说明：该文档以调换FFF0服务下的FFF1和FFF2通道为例子进行说明，客户可根据此文档对其他服务下的通道进行修改。

1. 在fff0s.h文件下做如下图修改。

```
.c (projects\ble_app_gatt\app)  app_fff0.c (projects\ble_app_gatt\app)  fff0s.c (sdk\...\src)  fff0s.h (sdk\...\api) * ×  ff
63: };
64:
65: /// Battery Service Attributes Indexes
66: enum
67: {
68:     FFF0S_IDX_SVC,
69:     #if 0
70:     FFF0S_IDX_FFF2_LVL_CHAR,
71:     FFF0S_IDX_FFF2_LVL_VAL,
72:     FFF0S_IDX_FFF1_LVL_CHAR,
73:     FFF0S_IDX_FFF1_LVL_VAL,
74:     FFF0S_IDX_FFF1_LVL_NTF_CFG,
75:     #else
76:     FFF0S_IDX_FFF1_LVL_CHAR,
77:     FFF0S_IDX_FFF1_LVL_VAL,
78:
79:     FFF0S_IDX_FFF2_LVL_CHAR,
80:     FFF0S_IDX_FFF2_LVL_VAL,
81:     FFF0S_IDX_FFF2_LVL_NTF_CFG,
82:     #endif
83:     FFF0S_IDX_NB,
84: };
```

2. 在fff0s.c文件下的fff0_att_db数组里做如下图修改。

```
/// Full FFF0 Database Description - Used to add attributes into the database
const struct attm_desc fff0_att_db[FFF0S_IDX_NB] =
{
    // FFF0 Service Declaration
    [FFF0S_IDX_SVC]              =   {ATT_DECL_PRIMARY_SERVICE, PERM(RD, ENABLE), 0, 0},

    [FFF0S_IDX_FFF1_LVL_CHAR]  =   {ATT_DECL_CHARACTERISTIC, PERM(RD, ENABLE), 0, 0},
    //  Characteristic Value
    [FFF0S_IDX_FFF1_LVL_VAL]   =   {ATT_USER_SERVER_CHAR_FFF1, PERM(WRITE_COMMAND, ENABLE), PERM(RI, ENABLE), FFF0_FFF1_DATA_LEN *sizeof(uint8_t)},

    // fff1 Level Characteristic Declaration
    [FFF0S_IDX_FFF2_LVL_CHAR]  =   {ATT_DECL_CHARACTERISTIC, PERM(RD, ENABLE), 0, 0},
    // fff1 Level Characteristic Value
    [FFF0S_IDX_FFF2_LVL_VAL]   =   {ATT_USER_SERVER_CHAR_FFF2, PERM(WRITE_COMMAND, ENABLE) , PERM(RI, ENABLE), FFF0_FFF2_DATA_LEN * sizeof(uint8_t)},
    // fff1 Level Characteristic - Client Characteristic Configuration Descriptor
    [FFF0S_IDX_FFF2_LVL_NTF_CFG] = {ATT_DESC_CLIENT_CHAR_CFG,  PERM(RD, ENABLE)|PERM(WRITE_REQ, ENABLE), 0, 0},

};/// Macro used to retrieve permission value from access and rights on attribute.
```

3. 在fff0s.c文件下的fff0s_init函数下做如下图修改。

```
53: static uint8_t fff0s_init (struct prf_task_env* env, uint16_t* start_hdl, uint16_t app_task, uint8_t sec_lvl,  struct fff0s_
54: {
55:     uint16_t shdl;
56:     struct fff0s_env_tag* fff0s_env = NULL;
57:     // Status
58:     uint8_t status = GAP_ERR_NO_ERROR;
59:     //-------------------- allocate memory required for the profile  ---------------------
60:     fff0s_env = (struct fff0s_env_tag* ) ke_malloc(sizeof(struct fff0s_env_tag), KE_MEM_ATT_DB);
61:     memset(fff0s_env, 0 , sizeof(struct fff0s_env_tag));
62:     // Service content flag
63:     uint8_t cfg_flag = FFF0S_CFG_FLAG_MANDATORY_MASK;
64:     // Save database configuration
65:     fff0s_env->features |= (params->features) ;
66:     // Check if notifications are supported
67:     if (params->features == FFF0_FFF1_LVL_NTF_SUP)
68:     {
69:         cfg_flag |= FFF0_CFG_FLAG_NTF_SUP_MASK;
70:     }
71:     shdl = *start_hdl;
72:     //Create FFF0 in the DB
73:     //------------------ create the attribute database for the profile -------------------
74:     status = attm_svc_create_db128(&(shdl), ATT_USER_SERVER_FFF0, (uint8_t *)&cfg_flag,
75:             FFF0S_IDX_NB, NULL, env->task, &fff0_att_db[0],
76:             (sec_lvl & (PERM_MASK_SVC_DIS | PERM_MASK_SVC_AUTH | PERM_MASK_SVC_EKS)));
77:     //Set optional permissions
78:     if (status == GAP_ERR_NO_ERROR)
79:     {
80:         //Set optional permissions
81:         if(params->features == FFF0_FFF1_LVL_NTF_SUP)
82:         {
83:             // Battery Level characteristic value permissions
84:             uint16_t perm = PERM(RD, ENABLE) | PERM(NTF, ENABLE);
85:
86:             attm_att_set_permission(shdl + FFF0S_IDX_FFF2_LVL_VAL, perm, 0);
87:         }
88:     }
```

