uses different OTP controller and memory layout than DA14580. DA14585/586 has a bigger OTP region (64 KB) which is supported by the latest SmartSnippets Toolbox.

### 2.1.4 Custom Profile

In SDK 6 multiple Primary Services can be implemented in a single custom profile.

### 2.1.5 Service Database Creation for Custom Profile

In this section the creation of custom database service in SDK 6 is presented. Examples from the SDK 6 for the creation of 128bit service database are used along with some definitions from [6]. For more info about the database creation please also refer to [7]. The differences that exist between SDK 6 and SDK 5.0.4 are highlighted.

**Note 1** Custom profile demo application has slightly changed in SDK 6. For more information please check section 8.3 of [2].

### 2.1.6 Creation of service database for 128bits long UUIDs

The GATT Profile requires the implementation of the Attribute Protocol (ATT). Attribute Protocol commands and requests act on values stored in Attributes on the server device. An Attribute is composed of four parts: Attribute Handle, Attribute Type, Attribute Value, and Attribute Permissions. The Attribute Handle is an index corresponding to a specific Attribute. The Attribute Type is a UUID that describes the Attribute Value. The Attribute Value is the data described by the Attribute Type and indexed by the Attribute Handle. Attribute Permissions is part of the Attribute that cannot be read from or written to using the Attribute Protocol. It is used by the server to determine whether read or write access is permitted for a given attribute. A profile is composed of one or more services

 

necessary to fulfill a use case. A service is composed of characteristics or references to other services. Each characteristic contains a value and may contain optional information about the value. The service and characteristic and the components of the characteristic (i.e. value and descriptors) contain the profile data and are all stored in Attributes on the server.

`struct attm_desc_128`, located in file *attm_db_128.h*, is used for attributes database creation (128 bit UUIDs) both in SDK 6 and SDK 5.0.4.

**SDK6**

```
/// Attribute description (used to create database)
struct attm_desc_128
{
    /// Element UUID
    uint8_t *uuid;
    /// UUID size
    uint8_t uuid_size;
    /// Attribute Permissions (@see enum attm_perm_mask)
    att_perm_type perm;
    /// Attribute Max Size (@see enum attm_value_perm_mask)
    /// note: for characteristic declaration contains handle offset
    /// note: for included service, contains target service handle
    att_size_t max_length;

    /// Current length of the element
    att_size_t length;
    /// Element value array
    uint8_t *value;
};
```

**SDK 5.0.4**

```
struct attm_desc_128
 {
     /// Element UUID
     uint8_t* uuid;
     /// UUID size
     uint8_t uuid_size;
     /// Attribute permission
     uint16_t perm;
     /// Maximum length of the element
     att_size_t max_length;
     /// Current length of the element
     att_size_t length;
```

 

```
    /// Element value array
    uint8_t* value;
};
```

**Comments**

1. There is a change in the database creation between SDK 6 and SDK 5.0.4.
   - In SDK 5.0.4 the `max_length` field contains the max size of the attribute. Especially if the attribute is characteristic declaration then this field contains the handle offset. If the attribute is included service then the `max_length` field contains the target service handle. The max_length/offset applies for the bits `[0-14]` of the member `max_length` of `struct attm_desc_128`.
   - In SDK 6, the `enum attm_value_perm_mask` in *attm.h* file has a new bit field added (RI). The new bit `[15]` is Trigger Read Indication. The usage of Trigger Read Indication is explained below:

     When RI is Disabled (0), the value is present in Database. When reading, lower layers will send `GATTC_READ_CFM` with the value that is already stored in the database. The upper layers will **not** receive a `GATTC_READ_REQ_IND`, since the value is already in the database.

     When RI is Enabled (1), the value is not present in the database. When reading, upper layers will receive a `GATTC_READ_REQ_IND` and are responsible for replying with a `GATTC_READ_CFM` that contains the value.
2. An example of an 128bit database creation is `custs1_att_db` from ALL IN ONE Ble project. The database is described in *users_custs1_def.c* file located in `<sdk_root>\projects\target_apps\ble_examples\ble_app_all_in_one\src\custom_profile`

### 2.1.7 Attribute Permissions

To retrieve the permission field of an attribute in **SDK 6**, the following macro, that can be found in *attm.h* file located at `<sdk_root>\sdk\ble_stack\host\att\attm`:, is used:

```
PERM(access, right)
```

- The available server access permissions (`enum attm_perm_mask`) are:

  `ALL` = Retrieve all permission info

  `RD` = Read Access

  `WR` = Write Access

  `IND` = Indication Access

  `NTF` = Notification Access

  `EXT` = Extended properties descriptor present

  `BROADCAST` = Broadcast descriptor present

  `EKS` = Check Encryption key size

  `WRITE_COMMAND` = Write Command Enabled attribute

  `WRITE_SIGNED` = Write Signed Enabled attribute

  `WRITE_REQ` = Write Request Enabled attribute

  `UUID_LEN` = UUID Length

- The available server Attribute & Service access rights are :

  `DISABLE` = Disable access

 

ENABLE = Enable access

UNAUTH = Access Requires Unauthenticated link

AUTH = Access Requires Authenticated link

SECURE = Access Requires Secure Connection

- To add more than one access permissions to an attribute OR the macros like the following examples:

```
// Control Point Characteristic Value

    [CUST1_IDX_CONTROL_POINT_VAL] = {CUST1_CTRL_POINT_UUID_128, ATT_UUID_128_LEN,
PERM(WR, ENABLE) | PERM(WRITE_REQ, ENABLE), DEF_CUST1_CTRL_POINT_CHAR_LEN, 0, NULL}


// LED State Characteristic Value

    [CUST1_IDX_LED_STATE_VAL] = {CUST1_LED_STATE_UUID_128, ATT_UUID_128_LEN, PERM(WR,
ENABLE) | PERM(WRITE_COMMAND, ENABLE), DEF_CUST1_LED_STATE_CHAR_LEN, 0, NULL}


 // ADC Value 1 Client Characteristic Configuration Descriptor

    [CUST1_IDX_ADC_VAL_1_NTF_CFG] = {(uint8_t*)&att_decl_cfg, ATT_UUID_16_LEN, PERM(RD,
ENABLE) | PERM(WR, ENABLE) | PERM(WRITE_REQ, ENABLE), sizeof(uint16_t), 0, NULL}
```

