设备云管家2.0 消息订阅、消息推送数据格式定义(对内)

前言

目前设备云管家不管是对内还是对外的消息流转，mq选型暂时统一用kafka。

数据流转消息协议

消息主题为：`DEV_CLOUD_MSG_TRANSFER`

msg消息的内容格式如下：

```
{
    "sId": 1526817642290102273,// 消息序列id
    "bizCode": 4, // 代表不同的业务码，不同的类型消息，bizCode的值会不一样目前bizCode的值有（1~6）
    "ver":"1.0" , // 协议的版本号1.0，预留着后面拓展升级结构体的时候用的上，同个业务码版本不同，业务
    "encrypt":0, // 是否加密(0-未加密，1-加密)，加密算法采用aes 对称加密
    "data":"*********", // 消息体内容，为了安全，对此字段做（对称）加密，解密后是json字符串格式或是报文字符串
    "t":1653009853587, // 发送消息时的时间戳
    “sign”: "" // 消息签名，为了安全考虑，可以避免后台人为恶意发送,签名算法可以简单点（内部服务可以自行决定是否进行校验）
}
```

前期由于是内部服务订阅，消息内容就先不加密，encrypt值为0，代表不加密。同时也可以不对签名进行校验，

data加密算法：

采用AES对称加密，secretKey秘钥是长度为16个字符的字符串，秘钥由设备云管家约定分配。

解密：

1. 对data进行base64解码
2. 用aes算法和secretKey秘钥 ，对字节流进行解密，
3. 用UTF8编码对字节流转成字符串

加解密示例工具类如下：

```
public class AESBase64Utils {

    private static final String AES = "AES";

    // 加密算法
    private String              ALGO;

    // 16位的加密密钥
    private byte[]              keyValue;

    /**
     * 用来进行加密的操作
     *
     * @param data
     * @return
     * @throws Exception
     */
    public String encrypt(String data) throws Exception {
        Key key = generateKey();
        Cipher c = Cipher.getInstance(ALGO);
        c.init(Cipher.ENCRYPT_MODE, key);
        byte[] encVal = c.doFinal(StringUtils.getBytesUtf8(data));
        String encryptedValue = Base64.encodeBase64String(encVal);
        return encryptedValue;
    }

    /**
     * 用来进行解密的操作
     *
     * @param encryptedData
     * @return
     * @throws Exception
     */
    public String decrypt(String encryptedData) throws Exception {
        Key key = generateKey();
        Cipher c = Cipher.getInstance(ALGO);
        c.init(Cipher.DECRYPT_MODE, key);
        byte[] decodedValue = Base64.decodeBase64(encryptedData);
        byte[] decValue = c.doFinal(decodedValue);
        String decryptedValue = StringUtils.newStringUtf8(decValue);
        return decryptedValue;
    }

    /**
     * 根据密钥和算法生成Key
     *
     * @return
     * @throws Exception
     */
    private Key generateKey() throws Exception {
        Key key = new SecretKeySpec(keyValue, ALGO);
        return key;
    }
```

```
    public String getALGO() {
        return ALGO;
    }

    public void setALGO(String aLGO) {
        ALGO = aLGO;
    }

    public byte[] getKeyValue() {
        return keyValue;
    }

    public void setKeyValue(byte[] keyValue) {
        this.keyValue = keyValue;
    }

    public static String decrypt(String data, String secretKey) throws Exception
{
        // 创建加解密
        AESBase64Utils aes = new AESBase64Utils();
        // 设置加解密算法
        aes.setALGO(AES);
        // 设置加解密密钥
        aes.setKeyValue(secretKey.getBytes());
        // 进行解密后的字符串
        return aes.decrypt(data);
    }

    public static String encrypt(String data, String secretKey) throws Exception
{
        // 创建加解密
        AESBase64Utils aes = new AESBase64Utils();
        // 设置加解密算法
        aes.setALGO(AES);
        // 设置加解密密钥
        aes.setKeyValue(secretKey.getBytes());
        // 进行解密后的字符串
        return aes.encrypt(data);
    }

}
```

