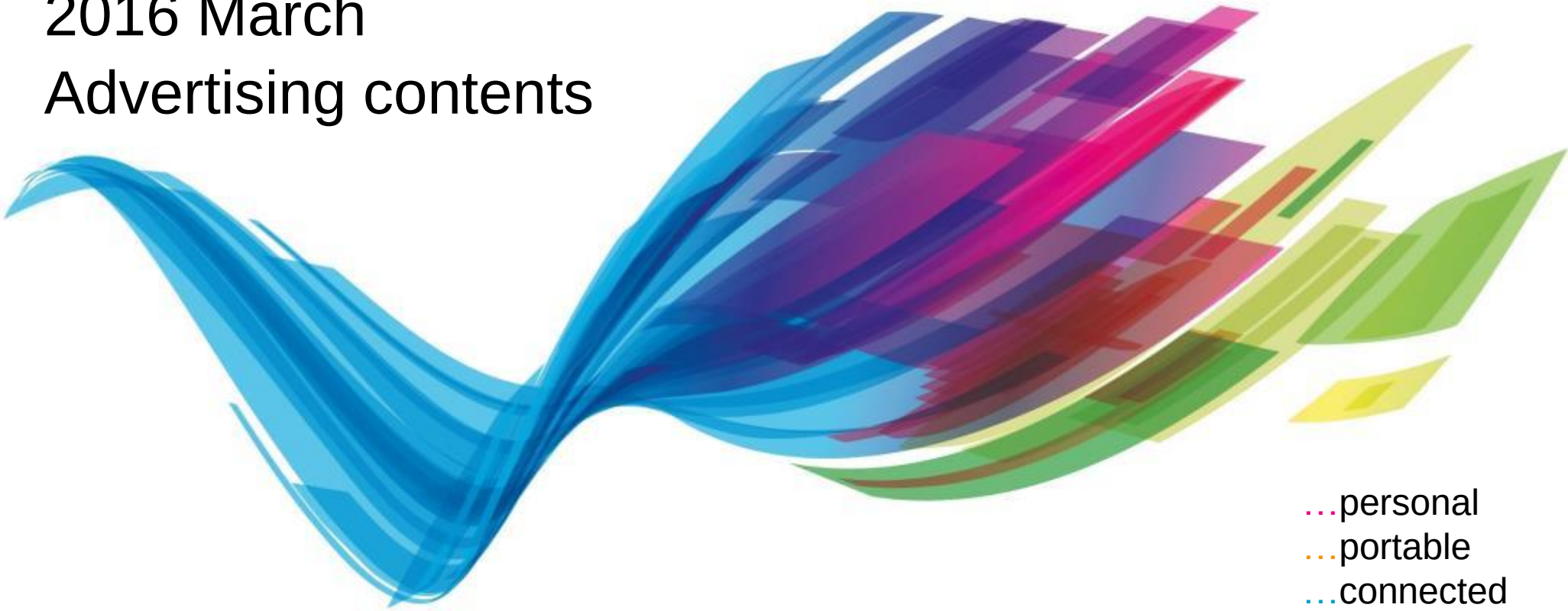
Dialog SDK 5.0.3 Training Materials

2016 March

Advertising contents



...personal
...portable
...connected

Advertising Contents

Advertising Contents

What will you learn today …

- **Before we start …**
  - Install/unpack SDK5.0.3 on the C:\ drive
  - Take a quick look at the Keil barebone project in the SDK

- **What are going to learn from this training …**

  - Basic understanding of BLE Advertising contents
  - Overview of the Dialog BLE SDK 5.0.3 project
  - How to change the advertising contents

Advertising Contents

What is an Advert?

- Bluetooth Low Energy has two primary modes of operation:
  - a connected mode which is defined by Profiles (see next training), and
  - an unconnected broadcast mode

  An Advert defines the format of the data transmitted in this broadcast mode.

- There are two common uses for the Advertising mode:
  - Transmit Beacons
  - Publish your identity and services

- Services are used to define the interface to a device once you are connected to it and are discussed in other training session.