In **SDK 5.0.4** the same macro as in SDK 6 is used to retrieve the permission field of an attribute.

```
PERM(access, right)
```

- The available server access permissions (`enum attm_perm_mask`) are:

  RD = Read Access

  WR = Write Access

  IND = Indication Access

  NTF = Notification Access

  EKS = Check Encryption key size

  HIDE = Hide/Show attribute

  WRITE_SIGNED = Write Signed Enabled attribute

- The available server Attribute & Service access rights are :

  DISABLE = Disable access

  ENABLE = Enable access

  UNAUTH = Access Requires Unauthenticated link

  AUTH = Access Requires Authenticated link

  AUTHZ = Access Requires authorization

**Notice**

1. SDK 6 has more server access permissions than SDK 5.0.4. There is also a difference in the available access rights between the two SDKs.
2. In SDK6 to enable writing on an attribute, apart from the Write Access, `WRITE_COMMAND` or `WRITE_SIGNED` or `WRITE_REQ` or a combination of them should be enabled as shown in the next figure:

 

```
/**
 * 15  14  13  12  11  10  9   8   7   6   5   4   3   2   1   0
 * +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
 * |UUID_LEN |EKS| WR | WS | WC | B  |EXT |   N   |   I   |  WP   |  RD   |
 * +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
 *
 * Bit [0-1]  : Read Permission          (0 = Disable, 1 = Enable, 2 = UNAUTH, 3 = AUTH, 4 = SECURE)
 * Bit [2-3]  : Write Permission         (0 = Disable, 1 = Enable, 2 = UNAUTH, 3 = AUTH, 4 = SECURE)
 * Bit [4-5]  : Indication Permission    (0 = Disable, 1 = Enable, 2 = UNAUTH, 3 = AUTH, 4 = SECURE)
 * Bit [6-7]  : Notification Permission  (0 = Disable, 1 = Enable, 2 = UNAUTH, 3 = AUTH, 4 = SECURE)
 * Bit [8]    : Extended properties present (only relevant for a characteristic value)
 * Bit [9]    : Broadcast permission     (only relevant for a characteristic value)
 * Bit [10]   : Write Command accepted
 * Bit [11]   : Write Signed accepted
 * Bit [12]   : Write Request accepted
 * Bit [13]   : Encryption key Size must be 16 bytes
 * Bit [15-14]: UUID Length              (0 = 16 bits, 1 = 32 bits, 2 = 128 bits, 3 = RFU)
 */
```

### 2.1.8 Service attribute

A service definition contains a service declaration and may contain include definitions and characteristic definitions. Service definitions appear on the server in an order based on Attribute Handle. A service declaration is an Attribute with the Attribute Type set to the UUID for «Primary Service» or «Secondary Service». The Attribute Value shall be the 16-bit Bluetooth UUID or 128-bit UUID for the service, known as the service UUID. All Attributes on a Server shall either contain a service declaration or exist within a service definition.

#### 2.1.8.1 Service definition example (128bit UUID)

Below the custom service declaration located in *user_custs1_def.c* file of ALL IN ONE Ble project is shown for:

**SDK6**

```
// CUSTOM1 Service Declaration

    [CUST1_IDX_SVC] =   {(uint8_t*)&att_decl_svc, ATT_UUID_16_LEN, PERM(RD, ENABLE),
                        sizeof(custs1_svc), sizeof(custs1_svc), (uint8_t*)&custs1_svc}
```

**SDK 5.0.4**

```
// CUSTOM1 Service Declaration

    [CUST1_IDX_SVC] =   {(uint8_t*)&att_decl_svc, ATT_UUID_16_LEN, PERM(RD, ENABLE),
                        sizeof(custs1_svc), sizeof(custs1_svc), (uint8_t*)&custs1_svc}
```

**Notice**

1. There are no changes between the two SDKs.

### 2.1.9 Characteristic attribute

A characteristic definition contains a characteristic declaration, a Characteristic Value declaration and may contain characteristic descriptor declarations. A characteristic definition ends at the start or the next characteristic declaration or service declaration or after the maximum Attribute Handle. Characteristic definitions appear on the server. The two required declarations are the characteristic declaration and the Characteristic Value declaration. Characteristic descriptors are used to contain related information about the Characteristic Value.

 

#### 2.1.9.1 Characteristic definition example (128bit UUID)

The declaration, value and descriptor of the custom characteristic "Control Point" is presented below. It is in *user_custs1_def.c* file of ALL IN ONE Ble project which is located in
<sdk_root>\projects\target_apps\ble_examples\ble_app_all_in_one

**SDK6**

```
// Control Point Characteristic Declaration

[CUST1_IDX_CONTROL_POINT_CHAR] =      {(uint8_t*)&att_decl_char, ATT_UUID_16_LEN,
                                      PERM(RD, ENABLE),0, 0, NULL}

// Control Point Characteristic Value

[CUST1_IDX_CONTROL_POINT_VAL]  =      {CUST1_CTRL_POINT_UUID_128, ATT_UUID_128_LEN,
                                      PERM(WR, ENABLE) | PERM(WRITE_REQ, ENABLE),
                                      DEF_CUST1_CTRL_POINT_CHAR_LEN, 0, NULL}

// Control Point Characteristic User Description

[CUST1_IDX_CONTROL_POINT_USER_DESC]=  {(uint8_t*)&att_decl_user_desc, ATT_UUID_16_LEN,
                                      PERM(RD, ENABLE),
                                      sizeof(CUST1_CONTROL_POINT_USER_DESC) - 1,
                                      sizeof(CUST1_CONTROL_POINT_USER_DESC) - 1,
                                      CUST1_CONTROL_POINT_USER_DESC}
```

