



## **GR551x Sample Service Application and Customization**

**Version: 1.7**

**Release Date: 2020-12-15**

**Copyright © 2020 Shenzhen Goodix Technology Co., Ltd. All rights reserved.**

Any excerpt, backup, modification, translation, transmission or commercial use of this document or any portion of this document, in any form or by any means, without the prior written consent of Shenzhen Goodix Technology Co., Ltd is prohibited.

## **Trademarks and Permissions**

**GOODIX** and other Goodix trademarks are trademarks of Shenzhen Goodix Technology Co., Ltd. All other trademarks and trade names mentioned in this document are the property of their respective holders.

## **Disclaimer**

Information contained in this document is intended for your convenience only and is subject to change without prior notice. It is your responsibility to ensure its application complies with technical specifications.

Shenzhen Goodix Technology Co., Ltd. (hereafter referred to as “Goodix”) makes no representation or guarantee for this information, express or implied, oral or written, statutory or otherwise, including but not limited to representation or guarantee for its application, quality, performance, merchantability or fitness for a particular purpose. Goodix shall assume no responsibility for this information and relevant consequences arising out of the use of such information.

Without written consent of Goodix, it is prohibited to use Goodix products as critical components in any life support system. Under the protection of Goodix intellectual property rights, no license may be transferred implicitly or by any other means.

## **Shenzhen Goodix Technology Co., Ltd.**

Headquarters: 2F. & 13F., Tower B, Tengfei Industrial Building, Futian Free Trade Zone, Shenzhen, China

TEL: +86-755-33338828

FAX: +86-755-33338099

Website: [www.goodix.com](http://www.goodix.com)

## Preface

### Purpose

This document introduces how to use and modify the sample service in a GR551x SDK, to help users with secondary development.

### Audience

This document is intended for:

- GR551x user
- GR551x developer
- GR551x tester
- Hobbyist developer
- Technical writer

### Release Notes

This document is the fifth release of *GR551x Sample Service Application and Customization*, corresponding to GR551x SoC series.

### Revision History

Version	Date	Description
1.0	2019-12-08	Initial release
1.3	2020-03-16	Updated descriptions in “Section 3.3 Add Sample Service to a New Project”.
1.5	2020-05-30	Updated pictures related to the project directory in “Chapter 3 Application of Goodix Sample Service”.
1.6	2020-06-30	Updated the document version based on SDK changes.
1.7	2020-12-15	Updated GRToolbox UI figures based on software update.

# Contents

<b>Preface.....</b>	<b>I</b>
<b>1 Introduction.....</b>	<b>1</b>
<b>2 Sample Service Overview.....</b>	<b>2</b>
<b>3 Application of Goodix Sample Service.....</b>	<b>3</b>
3.1 Preparation.....	3
3.2 Create a Project Based on Template.....	3
3.3 Add Sample Service to a New Project.....	4
3.4 Apply Sample Service.....	7
3.4.1 Initialize Sample Service.....	7
3.4.2 Test and Verification.....	8
<b>4 Create a Custom Service.....</b>	<b>10</b>
4.1 Add a New Characteristic.....	10
4.2 Read and Write a New Characteristic.....	11
4.3 Add Notify Function to a New Characteristic.....	13
4.4 Apply the Custom Service.....	14
4.4.1 Add a Timer.....	14
4.4.2 Test and Verification.....	15

# 1 Introduction

To maintain the compatibility between all kinds of Bluetooth devices, Bluetooth Special Interest Group (Bluetooth SIG) defines a series of universal standard services in Bluetooth-related fields.

Bluetooth devices of all kinds are able to control the peer Bluetooth devices, or access relevant data easily based on these standard services.

However, in some circumstances, it is necessary to implement your own services. For example, your application may require some new functions which are not supported by these standard services.

This document focuses on the application and modification of a Goodix sample service.

Before you get started, it is recommended to refer to the documents as shown in [Table 1-1](#).

Table 1-1 Reference documents

Name	Description
GR551x Developer Guide	Introduces the software/hardware and quick start guide of GR551x SoCs.
Bluetooth Core Spec v5.1	Offers official Bluetooth standard and core specification (v5.1) from Bluetooth SIG. Available at <a href="https://www.bluetooth.com/specifications/bluetooth-core-specification/">https://www.bluetooth.com/specifications/bluetooth-core-specification/</a> .
J-Link/J-Trace User Guide	Provides J-Link operational instructions. Available at <a href="http://www.segger.com/downloads/jlink/UM08001_JLink.pdf">www.segger.com/downloads/jlink/UM08001_JLink.pdf</a> .
Keil User Guide	Offers detailed Keil operational instructions. Available at <a href="http://www.keil.com/support/man/docs/uv4/">www.keil.com/support/man/docs/uv4/</a> .

## 2 Sample Service Overview

GR551x SoCs provide a sample service in compliance with Bluetooth SIG standards for your reference, to implement the basic Write and Notify communications between the Master and the Slave. The information interaction process is presented as below.

