点播资源统一选链服务

文档的目的与受众

目的：描述资源选链API的调用方式，供相关调用方使用

受众：所有关注资源选链API的调用方以及资源选链API开发人员等

1 请求协议描述

资源选链服务接口设计遵从OpenApi设计规范

1.1 请求地址

环境	HTTP请求地址
测试环境	http://10.20.222.143:8068/unios-data-uslink

1.2 请求参数

HTTP Method

采用GET、POST请求方式

通用参数说明

参数名	参数类型	是否必填	描述

1.3 返回结果

返回结果为JSON串，具体格式如下

返回结果格式

```
{
    "errorCode": "1010000",
    "errorMsg": "",
    "result": {}
}
```

返回字段说明

参数名	参数类型	描述
errorCode	String	调用接口返回码
errorMsg	String	错误描述

result	JSON	json中返回具体业务调用的结果信息

1.4 签名说明

选链服务支持一下几种签名方式：MD5（签名方式为空默认是MD5）、SHA1、AES、DES、HMACSHA256

1、MD5、HMACSHA256、AES、DES

内容签名生成规则：
A）将所有请求参数（除encryptMethod）放入Map中（MD5签名将appSecret也一并放入Map），注意：计算signature时所有参数不应进行URLEncode；
B）将格式化后的参数以字典序升序排列，拼接在一起，注意字典序中大写字母在前，空值不参与签名；
C）将B形成字符串获取摘要，即为本次请求signature（签名）的值；

MD5签名Java代码：

说明：params：含有appSecret的Map；

MD5签名Java代码

```
private static final String SYMBOL_AND = "&";

/**
 *
 *
 * @param params
 *
 * @return
 * @throws IllegalAccessException
 */
public static String getSign(Map<String, String> params) throws Exception {
    if (null == params || params.isEmpty() ) {
        return null;
    }
    params.remove("signature");
    //
    String result = formatAndSortParams(params);
    // Base64
    result = CoderUtil.encryptBASE64(result).replace("\r\n", "");
    // MD5
    result = new String(DigestUtils.md5Hex(result));
    return result;
}

/**
 *
 * @param map
 * @return
 * @throws Exception
 */
private static String formatAndSortParams(Map<String, String> map) throws
Exception {
    if (null == map || map.size() == 0) {
        return null;
```

```
    }

    ArrayList<String> list = new ArrayList<String>();
    for (Map.Entry<String, String> entry : map.entrySet()) {
        if (StringHandler.isNullOrEmpty(entry.getKey()) ||
StringHandler.isNullOrEmpty(entry.getValue())) {
            continue;
        }
        list.add(entry.getKey().trim() + "=" +
UrlCoder.encode(entry.getValue().trim()) + "&");
    }
    int size = list.size();
    String[] arrayToSort = list.toArray(new String[size]);
    Arrays.sort(arrayToSort, String.CASE_INSENSITIVE_ORDER);
    StringBuilder sb = new StringBuilder();
    for (int i = 0; i < size; i++) {
        sb.append(arrayToSort[i]);
    }
    String params = sb.toString();

    if (params.endsWith(SYMBOL_AND)) {
        params = params.substring(0, params.length() - 1);
```

```
    }
    return params;
}
```

HMACSHA256签名Java代码：

说明：data：Map参数排序后的字符串；key：appSecret

HMACSHA256签名Java代码

```
public static String HMACSHA256(String data, String key) throws Exception {
    Mac sha256_HMAC = Mac.getInstance("HmacSHA256");
    SecretKeySpec secret_key = new SecretKeySpec(key.getBytes("UTF-8"),
"HmacSHA256");
    sha256_HMAC.init(secret_key);
    byte[] array = sha256_HMAC.doFinal(data.getBytes("UTF-8"));
    StringBuilder sb = new StringBuilder();
    byte[] var6 = array;
    int var7 = array.length;

    for(int var8 = 0; var8 < var7; ++var8) {
        byte item = var6[var8];
        sb.append(Integer.toHexString(item & 255 | 256).substring(1, 3));
    }
    return sb.toString().toUpperCase();
}
```

AES签名Java代码：

说明：sSrc：Map参数排序后的字符串；encodingFormat：UTF-8；sKey：appSecret的前16位；ivParameter：appSecret 从17位到最后

AES签名Java代码

```
//
public static String cbc_encrypt(String sSrc, String encodingFormat, String
sKey, String ivParameter) throws Exception {
   Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");
   byte[] raw = sKey.getBytes();
   SecretKeySpec skeySpec = new SecretKeySpec(raw, "AES");
   IvParameterSpec iv = new IvParameterSpec(ivParameter.getBytes());//
CBCiv
   cipher.init(Cipher.ENCRYPT_MODE, skeySpec, iv);
   byte[] encrypted = cipher.doFinal(sSrc.getBytes(encodingFormat));
   String ret =  new BASE64Encoder().encode(encrypted);// BASE64
   ret = ret.replaceAll("\r\n", "");
   ret = ret.replaceAll("\n", "");
   return ret;
}
```

DES签名Java代码：

说明：src：Map参数排序后的字符串；key：appSecret

DES签名Java代码

```
/**
 * 3DESECB,key 3*8 = 24
 *
 * @param src String
 * @param key   String
 * @return String
 */
public static String encryptThreeDESECB(String src, String key) throws
Exception {
    if (src == null || key == null) {
        return null;
    }
    final DESedeKeySpec dks = new DESedeKeySpec(key.getBytes("UTF-8"));
    final SecretKeyFactory keyFactory =
SecretKeyFactory.getInstance("DESede");
    final SecretKey securekey = keyFactory.generateSecret(dks);

    final Cipher cipher = Cipher.getInstance("DESede/ECB/PKCS5Padding");
    cipher.init(Cipher.ENCRYPT_MODE, securekey);
    final byte[] b = cipher.doFinal(src.getBytes("UTF-8"));

    final BASE64Encoder encoder = new BASE64Encoder();
    return encoder.encode(b).replaceAll("\r", "").replaceAll("\n", "");

}
```