Advertising Contents

What is an Advert? Continued …

- The Advert packets are small and have a well-defined format so only a restricted amount of user data can be carried.

- The Advertising mode also supports a secondary Scan Response packet which can contain additional data. This data can be requested by a potential client using a Scan Request without establishing a permanent connection to the device.

- The Advert packet is made up of a number of AD fields which typically include:
  - The name of the device
  - Some or all of the Services supported by the device

- The Advert may also contain proprietary Manufacturer Specific Data and Flags declaring the capabilities of the device.

Advertising Contents

Beacons

- A Beacon typically operates solely as a broadcast device. It uses the BLE Advertising packet to transmit static or dynamic data which can be used to push location-specific or context-specific content to enabled Smartphones

- There are a number of different specifications for Beacons including the iBeacon from Apple and the Eddystone Beacon from Google. The Dialog SDK can support all of these common variations or even a combination of them within the same device

- Typically, Beacons will not be connectable and so will not advertise Services, although there are some exceptions

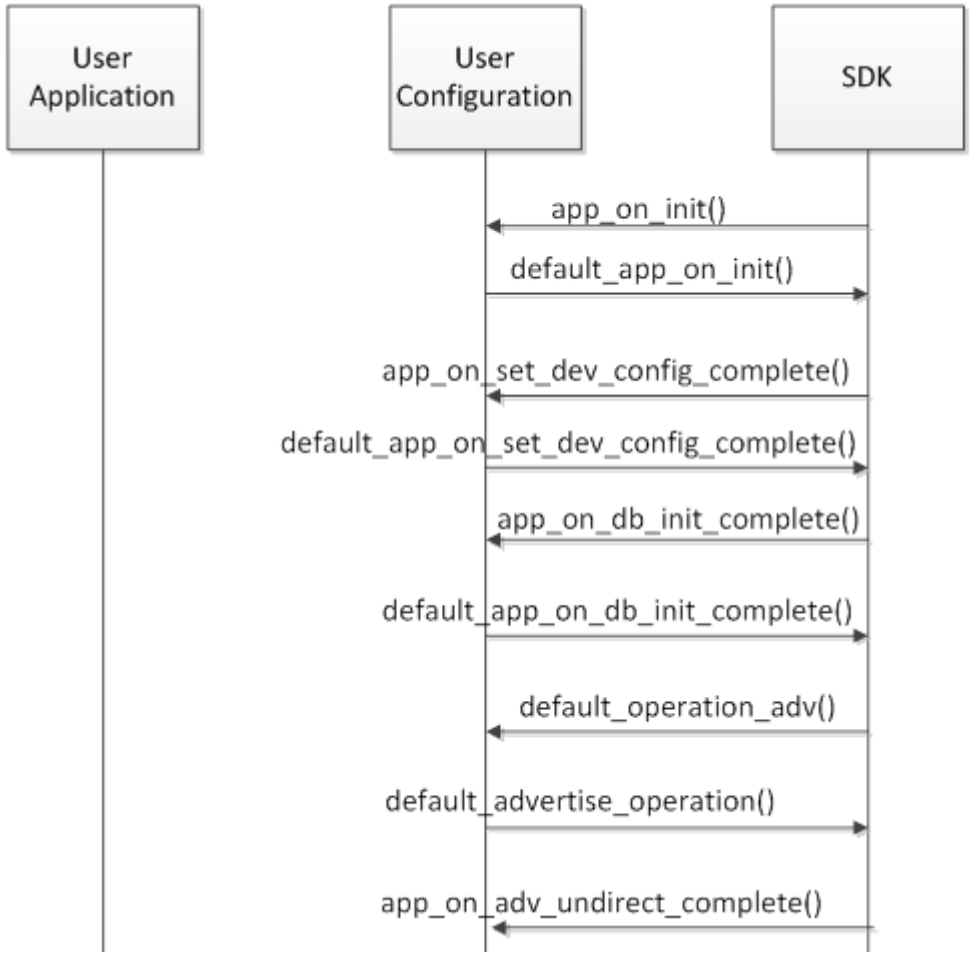
Advertising Contents

Barebone example

- The first example demonstrates how to customize the simple Barebone project. This will make the device to advertise its presence with some custom data.
- Project location (example): C:\DA1458x_SDK\5.0.3\projects\target_apps\ble_examples\ble_app_barebone
- Each beacon type is a custom advertisement which uses the Manufacturer Specific Data field to broadcast custom data.
- The subsequent sections describe the modifications required in each source file.