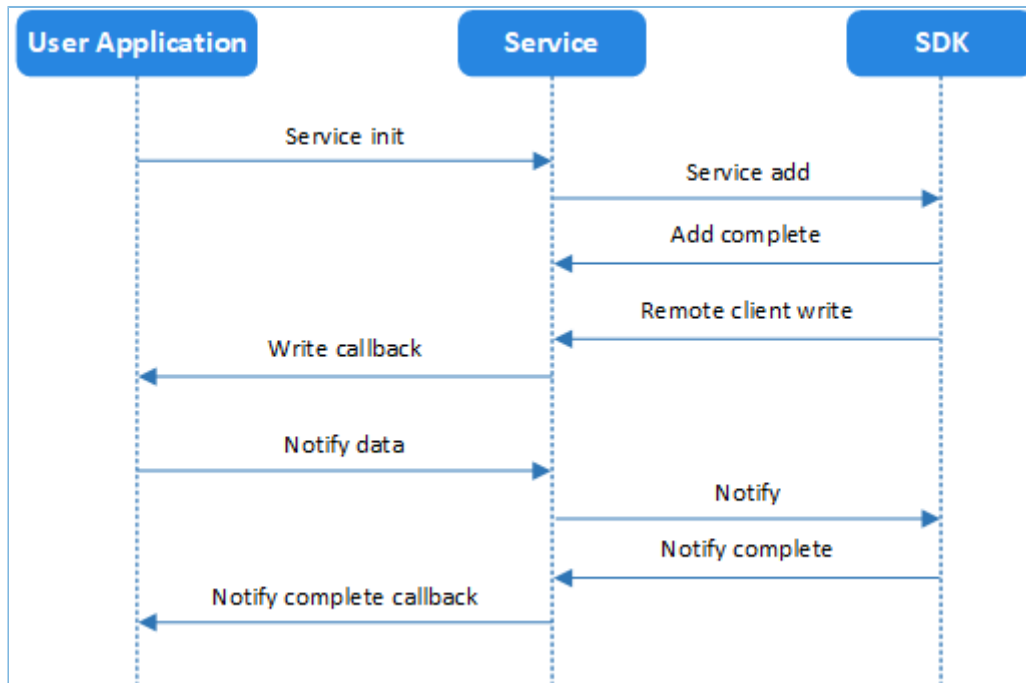


Figure 2-1 Information interaction process of the sample service

The sample service provides two characteristics: RX and TX, with the property as Write and Notify respectively. Detailed comparison between two characteristics is presented in [Table 2-1](#).

Table 2-1 Sample service characteristics

Description	UUID	Property
Service	A6ED0101-D344-460A-8075-B9E8EC90D71B	-
RX Characteristic	A6ED0102-D344-460A-8075-B9E8EC90D71B	Write without Response
TX Characteristic	A6ED0103-D344-460A-8075-B9E8EC90D71B	Notify

## 3 Application of Goodix Sample Service

This chapter introduces how to create a project for the Goodix sample service in Keil.

### Note:

SDK\_Folder is the root directory of GR551x SDK.

### 3.1 Preparation

Perform the following tasks before applying the sample service.

- Hardware preparation**

Table 3-1 Hardware preparation

Name	Description
J-Link debug probe	JTAG emulator launched by SEGGER. For more information, visit <a href="http://www.segger.com/products/debug-probes/j-link/">www.segger.com/products/debug-probes/j-link/</a> .
Development board	GR5515 Starter Kit Board (GR5515 SK Board)

- Software preparation**

Table 3-2 Software preparation

Name	Description
Windows	Windows 7/Windows 10
J-Link driver	A J-Link driver. Available at <a href="http://www.segger.com/downloads/jlink/">www.segger.com/downloads/jlink/</a> .
Keil MDK5	An integrated development environment (IDE). Available at <a href="http://www.keil.com/download/product/">www.keil.com/download/product/</a> .
LightBlue (iOS)	A Bluetooth Low Energy (Bluetooth LE) debugging tool running on iOS. Download from App Store.
GRTtoolbox (Android)	A Bluetooth LE debugging tool for GR551x. Available in SDK_Folder\tools\GRTtoolbox.

### 3.2 Create a Project Based on Template

Open SDK\_Folder\projects\ble\ble\_peripheral, and copy the ble\_app\_template folder to the current directory. Rename the copied folder and the folder in the Keil directory as ble\_app\_template\_mine, and open it in Keil. The file structure of the ble\_app\_template\_mine project is shown in [Figure 3-1](#).

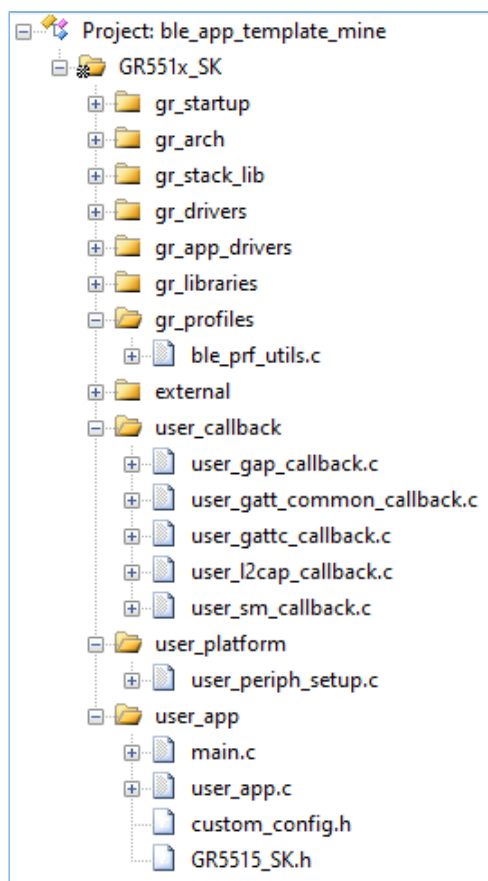


Figure 3-1 ble\_app\_template\_mine project structure directory


Related files are described in [Table 3-3](#).