**SDK 5.0.4**

```
// Control Point Characteristic Declaration

[CUST1_IDX_CONTROL_POINT_CHAR] =      {(uint8_t*)&att_decl_char, ATT_UUID_16_LEN,
                                      PERM(RD, ENABLE),
                                      sizeof(custs1_ctrl_point_char),
                                      sizeof(custs1_ctrl_point_char),
                                      (uint8_t*)&custs1_ctrl_point_char}

// Control Point Characteristic Value

[CUST1_IDX_CONTROL_POINT_VAL]  =      {CUST1_CTRL_POINT_UUID_128, ATT_UUID_128_LEN,
                                      PERM(WR, ENABLE),
                                      DEF_CUST1_CTRL_POINT_CHAR_LEN, 0, NULL}


// Control Point Characteristic User Description

[CUST1_IDX_CONTROL_POINT_USER_DESC]=  {(uint8_t*)&att_decl_user_desc, ATT_UUID_16_LEN,
                                      PERM(RD, ENABLE),
                                      sizeof(CUST1_CONTROL_POINT_USER_DESC) - 1,
                                      sizeof(CUST1_CONTROL_POINT_USER_DESC) - 1,
                                      CUST1_CONTROL_POINT_USER_DESC}
```

 

**Notice**

1. `[CUST1_IDX_CONTROL_POINT_CHAR]`: This is a custom SDK 6 characteristic declaration attribute. The `max_length` field of this attribute contains the handle offset. This custom characteristic declaration does not define descriptors (as SDK 5.0.4 does) for the max (`att_size_t max_length`) and the current (`att_size_t length`) length. Value field is `NULL` (SDK 6). In SDK 5.0.4 the value field contains `custs1_ctrl_point_char` struct which is a value desciptor. From the definition of this struct user can observe that it has its attribute handle equal to zero.
2. `[CUST1_IDX_CONTROL_POINT_USER_DESC]`: This is a custom SDK 6 characteristic value attribute. The perm field of this attribute is `PERM(WR, ENABLE) | PERM(WRITE_REQ, ENABLE)`. This means that this characteristic value attribute has enabled two writing attributes: Write Request and write. SDK 5.0.4 has write access (writing attribute) enabled.

 

### 2.1.10 Non-Volatile Data Storage (NVDS)

The Non-Volatile Data Storage (NVDS) can be used to keep system configuration settings such as the Bluetooth device address. This is achieved with the `struct nvds_data_struct nvds_data_storage`.

**SDK6**

```
static const struct nvds_data_struct nvds_data_storage =
{
    .lpclk_drift                    = CFG_NVDS_TAG_LPCLK_DRIFT,
    .bd_address                     = CFG_NVDS_TAG_BD_ADDRESS,
    /// Default Channel Assessment Timer duration (20s - Multiple of 10ms)
    .ble_ca_timer_dur               = CFG_NVDS_TAG_BLE_CA_TIMER_DUR,
    /// Default Channel Reassessment Timer duration (Multiple of Channel Assessment
Timer duration)
    .ble_cra_timer_dur              = CFG_NVDS_TAG_BLE_CRA_TIMER_DUR,
    /// Default Minimal RSSI Threshold - -48dBm
    .ble_ca_min_rssi                = CFG_NVDS_TAG_BLE_CA_MIN_RSSI,
    /// Default number of packets to receive for statistics
    .ble_ca_nb_pkt                  = CFG_NVDS_TAG_BLE_CA_NB_PKT,
    /// Default number of bad packets needed to remove a channel
    .ble_ca_nb_bad_pkt              = CFG_NVDS_TAG_BLE_CA_NB_BAD_PKT,
};
```

**SDK 5.0.4**

```
const struct nvds_data_struct nvds_data_storage
__attribute__((section("nvds_data_storage_area")))
#ifdef CFG_INITIALIZE_NVDS_STRUCT
=   {
    .NVDS_VALIDATION_FLAG                   = (BD_ADDRESS_VALID | DEVICE_NAME_VALID |
LPCLK_DRIFT_VALID | APP_BLE_ADV_DATA_VALID \
                                              | APP_BLE_SCAN_RESP_DATA_VALID |
UART_BAUDRATE_VALID | SLEEP_ENABLE_VALID | EXT_WAKEUP_ENABLE_VALID \
                                              | DIAG_BLE_HW_VALID | DIAG_SW_VALID |
SECURITY_ENABLE_VALID | NEB_ID_VALID \
                                              | NVDS_BLE_CA_TIMER_DUR_VALID |
NVDS_BLE_CRA_TIMER_DUR_VALID | NVDS_BLE_CA_MIN_RSSI_VALID | NVDS_BLE_CA_NB_PKT_VALID \
                                              | NVDS_BLE_CA_NB_BAD_PKT_VALID),
    .NVDS_TAG_UART_BAUDRATE                 = 115200,
    .NVDS_TAG_DIAG_SW                       = 0,
    .NVDS_TAG_DIAG_BLE_HW                   = 0,
    .NVDS_TAG_NEB_ID                        = 0,
    .NVDS_TAG_LPCLK_DRIFT                   = CFG_NVDS_TAG_LPCLK_DRIFT,
```

