WS73V100 BLE 和 Wi-Fi 小型化

开发指南

文档版本　03

发布日期　2025-01-21

前言

概述

本文档详细的描述了 WS73V100 BLE 蓝牙和 Wi-Fi 小型化开发指导，同时提供了常见的问题解答及故障处理方法。

读者对象

本文档主要适用以下工程师:

- 技术支持工程师
- 软件开发工程师

符号约定

在本文中可能出现下列标志，它们所代表的含义如下。

符号	说明
危险	表示如不避免则将会导致死亡或严重伤害的具有高等级风险的危害。
警告	表示如不避免则可能导致死亡或严重伤害的具有中等级风险的危害。
注意	表示如不避免则可能导致轻微或中度伤害的具有低等级风险的危害。
须知	用于传递设备或环境安全警示信息。如不避免则可能会导致设备损坏、数据丢失、设备性能降低或其它不可预知的结果。

符号	说明
	“须知”不涉及人身伤害。
说明	对正文中重点信息的补充说明。 “说明”不是安全警示信息，不涉及人身、设备及环境伤害信息。

修改记录

文档版本	发布日期	修改说明
03	2025-01-21	• 更新“1.6 sample 示例”章节内容。 • 新增“2.5 小型化 V2.0INI 配置”章节内容。
02	2024-10-25	• 更新“1 BLE 小型化使用和开发指导”章节内容。 • 更新“2.1 小型化的编译配置”章节内容。 • 更新“2.2 小型化版本编译”章节内容。 • 更新“2.4 小型化版本特性差异”章节内容。
01	2024-08-09	第一次正式版本发布。

目 录

1 BLE 小型化使用和开发指导

WS73V100 通过 API（Application Programming Interface）面向开发者提供 BLE 功能的开发和应用接口，包括 GAP、GATT server 和 GATT client 接口。

各组件功能说明如下：

- GAP：通用访问协议（Generic Access Profile），包含蓝牙本地设置和低功耗蓝牙的发现和连接接口。
- GATT：通用属性协议（Generic Attribute Profile），包含服务注册、服务发现等功能相关接口。

1.1 驱动编译

参考《WS73V100 Linux 平台驱动移植 用户指南》。

1.2 协议栈获取

查看 WS73 SDK 中"application/lib"文件夹下是否有对应主控文件夹，若有可使用对应主控中的协议栈；否则需要提供主控对应的交叉编译工具链给原厂，原厂编译协议栈后提供给客户。

1.3 开发流程

1.3.1 gap 层接口

1.3.1.1 概述

GAP 实现蓝牙设备开关控制、设备信息管理、广播管理、主动连接和断开连接等功能。

1.3.1.2 使用场景

打开蓝牙设备开关是使用蓝牙功能的首要条件，蓝牙启动后可进行设备信息管理，包括获取与设置本地设备名称、获取本地设备地址、获取配对信息、获取远端设备名称/设备类型/接收信号强度等。

当蓝牙设备需要被动与对端设备建立连接时，可设置广播参数并启动广播等待对端连接；当蓝牙设备需要主动与对端设备建立连接时，可向对端发起主动连接；当对端地址已知时，用户可直接向对端发起主动连接；当对端地址未知时，可打开蓝牙设备的扫描功能，获取正在广播的设备信息，并向对端发起主动连接；当蓝牙设备处于连接状态时，可获取设备连接信息；当蓝牙设备不需要与对端设备保持连接时，可主动断开连接。

1.3.1.3 功能

GAP 提供的接口如表 1-1所示。

表1-1GAP 接口说明

接口名称	描述	入参说明	返回信息说明
enable_ble	使能 BLE。	-	接口返回值：错误码。
disable_ble	去使能 BLE。	-	接口返回值：错误码。

接口名称	描述	入参说明	返回信息说明
bts_dev_manager_register_callbacks	enable 和 disable 结果回调注册函数。	func：enable 和 disable 回调函数。	接口返回值：错误码。
gap_ble_set_local_addr	设置本地设备地址。	mac：本地设备地址指针； len：本地设备地址长度。	接口返回值：错误码。
gap_ble_get_local_addr	获取本地设备地址。	mac：本地设备地址指针； len：本地设备地址长度。	本地设备地址存储在入参 mac 中； 接口返回值：错误码。
gap_ble_set_local_name	设置本地设备名称。	local_name：本地设备名称指针； length：本地设备名称长度。	接口返回值：错误码。
gap_ble_set_local_appearance	设置本地设备 appearance。	appearance：本地设备外貌。	接口返回值：错误码。
gap_ble_get_local_name	获取本地设备名称。	local_name：本地设备名称指针； length：本地设备名称长度。	本地设备名称存储在入参 local_name 中； 接口返回值：错误码。
gap_ble_get_paired_devices_num	获取 BLE 配对设备数量。	number：配对设备数量指针。	配对设备数量存储在入参 number 中； 接口返回值：错误码。

接口名称	描述	入参说明	返回信息说明
gap_ble_get_paired_devices	获取 BLE 配对设备。	number：配对设备数量指针; addr：配对设备地址指针。	配对设备数量存储在入参 number 中; 配对设备地址存储在入参 addr 中; 接口返回值：错误码。
gap_ble_get_pair_state	获取 BLE 设备的配对状态。	addr：对端设备地址。	接口返回值：配对状态（GAP_PAIR_NONE、GAP_PAIR_PAIRING、GAP_PAIR_PAIRED）。
gap_ble_remove_all_pairs	删除所有配对设备。	-	接口返回值：错误码。
gap_ble_remove_pair	删除配对设备。	addr：配对设备地址。	接口返回值：错误码。
gap_ble_disconnect_remote_device	断开 BLE 设备连接。	addr：对端设备地址。	接口返回值：错误码。
gap_ble_connect_remote_device	与设备建立 ACL 连接。	addr：对端设备地址。	接口返回值：错误码。
gap_ble_pair_remote_device	与已连接设备进行配对。	addr：对端设备地址。	接口返回值：错误码。
gap_ble_connect_param_update	连接参数更新。	params：待更新连接参数。	接口返回值：错误码。
gap_ble_set_adv_data	设置 BLE 广播数据。	adv_id：广播 id; data：设置的广播数据。	接口返回值：错误码。
gap_ble_set_adv_param	设置广播参数。	adv_id：广播 id; param：设置	接口返回值：错误码。