Table 3-3 File description of ble\_app\_template\_mine

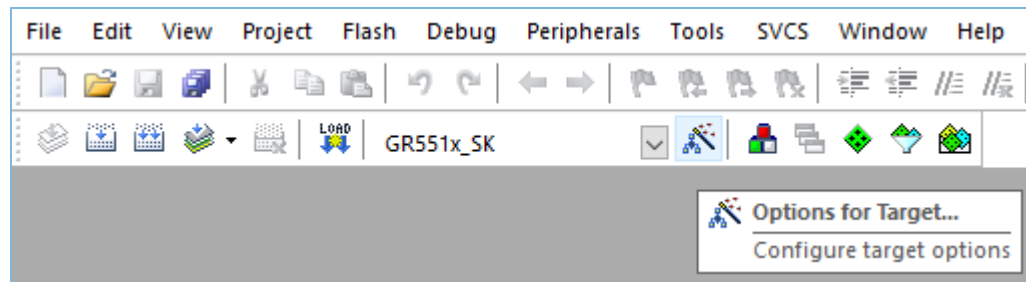
Group	Description
gr_profiles	This folder contains profile source files.
user_callback	This folder contains user-defined Bluetooth LE callback functions.
user_platform	This folder initializes user peripherals.
user_app	This folder implements user application logics.

### 3.3 Add Sample Service to a New Project

Copy the directory SDK\_Folder\components\profiles\sample to SDK\_Folder\projects\ble\ble\_peripheral\ble\_app\_template\_mine\Src. Follow the steps below to add *sample\_service.c* from the copied sample directory to the new project, and start compiling.

1. Add *sample\_service.c* to the new project.
  - (1). Click the **Options for Target...** icon  in the tool bar of Keil.



Figure 3-2 Clicking the **Options for Target...** icon

- (2). When the **Options for Target GR551x\_SK** window pops up, select the **C/C++** tab page, and click **...** next to **Include Paths** to browse paths.

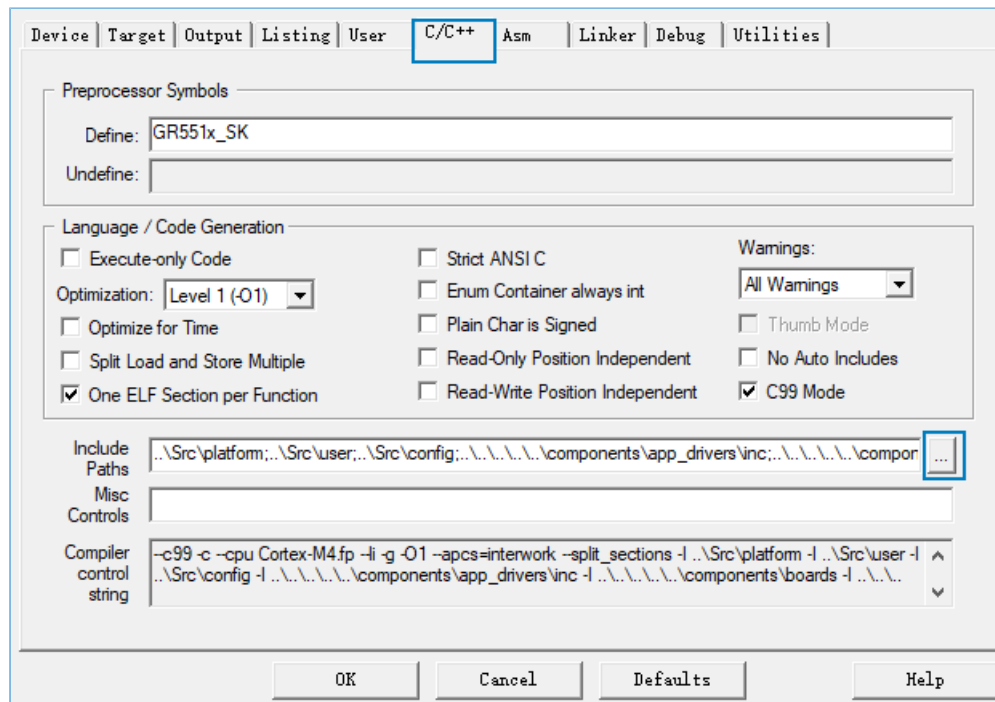


Figure 3-3 Browsing paths

- (3). Drag the scroll bar to the bottom of the **Include Paths** list in the **Folder Setup** window, and enter the path of the sample service ( `.. \Src\sample`); or double-click any empty line at the bottom, click **...** in the same line to browse the path for the sample service ( `.. \Src\sample`), and click **OK**.

