

Bluetooth Mesh 数据收发与 DP 控制

概述

Bluetooth mesh spec 中规定设备的通信使用 mesh model 中相应的 opcode（详细参考 bluetooth mesh 官方文档 MshMDL_v1.1.pdf），Tuya IoT 平台使用 DP 模型通信（详细可在 Tuya IoT 平台创建产品后查看）。所以在平台 DP 数据与标准 Mesh 数据协议中就存在一个转换规则，针对不同品类有不同的转换模板与协议规则。

品类的划分根据 Mesh 能力值确定，可参考 [能力地图 - Mesh Category](#) 文档。

照明品类

Opcode 具体数据格式的定义在 mesh spec 中，由于文档较大不方便查找，开发者可以参考 `tal_bluetooth_mesh_def.h` 文件中结构体定义，例如 Light lightness set 命令的结构体格式为以下格式：

```
typedef struct{
    USHORT_T lightness;           /**< The target value of the Light Lightness
Actual state */
    UCHAR_T tid;                 /**< Transaction Identifier */
    UCHAR_T transit_t;           /**< Generic Default Transition Time(optional)
*/
    UCHAR_T delay;               /**< If the transit_t field is present, the
Delay field shall also be present; otherwise these fields shall not be present. */
}PACKED TAL_MESH_LIGHT_LIGHTNESS_SET_T;
```

照明类设备，是使用 Bluetooth mesh 标准 model 最多的，DP 点与 mesh opcode 的对应规则如下：

DP ID	DP功能	Model	Opcode	Element index	备注
1	开关	Generic OnOff Model	Generic onoff get Generic onoff set Generic onoff set unack Generic onoff status	0	-
2	模式	Tuya Vendor Model	Vendor read Vendor write Vendor write unack Vendor data	0	用于白光彩光模式切换 注意：照明品类模式DP较为特殊，不使用Vendor透传协议，只有一个Byte数据用来表示照明模式（详细参考Demo实现）。
3	亮度	Light Lightness Model	Light lightness get Light lightness set Light lightness set unack Light lightness status	0	SIG MESH 数据默认范围：0-65535

DP ID	DP功能	Model	Opcode	Element index	备注
4	色温	Light CTL Tempture Model	Light ctl tempture get Light ctl tempture set Light ctl tempture set unack Light ctl tempture status	0	SIG MESH 数据默认范围： 800-20000
5	颜色	Light HSL Model	Light hsl get Light hsl set Light hsl set unack Light hsl status	0	Mesh 规定使用 HSL 模型， Tuya DP 点为 HSV 模型， 中间需要转换
6	情景	Tuya Vendor Model	Vendor read Vendor write Vendor write unack Vendo data	0	DP 数据，情景模式数据经过特殊压缩处理，降低数据长度，详细参考照明品类包中相关介绍

DP ID	DP功能	Model	Opcode	Element index	备注
7	音乐	Tuya Vendor Model	Vendor read Vendor write Vendor write unack Vendo data	0	DP 数据，音乐灯数据经过特殊压缩处理，降低数据长度
其他	其他 DP	Tuya Vendor Model	Vendor read Vendor write Vendor write unack Vendo data	0	其他 DP 点数据均采用 Vendor Model 透传协议处理

电工品类

电工类设备也是较为特殊的，由于存在多路的开关，设备需要具备多个 element。

DP ID	DP功能	Model	Opcode	Element index	备注
1	开关1	Generic OnOff Model	Generic onoff get Generic onoff set Generic onoff set unack Generic onoff status	0	第1路设备的开关控制与状态
2	开关2	Generic OnOff Model	Generic onoff get Generic onoff set Generic onoff set unack Generic onoff status	1	第2路设备的开关控制与状态
3	开关3	Generic OnOff Model	Generic onoff get Generic onoff set Generic onoff set unack Generic onoff status	2	第3路设备的开关控制与状态
4	开关4	Generic OnOff Model	Generic onoff get Generic onoff set Generic onoff set unack Generic onoff status	3	第4路设备的开关控制与状态
5	开关5	Generic OnOff Model	Generic onoff get Generic onoff set Generic onoff set unack Generic onoff status	4	第5路设备的开关控制与状态
6	开关6	Generic OnOff Model	Generic onoff get Generic onoff set Generic onoff set unack Generic onoff status	5	第6路设备的开关控制与状态