接口名称	描述	入参说明	返回信息说明
		的广播数据。 **注：使用板端地址发送广播时，own_addr 应设置成全 0。**	
gap_ble_start_adv	启动 BLE 广播。	adv_id：广播 id。	接口返回值：错误码。
gap_ble_stop_adv	停止 BLE 广播。	adv_id：广播 id。	接口返回值：错误码。
gap_ble_set_scan_parameters	设置扫描参数。	param：设置的扫描参数。	接口返回值：错误码。
gap_ble_start_scan	启动扫描。	-	接口返回值：错误码。
gap_ble_stop_scan	停止扫描。	-	接口返回值：错误码。
gap_ble_register_callbacks	注册 BLE GAP 回调。	func：用户回调函数。	接口返回值：错误码。
bth_ota_init	初始化 bth ota 通道。 **注：在收到 ble 使能成功的回调之后调用。**	-	接口返回值：错误码。

1.3.1.4 开发流程

GAP 开发的具体编程实例可参考 application/samples/bt。

GAP 开发的典型流程（指令中的数据请参见《WS73V100 命令使用指南》自行修改）。

Slave

步骤 1 调用 bts_dev_manager_register_callbacks 注册蓝牙 enable 和 disable 执行结果回调函数。

步骤 2 调用 gap_ble_register_callbacks 注册用户回调函数。

步骤 3 调用 enable_ble，打开蓝牙开关。

步骤 4 调用 gap_ble_set_local_addr，设置本地蓝牙地址。

步骤 5 调用 gap_ble_set_local_name，设置本地设备名称。

步骤 6 调用 gap_ble_set_adv_param，设置广播参数。

步骤 7 调用 gap_ble_set_adv_data，设置广播数据。

步骤 8 调用 gap_ble_start_adv，启动广播。

----结束

Master

步骤 1 调用 bts_dev_manager_register_callbacks 注册蓝牙 enable 和 disable 执行结果回调函数。

步骤 2 调用 gap_ble_register_callbacks 注册用户回调函数。

步骤 3 调用 enable_ble，打开蓝牙开关。

步骤 4 调用 gap_ble_set_local_addr，设置本地蓝牙地址。

步骤 5 调用 gap_ble_set_local_name，设置本地设备名称。

步骤 6 调用 gap_ble_set_scan_parameters，设置扫描参数。

步骤 7 调用 gap_ble_start_scan，启动扫描。

步骤 8 调用 gap_connect_remote_device，连接到目标设备。

步骤 9 调用 gap_ble_pair_remote_device，与目标设备配对。

----结束

1.3.1.5 返回值

获取配对状态返回值如表 1-2所示。

表1-2 返回值描述

序号	定义	实际数值	描述
1	GAP_PAIR_NONE	1	未配对。
2	GAP_PAIR_PAIRING	2	配对中。
3	GAP_PAIR_PAIRED	3	已配对。

1.3.1.6 注意事项

- WS73V100 产品可支持 8 路蓝牙连接。
- 若扫描不到设备，请先检查设备是否已在配对设备列表中，或者设备是否已与其他设备配对（此情况下需要先清除设备端配对信息）。

1.3.2 gatt server 层接口

1.3.2.1 概述

GATT 是一个基于蓝牙 GAP 连接的发送和接收数据的通用规范，支持在两个蓝牙设备间进行数据传输。

1.3.2.2 使用场景

GATT server 主要接收对端设备的命令和请求，给对端设备发送响应、指示或者通知。

1.3.2.3 功能

GATT server 提供的接口如表 1-3所示。

表1-3 GATT server 接口说明

接口名称	描述	入参说明	返回信息说明
gatts_register_server	Gatt server 注册。根据传入的 UUID 注册 server，回调函数返回 server 接口 ID。 **注：目前只支持注册一个 GATT server。**	app_uuid：应用 UUID 指针； server_id：服务端 ID 指针。	服务端 ID 存储在 server_id 中； 接口返回值：错误码。

接口名称	描述	入参说明	返回信息说明
gatts_unregister_server	注销 gatt 服务端。	server_id：服务器 ID。	接口返回值：错误码。
gatts_add_service	添加 service。	server_id：服务器 ID; service_uuid：服务 UUID; is_primary：是否是首要服务。	接口返回值：错误码。
gatts_add_characteristic	添加 characteristic 到指定的 service。	server_id：服务器 ID; service_handle：服务句柄; character：特征信息。	接口返回值：错误码。
gatts_add_descriptor	添加 descriptor 到对应的 characteristic。包含当前 Characteristic 的描述信息、配置信息、表示格式信息等。	server_id：服务器 ID; service_handle：服务句柄; descriptor：描述符信息。	接口返回值：错误码。
gatts_add_service_sync	添加一个 gatt 服务同步接口，服务句柄同步返回。	server_id：服务器 ID; service_uuid：服务 UUID; is_primary：是否是首要服务; handle：服务句柄指针。	服务句柄存储在 handle 中; 接口返回值：错误码。
gatts_add_characteristic_sync	添加一个 gatt 特征同步接口，特征句柄同步	server_id：服务器 ID;	特征句柄存储在 handle 中;

接口名称	描述	入参说明	返回信息说明
	返回。	service_handle：服务 UUID； character：GATT 特征； handle：特征句柄指针。	接口返回值：错误码。
gatts_add_descriptor_sync	添加一个 gatt 特征描述符同步接口，特征描述符句柄同步返回。	server_id：服务器 ID； service_handle：服务 UUID； character：特征描述符； handle：特征描述符句柄指针。	特征描述符句柄存储在 handle 中； 接口返回值：错误码。
gatts_start_service	启动 service。	server_id：服务器 ID； service_handle：服务句柄。	接口返回值：错误码。
gatts_delete_all_services	删除所有 GATT 服务。	server_id：服务器 ID；	接口返回值：错误码。
gatts_send_response	用户发送响应。	server_id：服务器 ID； conn_id：连接 ID； param：响应参数。	接口返回值：错误码。
gatts_notify_indicate	给远端 client 发送 indication/notification 。	server_id：服务器 ID； conn_id：连接 ID； param：通知或指示参数。	接口返回值：错误码。
gatts_notify_indicate_by_u	按照 UUID 给远端	server_id：服务器	接口返回值：错