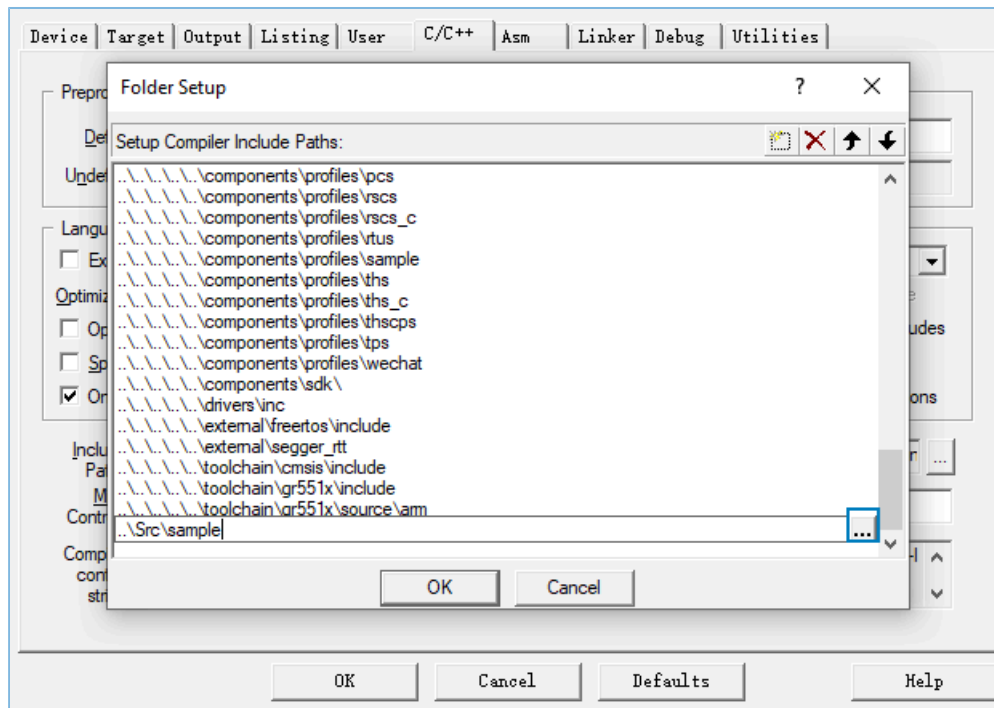


Figure 3-4 Entering the path of the sample service

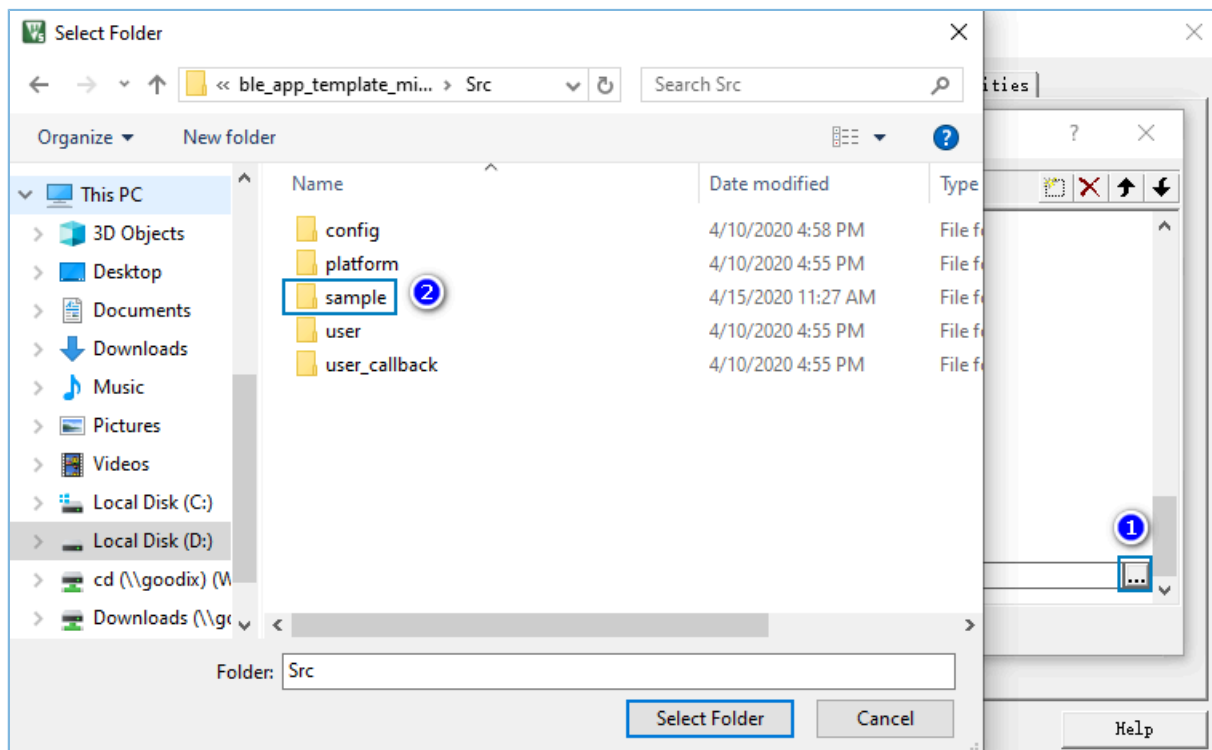


Figure 3-5 Browsing the path of the sample service

2. Add the Sample Service source file *sample\_service.c* to Keil project. Select **gr\_profiles** in the project directory, right-click and select **Add Existing Files to Groups 'gr\_profiles'**. Choose the *sample\_service.c* file in the *Src\Sample* directory, and add it to the *gr\_profiles* folder by clicking **Add**.