DP ID	DP功能	Model	Opcode	Element index	备注
其他	其他 DP	Tuya Vendor Model	Vendor read Vendor write Vendor write unack Vendo data	0	其他 DP 点数据均采用 Vendor Model 透传协议处理

对于电工类设备有多个开关时，控制不同开关时 opcode 是相同的，对于区分是哪一路开关就是根据不同 element 来区分，不同的 element 地址不同。element 0 的地址即为设备的单播地址，也可以称之为主 element 地址，使用 `tal_primary_ele_addr_get()` 可以获取。

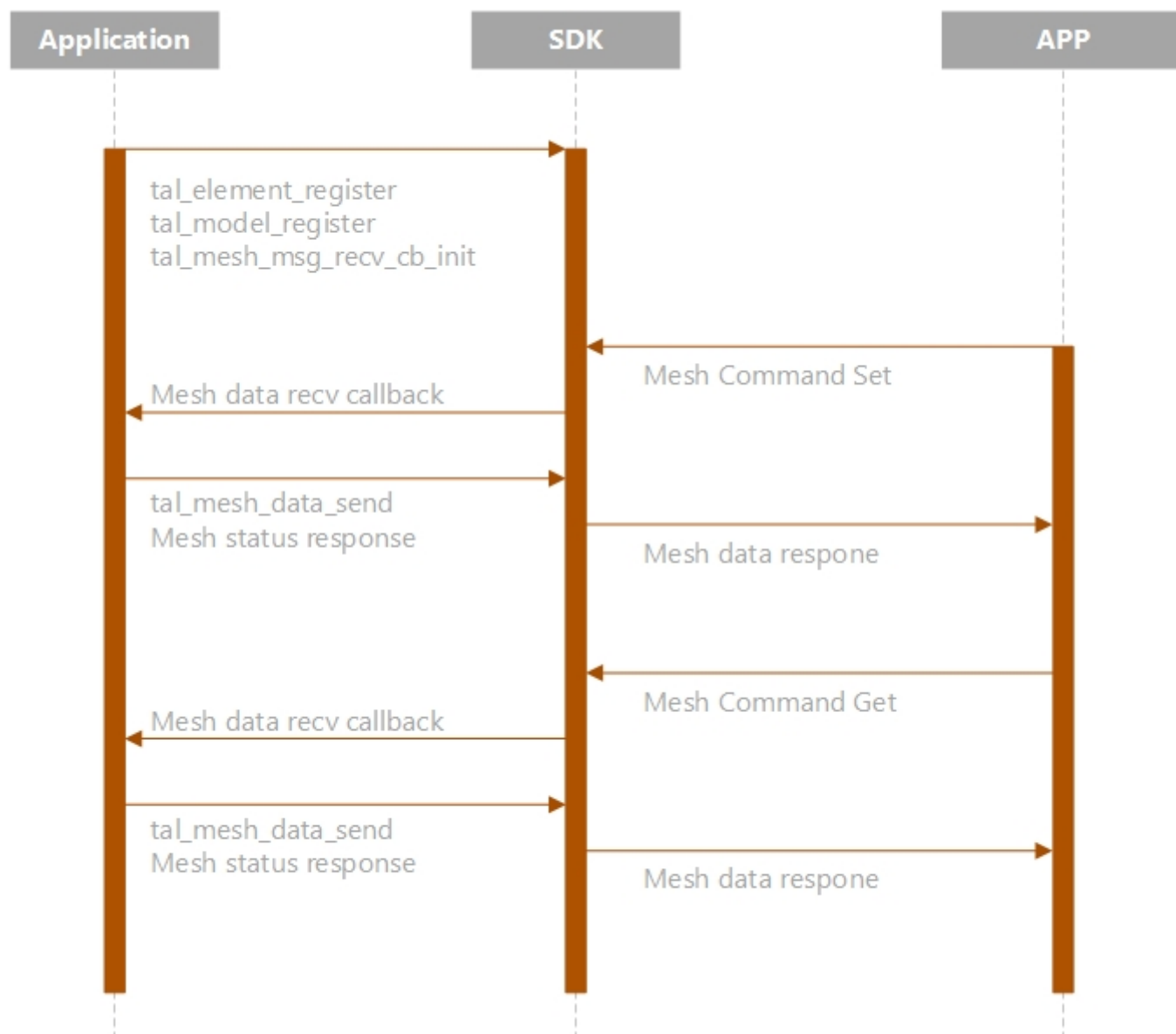
其他类与透传类

除照明品类与电工品类之外的产品，均采用 vendor 透传处理。所以能力值中大小类只是用来做相关记录，无特殊的品类处理。

DP ID	DP功能	Model	Opcode	Element index	备注
All	所有 DP	Tuya Vendor Model	Vendor read Vendor write Vendor write unack Vendor data	0	其他 DP 点数据均采用 Vendor Model 透传协议处理
无	无	Generic OnOff Model	Generic onoff get Generic onoff status	0	对于长供电设备，需要支持 Generic OnOff Model, Generic onoff get 与 Generic onoff status 命令用作心跳，设备如果没有开关类状态，则此处 Generic onoff status 可以回复默认值为 1 的状态。

