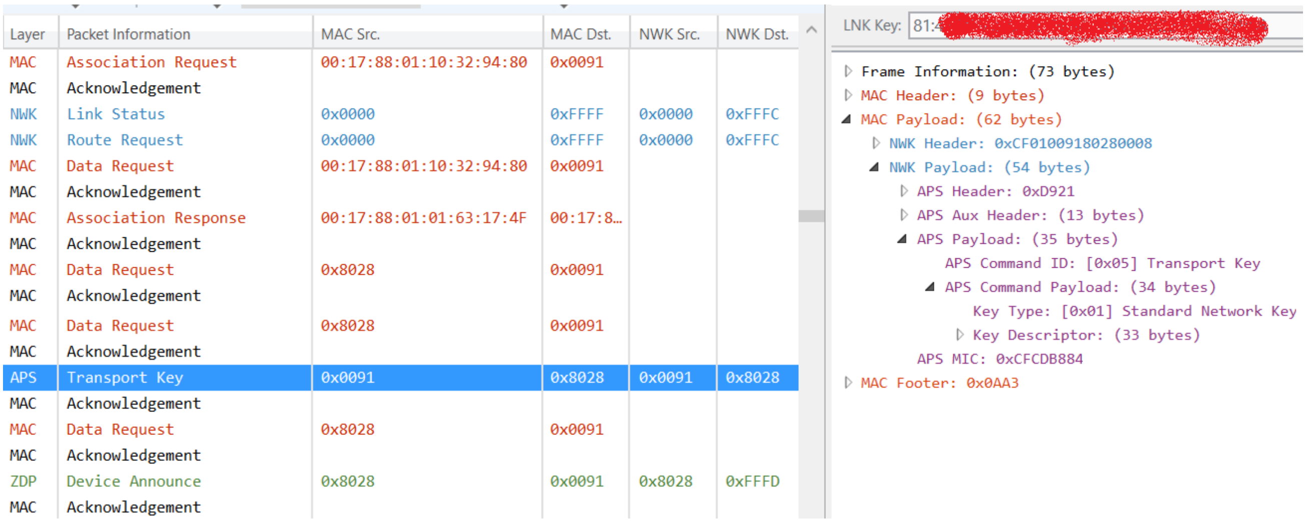
# ZigBee3.0 Bulb 如何加入 Philip HUE 网关

(shaozhong.liang@gmail.com)

Philips Hue 生态系统将全面支持 ZigBee 3.0 协议。在技术上，ZigBee 3.0 向前兼容 ZigBee Light Link。因此，消费者不会感觉到 ZigBee 3.0 和 ZigBee Light Link 设备之间有什么行为差异。

1. HUE 网关使用的 Link Key 是认证 Master Key{81:XX:XX:....}, 需要替换 ZigBee3.0 默认的 Trust Center Link Key{5A: 69:67:42:65:65:41:6C:6C:69:61:6E:63:65:30:39}，否则无法成功入网。



| Layer | Packet Information | MAC Src. | MAC Dst. | NWK Src. | NWK Dst. |
|---|---|---|---|---|---|
| MAC | Association Request | 00:17:88:01:10:32:94:80 | 0x0091 | | |
| MAC | Acknowledgement | | | | |
| NWK | Link Status | 0x0000 | 0xFFFF | 0x0000 | 0xFFFC |
| NWK | Route Request | 0x0000 | 0xFFFF | 0x0000 | 0xFFFC |
| MAC | Data Request | 00:17:88:01:10:32:94:80 | 0x0091 | | |
| MAC | Acknowledgement | | | | |
| MAC | Association Response | 00:17:88:01:01:63:17:4F | 00:17:8... | | |
| MAC | Acknowledgement | | | | |
| MAC | Data Request | 0x8028 | 0x0091 | | |
| MAC | Acknowledgement | | | | |
| MAC | Data Request | 0x8028 | 0x0091 | | |
| MAC | Acknowledgement | | | | |
| APS | Transport Key | 0x0091 | 0x8028 | 0x0091 | 0x8028 |
| MAC | Acknowledgement | | | | |
| MAC | Data Request | 0x8028 | 0x0091 | | |
| MAC | Acknowledgement | | | | |
| ZDP | Device Announce | 0x8028 | 0x0091 | 0x8028 | 0xFFFD |
| MAC | Acknowledgement | | | | |