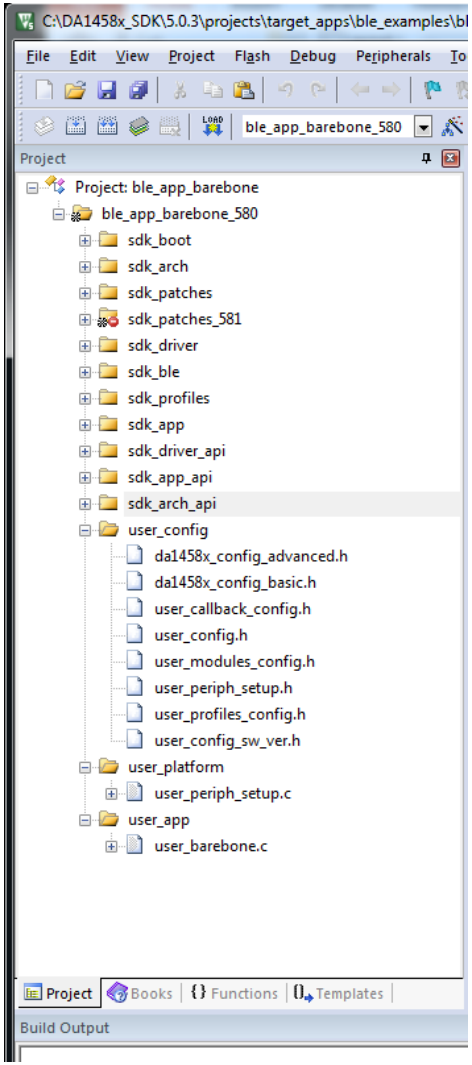
Advertising Contents

User application code flowgraph



Advertising contents

Barebone project overview (location of project files)



Advertising Contents

Source code discussion: user_config/user_config.h

- The USER_DEVICE_NAME can be customized.
- ***Exercise***: change the device name to “DIALOG-TRAINING”.

```
/// Advertising name
    #define USER_DEVICE_NAME     ("DIALOG-TRAINING")
```

- The advertising interval can be customized in this file.
- ***Exercise***: change the advertising interval to 1 second.

```
static const struct advertise_configuration user_undirected_advertise_conf ={
    /// Advertise operation type.
            .advertise_operation=ADV_NON_CONN,
    /// Own BD address source of the device:
            .address_src=GAPM_PUBLIC_ADDR,
    /// Advertise interval
            .intv = 1600,       //(1600 * 0.625ms) = 1 sec
```

Advertising Contents

Source code discussion: user_app/user_barebone.c/user_barebone.h

- A part of the advertised data is the manufacturer specific data. This data is fully customisable. This data can be customized in the file user_barebone.h.
- ***Exercise***: Change the company data to “AAAA”.

```
/* Manufacturer specific data constants */
#define APP_AD_MSD_COMPANY_ID                   (0xAAAA)
#define APP_AD_MSD_COMPANY_ID_LEN               (2)
#define APP_AD_MSD_DATA_LEN                     (sizeof(uint16_t))
```

- The manufacturer specific data can be automatically updated in a specified amount of time.
- ***Exercise***: change the update timer into 1 second.

```
/* Advertising data update timer */
#define APP_ADV_DATA_UPDATE_TO                  (100)    // 100*10ms = 1 second
```

