Enhanced Device Protocol（EDP）

欢迎访问设备云门户网站<http://open.iot.10086.cn/>注册用户，获取最新文档。

|  |  |  |  |
| --- | --- | --- | --- |
| **版本号** | **修订日期** | **修订内容** | **说明** |
| V1.1 | 2014.10.30 | EDP增加存储数据报文；  REST API 增加历史数据查询接口，用于上报数据点，或者上报的同时转发数据点。 |  |
| V1.1.1 | 2014/11/3 | EDP增加消息类型9，以支持对存储数据的确认。 |  |
| V1.2 | 2015/07/17 | 增加加密机制；  命令请求和响应报文；  存储数据点，新增3种格式；  修改登陆方式为2种； |  |
| V1.3 | 2015/10/14 | 添加连接关闭消息 |  |
| V1.4 | 2015/03/21 | 添加消息编号指示 |  |
| V1.5 | 2016/04/07 | 增加数据类型7 |  |

目录

[Enhanced Device Protocol（EDP） 1](#_Toc425932409)

[1 说明 2](#_Toc425932410)

[2 设备与业务接入模式 2](#_Toc425932411)

[3 接入流程 2](#_Toc425932412)

[4 消息格式 3](#_Toc425932413)

[4.1 消息类型 3](#_Toc425932414)

[4.2 剩余消息长度 3](#_Toc425932415)

[4.3 选项 4](#_Toc425932416)

[4.4 消息体 5](#_Toc425932417)

[5 消息类型 5](#_Toc425932418)

[5.1 连接请求 5](#_Toc425932419)

[5.2 连接响应 8](#_Toc425932420)

[5.3 转发(透传)数据 9](#_Toc425932421)

[5.4 存储(&转发)数据 10](#_Toc425932422)

[5.5 存储确认 13](#_Toc425932423)

[5.6 命令请求 14](#_Toc425932424)

[5.7 命令响应 14](#_Toc425932425)

[5.8 心跳请求 15](#_Toc425932426)

[5.9 心跳响应 15](#_Toc425932427)

[5.10 加密请求 16](#_Toc425932428)

[5.11 加密响应 16](#_Toc425932429)

[6 主要流程 17](#_Toc425932430)

[6.1 登录 17](#_Toc425932431)

[6.2 数据收发（透传） 18](#_Toc425932432)

[6.3 存储数据点（datapoint） 18](#_Toc425932433)

[6.4 存储数据点并获得确认 19](#_Toc425932434)

[6.5 存储数据点并转发 20](#_Toc425932435)

[6.6 命令请求及响应 20](#_Toc425932436)

[6.7 心跳保持 21](#_Toc425932437)

[6.8 数据加密 21](#_Toc425932438)

# 说明

该接口上的协议基于TCP，但只传输数据包到目的地，不保证传输的顺序与到达的顺序相同，事务机制需要在上层实现；若客户端同时发起两次请求，服务器返回时，不保障返回报文的顺序。

# 设备与业务接入模式

**设备层：**利用平台提供的EDPSDK，实现EDP协议，用于上报业务数据点到OnetNet。若需要实时接收业务层下发的控制命令，需要保持EDP长连接。

**业务应用层：**若要自定义实现业务平台，可通过HTTP 协议的RESTful API操作OnetNet提供的资源（设备、数据点、命令控制等资源的增删查改）。



适用场景：在充分分析业务数据模型的基础上，认为OneNet提供的设备-数据流-数据点模型适合业务数据存储。优先推荐新业务使用该模式。

# 接入流程

* 1. 访问平台<http://open.iot.10086.cn/>注册用户；
  2. 用户根据业务情况，在“连接请求”章节中选择EDP登录方式；
  3. 根据登录方式，填写设备相关属性，在产品下新增设备，获取产品ID、设备ID，以及api-key等信息；
  4. 设备发送TCP连接请求到以下地址，发送封装的报文与平台交互。

**平台服务器地址jjfaedp.hedevice.com，TCP 端口876,29876**

# 消息格式

消息包括三个部分：必选的消息头（绿色），可选的多个选项（黄色）以及可选的消息体（蓝色）。

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **字节\bit** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| Byte 1 | 消息类型 | | | | 保留位（全零） | | | |
| Multi-bytes | 剩余消息长度（1-4字节，指示选项+消息体的长度） | | | | | | | |
| Multi-bytes | 选项（根据消息类型0个或多个） | | | | | | | |
| Multi-bytes | 消息体（根据消息类型0或多个字节） | | | | | | | |

## 消息类型

占第一个字节的前4位，取值范围（0-15），定义如下：

|  |  |  |
| --- | --- | --- |
| **类型值** | **含义** | **方向** |
| 1 | CONN\_REQ：连接建立请求 | C(client)->S(server) |
| 2 | CONN\_RESP：连接建立响应 | S->C |
| 3 | PUSH\_DATA：转发（透传）数据 | 双向 |
| 4 | CONN\_CLOSE：连接关闭 | S->C |
| 8 | SAVE\_DATA：存储（&转发）数据 | 双向 |
| 9 | SAVE\_ACK：存储确认 | S->C |
| 10 | CMD\_REQ | S->C |
| 11 | CMD\_RESP | C->S |
| 12 | PING\_REQ: 心跳请求 | C->S |
| 13 | PING\_RESP: 心跳响应 | S->C |
| 14 | ENCRYPT\_REQ | C->S |
| 15 | ENCRYPT\_RESP | S->C |
| 其他值 | 保留 |  |

## 剩余消息长度

用于指示选项和消息体的字节数。目前平台限制EDP协议每条消息剩余长度不能超过**4M**。

该字段占用1-4个字节，**长度值的低位部分放在传输的前面字节，高位放在后面**。每个字节的最高位为延续指示位。延续指示位为1时，标示后面字节也是长度值，最多可延续4个字节。可表示数据范围如下：

|  |  |  |
| --- | --- | --- |
| **字节数** | **最小值** | **最大值** |
| 1 | 0(0x00) | 127(0x7F) |
| 2 | 128(0x80, 0x01) | 16383 (0xFF, 0x7F) |
| 3 | 16 384 (0x80, 0x80, 0x01) | 2 097 151 (0xFF, 0xFF, 0x7F) |
| 4 | 2 097 152 (0x80, 0x80, 0x80, 0x01) | 268 435 455 (0xFF, 0xFF, 0xFF, 0x7F) |