接口名称	描述	入参说明	返回信息说明
uid	client 发送 indication/notification 。	ID; conn_id：连接 ID; param：通知或指示参数。	误码。
gatts_set_mtu_size	在连接之前设置服务端接收 mtu。	server_id：服务器 ID; mtu_size：服务端接收 mtu 值。	接口返回值：错误码。
gatts_register_callbacks	注册 GATT server 回调函数。	func：用户回调函数。	接口返回值：错误码。

1.3.2.4 开发流程

开发流程

GATT server 开发具体编程实例可参考 application/samples/bt。

GATT server 开发的典型流程：添加服务和特征及描述信息并启动服务。

步骤 1 调用 bts_dev_manager_register_callbacks 注册蓝牙 enable 和 disable 执行结果回调函数。

步骤 2 调用 gatts_register_callbacks，注册 GATT server 用户回调函数。

步骤 3 调用 enable_ble，打开蓝牙开关。

步骤 4 调用 gatts_register_server，创建一个 server。

步骤 5 调用 gatts_add_service，根据 UUID 创建 service。

步骤 6 调用 gatts_add_characteristic，对创建的服务添加特征值。

步骤 7 调用 gatts_add_descriptor，对服务中的特征添加描述信息。

步骤 8 调用 gatts_start_service，启动 service。

步骤 9 启动广播，等待对端连接。

步骤 10 被对端使能为“可通知”后，调用 gatts_notify_indicate 或 gatts_notify_indicate_by_uuid 向对端发起特征值通知。

----结束

1.3.3 gatt client 层接口

1.3.3.1 概述

GATT 是一个基于蓝牙 GAP 连接的发送和接收数据的通用规范，支持在两个蓝牙设备间进行数据传输。

1.3.3.2 使用场景

GATT client 主要给对端发送命令和请求，接收对端回复的响应、指示和通知。

1.3.3.3 功能

GATT client 提供的接口如表 1-4所示。

表1-4 GATT client 接口说明

接口名称	描述	入参说明	返回信息说明
gattc_register_client	注册 GATT client。	app_uuid：应用 UUID； client_id：客户端 ID 指针。	客户端 ID 存储在 client_id 中； 接口返回值：错误码。
gattc_unregister_client	取消注册 GATT client。	client_id：客户端 ID。	接口返回值：错误码。
gattc_discovery_service	发现服务。	client_id：客户端 ID； conn_id：连接 ID； uuid：服务 UUID。	接口返回值：错误码。
gattc_discovery_character	发现特征。	client_id：客户端 ID； conn_id：连接 ID； param：发现的特征参数。	接口返回值：错误码。

接口名称	描述	入参说明	返回信息说明
gattc_discovery_descriptor	发现特征描述符。	client_id：客户端 ID； conn_id：连接 ID； character_handle：特征声明句柄。	接口返回值：错误码。
gattc_read_req_by_handle	发起按照句柄读取请求。	client_id：客户端 ID； conn_id：连接 ID； handle：句柄。	接口返回值：错误码。
gattc_read_req_by_uuid	发起按照 UUID 读取请求。	client_id：客户端 ID； conn_id：连接 ID； param：按照 UUID 读取请求参数。	接口返回值：错误码。
gattc_write_req	发起写请求。	client_id：客户端 ID； conn_id：连接 ID； param：写请求参数。	接口返回值：错误码。
gattc_write_cmd	发起写命令。	client_id：客户端 ID； conn_id：连接 ID； param：写命令参数。	接口返回值：错误码。
gattc_exchange_mtu_req	发送交换 mtu 请求。	client_id：客户端 ID； conn_id：连接 ID； mtu_size：客户端接收 mtu。	接口返回值：错误码。
gattc_register_callbacks	注册 GATT client 回调函数。	func：用户回调函数。	接口返回值：错误码。

1.3.3.4 开发流程

GATT client 开发的具体编程实例可参考 application/samples/bt。

GATT client 开发的典型流程：连接对端设备，发现对端设备的服务，读写对端特征值，订阅对端的通知或者指示。

步骤 1 调用 bts_dev_manager_register_callbacks 注册蓝牙 enable 和 disable 执行结果回调函数。

步骤 2 调用 gap_ble_register_callbacks 注册用户回调函数

步骤 3 调用 gattc_register_callbacks，注册 GATT client 用户回调函数。

步骤 4 调用 enable_ble，打开蓝牙开关。

步骤 5 调用 gattc_register_client，创建一个 client。

步骤 6 递归调用 gattc_discovery_service、gattc_discovery_character 和 gattc_discovery_descriptor，获取对端的属性数据库。

步骤 7 调用 gattc_write_req 或 gattc_write_cmd 将关注的对端特征的客户端特征配置写为 0x0001 或 0x0002，设置为前者时可收到关注特征的特征通知，设置为后者时可收到关注特征的特征指示。

步骤 8 调用相应读写接口操作 GATT server的特征和描述符。

----结束

1.3.3.5 错误码

序号	定义	实际数值	描述
1	ERRCODE_BT_SUCCESS	0x0	执行成功。
2	ERRCODE_BT_FAIL	0x80006000	执行失败错误码。
3	ERRCODE_BT_NOT_READY	0x80006001	执行状态未就绪错误码。
4	ERRCODE_BT_MALLOC_FAIL	0x80006002	内存不足错误码。
5	ERRCODE_BT_MEMCPY_FAIL	0x80006003	内存拷贝错误错误码。
6	ERRCODE_BT_BUSY	0x80006004	繁忙无法响应错误码。
7	ERRCODE_BT_DONE	0x80006005	执行完成错误码。