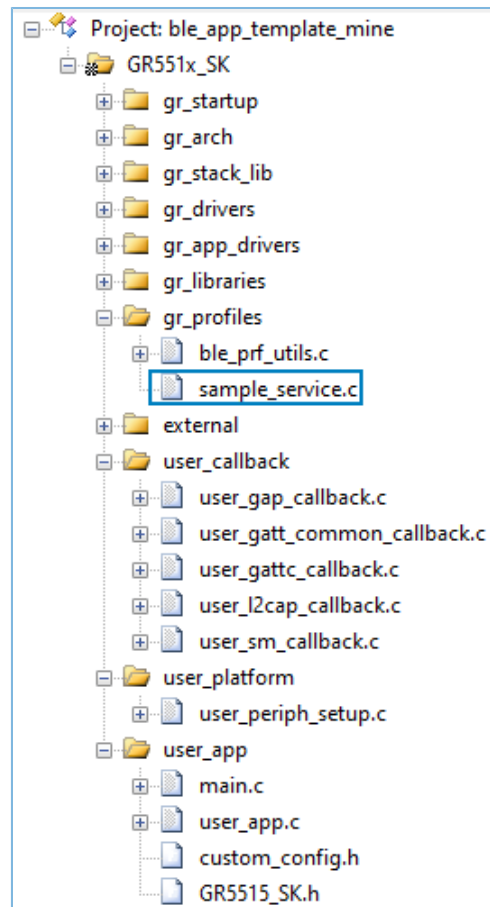


Figure 3-6 Adding the source file of the sample service to Keil project

## 3.4 Apply Sample Service

This section introduces how to initialize, test, and verify the sample service in the new project (ble\_app\_template\_mine).

### 3.4.1 Initialize Sample Service

Operations in the *user\_app.c* file are as below.

- Add header files of the sample service.

```
#include "user_app.h"
#include "gr55xx_sys.h"
#include "app_log.h"
#include "app_error.h"
#include "sample_service.h"
```

- Implement the callback function of the sample service.

```
static void sample_evt_process(samples_evt_t *p_evt)
{
    switch(p_evt->evt_type)
    {
```

```
    case SAMPLES_EVT_TX_NOTIFICATION_ENABLED:
        break;
    case SAMPLES_EVT_TX_NOTIFICATION_DISABLED:
        break;
    case SAMPLES_EVT_RX_RECEIVE_DATA:
        break;
    case SAMPLES_EVT_TX_NOTIFY_COMPLETE:
        break;
    default:break;
}
}
```

- Initialize the environment variables in the `services_init()` function and call the `samples_service_init()` function to add Service.

```
static void services_init(void)
{
    samples_init_t sample_init[1];
    sample_init[0].evt_handler = sample_envt_process;
    samples_service_init(sample_init, 1);
}
```

### 3.4.2 Test and Verification

After finishing the previous operations, compile the project, and download target files to the GR5515 SK Board. Follow the steps below to test and verify the sample service.

1. Start GRTtoolbox to scan and discover the device with the advertising name of Goodix\_Tem, as shown in [Figure 3-7](#).

---

#### Note:

LightBlue is available for test and verification on iOS devices.

---

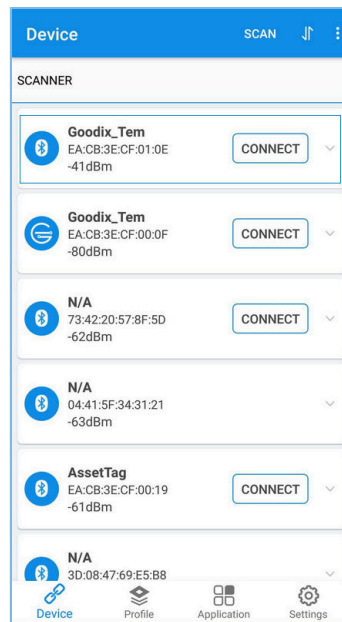


Figure 3-7 Discovering Goodix\_Tem

2. Tap **CONNECT** in the **Goodix\_Tem** pane to connect the device to the GR5515 SK Board through Bluetooth, and search for the sample service to check whether the two characteristics (RX and TX) are included in the service. If the values displayed under the discovered service are in accordance with the sample service characteristics listed in [Table 2-1](#), the sample service is applied successfully.

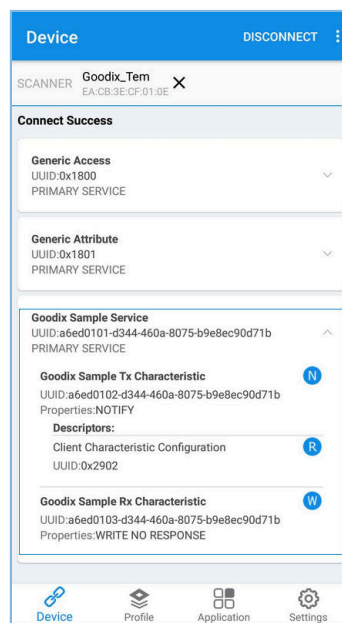


Figure 3-8 Successful application of the sample service

## 4 Create a Custom Service

In some applications, standard services cannot meet demands. This chapter introduces how to apply a custom service by adding a new characteristic (with properties of Notify and Write without Response) to the sample service.

### Note:

Custom services refer to modified services based on the sample service.