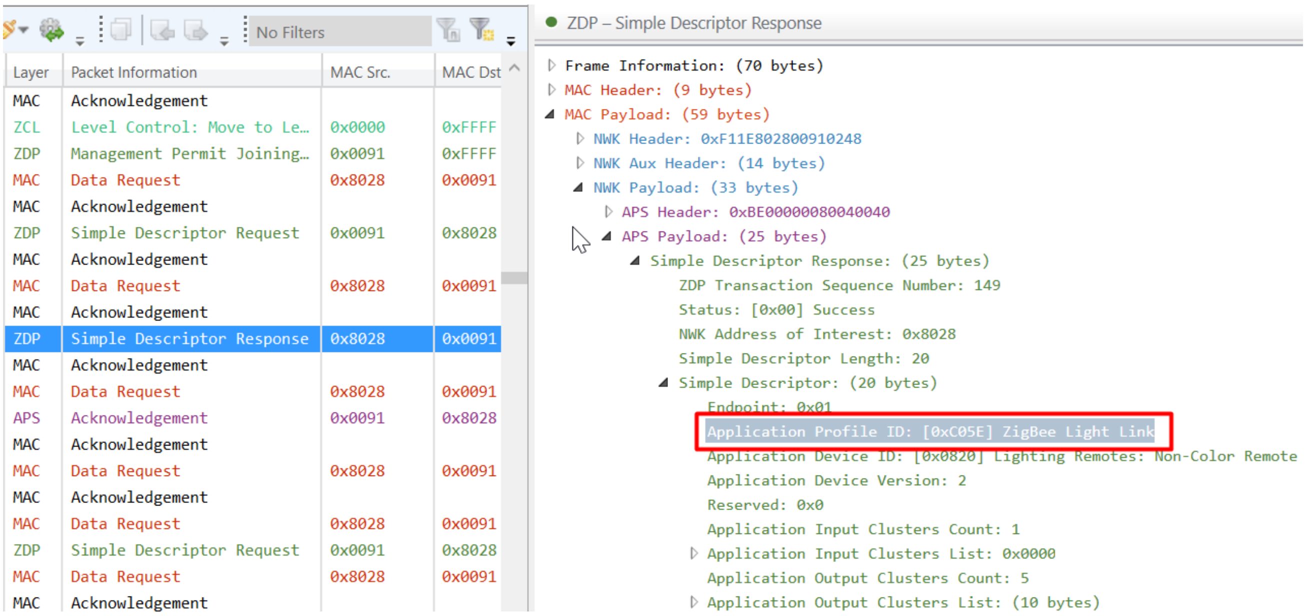
LNK Key: 81:[redacted]

```
▷ Frame Information: (73 bytes)
▷ MAC Header: (9 bytes)
◢ MAC Payload: (62 bytes)
    ▷ NWK Header: 0xCF01009180280008
    ◢ NWK Payload: (54 bytes)
        ▷ APS Header: 0xD921
        ▷ APS Aux Header: (13 bytes)
        ◢ APS Payload: (35 bytes)
              APS Command ID: [0x05] Transport Key
            ◢ APS Command Payload: (34 bytes)
                  Key Type: [0x01] Standard Network Key
                ▷ Key Descriptor: (33 bytes)
          APS MIC: 0xCFCDB884
▷ MAC Footer: 0x0AA3
```

2. 目前, HUE 网关对已支持 ZigBee 3.0 协议，但略做了限制性修改:

HUE 网关要求灯在回复 Simple Descriptor Request 时，Profile ID 必须为 0xC05E(ZLL)。但正常的控制和读取时，需要用 0x0104(HA)作为 Profile ID。如果不满足以上的要求，则 HUE 网关不识别这个 ZigBee Bulb 灯。

所以 ZigBee3.0 Bulb 如果需要加入到 HUE/HUE2 网关,必须保持主协议为 0x0104(HA)的情况下, 回复 Simple Descriptor Response 中 Profile ID 为 0xC05E(ZLL)。