Advertising Contents

Source code discussion: user_callback_config.h

- When the device is ready to start advertising, we want it to call our custom function. This example shows how to set up advertising using the default function. File that has to be modified for this purpose is user_callback_config.h

```
// Default Handler Operations
static const struct default_app_operations user_default_app_operations = {
    .default_operation_adv = user_app_adv_start,
};
```

Advertising Contents

Source code discussion: user_app/user_barebone.c

Custom function user_app_adv_start()

- When the device is ready to start advertising, it will call this function. The function sets some standard parameters such that the device will publicly broadcast its advertising data.
- The data to be advertised is copied from the structure defined above.
- Finally, the device is asked to start advertising based on this configuration.

```
void user_app_adv_start(void)
{
    // Schedule the next advertising data update
    app_adv_data_update_timer_used = app_easy_timer(APP_ADV_DATA_UPDATE_TO, adv_data_update_timer_cb);

    struct gapm_start_advertise_cmd* cmd;
    cmd = app_easy_gap_undirected_advertise_get_active();

    // add manufacturer specific data dynamically
    mnf_data_update();
    app_add_ad_struct(cmd, &mnf_data, sizeof(struct mnf_specific_data_ad_structure));

    app_easy_gap_undirected_advertise_start();
}
```

Advertising Contents

Source code discussion: user_app/user_barebone.c

- The function mnf_data_update() updates the manufacturer specific data in a specified amount of time (specified in previous slide).

```
static void mnf_data_update()
{
    uint16_t data;

    data = mnf_data.proprietary_data[0] | (mnf_data.proprietary_data[1] << 8);
    data += 1;
    mnf_data.proprietary_data[0] = data & 0xFF;
    mnf_data.proprietary_data[1] = (data >> 8) & 0xFF;

    if (data == 0xFFFF) {
        mnf_data.proprietary_data[0] = 0;
        mnf_data.proprietary_data[1] = 0;
    }
}
```

Advertising Contents

Source code discussion: da1458x_config_advanced.h

- The advertised BD Address can be changed in this file.
- ***Exercise***: change the default BD address into 01:02:03:04:05:06.

```
#define CFG_NVDS_TAG_BD_ADDRESS                {0x01, 0x02, 0x03, 0x04, 0x05, 0x06}
```

Advertising Contents

What would you see as output

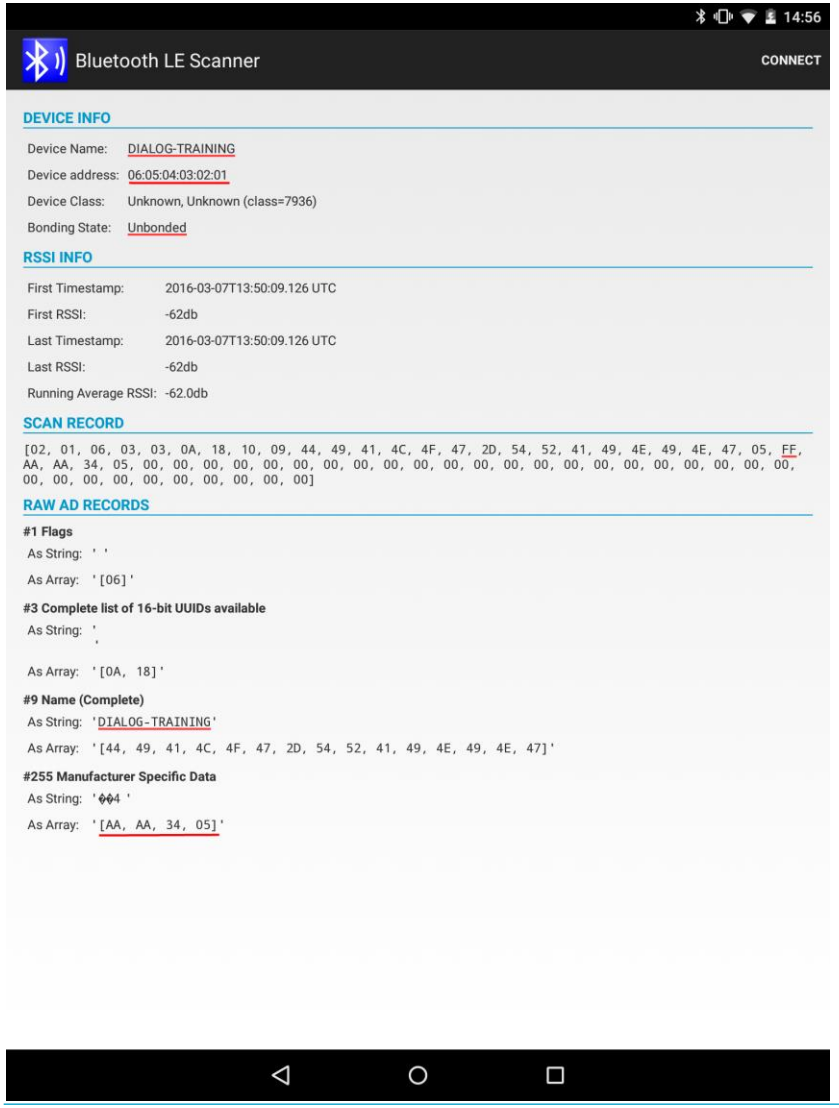
- When this exercise is run, the device will show up on the BLE scanner on your phone. Depending on the scanner used, it should be listed as DIALOG-TRAINING, or by the device's BD address (06:05:04:03:02:01). The device will not be in a connectable mode.