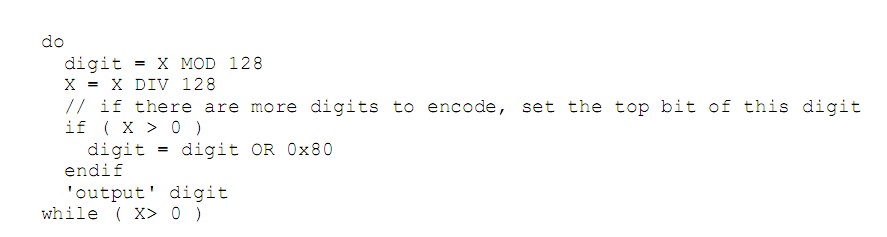
注意：消息剩余长度 = 选项所占字节数 + 消息体所占字节数，根据该值的大小来确定消息剩余长度字段在EDP数据包中占用多少个字节，比如从上表可以看出，当0<剩余长度<=127的时候，消息剩余长度字段在EDP数据包中只占一个字节，而不是四个字节；当 127<剩余长度<=16383的时候，消息剩余长度字段在EDP数据包中占两个字节，而不是四个字节；依此类推。

例如，若选项+消息体长度共321=65+2\*128字节，则该域（剩余长度）需要两个字节才能表示，第一个字节的延续位置1，按照传输顺序格式如下：

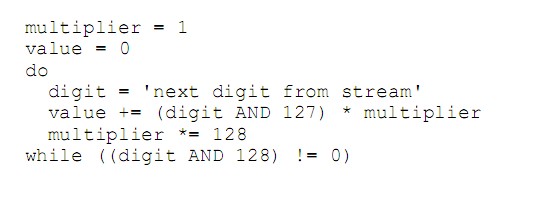
|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **字节\bit** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| Byte 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| Byte 2 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |

解析该长度值的C语言算法可表示为：( ( Byte2 & 127 ) << 7 ) | ( Byte1& 127 )

编码算法:



解码算法:



## 选项

根据消息类型，选项的格式不同；详见后面的命令类型说明。某些选项为固定格式的几个字节，另一些采用length+value的字符串格式，用两个字节指示后面值的长度，字符串最长0xFFFF。

字符串详细格式如下：

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **字节\bit** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| Byte 1 | 长度高位字节 | | | | | | | |
| Byte 2 | 长度低位字节 | | | | | | | |
| 0-mulit bytes | 0或多个字节的内容，最长0xFFFF字节 | | | | | | | |

## 消息体

根据消息类型，消息体可选，详见后面命令类型说明。

# 消息类型

## 连接请求

连接请求包含三部分：消息头，一些选项和消息体。其中，选项包括协议名称、协议版本、连接标志、保持连接时间。

消息体中可能包含设备ID、产品ID（可选）、鉴权信息。三项内容都为长度+内容的字符串格式。设备ID必须传递，若认证方式中不使用设备ID，应将设备ID长度设置为0；产品ID和鉴权信息，根据标志位若存在必须按顺序出现。

|  |  |
| --- | --- |
| **产品ID获取方式** | 在OneNet添加产品时，平台生成产品ID。 |
| **设备ID获取方式** | 在OneNet平台创建设备时平台生成的设备ID号。 |
| **api-key获取方式** | 在OneNet注册的产品，管理整个产品的api-key,或者用于自己新增具有该设备操作权限的api-key |

可选的登录认证方式：

|  |  |  |  |
| --- | --- | --- | --- |
| **登陆认证方式** | **携带信息** | **说明** | **消息示例** |
| 1 | 设备ID + 鉴权信息（api-key） | 设备ID：申请设备时平台返回的ID；  鉴权信息：在平台申请的可以操作该设备的api-key字符串； | 消息格式见示例1 |
| 2 | 产品ID + 鉴权信息（auth\_info） | 产品ID：在平台添加产品时平台生成的ID；  鉴权信息：在平台申请设备时填写设备的auth\_info属性（json对象字符串），该属性需要产品内具备唯一性； | 消息格式见示例2 |

**示例1：**采用登陆方式1：设备ID“43101”和api-key“abcdefg”，登录平台的连接请求消息格式如下：

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **字节** | **说明\bit** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| 消息头 | | | | | | | | | |
| Byte 1 | 第一字节：  Bit（4-7）：消息类型，值为1；  Bit（0-3）：保留位，值为0； | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 变长剩余消息长度(25编码后需要占用1个字节) | | | | | | | | | |
| Byte 2 | 第二字节：  消息剩余字节长度，值为25 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 |
| 选项1：协议描述（字符串格式） | | | | | | | | | |
| Byte 3 | 长度高位字节，值为0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Byte 4 | 长度低位字节，值为3 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| Byte 5 | 字母’E’ | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 |
| Byte 6 | 字母’D’ | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| Byte 7 | 字母’P’ | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| 选项2：协议版本 | | | | | | | | | |
| Byte 8 | 一个字节表示，值为1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 选项3：连接标志 | | | | | | | | | |
| Byte 9 | Bit（7）：产品ID标志位，值0，  Bit（6）：鉴权信息标志位，值1，表示后面消息体有该项  Bit（0-5）：系统保留位，填0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 选项4：保持连接时间（256秒=0x0100） | | | | | | | | | |
| Byte 10 | 第一字节，时间值的高位字节，值1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| Byte 11 | 第二字节，时间值的低位字节，值0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 消息体-设备ID（字符串格式） | | | | | | | | | |
| Byte 12 | 长度高位字节，值为0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Byte 13 | 长度低位字节，值为5 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| Byte 14 | 字符’4’ | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 |
| Byte 15 | 字符’3’ | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| Byte 16 | 字符’1’ | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 |
| Byte 17 | 字符’0’ | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| Byte 18 | 字符’1’ | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 |
| 消息体-鉴权信息（字符串格式） | | | | | | | | | |
| Byte 19 | 长度高位字节，值为0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Byte 20 | 长度低位字节，值为7 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| Byte 21 | 字符’a’ | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 |
| Byte 22 | 字符’b’ | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 |
| Byte 23 | 字符’c’ | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 |
| Byte 24 | 字符’d’ | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 |
| Byte 25 | 字符’e’ | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 |
| Byte 26 | 字符’f’ | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 |
| Byte 27 | 字符’g’ | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 |