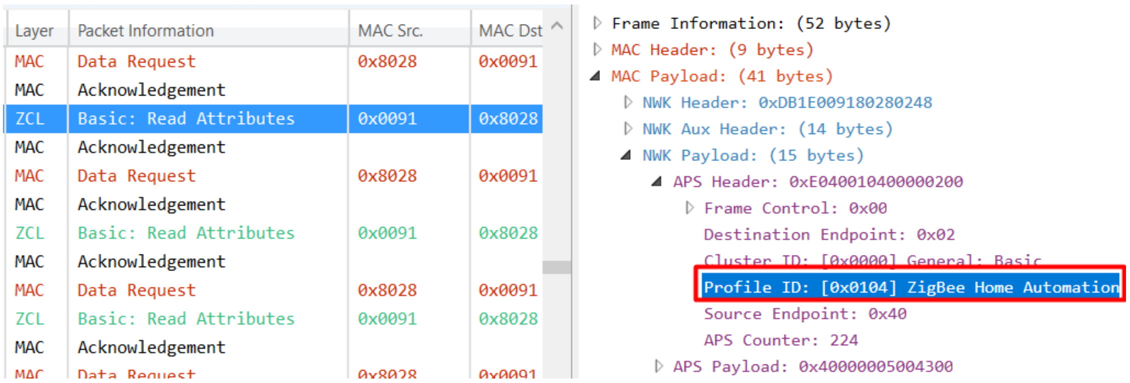


| Layer | Packet Information | MAC Src. | MAC Dst |
|---|---|---|---|
| MAC | Acknowledgement | | |
| ZCL | Level Control: Move to Le... | 0x0000 | 0xFFFF |
| ZDP | Management Permit Joining... | 0x0091 | 0xFFFF |
| MAC | Data Request | 0x8028 | 0x0091 |
| MAC | Acknowledgement | | |
| ZDP | Simple Descriptor Request | 0x0091 | 0x8028 |
| MAC | Acknowledgement | | |
| MAC | Data Request | 0x8028 | 0x0091 |
| MAC | Acknowledgement | | |
| ZDP | Simple Descriptor Response | 0x8028 | 0x0091 |
| MAC | Acknowledgement | | |
| MAC | Data Request | 0x8028 | 0x0091 |
| APS | Acknowledgement | 0x0091 | 0x8028 |
| MAC | Acknowledgement | | |
| MAC | Data Request | 0x8028 | 0x0091 |
| MAC | Acknowledgement | | |
| MAC | Data Request | 0x8028 | 0x0091 |
| ZDP | Simple Descriptor Request | 0x0091 | 0x8028 |
| MAC | Data Request | 0x8028 | 0x0091 |
| MAC | Acknowledgement | | |

● ZDP – Simple Descriptor Response

```
▷ Frame Information: (70 bytes)
▷ MAC Header: (9 bytes)
◢ MAC Payload: (59 bytes)
    ▷ NWK Header: 0xF11E802800910248
    ▷ NWK Aux Header: (14 bytes)
    ◢ NWK Payload: (33 bytes)
        ▷ APS Header: 0xBE00000080040040
        ◢ APS Payload: (25 bytes)
            ◢ Simple Descriptor Response: (25 bytes)
                  ZDP Transaction Sequence Number: 149
                  Status: [0x00] Success
                  NWK Address of Interest: 0x8028
                  Simple Descriptor Length: 20
                ◢ Simple Descriptor: (20 bytes)
                      Endpoint: 0x01
                      Application Profile ID: [0xC05E] ZigBee Light Link
                      Application Device ID: [0x0820] Lighting Remotes: Non-Color Remote
                      Application Device Version: 2
                      Reserved: 0x0
                      Application Input Clusters Count: 1
                    ▷ Application Input Clusters List: 0x0000
                      Application Output Clusters Count: 5
                    ▷ Application Output Clusters List: (10 bytes)
```

正常操作时，ProfileID 仍为 0x0104(HA)。



| Layer | Packet Information | MAC Src. | MAC Dst |
|---|---|---|---|
| MAC | Data Request | 0x8028 | 0x0091 |
| MAC | Acknowledgement | | |
| ZCL | Basic: Read Attributes | 0x0091 | 0x8028 |
| MAC | Acknowledgement | | |
| MAC | Data Request | 0x8028 | 0x0091 |
| MAC | Acknowledgement | | |
| ZCL | Basic: Read Attributes | 0x0091 | 0x8028 |
| MAC | Acknowledgement | | |
| MAC | Data Request | 0x8028 | 0x0091 |
| ZCL | Basic: Read Attributes | 0x0091 | 0x8028 |
| MAC | Acknowledgement | | |
| MAC | Data Request | 0x8028 | 0x0091 |

```
▷ Frame Information: (52 bytes)
▷ MAC Header: (9 bytes)
◢ MAC Payload: (41 bytes)
    ▷ NWK Header: 0xDB1E009180280248
    ▷ NWK Aux Header: (14 bytes)
    ◢ NWK Payload: (15 bytes)
        ◢ APS Header: 0xE040010400000200
            ▷ Frame Control: 0x00
              Destination Endpoint: 0x02
              Cluster ID: [0x0000] General: Basic
              Profile ID: [0x0104] ZigBee Home Automation
              Source Endpoint: 0x40
              APS Counter: 224
        ▷ APS Payload: 0x40000005004300
```