4. 在fff0s.c文件下的fff0s_notify_fff1_lvl函数下做如下图修改。

```
void fff0s_notify_fff1_lvl(struct fff0s_env_tag* fff0s_env, struct fff0s_fff1_level_upd_req const *param)
{
    // Allocate the GATT notification message
    struct gattc_send_evt_cmd *fff1_lvl = KE_MSG_ALLOC_DYN(GATTC_SEND_EVT_CMD,
            KE_BUILD_ID(TASK_GATTC, 0), prf_src_task_get(&(fff0s_env->prf_env),0),
            gattc_send_evt_cmd, sizeof(uint8_t)* (param->length));
    // Fill in the parameter structure
    fff1_lvl->operation = GATTC_NOTIFY;
    fff1_lvl->handle = fff0s_get_att_handle(FFF0S_IDX_FFF2_LVL_VAL);
    // pack measured value in database
    fff1_lvl->length = param->length;
    //fff1_lvl->value[0] = fff0s_env->fff1_lvl[0];
    memcpy(&fff1_lvl->value[0],&param->fff1_level[0],param->length);
    // send notification to peer device
    ke_msg_send(fff1_lvl);
}
```

5. 在fff0s.c文件下的fff0s_get_att_handle函数下做如下图修改。

```
uint16_t fff0s_get_att_handle( uint8_t att_idx)
{

    struct fff0s_env_tag *fff0s_env = PRF_ENV_GET(FFF0S, fff0s);
    uint16_t handle = ATT_INVALID_HDL;
    handle = fff0s_env->start_hdl;
    // increment index according to expected index
    if(att_idx < FFF0S_IDX_FFF2_LVL_NTF_CFG)
    {
        handle += att_idx;
    }
    // FFF1 notification
    else if((att_idx == FFF0S_IDX_FFF2_LVL_NTF_CFG) && (((fff0s_env->features ) & 0x01) == FFF0_FFF1_LVL_NTF_SUP))
    {
        handle += FFF0S_IDX_FFF2_LVL_NTF_CFG;
    }
    else
    {
        handle = ATT_INVALID_HDL;
    }
    return handle;
}
```

6. 在fff0s.c文件下的fff0s_get_att_idx函数下做如下图修改。

```
uint8_t fff0s_get_att_idx(uint16_t handle, uint8_t *att_idx)
{
    struct fff0s_env_tag* fff0s_env = PRF_ENV_GET(FFF0S, fff0s);
    uint16_t hdl_cursor = fff0s_env->start_hdl;
    uint8_t status = PRF_APP_ERROR;
    // Browse list of services
    // handle must be greater than current index
    // check if it's a mandatory index
    if(handle <= (hdl_cursor + FFF0S_IDX_FFF2_LVL_VAL))
    {
        *att_idx = handle -hdl_cursor;
        status = GAP_ERR_NO_ERROR;
    }
    hdl_cursor += FFF0S_IDX_FFF2_LVL_VAL;
    // check if it's a notify index
    if(((fff0s_env->features ) & 0x01) == FFF0_FFF1_LVL_NTF_SUP)
    {
        hdl_cursor++;
        if(handle == hdl_cursor)
        {
            *att_idx = FFF0S_IDX_FFF2_LVL_NTF_CFG;
            status = GAP_ERR_NO_ERROR;
        }
    }
    hdl_cursor++;
    return (status);
}
#endif // (BLE_fff0_SERVER)
```

7. 在fff0s_task.c文件下的gattc_att_info_req_ind_handler函数下做如下图修改。

```
static int gattc_att_info_req_ind_handler(ke_msg_id_t const msgid,
        struct gattc_att_info_req_ind *param,
        ke_task_id_t const dest_id,
        ke_task_id_t const src_id)
{
    struct gattc_att_info_cfm * cfm;
    uint8_t  att_idx = 0;
    // retrieve handle information
    uint8_t status = fff0s_get_att_idx(param->handle, &att_idx);
    //Send write response
    cfm = KE_MSG_ALLOC(GATTC_ATT_INFO_CFM, src_id, dest_id, gattc_att_info_cfm);
    cfm->handle = param->handle;
    if(status == GAP_ERR_NO_ERROR)
    {
        // check if it's a client configuration char
        if(att_idx == FFF0S_IDX_FFF2_LVL_NTF_CFG)
        {
            // CCC attribute length = 2
            cfm->length = 2;
        }
        // not expected request
        else
        {
            cfm->length = 0;
            status = ATT_ERR_WRITE_NOT_PERMITTED;
        }
    }
    cfm->status = status;
    ke_msg_send(cfm);
    return (KE_MSG_CONSUMED);
}
```