2、SHA1

内容签名生成规则：
A）将所有请求参数（除encryptMethod）和appSecret的值放入List中，注意：计算signature时所有参数不应进行URLEncode；
B）将格式化后的参数以字典序升序排列，拼接在一起，注意字典序中大写字母在前，空值（null）使用空字符串代替；
C）将B形成字符串获取SHA1摘要，形成一个40位的十六进制（字母大写）字符串，即为本次请求signature（签名）的值；

然后再GET请求时，所有参数的值都应进行URLEncode；(参数值是数字或字母的可以不进行URLEncode)

该签名值基本可以保证请求是合法者发送且参数没有被修改，但无法保证不被偷窥。

SHA1签名Java代码：

SHA1签名Java代码

```
/**
  *
  *
  * @param params
  * @return
  */
 public static String buildSignature(List<String> params) {
     if (params == null || params.isEmpty()) {
```

```
            return "";
        }

        //
        Collections.sort(params);

        StringBuilder sb = new StringBuilder();
        for (String param : params) {
            sb.append(param == null ? "" : param);
        }

        return getSHA1Digest(sb.toString());
    }

    /**
     * SHA1
     *
     * @param data
     * @return
     * @throws Exception
     */
    public static String getSHA1Digest(String data) {
        String digest = null;
        try {
            MessageDigest md = MessageDigest.getInstance("SHA-1");
            byte[] bytes = md.digest(data.getBytes("UTF-8"));
            digest = byte2hex(bytes);
        } catch (Exception e) {
            logger.error(e.getMessage(), e);
        }

        return digest;
    }

    /**
     *
     *
     * @param bytes
     * @return
     */
    private static String byte2hex(byte[] bytes) {
        StringBuilder sign = new StringBuilder();
        for (int i = 0; i < bytes.length; i++) {
            String hex = Integer.toHexString(bytes[i] & 0xFF);
            if (hex.length() == 1) {
                sign.append("0");
            }
            sign.append(hex.toUpperCase());
        }
```

```
        return sign.toString();
    }
```

2 API定义

数据类型dataType和对应的数据源dataSourceCode列表

数据类型code	数据类型名称	数据源code	数据源名称
child	儿童资源数据类型	child	儿童资源统一数据源
music	音乐数据类型	kuwo	酷我音乐数据源
news	新闻数据类型	leting	乐听新闻数据源
audio	有声读物数据类型	qingtingFM	蜻蜓FM数据源
		xmly	喜马拉雅数据源
channel	广播电台数据类型	qingtingFM	蜻蜓FM数据源

2.1 资源选链（/rest/v1/link/get_data_link）

请求类型：GET

请求参数：除了通用参数外，还包括如下业务参数

参数名	类型	必需	默认值	描述
appKey	String	Y	无	应用Key
udid	String	Y	无	设备号
deviceType	String	Y	无	设备类型（android ，ios，chip ，web）
dataType	String	Y	无	数据类型（child/music/audio）
dataSourceCode	String	Y	无	数据源（child/xmly/kuwo）
id	String	Y	无	资源id
resourceType	Integer	Y	无	资源类型（1：视频 2：音频 ）
timestamp	Long	Y	无	时间戳，有效性（10分钟内）
encryptMethod	String	Y	MD5	加密方式(MD5/AES/DES/SHA1/HMACSHA256)
signature	String	Y	无	签名

选链返回值

名称	类型	必须	默认值	描述
errorCode	String	Y	无	错误码
errorMsg	String	Y	无	错误描述
result	JSON	Y	无	业务返回值

└data	JSON	Y	无	儿童资源列表
└└id	String	Y	无	资源ID，唯一标识
└└url	String	Y	无	播放链接
└└totalContentPlayTime	Integer	Y	无	播放时长,单位秒
└└expiryTime	Integer	N	无	音频地址过期时间,单位秒

http://10.20.222.143:8068/unios-data-uslink/rest/v1/link/get_data_link?appKey=oa7bnqilgjv6glj3utgstbink7lahd3m7refcbi2&udid=uni_uid&deviceType=android&signature=0268a4ca5f6b1155367304216adff48c&id=2000130210&timestamp=1558347389&encryptMethod=MD5&dataType=child&dataSourceCode=child&resourceType=2

选链返回结果示例


```
{
 "errorCode": "0",
 "errorMsg": "",
 "result": {
  "data": {
   "id": "2000130210",
   "totalContentPlayTime": 161,
   "url":
"http://resource.hivoice.cn/dcs-resources/audio/44fff002ec33f4c5ada5af4c23
7af6d4.mp3"
  }
 }
}
```


3 错误码列表

错误码格式

格式：BBBCCCC

- BBB: 3位数字，是服务编号；
- CCCC:4位数字，服务内部错误码；

业务系统代号定义

服务编号	系统名称	服务描述
302	资源选链服务	资源选链服务

详细错误列表

errorCode	errorMsg	描述
0	请求成功	请求成功

3020001	请求参数不合法	请求参数不合法
3024444	调用内容服务异常	调用第三方内容服务异常
3029999	系统异常	系统异常
3020002	请求参数错误	业务校验参数错误
3020003	请求时间戳超出了请求有效期	时间戳与当前时间差超过10分钟
3020004	签名错误	签名校验错误
3020005	调用第三方内容服务结果为空	调用第三方内容服务结果为空