**示例2：**采用登陆方式2：产品ID“101”和鉴权信息“{“mac”:“FF”}”，登录平台的连接请求消息格式如下：

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **字节** | **说明\bit** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| 消息头 | | | | | | | | | |
| Byte 1 | 第一字节：  Bit（4-7）：消息类型，值为1；  Bit（0-3）：保留位，值为0； | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 变长剩余消息长度(25编码后需要占用1个字节) | | | | | | | | | |
| Byte 2 | 第二字节：  消息剩余字节长度，值为30 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 |
| 选项1：协议描述（字符串格式） | | | | | | | | | |
| Byte 3 | 长度高位字节，值为0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Byte 4 | 长度低位字节，值为3 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| Byte 5 | 字母’E’ | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 |
| Byte 6 | 字母’D’ | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| Byte 7 | 字母’P’ | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| 选项2：协议版本 | | | | | | | | | |
| Byte 8 | 一个字节表示，值为1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 选项3：连接标志 | | | | | | | | | |
| Byte 9 | Bit（7）：产品ID标志位，值1，表示后面消息体有该项  Bit（6）：鉴权信息标志位，值1，表示后面消息体有该项  Bit（0-5）：系统保留位，填0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 选项4：保持连接时间（256秒=0x0100） | | | | | | | | | |
| Byte 10 | 第一字节，时间值的高位字节，值1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| Byte 11 | 第二字节，时间值的低位字节，值0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 消息体-设备ID（字符串格式） | | | | | | | | | |
| Byte 12 | 长度高位字节，值为0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Byte 13 | 长度低位字节，值为0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 消息体-产品ID（字符串格式） | | | | | | | | | |
| Byte 14 | 长度高位字节，值为0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Byte 15 | 长度低位字节，值为3 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| Byte 16 | 字符’1’ | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 |
| Byte 17 | 字符’0’ | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| Byte 18 | 字符’1’ | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 |
| 消息体-鉴权信息（字符串格式） | | | | | | | | | |
| Byte 19 | 长度高位字节，值为0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Byte 20 | 长度低位字节，值为9 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| Byte 21 | 字符’{’ | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 |
| Byte 22 | 字符’”’ | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| Byte 23 | 字符’m’ | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 |
| Byte 24 | 字符’a’ | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 |
| Byte 25 | 字符’c’ | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 |
| Byte 26 | 字符’”’ | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| Byte 27 | 字符’:’ | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 |
| Byte 28 | 字符’”’ | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| Byte 29 | 字符’F’ | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 |
| Byte 30 | 字符’F’ | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 |
| Byte 31 | 字符’”’ | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| Byte 32 | 字符’}’ | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 |

## 连接响应

连接响应报文包含：消息头，2个必选项和可能的消息体。其中，选项包括一个字节的响应标志和一个字节的返回码选项；消息体根据授权码标志位，可能携带平台返回的授权码。

举例格式如下：

1.连接成功无授权码

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **字节** | **说明\bit** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| 消息头 | | | | | | | | | |
| Byte 1 | 第一字节：  Bit（4-7）：消息类型，**值为2**；  Bit（0-3）：保留位，值为0； | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| Byte 2 | 第二字节：  消息剩余字节长度，值为2 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 选项1：选项-标志 | | | | | | | | | |
| Byte 3 | Bit（1-4）：系统保留位  Bit（0）：授权码标志位，值1，表示后面消息体携带有授权码 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 选项2：选项-返回码 | | | | | | | | | |
| Byte 4 | 一个字节表示，根据验证情况，枚举值如下：  0：连接成功；  1：验证失败-协议错误；  2：验证失败-设备ID鉴权失败；  3：验证失败-服务器失败；  4：验证失败-产品ID鉴权失败；  5：验证失败-未授权；  6：验证失败-授权码无效；  7：激活失败-激活码未分配；  8：激活失败-该设备已被激活；  9：验证失败-重复发送连接请求包；  10-255：保留值； | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

2.连接成功有授权码，授权码为abcd

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **字节** | **说明\bit** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| 消息头 | | | | | | | | | |
| Byte 1 | 第一字节：  Bit（4-7）：消息类型，**值为2**；  Bit（0-3）：保留位，值为0； | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| Byte 2 | 第二字节：  消息剩余字节长度，值为2 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 选项1：选项-标志 | | | | | | | | | |
| Byte 3 | Bit（1-4）：系统保留位  Bit（0）：授权码标志位，值1，表示后面消息体携带有授权码 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 选项2：选项-返回码 | | | | | | | | | |
| Byte 4 | 一个字节表示，根据验证情况，枚举值如下：  0：连接成功；  1：验证失败-协议错误；  2：验证失败-设备ID鉴权失败；  3：验证失败-服务器失败；  4：验证失败-产品ID鉴权失败；  5：验证失败-未授权；  6：验证失败-授权码无效；  7：激活失败-激活码未分配；  8：激活失败-该设备已被激活；  9：验证失败-重复发送连接请求包；  10-255：保留值； | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 消息体-鉴权信息（字符串格式） | | | | | | | | | |
| Byte 5 | 长度高位字节，值为0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Byte 6 | 长度低位字节，值为4 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| Byte 7 | 字符’a’ | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 |
| Byte 8 | 字符’b’ | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 |
| Byte 9 | 字符’c’ | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 |
| Byte 10 | 字符’d’ | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 |

## 转发(透传)数据

该消息是一个双向消息，可以从设备到云，也可以由设备云发向设备。由消息头、一个选项和消息体组成，其中选项必须包括一个地址。

|  |  |
| --- | --- |
| **Push\_data消息方向** | **选项（地址）说明** |
| C->S(设备到平台) | 平台收到该消息，选项中的地址是该数据转发的目的地址（目的设备ID号）；  若目的地址的长度为零，即没有目的地址，则平台将该消息转发到设备注册时的默认目的地址（设备申请时，填写的route\_to字段）。 |
| S->C (平台到设备) | 设备收到该消息，选项中的地址是该数据发送的发送者（源）地址； |