序号	定义	实际数值	描述
8	ERRCODE_BT_UNSUPPORTED	0x80006006	不支持错误码。
9	ERRCODE_BT_PARAM_ERR	0x80006007	无效参数错误码。
10	ERRCODE_BT_STATE_ERR	0x80006008	状态错误。
11	ERRCODE_BT_UNHANDLED	0x80006009	未处理错误码。
12	ERRCODE_BT_AUTH_FAIL	0x8000600A	鉴权失败错误码。
13	ERRCODE_BT_RMT_DEV_DOWN	0x8000600B	远端设备关闭错误码。
14	ERRCODE_BT_AUTH_REJECTED	0x8000600C	鉴权被拒错误码。

1.4 错误码

序号	定义	实际数值	描述
1	ERRCODE_BT_SUCCESS	0x0	执行成功错误码。
2	ERRCODE_BT_FAIL	0x80006000	执行失败错误码。
3	ERRCODE_BT_NOT_READY	0x80006001	执行状态未就绪错误码。
4	ERRCODE_BT_MALLOC_FAIL	0x80006002	内存不足错误码。
5	ERRCODE_BT_MEMCPY_FAIL	0x80006003	内存拷贝错误错误码。
6	ERRCODE_BT_BUSY	0x80006004	繁忙无法响应错误码。
7	ERRCODE_BT_DONE	0x80006005	执行完成错误码。
8	ERRCODE_BT_UNSUPPORTED	0x80006006	不支持错误码。
9	ERRCODE_BT_PARAM_ERR	0x80006007	无效参数错误码。

序号	定义	实际数值	描述
10	ERRCODE_BT_STATE_ERR	0x80006008	状态错误。
11	ERRCODE_BT_UNHANDLED	0x80006009	未处理错误码。
12	ERRCODE_BT_AUTH_FAIL	0x8000600A	鉴权失败错误码。
13	ERRCODE_BT_RMT_DEV_DOWN	0x8000600B	远端设备关闭错误码。
14	ERRCODE_BT_AUTH_REJECTED	0x8000600C	鉴权被拒错误码。

1.5 BLE 功率档位定制

1.5.1 概述

WS73V100 支持 8 个档位：-6、-2、2、6、 10、 14、 16、20。

支持通过 ws73_cfg.ini 的 bt_maxpower 配置最高功率档位：

```
bt_maxpower=7，（默认值）表示有8个档位，分别是-6, -2, 2, 6, 10, 14, 16, 20

bt_maxpower=5，表示有6个档位，分别是-6, -2, 2, 6, 10, 14
```

1.5.2 开发流程

根据 BLE 官方协议要求，通过广播下发的 tx_power 值，选择≤tx_power 的最大档位。tx_power 在广播参数中下发，具体位于 gap_ble_adv_params_t 结构体中，如图 1-1 所示。

图1-1 gap_ble_adv_params_t 结构体

```
/**
 * @if Eng
 * @brief Struct of BLE advertising parameters.
 * @else
 * @brief BLE广播参数。
 * @endif
 */
typedef struct {
    uint32_t min_interval;     /*!< @if Eng Min interval[N * 0.625ms]
                                    @else   最小的广播间隔[N * 0.625ms] @endif */
    uint32_t max_interval;     /*!< @if Eng Max interval[N * 0.625ms]
                                    @else   最大的广播间隔[N * 0.625ms] @endif */
    uint8_t adv_type;          /*!< @if Eng Advertising type { @ref gap_ble_adv_type_t }
                                    @else   广播类型 { @ref gap_ble_adv_type_t } @endif */
    bd_addr_t own_addr;       /*!< @if Eng own address
                                    @else   本端地址 @endif */
    bd_addr_t peer_addr;       /*!< @if Eng Peer address
                                    @else   对端地址 @endif */
    uint8_t channel_map;       /*!< @if Eng channel bitmap
                                    @else   广播通道选择
                                            0x01----使用37通道
                                            0x07----使用37/38/39三个通道 @endif */
    uint8_t adv_filter_policy; /*!< @if Eng Advertising filter policy
                                        { @ref gap_ble_adv_filter_allow_scan_t }
                                    @else   白名单过滤策略
                                        { @ref gap_ble_adv_filter_allow_scan_t } @endif */
    int8_t  tx_power;          /*!< @if Eng Transmissive power
                                    @else   发送功率,单位dbm,范围-127~20 @endif */
    uint32_t duration;         /*!< @if Eng Duration
                                    @else   广播持续发送时长,单位dbm @endif */
} gap_ble_adv_params_t;
```

- 当 ws73_cfg.ini 的 bt_maxpower=7 时：

  tx_power=10，则选中第 4 档发送广播的功率为 10dBm

  tx_power=9，则选中第 3 档发送广播的功率为 6dBm

  tx_power=0x7F，则选中最高档位发送广播的功率为 20dBm

- 当 ws73_cfg.ini 的 bt_maxpower=3 时：

  tx_power=10，则选中第 4 档发送广播的功率为 6dBm

  tx_power=9，则选中第 3 档发送广播的功率为 6dBm

  tx_power=0x7F，则选中最高档位发送广播的功率为 6dBm

1.6 sample 示例

说明

sample 运行依赖的驱动 target 为 ble_android，需使用 make ble_android 编译出的驱动。

1.6.1 ble_gatt_client 使用指导

该 sample 用于验证蓝牙的 client 能力，包括打开蓝牙、client 注册、设置扫描参数、启动扫描、连接、配对、设置 MTU、数传（发送、接收）、停止扫描、取消配对、关闭蓝牙功能。其中，**连接、设置 MTU、数传、取消配对功能默认关闭**，可在"inc/ble_gatt_client.h"中将宏 **SUPPORT_GATT_C** 打开（#define SUPPORT_GATT_C TRUE）来使能 client 的连接、数传能力。

sample 编译

sample 编译如下：

步骤 1 更改 Makefile 中的 CROSS 宏为主控使用的工具链。

步骤 2 在 sample 的 lib 文件夹中放置 BTH 协议栈静态库文件（若 WS73 SDK 中"application\lib"文件夹下有对应主控文件夹，可使用对应主控中的协议栈；否则需要提供主控对应的交叉编译工具链给原厂，原厂编译 BTH 协议栈后提供给客户）。