In this chapter, code in bold refers to those of the newly added characteristic, whereas the code not in bold is original.

### 4.1 Add a New Characteristic

Follow the steps below to add a new characteristic.

1. Add the UUID of the new characteristic (in the *sample\_service.c* file).

```
/**@brief The UUIDs of GUS characteristics. */
#define SAMPLE_SERVER_TX_UUID      {0x1B, 0xD7, 0x90, 0xEC, 0xE8, 0xB9, 0x75, 0x80, 0x0A, 0x46, 0x44, 0xD3, 0x02, 0x01, 0xED, 0xA6}
#define SAMPLE_SERVER_RX_UUID      {0x1B, 0xD7, 0x90, 0xEC, 0xE8, 0xB9, 0x75, 0x80, 0x0A, 0x46, 0x44, 0xD3, 0x03, 0x01, 0xED, 0xA6}
#define SAMPLE_SERVER_ADD_UUID      {0x1B, 0xD7, 0x90, 0xEC, 0xE8, 0xB9, 0x75, 0x80, 0x0A, 0x46, 0x44, 0xD3, 0x04, 0x01, 0xED, 0xA6}
```

2. Add Notify configuration variable of the new characteristic in the environment variable structure system (in the *sample\_service.c* file).

```
/**@brief The UUIDs of GUS characteristics. */
/**@brief Samples Service environment variable. */
typedef struct
{
    /**< Sample Service initialization variables. */
    samples_init_t samples_init;
    /**< Service start handle. */
    uint16_t start_hdl;
    /**< TX Character Notification configuration of peer devices. */
    uint16_t tx_ntf_cfg[SAMPLES_CONNECTION_MAX];
    /**< ADD Character Notification configuration of peer devices. */
    uint16_t add_ntf_cfg[SAMPLES_CONNECTION_MAX];
} samples_env_t;
```

3. Add index enumerations of the new characteristic (in the *sample\_service.c* file).

```
/**@brief Sample Service Attributes Indexes. */
enum samples_attr_idx_t
{
    SAMPLES_IDX_SVC,

    SAMPLES_IDX_TX_CHAR,
    SAMPLES_IDX_TX_VAL,
    SAMPLES_IDX_TX_CFG,
    SAMPLES_IDX_RX_CHAR,
    SAMPLES_IDX_RX_VAL,
    SAMPLES_IDX_ADD_CHAR,
    SAMPLES_IDX_ADD_VAL,
    SAMPLES_IDX_ADD_CFG,
}
```

```
SAMPLES_IDX_NB,
};
```

4. Add return event type of the new characteristic at the application layer (in the *sample\_service.h* file).

```
/**@brief Sample Service event type. */
typedef enum
{
    SAMPLES_EVT_INVALID,
    SAMPLES_EVT_TX_NOTIFICATION_ENABLED,
    SAMPLES_EVT_TX_NOTIFICATION_DISABLED,
    SAMPLES_EVT_RX_RECEIVE_DATA,
    SAMPLES_EVT_TX_NOTIFY_COMPLETE,
    SAMPLES_EVT_ADD_NOTIFICATION_ENABLED,
    SAMPLES_EVT_ADD_NOTIFICATION_DISABLED,
    SAMPLES_EVT_ADD_RECEIVE_DATA,
    SAMPLES_EVT_ADD_NOTIFY_COMPLETE,
} samples_evt_type_t
```

5. Define attribute descriptions of the new characteristic (in the *sample\_service.c* file).

After completing all steps above, a new characteristic is generated and displayed on the **Goodix\_Tem** connection interface, as shown in [Figure 4-2](#).

```
/**@brief Full SAMPLES Database Description - Used to add attributes into the database. */
static const attm_desc_128_t samples_att_db[SAMPLES_IDX_NB] =
{
    ...
    [SAMPLES_IDX_RX_VAL] = {SAMPLE_SERVER_RX_UUID,
                           WRITE_CMD_PERM_UNSEC,
                           (ATT_VAL_LOC_USER |
                            ATT_UUID_TYPE_SET(UUID_TYPE_128)),
                           SAMPLES_MAX_DATA_LEN},

    //SAMPLE ADD Characteristic Declaration
    [SAMPLES_IDX_ADD_CHAR] = {ATT_128_CHARACTERISTIC, READ_PERM_UNSEC, 0, 0},
    //SAMPLE ADD Characteristic Value
    [SAMPLES_IDX_ADD_VAL] = {SAMPLE_SERVER_ADD_UUID,
                             NOTIFY_PERM_UNSEC | WRITE_CMD_PERM_UNSEC,
                             (ATT_VAL_LOC_USER |
                              ATT_UUID_TYPE_SET(UUID_TYPE_128)),
                             SAMPLES_MAX_DATA_LEN},

    //SAMPLE ADD Characteristic - Client Characteristic Configuration Descriptor
    [SAMPLES_IDX_ADD_CFG] = {ATT_128_CLIENT_CHAR_CFG,
                             READ_PERM_UNSEC | WRITE_REQ_PERM_UNSEC,
                             0,
                             0},
};
```