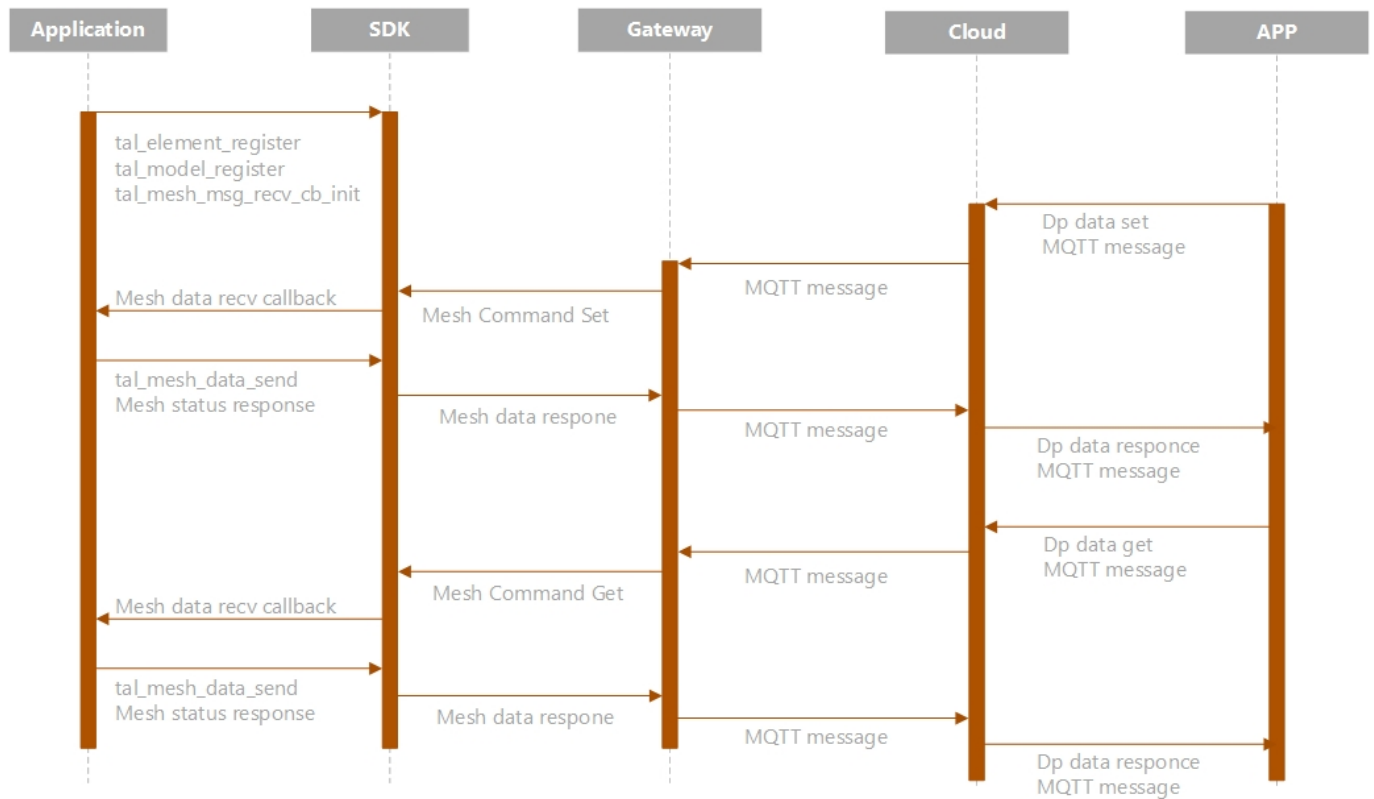
通信流程

APP 与设备通信流程



注意：App 控制设备后，不会将设备的状态同步到云端。如果需要在云端同步更新，则需要通过网关控制。

网关与设备通信流程



数据结构

TAL_MESH_ACCESS_MSG_T

```

typedef struct {
    UINT_T    opcode;           /**< Mesh opcode. */
    UCHAR_T   *data;           /**< Mesh data. */
    USHORT_T  data_len;        /**< Mesh data length. */
} TAL_MESH_ACCESS_MSG_T;
  
```

opcode: Mesh spec 定义的命令字

data: 数据内容指针，详细数据格式为 spec 中定义的格式，可参考 `tal_bluetooth_mesh_def.h`

data_len: 数据内容长度

TAL_MESH_ACCESS_MSG_T

```
typedef struct {
    USHORT_T  src_addr;           /**< Source unicast address. */
    USHORT_T  dst_addr;          /**< Destination unicast address. */
    UINT_T    seq;               /**< Sequence num of this msg. */
    UCHAR_T   ttl;               /**< Time To Live. */
    USHORT_T  app_key_index;      /**< The appkey index of this msg ues. */
    USHORT_T  net_key_index;      /**< The networkkey index of this msg ues. */
    CHAR_T    rssi;              /**< used when rx in adv bearer. */