**示例：**通过设备云发送数据到ID为21573的设备。

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **字节** | **说明\bit** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| 消息头 | | | | | | | | | |
| Byte 1 | 第一字节：  Bit（4-7）：消息类型，**值为3**；  Bit（0-3）：保留位，值为0； | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| 剩余消息长度(283编码后需要占用2个字节) | | | | | | | | | |
| Byte 2 | 消息剩余字节长度(283)-编码第一字节（低） | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 |
| Byte 3 | 消息剩余字节长度(283)-编码第二字节（高） | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 选项1：目的或源地址（字符串格式） | | | | | | | | | |
| Byte 4 | 固定两字节长度高位字节，值为0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Byte 5 | 固定两字节长度低位字节，值为5 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| Byte 6 | 字符’2’ | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 |
| Byte 7 | 字母’1’ | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 |
| Byte 8 | 字母’5’ | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| Byte 9 | 字母’7’ | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 |
| Byte 10 | 字母’3’ | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| 消息体（用户数据） | | | | | | | | | |
| Byte 11 | //最大支持3M  用户自定义的数据，本例中为276字节 |  |  |  |  |  |  |  |  |
| …  …  …  …  …  …  …  …  …  … |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
| Byte 286 |  |  |  |  |  |  |  |  |

## 连接关闭

当平台解析协议时，如果发现协议格式错误或者未按要求封装，则会向设备发送连接关闭消息，然后断开连接。此消息的消息体为一个字节，表示错误原因。

需要注意的是，错误码并一定能完全正确地反应真正出错的字段。比如有相邻的两个字段A和B，A要求长度为4字节，但是实际上封装了5个字节在里面，这样第五个字节就会被当作B的内容被读取解析，平台解析到B字段时，有可能认为字段B封装出错，但是实际上是因为A字段封装错误引起的。

消息格式如下：

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **字节** | **说明\bit** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| 消息头 | | | | | | | | | |
| Byte 1 | 第一字节：  Bit（4-7）：消息类型，值为0x40；  Bit（0-3）：保留位，值为0； | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| Byte 2 | 第二字节：  消息剩余字节长度，值为1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| Byte 3 | 第三字节：  连接关闭错误码。 |  |  |  |  |  |  |  |  |

错误码及含义如下：

|  |  |  |
| --- | --- | --- |
| **错误码** | **出错消息** | **含义** |
| 全局错误码，不与某条消息关联 | | |
| 1 |  | 未知错误 |
| 2 |  | 服务不可用，当服务器端因分配内存失败等内部原因导致需要断开连接时，会返回此错误。 |
| 3 |  | 解析消息类型时出错。 |
| 4 |  | 消息类型不属于4.1节消息类型中定义的任何一种消息。 |
| 5 |  | 解析剩余消息长度时出错，或者剩余消息长度不合法。 |
| 6 |  | 连接超时。 |
| 加解密 | | |
| 10 | ENCRYPT\_REQ | RSA加密算法的E、N等长度不合法。 |
| 11 | 在CONN\_REQ消息之后收到ENCRYPT\_REQ消息。 |
| 12 | 解析对称加密算法类型时出错。 |
| 13 | 指定的对称加密算法暂时不支持。 |
| 14 | 利用收到E、N初始化RSA密钥失败。 |
| 15 |  | 解密收到的数据时出错。 |
| 连接请求 | | |
| 20 | CONN\_REQ | 解析协议名称出错，或者不支持给定的协议名称。 |
| 21 | 解析协议版本出错，或者不支持给定的协议版本。 |
| 22 | 解析连接标志时出错。 |
| 23 | 连接标志中未将鉴权信息标志位（第6位）置上。 |
| 24 | 解析保活时间时出错。 |
| 25 | 解析设备ID时出错，或者设备ID长度大于限定值。 |
| 26 | 指定了产品ID标志位，但是解析产品ID出错，或者产品ID长度大于限定值，或者产品ID为0。 |
| 27 | 解析鉴权信息时出错或者鉴权信息为空。 |
| 28 | 从不同的TCP连接重复登录。 |
| 29 | 利用激活码激活失败。 |
| 30 | 鉴权码不是有效的json格式。 |
| 31 | 利用鉴权码鉴权失败。 |
| 32 | 利用devid+apikey的方式鉴权失败。 |
| 33 | 利用devid+apikey的方式鉴权，指定了产品ID，但是查询到的产品ID与指定的不一致。 |
| 34 | 利用产品ID+鉴权信息鉴权，但是鉴权信息不是有效的Json格式。 |
| 35 | 利用产品ID+鉴权信息鉴权失败。 |
| 36 | 利用产品ID+鉴权信息鉴权，未查询到相应的设备。 |
| 37 | 连接标志不合法。 |
| 38 |  | 用不同的鉴权信息重复连接。 |
| 转发（透传） | | |
| 50 | PUSH\_DATA | 解析目的设备ID时出错。 |
| 51 | 在发送连接请求以前发送了PUSH\_DATA消息。 |
| 存储（转发） | | |
| 60 | SAVE\_DATA | 在发送连接请求以前发送了SAVE\_DATA消息。 |
| 61 | 解析存储转发标志消息标志位时出错。 |
| 62 | 标志位中指定了目的设备ID，但是解析目的设备ID时出错。 |
| 63 | 解析存储转发数据封装类型时出错。 |
| 64 | 数据类型为1，解析其消息体时出错。 |
| 65 | 数据类型为1，其消息体不是有效的Json格式。 |
| 66 | 数据类型为1，缺少‘datastreams’字段，或者此字段不是数组。 |
| 67 | 数据类型为1，缺少‘id’字段，或者此字段不是字符串类型，或者此字段内容为空。 |
| 68 | 数据类型为1，缺少‘datapoints’字段，或者此字段不是数组类型。 |
| 69 | 数据类型为1，缺少‘value’字段。 |
| 70 | 数据类型为2，解析二进制数据描述项时出错。 |
| 71 | 数据类型为2，解析二进制数据长度时出错，或者数据长度不合法。 |
| 72 | 数据类型为2，二进制长度格式错误或者长度不合法。 |
| 73 | 数据类型为2，二进制描述项中不包含‘ds\_id’字段，或者其不是字符串类型，或者其值为空。 |
| 74 | 数据类型为3或者4，解析消息体时出错。 |
| 75 | 数据类型为3或者4，消息体不是有效的Json格式。 |
| 76 | 数据类型为3，数据流名称为空。 |
| 77 | 数据类型为4，消息体不是对象类型。 |
| 78 | 数据类型为4，数据流名称为空。 |
| 79 | 数据类型为5，解析消息体时出错。 |
| 80 | 数据类型为5，消息体格式有误。 |
| 81 | 不支持的数据类型。 |
| 82 | 解析消息标志出错或者消息标志不合法。 |
| 83 | 数据类型为6，解析默认时间出错 |
| 84 | 数据类型为6，解析数据流出错 |
| 85 | 数据类型为7，解析默认时间出错 |
| 86 | 数据类型为7，解析数据流出错 |
| 命令响应 | | |
| 100 | CMD\_RESP | 在发送连接请求以前发送了CMD\_RESP消息。 |
| 101 | 解析命令响应的cmdid时出错。 |
| 102 | 命令响应消息体长度不合法。 |
| 心跳请求 | | |
| 110 | PING\_REQ | 在发送连接请求以前发送了PING\_REQ消息。 |