## 4.2 Read and Write a New Characteristic

1. Add the function of reading the new characteristic to the *samples\_read\_att\_cb()* function.

```
static void samples_read_att_cb(uint8_t conn_idx, const gatts_read_req_cb_t *p_param)
{
```

```

...
switch (tab_index)
{
    case SAMPLES_IDX_TX_CFG:
        cfm.length = sizeof(uint16_t);
        cfm.value = (uint8_t *)(&s_samples_env[i].tx_ntf_cfg[conn_idx]);
        break;

    case SAMPLES_IDX_ADD_CFG:
        cfm.length = sizeof(uint16_t);
        cfm.value = (uint8_t *)(&s_samples_env[i].add_ntf_cfg[conn_idx]);
        break;

    default:
        break;
}

ble_gatts_read_cfm(conn_idx, &cfm);
}

```

## 2. Add the function of writing the new characteristic to the samples\_write\_att\_cb() function.

```

static void samples_write_att_cb(uint8_t conn_idx, const gatts_write_req_cb_t *p_param)
{
    ...
    switch (tab_index)
    {
        case SAMPLES_IDX_RX_VAL:
            event.conn_idx = conn_idx;
            event.evt_type = SAMPLES_EVT_RX_RECEIVE_DATA;
            break;

        case SAMPLES_IDX_TX_CFG:
            cccd_value = le16toh(&p_param->value[0]);
            event.conn_idx = conn_idx;
            event.evt_type = (PRF_CLI_START_NTF == cccd_value) ? \
                SAMPLES_EVT_TX_NOTIFICATION_ENABLED : \
                SAMPLES_EVT_TX_NOTIFICATION_DISABLED;
            s_samples_env[i].tx_ntf_cfg[conn_idx] = cccd_value;

            break;

        case SAMPLES_IDX_ADD_VAL:
            event.conn_idx = conn_idx;
            event.evt_type = SAMPLES_EVT_ADD_RECEIVE_DATA;
            break;

        case SAMPLES_IDX_ADD_CFG:
            cccd_value = le16toh(&p_param->value[0]);
            event.conn_idx = conn_idx;
            event.evt_type = (PRF_CLI_START_NTF == cccd_value) ? \
                SAMPLES_EVT_ADD_NOTIFICATION_ENABLED : \
                SAMPLES_EVT_ADD_NOTIFICATION_DISABLED;
            s_samples_env[i].add_ntf_cfg[conn_idx] = cccd_value;

            break;

        default:
            cfm.status = BLE_ATT_ERR_INVALID_HANDLE;
            break;
    }
    ...
}

```

## 3. Add Client Characteristic Configuration Descriptor (CCCD) settings in the samples\_cccd\_set\_cb() function.



```

static void samples_cccd_set_cb(uint8_t conn_idx, uint16_t handle, uint16_t cccd_value)
{
    ...
    switch (tab_index)
    {
        case SAMPLES_IDX_TX_CFG:
            event.conn_idx = conn_idx;
            event.evt_type = (PRF_CLI_START_NTF == cccd_value) ?\
                SAMPLES_EVT_TX_NOTIFICATION_ENABLED :\
                SAMPLES_EVT_TX_NOTIFICATION_DISABLED;
            s_samples_env[i].tx_ntf_cfg[conn_idx] = cccd_value;
            break;
        case SAMPLES_IDX_ADD_CFG:
            event.conn_idx = conn_idx;
            event.evt_type = (PRF_CLI_START_NTF == cccd_value) ?\
                SAMPLES_EVT_ADD_NOTIFICATION_ENABLED :\
                SAMPLES_EVT_ADD_NOTIFICATION_DISABLED;
            s_samples_env[i].add_ntf_cfg[conn_idx]=cccd_value;
            break;
        default:
            break;
    }
    ...
}

```

### 4.3 Add Notify Function to a New Characteristic

Add the Notify function to the new characteristic, and make the declaration in the *sample\_service.h* file. The code below is for your reference.

```

sdk_err_t samples_notify_add_data(uint8_t conn_idx, uint8_t ins_idx, uint8_t *p_data,
    uint16_t length)
{
    sdk_err_t error_code = SDK_ERR_NTF_DISABLED;
    gatts_noti_ind_t send_cmd;

    if (PRF_CLI_START_NTF == s_samples_env[ins_idx].add_ntf_cfg[conn_idx])
    {
        if (ins_idx <= s_samples_ins_cnt)
        {
            // Fill in the parameter structure
            send_cmd.type = BLE_GATT_NOTIFICATION;
            send_cmd.handle = prf_find_handle_by_idx(SAMPLES_IDX_ADD_VAL,
                s_samples_env[ins_idx].start_hdl,
                (uint8_t *)&s_samples_features);

            // pack measured value in database
            send_cmd.length = length;
            send_cmd.value = p_data;
            s_now_ins_cnt = ins_idx;
            s_now_notify_cmp_type = SAMPLES_EVT_ADD_NOTIFY_COMPLETE;
            // send notification to peer device
            error_code = ble_gatts_noti_ind(conn_idx, &send_cmd);
        }
    }
    return error_code;
}

```

## 4.4 Apply the Custom Service

To apply and verify the modified sample service, set a one-second timer to accumulate hexadecimal variables. Notify the accumulated value to the Master through the newly added characteristic (UUID: 0x1B, 0xD7, 0x90, 0xEC, 0xE8, 0xB9, 0x75, 0x80, 0x0A, 0x46, 0x44, 0xD3, 0x04, 0x01, 0xED, 0xA6; properties: Write Without Response/Notify).

### 4.4.1 Add a Timer

The steps to add a timer in the *user\_app.c* file are as below.