} TAL_MESH_NET_PARAM_T;
```

src_addr: Mesh 消息来源节点地址，即发送此消息节点的地址，只可能为单播地址

dst_addr: Mesh 消息目的地址，可以为节点地址，也可以为组播地址

seq: 此 Mesh 消息 sequence num，用于防重放

ttl: 此 Mesh 消息的 TTL，转发过程中会不断减小（Tuya App 与网关默认使用 TTL = 8 来发送消息，接收设备可以根据收到的 TTL 来判断消息被转发了几次）

app_key_index: 消息加密所使用的 app key index，默认为 0

net_key_index: 消息加密所使用的 net key index，默认为 0

rssi: 当前 Mesh 广播接收到时的信号强度（不一定为发送节点的信号强度，也可能是其他中级节点转发的广播）

tal_mesh_msg_rcv_cb

```
typedef OPERATE_RET (*tal_mesh_msg_rcv_cb)(TAL_MESH_ACCESS_MSG_T *msg_raw,
TAL_MESH_NET_PARAM_T *net_param);
```

Mesh 数据接收回调定义：

msg_raw: Mesh 消息数据，具体格式参考前文

net_param: Mesh 消息网络信息，具体格式参控前文

API 说明

Mesh 数据接收回调

```
OPERATE_RET tal_mesh_msg_rcv_cb_init(tal_mesh_msg_rcv_cb access_data_cb);
```

参考 `能力地图-初始化` 通过注册函数在初始化阶段将 Mesh 数据接收回调注册到 SDK 中。在 APP / 网关或者其他节点向设备发送数据时在此回调内即可收到对应的数据。

Mesh 数据发送

```
OPERATE_RET tal_mesh_data_send(USHORT_T src_addr, USHORT_T dst_addr, UINT_T opcode,
    UCHAR_T *data, USHORT_T data_len);
```