## 存储(&转发)数据

该消息是一个双向消息，可以从设备到云，也可以由设备云发向设备。由消息头、1到2个选项和消息体组成，其中选项标志位为必填，其他根据标志位选填。

固定选项标志位第7位置1，表示后面携带有地址信息，否则置0。

固定选项标志位第6位置1，表示后面携带有消息编号。消息编号是一个非零值。若携带有消息编号，服务器收到此消息后，会响应一个SAVE\_ACK消息。

固定选项标志位其它位系统保留。

若同时发送多个数据包，平台最多一次处理100个。

|  |  |
| --- | --- |
| **save\_data消息方向** | **选项（地址）说明** |
| C->S(设备到平台) | 平台收到该消息，选项中的地址是该数据转发的目的地址（目的设备ID号）；  若目的地址的长度为零，即没有目的地址，则平台将该消息转发到设备注册时的默认目的地址（设备申请时，填写的route\_to字段）。 |
| S->C (平台到设备) | 设备收到该消息，选项中的地址是该数据发送的发送者（源）地址； |

**例A** :上报数据点报文，并转发数据到ID 为10011的设备。

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **字节** | **说明\bit** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| 消息头 | | | | | | | | | |
| Byte 1 | 第一字节：  Bit（4-7）：消息类型，**值为8**；  Bit（0-3）：保留位，值为0； | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 剩余消息长度(16664编码后需要占用3个字节) | | | | | | | | | |
| Byte 2 | 消息剩余字节长度(16664)-编码第一字节（低） | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| Byte 3 | 消息剩余字节长度(16664)-编码第二字节 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| Byte 4 | 消息剩余字节长度(16664)-编码第三字节（高） | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 固定选项：标志 | | | | | | | | | |
| Byte 5 | Bit 7：转发地址指示位，置1，后面有地址信息  Bit 6：消息编号指示位，置1，后面有2字节消息编号  Bit 5-0：系统保留，全零。 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 目的或源地址（根据上面的标志位确定存在与否） | | | | | | | | | |
| Byte 6 | 固定两字节长度高位字节，值为0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Byte 7 | 固定两字节长度低位字节，值为5 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| Byte 8 | 字符’1’ | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 |
| Byte 9 | 字母’0’ | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| Byte 10 | 字母’0’ | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| Byte 11 | 字母’1’ | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 |
| Byte 12 | 字母’1’ | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 |
| 消息编号（固定2字节，高字节在前，服务器用此编号返回存储确认） 示例：消息编号为261 | | | | | | | | | |
| Byte 13 | 消息编号261高位字节 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| Byte 14 | 消息编号261低位字节 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| 消息体（设备云规定的数据类型格式） | | | | | | | | | |
| Byte 15 | 7种数据结构，具体格式见后续说明：  Type = 7：带统一时间戳的浮点数数据流  Type = 6：带默认时间戳的自定义间隔字符串格式  type = 5：自定义间隔字符串格式；  type = 4：JSON格式3字符串；  type = 3：JSON格式2字符串；  type = 2：二进制数据点；  type = 1：JSON格式1字符串； |  |  |  |  |  |  |  |  |
| …  … |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
| Byte n |  |  |  |  |  |  |  |  |

数据类型7格式说明：

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Byte 15 | 数据类型指示：type=7 //浮点数数据流 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| Byte 16 | 年（后两位），例如2016年，则该字节为16 |  |  |  |  |  |  |  |  |
| Byte 17 | 月（1-12） |  |  |  |  |  |  |  |  |
| Byte 18 | 日（1-31） |  |  |  |  |  |  |  |  |
| Byte 19 | 小时（0~23） |  |  |  |  |  |  |  |  |
| Byte 20 | 分钟（0~59） |  |  |  |  |  |  |  |  |
| Byte 21 | 秒（0~59） |  |  |  |  |  |  |  |  |
| Byte 22 | //数据点个数，每次最多1000个浮点类型数据点  // 假设当前有2个浮点类型数据点  高位字节，值为0x00 |  |  |  |  |  |  |  |  |
| Byte 23 | 低位字节，值为0x02 |  |  |  |  |  |  |  |  |
| Byte 24 | //数据流名称（本类型中名称限定为1-65535的数字，平台会自动转为字符串类型存储。）  高位字节，值为0x00 |  |  |  |  |  |  |  |  |
| Byte 25 | 低位字节，值为0x01 |  |  |  |  |  |  |  |  |
| Byte 26 | 4字节float类型，低位在前，高位在后 |  |  |  |  |  |  |  |  |
| Byte 27 |  |  |  |  |  |  |  |  |
| Byte 28 |  |  |  |  |  |  |  |  |
| Byte 29 |  |  |  |  |  |  |  |  |
| ... |  |  |  |  |  |  |  |  |  |
| Byte n | //数据流名称（本类型中名称限定为1-65535的数字，平台会自动转为字符串类型存储。）  高位字节，值为0x24 |  |  |  |  |  |  |  |  |
| Byte n+1 | 低位字节，值为0x37 |  |  |  |  |  |  |  |  |
| Byte n+2 | 4字节float类型，低位在前，高位在后 |  |  |  |  |  |  |  |  |
| Byte n+3 |  |  |  |  |  |  |  |  |
| Byte n+4 |  |  |  |  |  |  |  |  |
| Byte n+5 |  |  |  |  |  |  |  |  |