步骤 3 在 WS73 SDK 的"application\sample\ble\ble_gatt_client"路径下执行"make clean;make"即可编译出 ble_client_sample 应用（不要更改 sample 的路径，Makefile 中定义了 sample 需要的头文件的相对路径）。

----结束

sample 使用

1. 扫描：sample 会在首次运行、断连时自动启动扫描，在连接后停止扫描。用户可在 ble_gatt_client_scan_result_cbk 接口中处理扫描到的蓝牙设备，每扫描到一个 BLE 设备都会回调一次 ble_gatt_client_scan_result_cbk 接口。
2. 连接：可在扫描到蓝牙设备时，使用 ble_gatt_client_connect_device 接口连接对端设备，**不要求必须在 ble_gatt_client_scan_result_cbk 接口中连接对端蓝牙设备，但需要保证对端蓝牙设备正在发广播。**连接状态的改变会在 ble_gatt_client_connect_state_change_cbk 回调中上报，当前 sample 会在连接后停止扫描、主动协商 MTU、服务发现等操作。
3. 数据发送：调用 ble_gatt_client_send_report 接口发送数据给 server。sample 中通过一个全局变量 g_tx_handle 与 server 进行数传（**不代表只有一个特征能进行数传，只要特征在 server 端配置了 WRITE、WRITE_NO_RSP 即可进行数传**），用户可以在 ble_gatt_client_discover_character_cbk 回调中获取 g_tx_handle，将符合要求的特征记录下来用于与 client 的数传，代码如下：

```
g_tx_handle = character->value_handle;
```

4. 数据接收：在 ble_gatt_client_notification_cbk 回调或者 ble_gatt_client_indication_cbk 回调中处理接收到的 server 数据。sample 为验证数传功能，会在该回调中调用 ble_gatt_client_send_report 接口将接收到的数据重新发送给 server。

L口说明

对于回调中的入参，不需要主动释放内存，回调结束后，协议栈自身会进行释放。

1.6.2 ble_uuid_server 使用指导

该 sample 用于验证蓝牙的 server 能力，包括打开蓝牙、server 注册、设置广播参数、启动广播、连接、数传、移除配对、停止广播、关闭蓝牙功能。

sample 编译

sample 编译如下：

步骤 1 更改 Makefile 中的 CROSS 宏为主控使用的工具链。

步骤 2 在 sample 的 lib 文件夹中放置 BTH 协议栈静态库文件（若 WS73 SDK 中"application\lib"文件夹下有对应主控文件夹，可使用对应主控中的协议栈；否则需要提供主控对应的交叉编译工具链给原厂，原厂编译 BTH 协议栈后提供给客户）。

步骤 3 在 SDK 的"application\sample\ble\ble_uuid_server"路径下执行"make clean;make"即可编译出 ble_server_sample 应用（不要更改 sample 的路径，Makefile 中定义了 sample 需要的头文件的相对路径）。

----结束

sample 使用

用户可使用 nRF Connect 工具进行连接测试，下载连接为"https://github.com/NordicSemiconductor/Android-nRF-Connect"。

1. 启动广播：调用 ble_start_adv 接口启动广播。当前 sample 会在首次运行、server 非主动断连时自动启动广播。
2. 停止广播：调用 ble_stop_adv 接口停止广播。当前 sample 会在反初始化蓝牙时停止广播。此外，当 server 被连接时，BTH 协议栈会自动停止广播。
3. 接收数据：在 ble_uuid_server_receive_write_req_cbk 接口中处理 client 发送的数据。

4. 发送数据：调用 ble_uuid_server_send_report_by_handle 接口发送数据。并**不要求 server 向 client 发送数据必须在 ble_uuid_server_receive_write_req_cbk 中进行**，只需要参考 ble_uuid_server_receive_write_req_cbk 中的代码示例，使用 ble_uuid_server_send_report_by_handle 接口发送数据即可。
5. 自定义服务、特征设置：自定义服务、特征、描述符的 UUID 定义在"ble_uuid_server/inc/ble_uuid_server.h"中。当前 sample 定义了一个私有服务 BLE_UUID_UUID_SERVER_SERVICE，该服务下面存在两个特征 BLE_UUID_CHARACTERISTIC_UUID_TX、BLE_UUID_CHARACTERISTIC_UUID_RX。为了验证数传功能，sample 会在 ble_uuid_server_receive_write_req_cbk 接口中接收到 client 发送的数据时，将 client 发送给 server 的数据通过 BLE_UUID_CHARACTERISTIC_UUID_TX 发送给 client。UUID 需要根据客户自身需求更改。UUID 的详细介绍如表 1-5所示。

表1-5 server UUID 介绍

名称	介绍	sample 默认值
BLE_UUID_UUID_SERVER_SERVICE	服务 UUID	0xABCD
BLE_UUID_CHARACTERISTIC_UUID_TX	特征 1UUID	0x2A90
BLE_UUID_CHARACTERISTIC_UUID_RX	特征 2UUID	0x2A8A

若 UUID 的 **16** 字节全为自定义的（比如为 ABCDEFGH-IJKL-MNOP-QRST-UVWXYZ012345），需要在 UUID 注册时，参考下面代码将字节数组转换为 bt_uuid_t。

```
void stream_data_to_uuid(bt_uuid_t *out_uuid)
{
    char uuids[] = {0x45, 0x23, 0x01, 0xYZ, 0xWX, 0xUV, 0xST, 0xQR, 0xOP, 0xMN, 0xKL, 0xIJ,
0xGH, 0xEF, 0xCD, 0xAB};
    out_uuid->uuid_len = 16;
    if (memcpy_s(out_uuid->uuid, out_uuid->uuid_len, uuids, 16) != EOK) {
        return;
    }
}
```