 

```
    .NVDS_TAG_SLEEP_ENABLE                  = 1,
    .NVDS_TAG_EXT_WAKEUP_ENABLE             = 0,
    .NVDS_TAG_SECURITY_ENABLE               = 1,
    .NVDS_TAG_APP_BLE_ADV_DATA              = USER_ADVERTISE_DATA,
    .NVDS_TAG_APP_BLE_SCAN_RESP_DATA        = USER_ADVERTISE_SCAN_RESPONSE_DATA,
    .NVDS_TAG_DEVICE_NAME                   = USER_DEVICE_NAME,
    .NVDS_TAG_BD_ADDRESS                    = CFG_NVDS_TAG_BD_ADDRESS,
    .ADV_DATA_TAG_LEN                       = USER_ADVERTISE_DATA_LEN,
    .SCAN_RESP_DATA_TAG_LEN                 = USER_ADVERTISE_SCAN_RESPONSE_DATA_LEN,
    .DEVICE_NAME_TAG_LEN                    = USER_DEVICE_NAME_LEN,
    /// Default Channel Assessment Timer duration (20s - Multiple of 10ms)
    .NVDS_TAG_BLE_CA_TIMER_DUR              = CFG_NVDS_TAG_BLE_CA_TIMER_DUR,
    /// Default Channel Reassessment Timer duration (Multiple of Channel Assessment
Timer duration)
    .NVDS_TAG_BLE_CRA_TIMER_DUR             = CFG_NVDS_TAG_BLE_CRA_TIMER_DUR,
    /// Default Minimal RSSI Threshold - -48dBm
    .NVDS_TAG_BLE_CA_MIN_RSSI               = CFG_NVDS_TAG_BLE_CA_MIN_RSSI,
    /// Default number of packets to receive for statistics
    .NVDS_TAG_BLE_CA_NB_PKT                 = CFG_NVDS_TAG_BLE_CA_NB_PKT,
    /// Default number of bad packets needed to remove a channel
    .NVDS_TAG_BLE_CA_NB_BAD_PKT             = CFG_NVDS_TAG_BLE_CA_NB_BAD_PKT,
}
#endif
;
```

**Notice**

1. In SDK 6 `struct nvds_data_struct nvds_data_storage` members have changed.
2. The most significant change is that the RO (read-only) `nvds_data_storage` variable is no longer placed at a fixed location in RAM, as it was done in the SDK 5.0.4. Its location may differ from build to build. The SDK 5.0.4 uses the following attribute that specifies that the `nvds_data_storage` must be placed in a particular section:

   ```
   struct nvds_data_struct nvds_data_storage
   __attribute__((section("nvds_data_storage_area")))
   ```

   The section to be placed is defined during the linking phase, by the scatter file:

   ```
   LR_IROM4 0x20000340 0x100 {
       ER_IROM4 0x20000340 0x100 {
           * (nvds_data_storage_area)
       }
   }
   ```

**Note 2** The above linker file is valid for DA14580 (SDK5) and **not** for DA14585 (SDK 6)

 

3. By placing the nvds module to the same location (as instructed by the linker file) the same behavior as for SDK 5.0.4 can be achieved.

### 2.1.11 External Processor Periodic Wakeup

The Periodic wakeup period if GTL interface is not enabled has increased to 600s.

**Note 3** For more info about the Wake-Up and External Processor Configuration please refer to [1].

### 2.1.12 Kernel Timer

The maximum valid timeout for Kernel Timer has increased. More specifically the `KE_TIMER_DELAY_MAX` define has increased to:

41943030 msec = 41943.030 sec ~ 699 minutes

**Note 4** The `KE_TIMER_DELAY_MAX` definition can be found in *app_easy_timer.h* header file which is located in: `<sdk_root>\sdk\app_modules\api`. For more info about the Real Time Kernel and the Timer please refer to [1].

 

## 2.2 Porting Instructions

In this section, the steps in porting two key example applications, the prox_reporter and all_in_one from SDK 5.0.4 to SDK 6 are presented. The steps depicted below can to some extend be used also as a guide in porting applications developed with SDK 5.0.4 to SDK 6.

### 2.2.1 Porting the prox_reporter example application

The following steps describe how to port the prox_reporter example application from SDK 5.0.4 to SDK 6. The procedure is as follows:

1. To port an application developed in SDK5.0.4, to SDK 6 the *empty_peripheral_template* located in `<sdk_root>\projects\target_apps\template` can be used. Any of the BLE demo projects located in `<sdk_root>\projects\target_apps\ble_examples` can also be used depending on the features of the application to be ported.
2. All the "*user_ x"* application files must be copied to the `<sdk_root>\projects\target_apps\template\empty_template_ext\src` folder of the project to their corresponding folders.
3. `da1458x_config_advanced.h` and `da1458x_config_basic.h` files must not be copied but keep the latest versions of these files located in the `user_config` folder as shown in Figure 1. Apply your settings in these files.

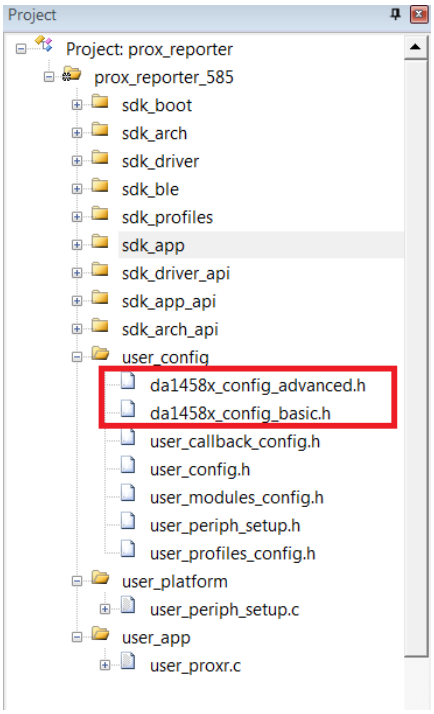


**Figure 1**

4. Instead of including files to enable specific profiles user must use configuration flags. For example, in `user_profiles_config.h` file of the prox_reporter demo application the following includes:

```
#include "proxr.h"
#include "spotar.h"
#include "bass.h"
#include "diss.h"
```

Have been replaced with the flags:

```
#define CFG_PRF_DISS
#define CFG_PRF_BASS
#define CFG_PRF_PXPR
#define CFG_PRF_SUOTAR
```

 