1. Add header files of the timer.

```
#include "user_app.h"
#include "gr55xx_sys.h"
#include "app_log.h"
#include "app_error.h"
#include "sample_service.h"
#include "app_timer.h"
```

2. Define the timer ID and a cumulative variable.

```
static app_timer_id_t s_add_timer_id;
static uint16_t s_add_count = 0;
```

3. Create a timer in *ble\_init\_cmp\_callback()*.

```
void ble_init_cmp_callback(void)
{
    ...
    error_code = ble_gap_adv_start(0, &s_gap_adv_time_param);
    APP_ERROR_CHECK(error_code);

    APP_LOG_INFO("Template application example started." );

    app_timer_create(&s_add_timer_id, ATIMER_REPEAT, add_time_out_handler);
}
```

4. Implement the event processing logics of the sample service.

#### Note:

The timer has a time base of 1 millisecond. Therefore, set the value of the timer to 1000, which means 1 second.

```
static void sample_envt_process(samples_evt_t *p_evt)
{
    switch(p_evt->evt_type)
    {
        case SAMPLES_EVT_TX_NOTIFICATION_ENABLED:
            break;
        case SAMPLES_EVT_TX_NOTIFICATION_DISABLED:
            break;
        case SAMPLES_EVT_RX_RECEIVE_DATA:
            break;
        case SAMPLES_EVT_TX_NOTIFY_COMPLETE:
            break;
        case SAMPLES_EVT_ADD_NOTIFICATION_ENABLED:
```

```

        app_timer_start(s_add_timer_id, 1000, NULL);
        break;
    case SAMPLES_EVT_ADD_NOTIFICATION_DISABLED:
        app_timer_stop (s_add_timer_id);
        break;
    case SAMPLES_EVT_ADD_RECEIVE_DATA:
        break;
    case SAMPLES_EVT_ADD_NOTIFY_COMPLETE:
        break;

    default:break;
}
}

```

5. Implement the timeout handler function of the timer.

```

static void add_time_out_handler(void *p_arg)
{
    s_add_count++;
    samples_notify_add_data(0,0,(uint8_t*)&s_add_count,2);
}

```

## 4.4.2 Test and Verification

After finishing all the operations above, compile the project, and download target files to the GR5515 SK Board. The steps for test and verification are as below.

1. Start the GRToolbox to scan and discover the device with the advertising name of **Goodix\_Tem**.

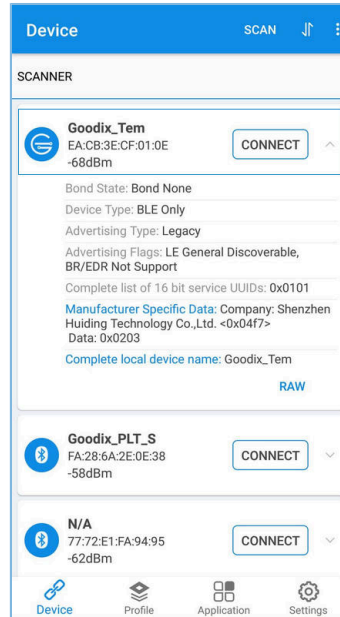


Figure 4-1 Discovering **Goodix\_Tem**

2. Tap **CONNECT** in the **Goodix\_Tem** pane, to connect the device to the GR5515 SK Board through Bluetooth. View the characteristics of the sample service to check whether the newly added characteristic is included.

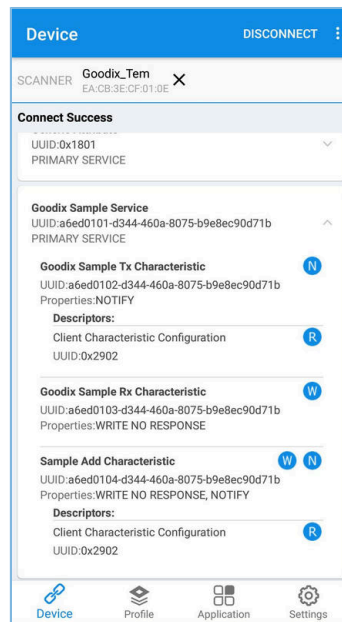


Figure 4-2 Viewing characteristics of the custom sample service

If the sample service includes a characteristic which shares the same UUID and properties with the newly added one, the modification to the sample service is successful.

3. Tap **N** on the right of **Sample Add Characteristic**. You can observe the value of the Notify characteristic ascends every second from the drop-down list.

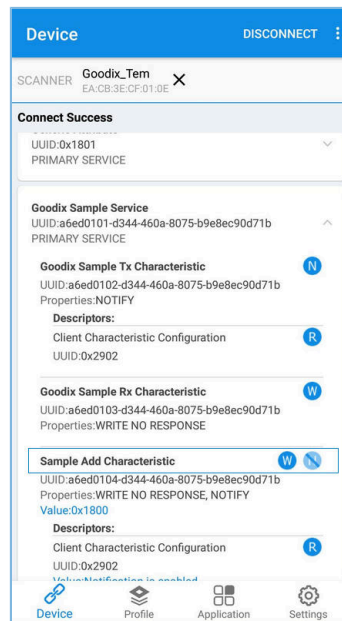


Figure 4-3 Successful application of the custom sample service

The custom sample service is applied successfully, as shown in [Figure 4-3](#).