同时，sample 中定义的 BLE_UUID_CHARACTERISTIC_UUID_TX 特征的 properties 配置为 GATT_CHARACTER_PROPERTY_BIT_NOTIFY 和

GATT_CHARACTER_PROPERTY_BIT_READ；

BLE_UUID_CHARACTERISTIC_UUID_RX 特征的 properties 配置为 GATT_CHARACTER_PROPERTY_BIT_READ 和 GATT_CHARACTER_PROPERTY_BIT_WRITE_NO_RSP。因此，特征 BLE_UUID_CHARACTERISTIC_UUID_RX 不具备发送数据的能力。

用户可根据自身需求更改特征的 properties，具体枚举定义在"bts_gatt_stru.h"的 gatt_characteristic_property_t 中。

此外，**默认的 MTU 大小为 23 字节，若数传的数据长度超过 20 字节（其中 3 字节为 GATT 包头），需要客户端主动发起 MTU 协商。**

若需要在广播数据中携带自定义 16 位 complete service uuid，需要在 ble_server_adv.h 中新建一个结构体

```
typedef struct {
    uint8_t length;
    uint8_t type;
    uint8_t value[2];
} ble_complete_service_uuid_st;
```

在 ble_server_adv.c 中新增函数（以 0x2211 为例）

```
static uint8_t ble_set_adv_complete_service_uuid(uint8_t *set_adv_data_position, uint8_t max_len)
{
    uint8_t idx = 0;
    ble_complete_service_uuid_st service = {
        .length = 3,
        .type = 0x03,
        .value = { u16_low_u8(0x2211), u16_high_u8(0x2211) },
    };
    memcpy_s(set_adv_data_position, 4, &service, 4);
    return 4;
}
```

在 ble_uuid_server_set_adv_data 函数中返回 idx 之前增加代码

```
    idx += ble_set_adv_complete_service_uuid(&set_adv_data[idx], adv_data_max_len - idx);
```

如图所示

```
static uint8_t ble_set_adv_complete_service_uuid(uint8_t *set_adv_data_position, uint8_t max_len)
{
    uint8_t idx = 0;
    ble_complete_service_uuid_st service = {
        .length = 3,
        .type = 0x03,
        .value = { u16_low_u8(0x2211), u16_high_u8(0x2211) },
    };
    memcpy_s(set_adv_data_position, 4, &service, 4);
    return 4;
}

static uint16_t ble_uuid_server_set_adv_data(uint8_t *set_adv_data, uint8_t adv_data_max_len)
{
    uint8_t idx = 0;

    if ((set_adv_data == NULL) || (adv_data_max_len == 0)) {
        return 0;
    }
    idx += ble_set_adv_flag_data(set_adv_data, adv_data_max_len);
    idx += ble_set_adv_appearance_data(&set_adv_data[idx], adv_data_max_len - idx);
    idx += ble_set_adv_complete_service_uuid(&set_adv_data[idx], adv_data_max_len - idx);
    return idx;
}
```

L口说明

对于回调中的入参，不需要主动释放内存，回调结束后，协议栈自身会进行释放。

1.6.3 测速

自动测速需要按照表 1-6将对应的宏打开后再编译 sample 运行。当前 sample 测速为 ble_gatt_client 给 ble_uuid_server 发送数据，速率打印在 ble_uuid_server 中。

以下三个宏打开后会关闭 sample 中的所有打印。

表1-6 sample 测速宏

sample	宏	值
ble_uuid_server	SPEED_TEST_RECV	TRUE
ble_gatt_client	SUPPORT_GATT_C	TRUE
	SPEED_TEST_SEND	TRUE

1.6.4 注意事项

异常断连时，需要重启广播以及重启扫描，以保证业务在干扰场景下能够尽可能自愈（建议在**断连原因为非本端断链**、以及**配对失败**时需要自愈）。

1.7 AT 指令指南

参考《WS73V100 命令使用指南》中的“BLE&SLE 模块 AT 指令”章节。

2 Wi-Fi 小型化使用说明

2.1 小型化的编译配置

步骤 1 对小型化配置文件 ws73_usb_light.config 或者 ws73_usb_light_v2.config 进行配置，配置内容为编译器存放路径，内核存放路径，通信总线类型等信息，具体配置含义请参见《WS73V100 Linux 平台驱动移植 用户指南》“config 配置方法”章节。

步骤 2 将配置好的小型化的文件内容覆盖到 build/config/ws73_default.config，然后输入编译命令 make 进行编译。

----结束

说明

- ws73_usb_light.config 是小型化 V1.0 的配置文件。
- ws73_usb_light_v2.config 是小型化 V2.0 的配置文件。

2.2 小型化版本编译

采用 make 可进行小型化版本的驱动编译，编译结果保存目录是“out/bin”。

采用 make tools_light 可进行小型化版本的工具编译，编译结果保存目录是”out/bin/open_source”。

说明

小型化版本驱动默认裁减掉了 wow 功能，flash 中可以不用放入 wow.bin

小型化版本驱动工具 wpa_supplicant 默认不再需要 libnl-route-3.so.200.26.0 库的支撑，flash 中可以不用放入。

2.3 工具自定义裁减

工具小型化版本的编译依赖于文件 open_source/open_source_light.mk，默认给出的是小型化的基线版本，如果客户需要开启某些 wpa_supplicant 的功能，可以通过配置 wpa_supplicant 目标中的宏开关进行配置。

比如通过将“@sed -i "s/CONFIG_DRIVER_WEXT=y/#CONFIG_DRIVER_WEXT=y/g” ${WPA_SUPPLICANT_DIR}/wpa_supplicant/.config”

修改为“@sed -i "s/#CONFIG_DRIVER_WEXT=y/CONFIG_DRIVER_WEXT=y/g” ${WPA_SUPPLICANT_DIR}/wpa_supplicant/.config"可开启 wpa_supplicant 的 WEXT 功能。

先执行清理“make tools_clean_light”

再执行编译“make tools_light”