5. All the ATT DB related flags can be found in the rwip_config.h located in <sdk_root>\sdk\platform\core_modules\rwip\api folder.
6. In gapm_task.h file, located in <sdk_root>\sdk\ble_stack\host\gap\gapm, user can find the GAPM messages which may have changed due to the BLE stack upgrade. For example, the advertise configuration section in the user_config.h file of the prox_reporter demo project.
7. In app_user_config.h file, located in <sdk_root>\sdk\app_modules\api, the central_configuration has changed (in Central configuration). For this reason in user_config.h file of the prox_reporter demo project Central configuration section must be updated.
8. SPOTAR has been renamed to SUOTAR. So any file, function, message… containing the word spotar must be renamed to suotar. For example, in user_profiles_config.h file of the prox_reporter demo project #include "spotar.h" must be renamed to #include "suotar.h"
9. For errors in user_callback_config.h file please check changes in app_callback.h file located in <sdk_root>\sdk\app_modules\api. For example, struct user_profile_callbacks has been removed from user_callback_config.h, of the prox_reporter demo project, because struct profile_callbacks has been removed respectively from app_callback.h file. Also struct app_callbacks has some new members added.
10. In user_callback_config.h file,of the SDK6.0.4 prox_reporter demo project, we can see the structs app_bass_cb, app_proxr_cb and app_suotar_cb that are replacing struct profile_callback. These are new structs that can been found in app_suotar.h, app_proxr.h and app_bass.h files respectively.
11. TASK_NONE renamed to TASK_ID_INVALID in rwip_config.h located in <sdk_root>\sdk\platform\core_modules\rwip\api. Please check for any other changes in tasks type definitions.
12. In user_profiles_config.h file, of the SDK6.0.4 prox_reporter demo project, DISS and SUOTAR application profiles configuration has been altered. These profile configurations must be used by the application to be ported (if it uses these two profiles).

### 2.2.2 Porting the all_in_one example application

The following steps describe how to port the all_in_one example application from SDK 5.0.4 to SDK 6. The procedure is as follows:

1. To port the all_in_one demo application from SDK 5.0.4 to SDK 6 the empty_peripheral_template located in <sdk_root>\projects\target_apps\template will be used. Any of the BLE demo projects located in <sdk_root>\projects\target_apps\ble_examples can also be used depending on the features of the application to be ported.
2. all_in_one demo project is located in <sdk_root>\projects\target_apps\ble_examples.
3. All the "*user_ x"* application files must be copied to the <sdk_root>\projects\target_apps\template\empty_template_ext\src folder of the project to their corresponding folders.
4. da1458x_config_advanced.h and da1458x_config_basic.h files must not be copied but keep the latest versions of these files located in the user_config folder as shown in Figure 1. Apply your settings on these files.
5. SPOTAR has been renamed to SUOTAR. So any file, function, message… containing the word spotar must be renamed to suotar. For example, in user_profiles_config.h file of the all_in_one demo project #include "spotar.h" must be renamed to #include "suotar.h"
6. Instead of including files to enable specific profiles user must use configuration flags. For example in user_profiles_config.h file of the all_in_one demo application the following includes:

```
#include "diss.h"

#include "suotar.h"

#include "custs1.h"
```

 

Have been replaced with the flags:

```
#define CFG_PRF_DISS

#define CFG_PRF_SUOTAR

#define CFG_PRF_CUST1
```

7. All the ATT DB related flags can be found in the `rwip_config.h` located in `<sdk_root>\sdk\platform\core_modules\rwip\api` folder.
8. Sleep modes have changed. In `user_config.h` make the appropriate change in `app_default_sleep_mode`. The possible values are:
   - `ARCH_SLEEP_OFF`
   - `ARCH_EXT_SLEEP_ON`
   - `ARCH_EXT_SLEEP_OTP_COPY_ON`
9. In `gapm_task.h` file, located in `<sdk_root>\sdk\ble_stack\host\gap\gapm`, user can find the GAPM messages which may have changed due to the BLE stack upgrade. For example, the advertise configuration in the `user_config.h` file of the all_in_one demo project.
10. struct `advertise_configuration` in `app_user_config.h` file has some changes. As a result advertising configuration in `user_config.h` file must change accordingly.
11. In `user_config.h` select memory offset for bond data storage file. For SUOTA-ready memory layout:

    ```
    #define USER_CFG_BOND_DB_DATA_OFFSET (0x39000)
    ```

12. struct `gapm_configuration` in `app_user_config.h` file has some changes. As a result GAPM configuration in `user_config.h` file must change accordingly.
13. In `app_user_config.h` file, located in `<sdk_root>\sdk\app_modules\api`, the `central_configuration` has changed (in Central configuration section). For this reason, in `user_config.h` file of the prox_reporter demo project Central configuration section must be updated.
14. `struct app_callbacks` has some new members added in file `app_callback.h` file located in `<sdk_root>\sdk\app_modules\api`. For this reason, `struct user_app_callbacks` must be corrected in `user_callback_config.h` to reflect these changes.
15. In `user_profiles_config.h` file in SUOTA configuration section `#define SUOTAR_PATCH_AREA` is not valid anymore and has been removed.
16. In DISS application profile configuration in `user_profiles_config.h` some definitions have changed and some new have been added.
17. In `rwip_config.h` file located in `<sdk_root>\sdk\platform\core_modules\rwip\api` please check for changes in task type definitions. For example, `TASK_NONE` has been renamed to `TASK_ID_INVALID` and custom profile server tasks `TASK_CUSTS1` while `TASK_CUSTS2` have been renamed to `TASK_ID_CUSTS1` and `TASK_ID_CUSTS2` respectively. In general task type definitions are in `TASK_ID_x` format.
18. Timer drivers can now be found in `<sdk_root>\sdk\platform\driver\timer` folder which has replaced the `\pwm` folder. The pwm driver has been replaced with two timer drivers timer0 and timer2.
19. In `user_custs1_def.c` there are some changes in the PERM field of the characteristics added to the database. For example, in the Control Point Characteristic Value the PERM field has been modified:

    from `PERM(WR, ENABLE)` to `PERM(WR, ENABLE) | PERM(WRITE_REQ, ENABLE)`

    In LED State Characteristic Value the PERM field has been modified:

    from `PERM(WR, ENABLE)` to `PERM(WR, ENABLE) | PERM(WRITE_COMMAND, ENABLE)`

# 3 Porting from SDK 6.0.4 to 6.0.6

The latest SDK 6.0.6 release for the DA14585/586 devices has changes and improvements that affect the backwards compatibility for existing applications developed in a previous DA14585/586 6.0.4 or 6.0.2 SDK release.