数据类型6格式说明：

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Byte 15 | 数据类型指示：type=6 //带时间自定义分隔符 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |
| Byte 16 | 年（后两位），例如2016年，则该字节为16 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| Byte 17 | 月（1-12） |  |  |  |  |  |  |  |  |
| Byte 18 | 日（1-31） |  |  |  |  |  |  |  |  |
| Byte 19 | 小时（0~23） |  |  |  |  |  |  |  |  |
| Byte 20 | 分钟（0~59） |  |  |  |  |  |  |  |  |
| Byte 21 | 秒（0~59） |  |  |  |  |  |  |  |  |
| Byte 22 | //指示后面字符串长度  固定两字节长度高位字节，值为0x00 |  |  |  |  |  |  |  |  |
| Byte 23 | 固定两字节长度低位字节，值为0x41 |  |  |  |  |  |  |  |  |
| Byte 24 | 消息中最前面两字节为用户自定义的域中分隔符和域间分隔符。  具体格式见type=5 说明，若相关域中没有时间戳，则采用本类型的默认时间戳当做数据点的时间来存储。 |  |  |  |  |  |  |  |  |
| …  … |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
| Byte n |  |  |  |  |  |  |  |  |

数据类型5格式说明：

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Byte 15 | 数据点类型指示：type=5 //自定义分隔符 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| Byte 16 | //指示后面字符串长度  固定两字节长度高位字节，值为0x00 |  |  |  |  |  |  |  |  |
| Byte 17 | 固定两字节长度低位字节，值为0x41 |  |  |  |  |  |  |  |  |
| Byte 18 | 消息中最前面两字节为用户自定义的域中分隔符和域间分隔符，这两个分隔符不能相同。比如采用逗号作为域中分隔符，分号作为域间分隔符的格式如下：  ,;feild0;feild1;…;feildn  其中，每个field格式支持3种：  field格式1： 3个子字段，分别是数据流ID,时间戳，数据值。通用格式：  Datastream\_id,datetime,value  field格式2： 2个子字段，分别是数据流ID和数据值，省略时间戳。通用格式：  Datastream\_id,value  field格式3： 1个子字段，省略了数据ID和时间戳，只传输数据值，平台将用该域(feild)所在的位置号（从0开始）作为数据流ID。通用格式：  value  示例：  (1),;temperature,2015-03-22 22:31:12,22.5;102;pm2.5,89;10  (2)#@temperature#2015-03-22 22:31:12#22.5@102@pm2.5#89@10 |  |  |  |  |  |  |  |  |
| …  …  …  …  …  …  …  …  …  … |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
| Byte n |  |  |  |  |  |  |  |  |

数据类型4格式说明：

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Byte 15 | 数据点类型指示：type=4 // JSON格式3字符串 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| Byte 16 | //指示后面字符串长度  固定两字节长度高位字节，值为0x00 |  |  |  |  |  |  |  |  |
| Byte 17 | 固定两字节长度低位字节，值为0x46 |  |  |  |  |  |  |  |  |
| Byte 18 | 通用格式：  {  “datastream\_id1”:{“datetime1”:”value1”},  “datastream\_id2”: {“datetime2”:”value2”},  …  }  示例：  {“temperature”:{“2015-03-22 22:31:12”:22.5}} |  |  |  |  |  |  |  |  |
| …  …  …  … |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
| Byte n |  |  |  |  |  |  |  |  |

数据类型3格式说明：

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Byte 15 | 数据点类型指示：type=3 // JSON格式2字符串 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| Byte 16 | //指示后面字符串长度  固定两字节长度高位字节，值为0x00 |  |  |  |  |  |  |  |  |
| Byte 17 | 固定两字节长度低位字节，值为0x46 |  |  |  |  |  |  |  |  |
| Byte 18 | 通用格式：  {  “datastream\_id1”:”value1”,  “datastream\_id2”:”value2”,  …  }  示例：  {“temperature”:22.5,”humidity”:”95.2%”} |  |  |  |  |  |  |  |  |
| …  …  …  … |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
| Byte n |  |  |  |  |  |  |  |  |

数据类型2格式说明：

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Byte 15 | 数据点类型指示：type=2 //二进制数据 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| Byte 16 | //指示后面json字符串长度  固定两字节长度-高位字节，值为0x00 |  |  |  |  |  |  |  |  |
| Byte 17 | 固定两字节长度-低位字节，值为0x10 |  |  |  |  |  |  |  |  |
| Byte 18 | {  ”ds\_id”:”image”, //创建数据流时定义的ID，（必填）  “at”:”2014-10-25 12:23:23”, //时间，（可选）  ”desc”:字符串或json对象//对该数据的描述（可选）  } |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
| … |  |  |  |  |  |  |  |  |
| Byte n |  |  |  |  |  |  |  |  |
| Byte n+1 | //指示后面二进制数据长度  固定四字节长度-第1字节(最高)，值为0x00 |  |  |  |  |  |  |  |  |
| Byte n+1 | 固定四字节长度-第2字节，值为0x00 |  |  |  |  |  |  |  |  |
| Byte n+2 | 固定四字节长度-第3字节，值为0x01 |  |  |  |  |  |  |  |  |
| Byte n+3 | 固定四字节长度-第4字节(最低)，值为0x00 |  |  |  |  |  |  |  |  |
| Byte n+4 | //该域目前最大支持3M  本例中的该域256字节数据 |  |  |  |  |  |  |  |  |
| … |  |  |  |  |  |  |  |  |
| Byte n+260 |  |  |  |  |  |  |  |  |

数据类型1格式说明：

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Byte 15 | 数据点类型值：1 //1： json格式1字符串 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| Byte 16 | //指示后面json字符串长度  固定两字节长度高位字节，值为0x00 |  |  |  |  |  |  |  |  |
| Byte 17 | 固定两字节长度低位字节，值为0x41 |  |  |  |  |  |  |  |  |
| Byte 18 | {  “datastreams”:[// 可以同时传递多个数据流  {  “id”:”temperature”,  “datapoints”:[  {  ”at”:”2013-04-22 22:22:22”,//可选  ”value”: 36.5//用户自定义  }  ]  },  {  “id”:”location”  “datapoints”:[…]  }, { … }  ]  } |  |  |  |  |  |  |  |  |
| …  …  …  …  …  …  …  …  …  … |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
| Byte n |  |  |  |  |  |  |  |  |