L口说明

WPA 的某些宏之间有依赖关系，需要在充分了解的基础上同时开启或者关闭相对应的宏。

2.4 小型化版本特性差异

全量版本：正常编译的版本。

小型化 V1.0 版本：将 ws73_usb_light.config 的内容拷贝到 ws73_default.config 进行编译的版本。

小型化 V2.0 版本：将 ws73_usb_light_v2.config 的内容拷贝到 ws73_default.config 进行编译的版本。

说明

在内容拷贝后，仅修改了工具链、内核路径、总线模式等信息，而不修改 WIFI 的配置宏等相关信息。

2.4.1 小型化 V1.0 版本

小型化 V1.0版本是在全量版本的基础上裁减了特性宏，如表 2-1所示。

表2-1小型化 V1.0 Wi-Fi 主要裁减特性宏

序号	特性宏说明	宏	简要描述	详细描述
1	Enable WIFI DEBUG	WIFI_DEBUG	使能 WIFI 内部调试功能，用于问题定位。	debug 命令和日志，WiFi 驱动串口日志打印。
2	Enable WPS	WIFI_WPS	WPS 关联功能，通常在一键连接，P2P 等场景中应用。	开启后支持 WPS 连接方式。
3	Enable Wireless EXT	_PRE_WLAN_WIRELESS_EXT	使能 Wireless Extension 接口支持，基于 IoCtrl 的 Wireless 命令通道。	开启后支持 iwconfig、iwlist 等工具。
4	Enable BSRP_NFRP	WIFI_BSRP_NFRP	使能 WiFi6 BSRP 和 NFRF 功能。	BSRP：11ax 特性 Buffer Status Reports。 NFRP：11ax 特性 NDP Feedback Report Poll。
5	Enable SLP	WIFI_SLP	使能与外部 SLP(星闪测距/定位)模块的对接功能。	需要额外一块 SLP 芯片。定位与测距的实现在 GLP 芯片和 host 侧 RM 模块完成，Wi-Fi 驱动层面完成数据透传、测距脉冲发出和数据通信等操作。

序号	特性宏说明	宏	简要描述	详细描述
6	Enable APF	WIFI_APF	使能过滤指定报文的功能。	一种低功耗特性。 开启 APF 功能后，在低功耗下，WiFi MCU 会根据配置的 APF 报文过滤规则，过滤 ipv6、ipv4、ARP、DHCP 等报文，防止主控芯片被不必要的报文频繁唤醒。
7	Enable repeater	WIFI_SINGLE_PROXYSTA	使能中继模式。	中继 WiFi Repeater 模式，设备同时支持 STA 与 AP，在 STA 与 AP 之间转发数据包。
8	Enable CSI	WIFI_CSI	使能 CSI(信道状态信息)的 WiFi 感知功能。	开启后支持衡量 WiFi 信道状态信息，从而做到人体感知、精细定位以及细粒度动作识别等功能。
9	Enable M2U	WIFI_M2U	组播转单播。	视频流通过组播转单播特性，提升组播流的视频质量。
10	Enable blacklist	WIFI_BLACKLIST	WiFi SoftAP 访问黑白名单功能。	开启后支持命令增、删、查看黑名单或白名单。
11	Enable WAPI	WIFI_WAPI	WAPI 加密功能。	开启后支持 WAPI 加密方式。
12	Enable SDP	WIFI_SDP	提供 WiFi Aware 基础服务，需二次开发后使用。	开启后支持 WiFi Aware 无感配网基础服务，通常搭配对端 WiFi Aware 设备二次开发使用。
13	Enable latency stat	WIFI_LATENCY_STAT	使能驱动内部报文传输时延统计功能。	统计报文在驱动内部各阶段的处理时长，内部优化用。

序号	特性宏说明	宏	简要描述	详细描述
14	Enable promisc	WIFI_PROMISC	提供 WiFi 混杂模式的空口抓包功能。	内部调试用，启用后开启混杂模式，可指定接收的报文类型。
15	Enable uapsd	WIFI_UAPSD	使能 WiFi UAPSD 节能模式。	一种 WiFi 节能模式，Unscheduled Automatic Power Save Delivery。
16	Enable PSD	WIFI_PSD	使能 WiFi 功率频谱探测数据上报功能。	需搭配 debugkit 工具使用，仅在无业务模式生效，开启后采集 2.4G 能量信息，可以在 debugkit 工具绘制信道频谱图。
17	Enable TWT	WIFI_TWT	使能 WiFi6 TWT 休眠特性。	11ax 新特性，节能使用。TWT （Target Wake Time），目标唤醒时间，核心功能就是在 STA 和 AP 建立 TWT SP（服务周期），将各 STA 访问媒体的时间错开，降低竞争冲突；STA 在 TWT SP 之外可以进入睡眠状态，降低功耗。
18	Enable DNB	WIFI_DNB	使能动态窄带特性。	配合支持此功能的路由器（如，华为 AX3），可增大 WiFi 覆盖。
19	Enable SR	WIFI_SR_STA	WiFi 6 STA 侧的 SR（信道空间复用）特性。	WiFi6 Spatial Reuse 特性。
20	Enable DFT	_PRE_WLAN_DFT_STAT	驱动维测功能。	开启后使能驱动收发包维测、掉线维测、内存维测等，内部定位问题用。

序号	特性宏说明	宏	简要描述	详细描述
21	Enable SNIFFER	_PRE_WLAN_FEATURE_SNIFFER	将混杂模式抓取的报文存为文件，与 Promisc 特性配合使用。	Linux 系统专用，与 Promisc 特性配合使用，将混杂抓取到的报文保存在特定文件目录。
22	Enable DAQ	WIFI_DAQ	Wifi 底层数采维测功能，通常用于定位 MAC PHY 收发包功能。	开启后使能数采功能，收集 MAC、PHY 的收发信息并存储，内部定位问题使用。
23	Enable ccpriv commands	_PRE_WLAN_SUPPORT_CCPRIV_CMD	CCPRIV 命令，调测使用。	使能调测命令，可以按需开放部分指令。
24	Enable DFX support sysfs	CONFIG_DFX_SUPPORT_SYSFS	WiFi DFX 维测功能。	HSO 工具自身的 DFX 功能，用于内部调试 debugkit 工具。
25	Enable dfx shell proc	CONFIG_DFX_SUPPORT_SHELL_PROC	HSO 工具的 DFX 维测功能。	通过 debugkit 将 shell 命令输入到单板执行，仅用于内部调试。此宏依赖于 Enable diag socket support 的开启 。