消息解析流程：

1. 消息消费者订阅接收到消息后，根据encrypt判断消息的data是否需要进行解密，如果不需要解密则为明文。
2. 根据bizCode和ver确定 消息类型以及消息模型，程序好进行下一步的处理。

建议： 消费端一旦消费过程中出现问题，需要把接收到的消息记录或转发到某个专门记录失败消息的消息主题，后期可以针对性的做重新消费或者排查失败原因。

消息类型bizCode

bizCode	ver	描述
1	1.0	透传消息（可先忽略）
2	1.0	设备上线
3	1.0	设备离线
4	1.0	设备数据上报
5	1.0	设备事件上报
6	1.0	设备模板变动通知
7	1.0	设备归属关系变动通知

- 透传消息（本期暂无透传消息上报，可先忽略） ： 1

```
#msg.bizCode= 1
#msg.data 解密后
{
    "deviceId": "xxxxx",  // 设备id,
    "deviceSn":"xxxx" , // 设备号
    "encode":"", // 编码类型 hex \ ascill\base64 \ 等等其他的，该属性主要用于报文字符串解码
    "msg":"xxxxxxx", // 消息报文字符串
    "ts": 1653013612400 // 消息报文接收的时间
}
```

- 设备上线 : 2

```
#msg.bizCode= 2
#msg.data 解密后
{
    "deviceId": "xxxxx",  // 设备id，设备id是否全局唯一
    "deviceSn":"xxxx" , // 设备号
    "ts" : 1653013612400  // 上线时间
}
```

- 设备离线 : 3

```
#msg.bizCode= 3
#msg.data 解密后
{
    "deviceId": "xxxxx",  // 设备id，设备id是否全局唯一
    "deviceSn":"xxxx" , // 设备号
    "ts" : 1653013612400  // 下线时间
}
```

- 设备数据上报：4

```
#msg.bizCode= 4
#msg.data 解密后
{
    "deviceId": "xxxxx",  // 设备id，
    "deviceSn":"xxxx" , // 设备号
    "templateId":12312312313,   // 模板id，long类型
    "props":{
            "key1" : "value1",   // key是要素标识符1，value 是值
            "key2" : "value1",   // key是要素标识符2，value 是值
            ...
            "keyn" : "value1",   // key是要素标识符n，value 是值
    },
    "illegal":{
            "key1" : "value1",   // key是要素标识符1，value 是值
            "key2" : "value1",   // key是要素标识符2，value 是值
            ...
            "keyn" : "value1",   // key是要素标识符n，value 是值
    },   // 不合规的要素
    "ts":1653013612400   // 上报时间
}
```

- 设备事件上报（本期暂无设备事件上报，可先忽略）：5

```
#msg.bizCode = 5
#msg.data 解密后
{
    "deviceId": "xxxxx",  // 设备id，
    "deviceSn":"xxxx" , // 设备号
    "event":"xxx_event",//  事件类型名称，物模型里定义的事件
    "props":{  // 事件附带参数
        "key1" : "xxxxx",
        "key2":"xxxx"
     }
    "ts":1653013612400  // 事件上报时间
}
```

- 设备模板变动通知：6

```
#msg.bizCode= 6
#msg.data 解密后
{
    "templateId":12312312313,   // 模板id, long类型
    "schema":[
        {
            "identifier":"xxxx1",   // 要素标识符 字符串
            "dataType":"xxxx",      // 数据类型 number , char
            "dataLength": 123 // 数据长度
        },
        {...} // 要素2
    ]
}
```

- 设备归属关系变动通知：7

```
#msg.bizCode= 7
#msg.data 解密后
{
    "deviceId":12312312313,   // 设备id long类型
    "tenantId": 123134564564566, //  租户id , long 类型
    "templateId":12312312313,   // 模板id, long类型
    "appIds":[123232323,23232323,2323232333] // 应用id数组,long 类型
}
```