注意：在封装存储（转发）数据包的时候，如果只是实现数据存储，目的地址设置为空；如果实现存储并转发，可以分为以下两种情况，第一，是在平台设备注册时设置了默认目的地址（设备申请时，填写的route\_to字段），数据包中的目的地址为空，此时数据存储后转发到默认目的地址对应的设备；第二，数据包中的目的地址不为空，数据包存储后被转发到数据包中设备ID对应的设备。

## 存储确认

当消息类型“存储数据”中消息编号指示位设置为1时，平台将下发该消息，用作存储消息的确认。该消息包含：消息头，1个必选项和可能的消息体。其中，选项包括一个字节的标志；消息体由选项决定。

若选项的第6位被置上，表示后面携带有固定两字节大小的消息编号和一字节大小的用于表示存储是否成功的消息体。消息编号与SAVE\_DATA类型的消息中携带的消息编号相同。

举例格式如下：

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **字节** | **说明\bit** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| 消息头 | | | | | | | | | |
| Byte 1 | 第一字节：  Bit（4-7）：消息类型，**值为9**；  Bit（0-3）：保留位，值为0； | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| Byte 2 | 第二字节：  消息剩余字节长度，值为4 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 选项：标志 | | | | | | | | | |
| Byte 3 | Bit（7）：系统保留位  Bit（6）：确认信息指示位，置1  Bit（5-0）：系统保留位 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 确认信息 | | | | | | | | | |
| Byte 4 | 消存储数据消息中携带的消息编号（固定2字节，大端序，示例：261） | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| Byte 5 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| Byte 6 | 成功指示：  0: 成功  1：失败  其他：保留 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## 命令请求

该消息由服务器发往客户端，以执行指定的命令。格式如下：

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **字节** | **说明\bit** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| 消息头 | | | | | | | | | |
| Byte 1 | 第一字节：  Bit（4-7）：消息类型，值为10；  Bit（0-3）：保留位，值为0； | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 剩余消息长度（长度不定1-4字节） | | | | | | | | | |
| Byte 2 | 剩余长度 |  |  |  |  |  |  |  |  |
| ... |  |  |  |  |  |  |  |  |
| Byte x |  |  |  |  |  |  |  |  |
| Byte x+1 | cmdid两字节长度高位字节 |  |  |  |  |  |  |  |  |
| Byte x+2 | cmdid两字节长度低位字节 |  |  |  |  |  |  |  |  |
| Byte x+3 | cmdid |  |  |  |  |  |  |  |  |
| ...  ...  ... |  |  |  |  |  |  |  |  |
| Byte y |  |  |  |  |  |  |  |  |
| Byte y+1 | 命令消息体四字节长度第1字节（高位） |  |  |  |  |  |  |  |  |
| Byte y+2 | 命令消息体四字节长度第2字节 |  |  |  |  |  |  |  |  |
| Byte y+3 | 命令消息体四字节长度第3字节 |  |  |  |  |  |  |  |  |
| Byte y+4 | 命令消息体四字节长度第4字节（低位） |  |  |  |  |  |  |  |  |
| Byte y+5 | 命令消息体（不超过64k） |  |  |  |  |  |  |  |  |
| ...  ...  ... |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
| Byte z |  |  |  |  |  |  |  |  |

## 命令响应

该消息由客户端发往服务器，以响应相应的命令。格式如下：

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **字节** | **说明\bit** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| 消息头 | | | | | | | | | |
| Byte 1 | 第一字节：  Bit（4-7）：消息类型，值为11；  Bit（0-3）：保留位，值为0； | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| 剩余消息长度（长度不定1-4字节） | | | | | | | | | |
| Byte 2 | 剩余长度 |  |  |  |  |  |  |  |  |
| ... |  |  |  |  |  |  |  |  |
| Byte x |  |  |  |  |  |  |  |  |
| Byte x+1 | cmdid两字节长度高位字节 |  |  |  |  |  |  |  |  |
| Byte x+2 | cmdid两字节长度低位字节 |  |  |  |  |  |  |  |  |
| Byte x+3 | cmdid |  |  |  |  |  |  |  |  |
| ...  ...  ... |  |  |  |  |  |  |  |  |
| Byte y |  |  |  |  |  |  |  |  |
| Byte y+1 | 命令响应消息体四字节长度第1字节（高位） |  |  |  |  |  |  |  |  |
| Byte y+2 | 命令响应消息体四字节长度第2字节 |  |  |  |  |  |  |  |  |
| Byte y+3 | 命令响应消息体四字节长度第3字节 |  |  |  |  |  |  |  |  |
| Byte y+4 | 命令响应消息体四字节长度第4字节（低位） |  |  |  |  |  |  |  |  |
| Byte y+5 | 命令响应消息体（不超过64k） |  |  |  |  |  |  |  |  |
| ...  ...  ... |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
| Byte z |  |  |  |  |  |  |  |  |

## 心跳请求

该消息只有消息头，由客户端发向服务器，格式如下：

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **字节** | **说明\bit** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| 消息头 | | | | | | | | | |
| Byte 1 | 第一字节：  Bit（4-7）：消息类型，值为12；  Bit（0-3）：保留位，值为0； | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| Byte 2 | 第二字节：  消息剩余字节长度，值为0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## 心跳响应

该消息只有消息头，从服务器返回客户端，格式如下：

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **字节** | **说明\bit** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| 消息头 | | | | | | | | | |
| Byte 1 | 第一字节：  Bit（4-7）：消息类型，值为13；  Bit（0-3）：保留位，值为0； | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| Byte 2 | 第二字节：  消息剩余字节长度，值为0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## 加密请求

该消息由客户端发向服务器端，表明此后客户端与服务器之间的通信数据需要加密，否则采用明文通信。加密内容为剩余消息长度之后的内容，被加密命令的剩余消息长度设置为加密后的数据长度。

服务器端收到此消息后，随机选择一个对称加密算法的密钥，利用接收到的RSA公钥，以RSA\_PKCS1\_PADDING模式加密，之后发送给客户端。此后服务器端与客户端之间传递的数据利用对称加密算法加密。

对称加密算法由客户端在本消息中指定，目前服务器端只支持AES加密算法，代码为1，密钥长度采用128位，加密模式为ECB模式，填充方式为ISO10126padding方式。

此消息需要在连接请求之前发送，若服务器端在收到连接请求之后再收到此消息，则认为消息序列混乱，关闭连接。

消息体包含客户端RSA公钥中的e和n，e为DWORD类型，按照大端字节序方式传输，n为128字节的BYTE类型，按照字节流的顺序传输。其中DWORD为32位的无符号四字节整形，BYTE为8位的无符号单字节整形。