8. 在fff0s_task.c文件下的gattc_write_req_ind_handler函数下做如下图修改。

```
static int gattc_write_req_ind_handler(ke_msg_id_t const msgid, struct gattc_write_req_ind const *param,
                                       ke_task_id_t const dest_id, ke_task_id_t const src_id)
{
    struct gattc_write_cfm * cfm;
    uint8_t att_idx = 0;
    uint8_t conidx = KE_IDX_GET(src_id);
    // retrieve handle information
    uint8_t status = fff0s_get_att_idx(param->handle,  &att_idx);
    // If the attribute has been found, status is GAP_ERR_NO_ERROR
    if (status == GAP_ERR_NO_ERROR)
    {
        struct fff0s_env_tag* fff0s_env = PRF_ENV_GET(FFF0S, fff0s);
        // Extract value before check
        uint16_t ntf_cfg = co_read16p(&param->value[0]);
        // Only update configuration if value for stop or notification enable
        if ((att_idx == FFF0S_IDX_FFF2_LVL_NTF_CFG)
                && ((ntf_cfg == PRF_CLI_STOP_NTFIND) || (ntf_cfg == PRF_CLI_START_NTF)))
        {
            // Conserve information in environment
            if (ntf_cfg == PRF_CLI_START_NTF)
            {
                // Ntf cfg bit set to 1
                fff0s_env->ntf_cfg[conidx] |= (FFF0_FFF1_LVL_NTF_SUP );
            }
            else
            {
                // Ntf cfg bit set to 0
                fff0s_env->ntf_cfg[conidx] &= ~(FFF0_FFF1_LVL_NTF_SUP );
            }
            // Inform APP of configuration change
            struct fff0s_fff1_level_ntf_cfg_ind * ind = KE_MSG_ALLOC(FFF0S_FFF1_LEVEL_NTF_CFG_IND,
                    prf_dst_task_get(&(fff0s_env->prf_env), conidx), dest_id,
                    fff0s_fff1_level_ntf_cfg_ind);
            ind->conidx = conidx;
            ind->ntf_cfg = fff0s_env->ntf_cfg[conidx];
            ke_msg_send(ind);
        }
        else if (att_idx == FFF0S_IDX_FFF1_LVL_VAL)
        {
            // Allocate the alert value change indication
            struct fff0s_fff2_writer_ind *ind = KE_MSG_ALLOC(FFF0S_FFF2_WRITER_REQ_IND,
```

9. 在fff0s_task.c文件下的gattc_read_req_ind_handler函数下做如下图修改。

```
static int gattc_read_req_ind_handler(ke_msg_id_t const msgid, struct gattc_read_req_ind const *param,
                                      ke_task_id_t const dest_id, ke_task_id_t const src_id)
{
    struct gattc_read_cfm * cfm;
    uint8_t  att_idx = 0;
    uint8_t conidx = KE_IDX_GET(src_id);
    // retrieve handle information
    uint8_t status = fff0s_get_att_idx(param->handle, &att_idx);
    uint16_t length = 0;
    struct fff0s_env_tag* fff0s_env = PRF_ENV_GET(FFF0S, fff0s);
    // If the attribute has been found, status is GAP_ERR_NO_ERROR
    if (status == GAP_ERR_NO_ERROR)
    {
        // read notification information
        if (att_idx == FFF0S_IDX_FFF2_LVL_VAL)
        {
            length = FFF0_FFF1_DATA_LEN * sizeof(uint8_t);
        }
        // read notification information
        else if (att_idx == FFF0S_IDX_FFF2_LVL_NTF_CFG)
        {
            length = sizeof(uint16_t);
        }
        else
        {
            status = PRF_APP_ERROR;
        }
    }
    //Send write response
    cfm = KE_MSG_ALLOC_DYN(GATTC_READ_CFM, src_id, dest_id, gattc_read_cfm, length);
    cfm->handle = param->handle;
    cfm->status = status;
    cfm->length = length;
    if (status == GAP_ERR_NO_ERROR)
    {
        // read notification information
        if (att_idx == FFF0S_IDX_FFF2_LVL_VAL)
        {
            cfm->value[0] = fff0s_env->fff1_lvl[0];
        }
        // retrieve notification config
        else if (att_idx == FFF0S_IDX_FFF2_LVL_NTF_CFG)
        {
            uint16_t ntf_cfg = (fff0s_env->ntf_cfg[conidx] & FFF0_FFF1_LVL_NTF_SUP) ? PRF_CLI_START_NTF : PRF_CLI_STOP_NTFIND;
            co_write16p(cfm->value, ntf_cfg);
        }
```

10. 修改完成后，原始SDK的FFF0服务下FFF1通道的write no response属性会被修改成notify、read属性， FFF2通道的notify、read属性会被修改成write no response属性