2.4.2 小型化 V2.0 版本

小型化 V2.0 版本是在小型化 V1.0 的版本的基础上裁减了特性宏，如表 2-2所示。

表2-2 小型化 V2.0 Wi-Fi 主要裁减特性宏

序号	特性宏说明	宏	简要描述	详细描述
1	Enable MFG TEST	WSCFG_MFG_TEST	产测功能。	开启后支持射频校准，读写 Efuse 校准数据等功能。

序号	特性宏说明	宏	简要描述	详细描述
2	Enable WIFI CSA	WIFI_CSA	WiFi CSA 信道切换宣告。	例如路由器 AP 在 DFS 信道检测到雷达信号，切换信道，通过 CSA 报文/IE 通知 STA 同步进行切换。
3	Enable WPA3	_PRE_WLAN_FEATURE_WPA3	WIFI WPA3 加密功能。	开启后支持 WPA3 加密方式。
4	Enable wifi-bt coexist	WIFI_BTCOEX	Wi-Fi 与内建蓝牙模块业务共存。	芯片支持 WiFi 和蓝牙同时通信的共存。
5	Enable Tcp Ack Filter	WIFI_TCP_ACK_FILTER	TCP ACK 报文过滤功能。	TCP 打流场景，在超时范围内丢掉部分 ACK，优化极限场景打流数据，可能会增大时延，导致打流不稳定。
6	Enable ROAM（含 kvr、mbo）	WIFI_BASE_ROAM WIFI_11KVR WIFI_MBO	基础漫游， 11K/11V/11R 协议， WiFi MBO 多频操作功能。	开启后能支持基础漫游、802.11kvr 漫游。
7	Enable auto adjust freq	WIFI_AUTO_ADJUST_FREQ	WiFi Device MCU 自动调频功能。	默认关闭，调节 73 Device CPU 的频率，目的是降保连时的功耗。
8	Enable tx amsdu	WIFI_TX_AMSDU	WiFi 发送 AMSDU 聚合。	聚合能提升 RX 打流性能，主要是短包聚合， 比如回复 TCP ACK 的聚合。
9	Enable alg anti interference	WIFI_ALG_ANTI_INTF	弱干扰免疫算法。	算法屏蔽过滤弱信号的报文。
10	Enable tx classify	_PRE_WLAN_FEATURE_TX_CLASSIF	发送流分类。	802.11 QoS 场景，使能后通过五元组映射 TID。

序号	特性宏说明	宏	简要描述	详细描述
		Y_LAN_TO_WLAN		关闭后，使用 DSCP 映射 TID。
11	Enable always tx	WIFI_ALWAYS_TX	常发常收测试命令。	产测和硬件射频测试使用。
12	Enable DFR	WIFI_BASE_DFR	WIFI DFR 自愈功能。	当系统发生 panic 重启 WiFi 模组 客户有自己的异常恢复方案，可裁剪；可能会和客户方案冲突。
13	Enable diag socket support	CONFIG_DIAG_SUPPORT_SOCKET	HSO 工具网口。	通过网口输出 log 至 debugkit，查看在线日志，仅用于 WIFI/BLE 驱动问题定位，客户不需要。
14	Enable diag uart support	CONFIG_DIAG_SUPPORT_UART	HSO 工具串口。	通过串口输出 log 至 debugkit，查看在线日志， 仅用于 WIFI/BLE 驱动问题定位，客户不需要。
15	Enable diag local log support	CONFIG_DIAG_SUPPORT_LOCAL_LOG	HSO 工具 LOG。	将需要删除到 debugkit 的日志存储到本地文件系统，仅用于 WIFI/BLE 驱动问题定位，客户不需要。
16	Enable Plat DFR	CONFIG_PLAT_SUPPORT_DFR	WiFi 平台领域 DFR 自愈功能。	当系统发生 panic 重启 WiFi 模组 客户有自己的异常恢复方案，可裁剪；可能会和客户方案冲突。
17	Enable diag log out	WSCFG_PLAT_DIAG_L	PLAT 打印开关。	PLAT 串口打印开关。

序号	特性宏说明	宏	简要描述	详细描述
		OG_OUT		

2.5 小型化 V2.0INI 配置

小型化 V2.0 开启了配置宏_PRE_WLAN_NO_SUPPORT_INI，此宏的含义为删除 INI 配置文件，不再使用 INI 配置文件作为接口，配置文件中配置的内容已经放到全局变量中，全局变量是 driver/wifi/common/customize_wifi.c 文件中被

_PRE_WLAN_NO_SUPPORT_INI 宏包裹的部分，一般情况下不需要客户手动修改。

针对大区功率做以下特殊说明（客户可根据实际情况更改大区功率），大区功率的全局变量为数组 g_default_cust_tx_powe，见下表：

表2-3 小型化版本全局变量设置大区功率

数组	数组下标	对应大区	对应国家码
g_default_cust_tx_powe[REGDOMAIN_FCC]	0	REGDOMAIN_FCC	RU,AU,MY,ID,TR,PL,FR,PT,IT,DE,ES,AR,ZA,MA,PH,TH,GB,CO,MX,EC,PE,CL,SA,EG,AE
g_default_cust_tx_powe[REGDOMAIN_ETSI]	1	REGDOMAIN_ETSI	US CA KH
g_default_cust_tx_powe[REGDOMAIN_JAPAN]	2	REGDOMAIN_JAPAN	JP
g_default_cust_tx_powe[REGDOMAIN_COMMON]	3	REGDOMAIN_COMMON	CN

启动时会根据国家码 g_ac_country_code_etc 去刷新对应大区功率，默认的国家码为全局变量

osal_char g_ac_country_code_etc[COUNTRY_CODE_LEN] = "CN";

当国家码更改后，代码会更新使用国家码对应的大区功率。