## 3.1 Changes in SDK 6

The areas where there have been changes that affect the compatibility in SDK 6.0.6 are due to that it introduced:

- new security features and improvements
- an enhanced custom service that now can support multiple primary services

These changes affect the environment structure, callbacks, behavior of the default handlers, security and user function definitions.

### 3.1.1 Security

Support on security has been updated in 6.0.6 as this includes and has:

- Added support for Secure Connections (4.2 feature)
- Added the extra pairing method of the Secure Connections, the Numeric Comparison.
- Changed user configuration file (user_config.h) to support all possible Security configurations.
- Updated the application bond database implementation and API.
- Added support for address resolution.
- Changed security related user callbacks.
- Updated app_easy_security API.
- Modified callback app_on_tk_exch() to support all TK types, including TK_KEY_CONFIRM for Numeric Comparison pairing.
- Enhanced the application security environment variables.
- Changed the implementation of the security default handlers to support all the possible security processes (e.g. major change in default_app_on_encrypt_req_ind())

### 3.1.2 Custom Profile

The custom profile has been improved in order to support multiple services. Due to the changes introduced, the following three global variables need now to be defined in the user space (user_custs1_def.c file):

- `const uint8_t custs1_services[]`
- `const uint8_t custs1_services_size`
- `const uint16_t custs1_att_max_nb`

## 3.2 Porting Instructions

In this section, the steps in porting the SDK example applications, from SDK 6.0.4 to SDK 6.0.6 are presented. Specifically, the modifications that need to be made in the Pillar examples, pxp reporter, AES and empty peripheral demo projects, are described.

The steps depicted below can to some extend be used also as a guide in porting applications developed with SDK 6.0.4 to SDK 6.0.6.

 

### 3.2.1 Porting Pillar 1 (Bare Bone)

Backwards compatibility is not affected. This demo project can be ported from SDK 6.0.4 to SDK 6.0.6.

**Note 5** In case the CFG_APP_SECURITY configuration flag is defined in the *da1458x_config_basic.h* file then the following should be done to port the project to SDK 6.0.6:

1. Files app_utils.c and app_utils.h have to be added in the project tree under *sdk_app*
2. Files app_bond_db.c and app_bond_db.h has to be added in the project tree under *sdk_app*
3. user_app_callbacks table shall be updated in the user_callback_config.h file :
   - app_on_mitm_passcode_req has been removed
   - app_on_tk_exch_nomitm has been renamed to app_on_tk_exch
   - app_on_pairing_succeded has been renamed to app_on_pairing_succeeded
4. user_app_bond_db_callbacks table has to be added in the user_callback_config.h file

### 3.2.2 Porting Pillar 2 (Custom Profile)

Backwards compatibility is not affected. This demo project can be ported from SDK 6.0.4 to SDK 6.0.6.

**Note 6** The custom service's compatibility should be checked.

**Note 7** In case the CFG_APP_SECURITY configuration flag is defined in the *da1458x_config_basic.h* file then the following should be done to port the project to SDK 6.0.6:

1. Files app_utils.c and app_utils.h have to be added in the project tree under *sdk_app*
2. Files app_bond_db.c and app_bond_db.h has to be added in the project tree under *sdk_app*
3. user_app_callbacks table shall be updated in the user_callback_config.h file :
   - app_on_mitm_passcode_req has been removed
   - app_on_tk_exch_nomitm has been renamed to app_on_tk_exch
   - app_on_pairing_succeded has been renamed to app_on_pairing_succeeded
4. user_app_bond_db_callbacks table has to be added in the user_callback_config.h file

### 3.2.3 Porting Pillar 3 (Peripheral)

Backwards compatibility is not affected. This demo project can be ported from SDK 6.0.4 to SDK 6.0.6.

**Note 8** The custom service's compatibility should be checked.

**Note 9** In case the CFG_APP_SECURITY configuration flag is defined in the *da1458x_config_basic.h* file then the following should be done to port the project to SDK 6.0.6:

1. Files app_utils.c and app_utils.h have to be added in the project tree under *sdk_app*
2. Files app_bond_db.c and app_bond_db.h has to be added in the project tree under *sdk_app*
3. user_app_callbacks table shall be updated in the user_callback_config.h file :
   - app_on_mitm_passcode_req has been removed.
   - app_on_tk_exch_nomitm has been renamed to app_on_tk_exch
   - app_on_pairing_succeded has been renamed to app_on_pairing_succeeded
4. user_app_bond_db_callbacks table has to be added in the user_callback_config.h file.

### 3.2.4 Porting Pillar 4 (Security)

Backwards compatibility is affected due to the new Security features and improvements and the updated Custom Service. The following steps should be done for the application to be ported from SDK 6.0.4 to SDK 6.0.6:

 

1. user_app_callbacks table shall be updated in the user_callback_config.h file:
   - app_on_mitm_passcode_req has been removed
   - app_on_tk_exch_nomitm has been renamed to app_on_tk_exch and supports all TK exchange cases
   - app_on_pairing_succeded has been renamed to app_on_pairing_succeeded and has an extra uint8_t conidx argument
2. user_app_bond_db_callbacks table has to be added in user_callback_config.h. Bond database functions should be used through the app_easy_security API. The callback table connects the bond database operations to either the default or custom implementation of the bond database.
3. user_app_on_tk_exch_nomitm() should be changed to the new user_app_on_tk_exch()
4. user_app_on_encrypt_req_ind() should be removed, since new default_app_on_encrypt_req_ind() implements all possible cases
5. user_app_on_pairing_succeeded() should be removed, since all bond data functions and app_sec_env are no longer valid. Can be replaced with default_app_on_pairing_succeeded()
6. app_sec_env_tag has been changed to app_sec_bond_data_env_tag
7. struct bond_db_data has been removed
8. default_app_on_tk_exch_nomitm() has been replaced with default_app_on_tk_exch()
9. app_easy_security_tk_exch() has an extra 'bool accept' argument

### 3.2.5 Porting Pillar 5 (Sleep Mode)

Backwards compatibility is not affected. This demo project can be ported from SDK 6.0.4 to SDK 6.0.6.