使用此回调即可将数据通过 Access 层接口发送到指定的目的地址。

src_addr：数据发送源地址，为设备自身地址可通过 `USHORT_T tal_primary_ele_addr_get(VOID)` 接口获取

dst_addr：数据发送的目的地址

opcode：发送数据的 opcode，可参考 `tal_bluetooth_mesh_def.h` 中的定义

data：数据指针

data_len：数据长度

Mesh 数据上报

```
#define TUYA_REPROT_PUB_ADDR    0xD000

tal_mesh_data_send(tal_primary_ele_addr_get(), TUYA_REPROT_PUB_ADDR,
    TAL_MESH_OPCODE_ON_OFF_STAT, (UINT8_T*)&generic_onoff_status, 1);
```

Tuya 设备可以被多个 App 与控制，也可以自定义绑定到指定的网关下。所以设备无法获取目前在线的 APP / 网关的地址。所以在主动上报数据时无法使用 App / 网关的地址。

Tuya 的 App / 网关都会订阅 `TUYA_REPROT_PUB_ADDR` 的地址，所以当设备有数据上报时调用数据发送接口，目的地址使用 `TUYA_REPROT_PUB_ADDR` 即可。

Tuya Mesh Vendor Model DP 透传协议

本协议通过规范定义涂鸦 Vendor model 中协议格式，实现数据透传的功能，可以实现蓝牙 mesh model 中标准无法实现的控制逻辑或者数据传输。

概述

涂鸦蓝牙 mesh 产品大类主要包括照明、电工、传感、遥控等。涂鸦在设计蓝牙 mesh 产品体系时，尽量使用蓝牙的标准 model 来实现功能。例如，对于照明品类的开关，使用的是 Generic_OnOff Model。

由于标准 model 实现的功能是有限的，此时需要用到 vendor model。例如，蓝牙照明产品会使用 vendor model 来实现场景配置和切换。

Vendor Model

Vendor Model ID

- Server: 0x07D00004
- Client: 0x07D00005

Vendor Model Opcode

CMD	Op_code	Client: 0x07D00005	Server: 0x07D00004
WRITE	0xC9D007	写数据请求	-
WRITE_UNACK	0xCAD007	不带ACK写数据请求	-
READ	0xCCD007	读状态请求	-
STATUS	0xCBD007	-	保留
DATA	0xCDD007	-	数据回复与上报

数据格式与基础控制协议

字段	字节数	说明
命令字	1	数据类型： 0x01: DP数据 0x02: 时间同步 ...
数据长度	1	注意：DP数据单元无此字段
数据	N	-

注意：为控制 access 曾数据长度，减少数据分包造成的控制延时，因此 DP 数据无数据长度字段。

Datapoint 单元

当使用 vendor model 实现 DP 数据传输功能时，App 或网关将 DP 按照约定格式进行透传。其透传规则如下：

数据段	字节数	说明			
dpid	1	功能点序号			
type	1	对应 IoT 平台上某功能点具体的数据类型，通过如下 表示值 标识			
		类型	表示值	长度（字节）	说明
		Raw	0x00	N	对应于 raw 型 datapoint
		Boolean	0x01	1	范围：0x00/0x01
		Value	0x02	4	对应数值类型，大端表示
		String	0x03	N	对应于具体字符串
		Enum	0x04	1	枚举类型，范围 0-0xFF
		Bitmap	0x05	4	长度大于1字节时，大端表示
len	1	长度对应 data 的字节数 注意只有 raw/string/bitmap 才需要此字段			
data	1/4/N	hex 表示，大于 1 字节采用大端传输			

命令下发

- op_code: 0xC9D007 WRITE
App 或网关发送 WRITE 命令到设备端，设备需要回复 DATA (0xCDD007) 数据。
- op_code: 0xCAD007 WRITE_UNACK
发送 WRITE_UNACK 命令时，设备无需回复。