消息体格式如下：

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **字节** | **说明\bit** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| 消息头 | | | | | | | | | |
| Byte 1 | 第一字节：  Bit（4-7）：消息类型，值为14；  Bit（0-3）：保留位，值为0； | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| 剩余消息长度(133编码后需要占用2个字节) | | | | | | | | | |
| Byte 2 | 消息剩余字节长度(133)-编码第一字节（低） | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| Byte 3 | 消息剩余字节长度(133)-编码第二字节（高） | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| Byte 4 | 公钥信息中的e编码后的第一字节（高） |  |  |  |  |  |  |  |  |
| Byte 5 | 公钥信息中的e编码后的第二字节 |  |  |  |  |  |  |  |  |
| Byte 6 | 公钥信息中的e编码后的第三字节 |  |  |  |  |  |  |  |  |
| Byte 7 | 公钥信息中的e编码后的第四字节（低） |  |  |  |  |  |  |  |  |
| Byte 8 | 公钥信息中的n，共128字节 |  |  |  |  |  |  |  |  |
| ...  ...  ... |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
| Byte 135 |  |  |  |  |  |  |  |  |
| Byte  136 | 1：AES加密算法代码 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

## 加密响应

该消息由服务器端发往客户端，以响应加密请求。格式如下：

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **字节** | **说明\bit** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| 消息头 | | | | | | | | | |
| Byte 1 | 第一字节：  Bit（4-7）：消息类型，值为15；  Bit（0-3）：保留位，值为0； | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 剩余消息长度（长度不定1-4字节） | | | | | | | | | |
| Byte 2 | 剩余长度 |  |  |  |  |  |  |  |  |
| ... |  |  |  |  |  |  |  |  |
| Byte x |  |  |  |  |  |  |  |  |
| Byte x+1 | 密钥加密后两字节长度高位字节 |  |  |  |  |  |  |  |  |
| Byte x+2 | 密钥加密后两字节长度低位字节 |  |  |  |  |  |  |  |  |
| Byte x+3 | 加密后密钥信息 |  |  |  |  |  |  |  |  |
| ...  ...  ... |  |  |  |  |  |  |  |  |
| Byte y |  |  |  |  |  |  |  |  |

# 主要流程

## 登录

EDP设备登录设备云流程：

1. 访问设备云门户<http://open.iot.10086.cn/>注册用户；
2. 用户根据业务情况，在”连接请求”章节中选择EDP登录方式；
3. 根据登录方式，填写设备相关属性，在产品下新增设备，获取产品ID、设备ID，以及api-key等信息；
4. 设备发送连接请求报文到设备云服务器地址：jjfaedp.hedevice.com，端口 876或29876

**EDP登陆消息流**



* 根据相关登陆方式，在CONN\_REQ消息中携带验证信息；
* 平台验证后，返回鉴权响应码。

## 数据收发（透传）

* 要实现透传，通信双方必须都要登陆到设备云；
* 设备1使用消息类型3“发送数据”发送数据到设备云，目的地址写明设备2的ID（若目的地址长度为零则使用创建设备时的route\_to为默认接收地址），设备云根据目的地址使用消息类型3（源地址填充为设备2的ID）转发数据给设备2；
* 设备2接收到数据（消息类型3），可以查看到源地址（设备1的ID），通过平台向设备1发送应答。

**EDP消息流：**



## 存储数据点（datapoint）

利用EDP上报数据点到设备云进行存储的流程：

* 在设备云门户注册用户，创建需要登录的设备（详见登录流程）；
* 通过设备云门户该设备的详情页，或REST API 创建需要存储的数据流（定义上传数据类型名称）；
* 设备进行EDP登录鉴权；
* 根据数据类型（json或二进制），数据流ID（名称）和当前值，时间（默认当前系统时间）等信息，封装消息类型8（”存储数据”，具体格式见消息定义中的示例）发送到设备云，设备云解析后将数据点保存在该设备对应数据流中。

**EDP消息流：**



## 存储数据点并获得确认

终端登录后，若发送消息类型8，且消息编号指示位被置1的消息并携带了有效的消息标志，则平台响应一个存储确认消息。消息标志不能为零。



## 存储数据点并转发

利用EDP上报数据点到设备云进行存储，同时转发数据到指定设备ID流程：

* 在设备云门户注册用户，创建需要登录的设备（详见登录流程）；
* 通过设备云门户该设备的详情页，或REST API 创建需要存储的数据流（定义上传数据类型名称）；
* 接收方和发送方设备都要进行EDP鉴权登录；
* 根据数据类型（json或二进制），数据流ID（名称）和当前值，时间（默认当前系统时间）等信息，封装消息类型8（”存储数据”，具体格式见消息定义中的示例），设置地址标志位，并填写目的地址（若目的地址长度为零，设备云转发到注册设备时route\_to字段指示的设备ID）；平台先将数据点保存在该设备对应数据流中，同时使用消息类型8（改设地址修改为源地址）将该数据点内容复制转发到目的设备。

**EDP消息流：**



## 命令请求及响应

平台向设备发送CMD\_REQ命令，携带要执行的命令及命令ID信息。设备收到此消息后处理此命令消息，并向平台回应CMD\_RESP消息。此消息中携带回应的命令ID及响应内容。



## 心跳保持

EDP连接默认超时时间为4分钟。设备登录后，在超时期内无数据传输时，需要定期向平台发送PING\_REQ消息以保持连接，平台响应PING\_RESP消息。



## 数据加密

如果设备期望加密与平台的交互信息，可以在发送连接请求之前发送RSA加密消息。消息类型为14，将设备端RSA公钥的e，n以及期望的对称加密算法代码（目前只支持AES，代码为1，密钥长度采用128位，加密模式为ECB模式，填充方式为ISO10126padding方式）填到消息体中发送给平台。平台收到此消息后，创建一个随机的对称加密算法密钥，利用接收到的RSA公钥，以RSA\_PKCS1\_PADDING填充模式加密后发送给设备。此后设备与平台的交互信息中，剩余消息长度之后的内容都是利用对称加密算法加密后的内容，剩余消息长度为加密后的内容长度。

需要加密的EDP设备登录设备云流程如下：