**Note 10** In case the CFG_APP_SECURITY configuration flag is defined in the *da1458x_config_basic.h* file then the following should be done to port the project to SDK 6.0.6:

1. Files app_utils.c and app_utils.h have to be added in the project tree under *sdk_app*
2. Files app_bond_db.c and app_bond_db.h has to be added in the project tree under *sdk_app*
3. user_app_callbacks table shall be updated in the user_callback_config.h file :
   - app_on_mitm_passcode_req has been removed
   - app_on_tk_exch_nomitm has been renamed to app_on_tk_exch
   - app_on_pairing_succeded has been renamed to app_on_pairing_succeeded
4. user_app_bond_db_callbacks table has to be added in the user_callback_config.h file

### 3.2.6 Porting Pillar 6 (OTA)

Backwards compatibility is affected due to the new Security features and. The following steps should be done for the application to be ported from SDK 6.0.4 to SDK 6.0.6:

1. Files app_utils.c and app_utils.h have to be added in the project tree under *sdk_app*
2. Files app_bond_db.c and app_bond_db.h has to be added in the project tree under *sdk_app*
3. user_app_callbacks table shall be updated in the user_callback_config.h file:
   - app_on_mitm_passcode_req has been removed
   - app_on_tk_exch_nomitm has been renamed to app_on_tk_exch and supports all TK exchange cases
   - app_on_pairing_succeded has been renamed to app_on_pairing_succeeded and has an extra uint8_t conidx argument.
4. user_app_bond_db_callbacks table has to be added in the user_callback_config.h file
5. default_app_on_tk_exch_nomitm() has been replaced with default_app_on_tk_exch()

 

### 3.2.7 Porting Pillar 7 (All in One)

Backwards compatibility is affected due to the new Security features and improvements and the updated Custom Service. The following steps should be done for the application to be ported from SDK 6.0.4 to SDK 6.0.6:

1. `user_app_callbacks` table shall be updated in the `user_callback_config.h` file:
   - `app_on_mitm_passcode_req` has been removed.
   - `app_on_tk_exch_nomitm` has been renamed to `app_on_tk_exch` and supports all TK exchange cases
   - `app_on_pairing_succeded` has been renamed to `app_on_pairing_succeeded` and has an extra `uint8_t conidx` argument
2. `user_app_bond_db_callbacks` table has to be added in `user_callback_config.h`. Bond database functions should be used through the `app_easy_security` API. The callback table connects the bond database operations to either the default or custom implementation of the bond database.
3. `user_app_on_tk_exch_nomitm()` should be changed to the new `user_app_on_tk_exch()`
4. `user_app_on_encrypt_req_ind()` should be removed, since new `default_app_on_encrypt_req_ind()` implements all possible cases.
5. `user_app_on_pairing_succeeded()` should be removed, since all bond data functions and `app_sec_env` are no longer valid. Can be replaced with `default_app_on_pairing_succeeded()`
6. `app_sec_env_tag` has been changed to `app_sec_bond_data_env_tag`.
7. `struct bond_db_data` has been removed.
8. `default_app_on_tk_exch_nomitm()` has been replaced with `default_app_on_tk_exch()`
9. `app_easy_security_tk_exch()` has an extra `'bool accept'` argument

### 3.2.8 Porting Proximity Reporter example application

Backwards compatibility is affected due to the new Security features and improvements. The following steps should be done for the application to be ported from SDK 6.0.4 to SDK 6.0.6:

1. Files `app_utils.c` and `app_utils.h` have to be added in the project tree under *sdk_app*
2. `user_app_callbacks` table shall be updated in the `user_callback_config.h` file:
   - `app_on_mitm_passcode_req` has been removed.
   - `app_on_tk_exch_nomitm` has been renamed to `app_on_tk_exch`
   - `app_on_pairing_succeded` has been renamed to `app_on_pairing_succeeded` and has an extra `uint8_t conidx` argument
3. `user_app_bond_db_callbacks` table has to be added in `user_callback_config.h`.
4. `default_app_on_tk_exch_nomitm()` has been replaced with `default_app_on_tk_exch()`

### 3.2.9 Porting AES Encryption example application

Backwards compatibility is not affected. This demo project can be ported from SDK 6.0.4 to SDK 6.0.6.

**Note 11** In case the `CFG_APP_SECURITY` configuration flag is defined in the *da1458x_config_basic.h* file then the following should be done to port the project to SDK 6.0.6:

1. Files `app_utils.c` and `app_utils.h` have to be added in the project tree under *sdk_app*
2. Files `app_bond_db.c` and `app_bond_db.h` has to be added in the project tree under *sdk_app*
3. `user_app_callbacks` table shall be updated in the `user_callback_config.h` file :
   - `app_on_mitm_passcode_req` has been removed

 

- app_on_tk_exch_nomitm has been renamed to app_on_tk_exch
- app_on_pairing_succeded has been renamed to app_on_pairing_succeeded

4. user_app_bond_db_callbacks table has to be added in the user_callback_config.h file

### 3.2.10 Porting empty peripheral template

Backwards compatibility is affected due to the new Security features and improvements. The following steps should be done for the application to be ported from SDK 6.0.4 to SDK 6.0.6:

1. Files app_utils.c and app_utils.h have to be added in the project tree under *sdk_app*
2. user_app_callbacks table shall be updated in the user_callback_config.h file:
   - app_on_mitm_passcode_req has been removed
   - app_on_tk_exch_nomitm has been renamed to app_on_tk_exch
   - app_on_pairing_succeded has been renamed to app_on_pairing_succeeded and has an extra uint8_t conidx argument.
3. user_app_bond_db_callbacks table has to be added in user_callback_config.h.
4. default_app_on_tk_exch_nomitm() has been replaced with default_app_on_tk_exch()

# 4 Porting from SDK 6.0.6 to 6.0.8

Backwards compatibility is affected **only** in applications that are not using a Public Address. Specifically for applications that are using Random Static or Host RPA BD address `USER_CFG_ADDRESS_MODE` flag must defined in the *user_config* file of the application. Below default value for `all_in_one` demo application is shown:

```
#define USER_CFG_ADDRESS_MODE        APP_CFG_ADDR_PUB
```

Available values are shown in the next list:

Privacy Capabilities and address configuration of local device:

- `APP_CFG_ADDR_PUB` No Privacy, Public BDA
- `APP_CFG_ADDR_STATIC` No Privacy, Random Static BDA
- `APP_CFG_HOST_PRIV_RPA` Host Privacy, RPA, Public Identity
- `APP_CFG_HOST_PRIV_NRPA` Host Privacy, NRPA
- `APP_CFG_CNTL_PRIV_RPA_PUB` Controller Privacy, RPA or PUB, Public Identity
- `APP_CFG_CNTL_PRIV_RPA_RAND` Controller Privacy, RPA, Public Identity

To port prox_reporter user must define `USER_CFG_ADDRESS_MODE` flag as described above and also comment out `.priv1_2 = 0` in `struct gapm_configuration` in *user_config.h* file of the project.

**Note 12** File and path of *thirdparty\hash* folder must be also included, if the Keil project file is from the previous SDK version.

Changes that are not affecting backwards compatibility are described in the next sections. The changes are in the configuration files of the demo applications provided with the SDK. The configuration files are located in the *config* folder of each application.

## 4.1 da1458x_config_advanced.h file

- `CFG_USE_DEFAULT_XTAL16M_TRIM_VALUE_IF_NOT_CALIBRATED` is defined

```
#define CFG_USE_DEFAULT_XTAL16M_TRIM_VALUE_IF_NOT_CALIBRATED
```

- In the definition `CFG_TRNG`, for enabling the TRNG, user must select the desirable Buffer size. The default value is 1024 bytes.

```
#define CFG_TRNG (1024)
```

  Available Buffer sizes: 32, 64, 128, 256, 512, 1024 bytes. For more info please check Appendix E.3 of [1].

- Define `CFG_ENABLE_SMP_SECURE` to enable creation of private and public keys using Elliptic Curve Diffie Hellman algorithms. Advised for applications using secure connections feature.

```
```

 

```
#define CFG_ENABLE_SMP_SECURE
```

- Temperature range selection has been added. Default value is `CFG_AMB_TEMPERATURE` definition. Available values are: `CFG_HIGH_TEMPERATURE`, `CFG_AMB_TEMPERATURE`, `CFG_MID_TEMPERATURE`, `CFG_EXT_TEMPERATURE`

```
#define CFG_AMB_TEMPERATURE
```

- To enable power optimizations using the XTAL16M adaptive settling algorithm the following definition must be added:

```
#define CFG_XTAL16M_ADAPTIVE_SETTLING
```

## 4.2 user_callback_config.h

- following check should be added

```
#if (BLE_APP_SEC)
#include "app_bond_db.h"
#endif
```

## 4.3 user_config.h

- To select the Controller Privacy Mode `USER_CFG_CNTL_PRIV_MODE` flag must be defined. Default value is

```
#define USER_CFG_CNTL_PRIV_MODE     APP_CFG_CNTL_PRIV_MODE_NETWORK
```

Available values can be found in `app_user_config.h` file located in `<sdk_root>\sdk\app_modules\api` folder.

- In `struct advertise_configuration` the `addr_src` value is calculated through `APP_CFG_ADDR_SRC` macro. Default value is

```
.addr_src = APP_CFG_ADDR_SRC(APP_CFG_ADDR_PUB)
```

Available values can be found in `app_user_config.h` file located in `<sdk_root>\sdk\app_modules\api` folder.

- In `struct advertise_configuration` the `addr_type` value is calculated through `APP_CFG_ADDR_TYPE` macro. Default value is

```

```

 

```
.addr_type = APP_CFG_ADDR_TYPE(APP_CFG_ADDR_PUB)
```

Available values can be found in `app_user_config.h` file located in `<sdk_root>\sdk\app_modules\api` folder.

- In `struct advertise_configuration` the `renew_dur` default value is 15000

```
.renew_dur = 15000
```

 

# Revision History

| Revision | Date | Description |
|---|---|---|
| 1.0 | 24-Mar-2017 | Initial version. |
| 2.0 | 15-Jun-2017 | First cleanup version. |
| 3.0 | 09-Nov-2017 | Update for the DA14585/586 SDK Release 6.0.6. |
| 4.0 | 14-May-2018 | Update for the DA14585/586 SDK Release 6.0.8. |
| Change details:<br>● Section 5: Porting from SDK 6.0.6 to 6.0.8<br>New Section | | |
| 4.1 | 25-Jan-2022 | Updated logo, disclaimer, copyright. |

## Status Definitions

| Status | Definition |
|---|---|
| DRAFT | The content of this document is under review and subject to formal approval, which may result in modifications or additions. |
| APPROVED or unmarked | The content of this document has been approved for publication. |

## RoHS Compliance

Dialog Semiconductor's suppliers certify that its products are in compliance with the requirements of Directive 2011/65/EU of the European Parliament on the restriction of the use of certain hazardous substances in electrical and electronic equipment. RoHS certificates from our suppliers are available on request.