App 或网关发送：

字段	字节数	说明
命令字	1	0x01 (DP数据, dataPoint)
数据	N	Datapoint 组合（格式见前文）

查询设备状态

op_code: 0xCCD007 READ

App 或网关主动查询设备 DP 状态。发送 DPID 为 0 时，设备收到后通过 DATA 通道主动上报所有 DP 状态。

App 或网关发送：

字段	字节数	说明
命令字	1	0x01（DP 数据）
数据长度	1	N
数据	N * DP_ID (N * 1byte)	DP 序号集合

设备回复与上报

op_code: 0xCDD007 DATA

子设备收到 WRITE 以及 READ 命令时回复此消息。

设备发送：

字段	字节数	说明
命令字	1	0x01（DP 数据）
数据	N	Datapoint 组合（格式见前文）

使用说明

Mesh 数据接收与回复

以下为 mesh 数据接收回调示例（更多的示例参考 SDK demo 中的实现）：

```

OPERATE_RET app_mesh_data_rcv(TAL_MESH_ACCESS_MSG_T *msg_raw, TAL_MESH_NET_PARAM_T
*net_param){
    TAL_MESH_GENERIC_ONOFF_STATUS_T generic_onoff_status;

    switch(msg_raw->opcode){
        case TAL_MESH_OPCODE_ON_OFF_GET:
            generic_onoff_status.present = onoff_data;
            tal_mesh_data_send(net_param->dst_addr, net_param->src_addr,
TAL_MESH_OPCODE_ON_OFF_STAT, (UINT8_T*)&generic_onoff_status, 1);
            break;
        case TAL_MESH_OPCODE_ON_OFF_SET:
        case TAL_MESH_OPCODE_ON_OFF_SET_UNACK:{
            TAL_MESH_GENERIC_ONOFF_SET_T *generic_onoff_set =
(TAL_MESH_GENERIC_ONOFF_SET_T*)msg_raw->data;
            onoff_data = generic_onoff_set->onoff;
            if(TAL_MESH_OPCODE_ON_OFF_SET == msg_raw->opcode){
                generic_onoff_status.present = generic_onoff_set->onoff;
                tal_mesh_data_send(net_param->dst_addr, net_param->src_addr,
TAL_MESH_OPCODE_ON_OFF_STAT, (UINT8_T*)&generic_onoff_status, 1);
            }
        }
        break;
        default:
            break;
    }

    return OPRT_OK;
}

```

Mesh 数据主动上报

```

TAL_MESH_GENERIC_ONOFF_STATUS_T generic_onoff_status;
generic_onoff_status.present = 0x01;
tal_mesh_data_send(tal_primary_ele_addr_get(), TUYA_REPROT_PUB_ADDR,
TAL_MESH_OPCODE_ON_OFF_STAT, (UINT8_T*)&generic_onoff_status, 1);

```

示例为主动上报设备自己的开关状态。

多 Element 控制

每个 Element 只有一个 onoff model，所以电工品类中多路开关需要多个 element。在多个 element 时就需要使用目的地址来区分是发送给哪个 Element 的消息。

```

STATIC UINT8_T onoff_data[6] = {0};

OPERATE_RET app_mesh_data_recv(TAL_MESH_ACCESS_MSG_T *msg_raw, TAL_MESH_NET_PARAM_T
*net_param){
    TAL_MESH_GENERIC_ONOFF_STATUS_T    generic_onoff_status;

    UINT8_T switch_index = net_param->dst_addr - tal_primary_ele_addr_get();
    switch(msg_raw->opcode){
        case TAL_MESH_OPCODE_ON_OFF_GET:
            generic_onoff_status.present = onoff_data[switch_index];
            tal_mesh_data_send(net_param->dst_addr, net_param->src_addr,
TAL_MESH_OPCODE_ON_OFF_STAT, (UINT8_T*)&generic_onoff_status, 1);
            break;
        case TAL_MESH_OPCODE_ON_OFF_SET:
        case TAL_MESH_OPCODE_ON_OFF_SET_UNACK:{
            TAL_MESH_GENERIC_ONOFF_SET_T *generic_onoff_set =
(TAL_MESH_GENERIC_ONOFF_SET_T*)msg_raw->data;
            onoff_data[switch_index] = generic_onoff_set->onoff;
            if(TAL_MESH_OPCODE_ON_OFF_SET == msg_raw->opcode){
                generic_onoff_status.present = generic_onoff_set->onoff;
                tal_mesh_data_send(net_param->dst_addr, net_param->src_addr,
TAL_MESH_OPCODE_ON_OFF_STAT, (UINT8_T*)&generic_onoff_status, 1);
            }
        }
        break;

        default:
            break;
    }

    return OPRT_OK;
}

```

以上示例为一个六路的开关被控制时的数据接收，switch_index 为开关的 index，也是 element 的 index。