为了满足上述要求，在 NXP JN5169 工程中修改 Source\zps_gen.c 文件，手工增加下面处理函数。注意 zps_gen.c 文件是由 Beyond Studio IDE 自动生成，每次 app.zpscfg 修改后或者项目工程被 clean 清除后，zps_gen.c 将会重新生成，需要重新添加下面代码。

```
PRIVATE bool zps_bAplZdoSimpleDescServer_HUE2(void *, void *, ZPS_tsAfEvent *);

/* ZDO Servers */
PRIVATE const zps_tsAplZdoServer s_asAplZdoServers[21] = {
    { zps_bAplZdoZdoClient, s_sZdoClientContext },
    { zps_bAplZdoDeviceAnnceServer, s_sDeviceAnnceServerContext },
    { zps_bAplZdoActiveEpServer, s_sActiveEpServerContext },
    { zps_bAplZdoNwkAddrServer, s_sNwkAddrServerContext },
    { zps_bAplZdoIeeeAddrServer, s_sIeeeAddrServerContext },
    { zps_bAplZdoSystemServerDiscoveryServer, s_sSystemServerDiscoveryServerContext },
    { zps_bAplZdoNodeDescServer, s_sNodeDescServerContext },
    { zps_bAplZdoPowerDescServer, s_sPowerDescServerContext },
    { zps_bAplZdoMatchDescServer, s_sMatchDescServerContext },
    { zps_bAplZdoSimpleDescServer_HUE2, s_sSimpleDescServerContext },
    { zps_bAplZdoMgmtLqiServer, s_sMgmtLqiServerContext },
    ……………………………
}

PRIVATE bool zps_bAplZdoSimpleDescServer_BUE2(void *a, void *b, ZPS_tsAfEvent *c)
{
    bool bRetVal = FALSE;
    ZPS_tsAplAfSimpleDescriptor *psSimpleDesc = NULL;

    zps_tsApl *psApl = (zps_tsApl *)a;
    ZPS_tsAfEvent* pszdoServerEvent = (ZPS_tsAfEvent*)c;

   if(ZPS_ZDP_SIMPLE_DESC_REQ_CLUSTER_ID==
       psZdoServerEvent->uEvent.sApsDataIndEvent.u16ClusterId)
    {
        ZPS_tsAplZdpSimpleDescReq sAplZdpSimpleDescReq;
        zps_tsAplAfSimpleDescCont *psSDC = NULL;
        uint32 nIndex;

        /* get the endpoint ID */
        PDUM_u16APduInstanceReadNBO(psZdoServerEvent->uEvent.sApsDataIndEvent.hAPduInst, 1, 
"hb", &sAplZdpSimpleDescReq);

        /* look at the list of simple descriptors to see if any matches could be found */
        for (nIndex = 0; nIndex < psApl->sAfContext.u32NumSimpleDescriptors ; nIndex++)
        {
           psSDC = &psApl->sAfContext.psSimpleDescConts[nIndex];

           if (psSDC->sSimpleDesc.u8Endpoint == sAplZdpSimpleDescReq.u8EndPoint)
           {
               if (psSDC->bEnabled)
               {
                   psSimpleDesc = &psSDC->sSimpleDesc;
               }
               break;
           }
        }
        if(psSimpleDesc != NULL)
        {
           /* Add your code here!!!!!!! */
           psSimpleDesc->u16ApplicationProfileId = 0xC05E; //Change it for HUE2
        }
    }

    /* call original function */
    bRetVal = zps_bAplZdoSimpleDescServer(a, b, c);

    /* Restore the profile id from 0xC05E to 0x0104, for Zigbee3.0 specification */
    if(psSimpleDesc != NULL)
    {
        /* Add your code here!!!!!!! */
        psSimpleDesc->u16ApplicationProfileId = 0x0104; //Change it for standard Zigbee3.0
    }

    return bRetVal;
}
```