- Selecting the device will show the manufacturer specific data, most likely as a block of hexadecimal data, in this example 4 bytes. The unique UUID entered in the config struct should be visible in the manufacturer's data. This number should be incrementing every second.

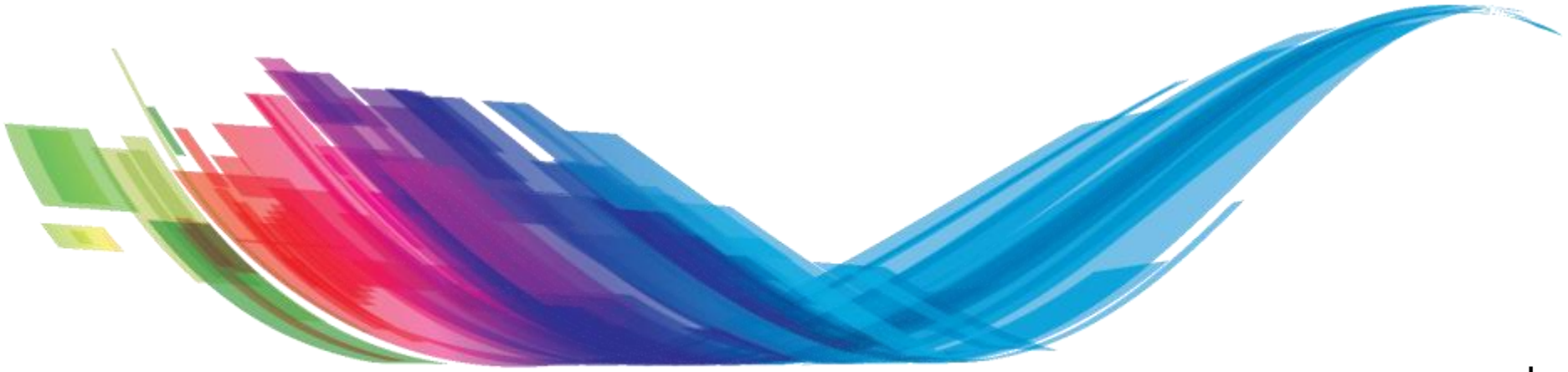
- **Further reading:**
  - **http://support.dialog-semiconductor.com/connectivity/product/da14580**
  - **http://support.dialog-semiconductor.com/resource/um-b-050-da1458x-software-development-guide-sdk5**

Advertising Contents

Screenshot Bluetooth LE Scanner



The Power To Be…



…personal
…portable
…connected