浙江省健康数据高铁安诊儿专线

接口规范

第 1 部分：总则

（试行）

目 次

前 言

《浙江省健康数据高铁安诊儿专线接口规范》分为以下部分：
——第 1 部分：总则；
——第 2 部分：消息接口；
——第 3 部分：服务接口；

本部分为浙江省健康数据高铁安诊儿专线接口规范的第 1 部分。
本部分起草单位：浙江省卫生健康信息中心。

1 范围

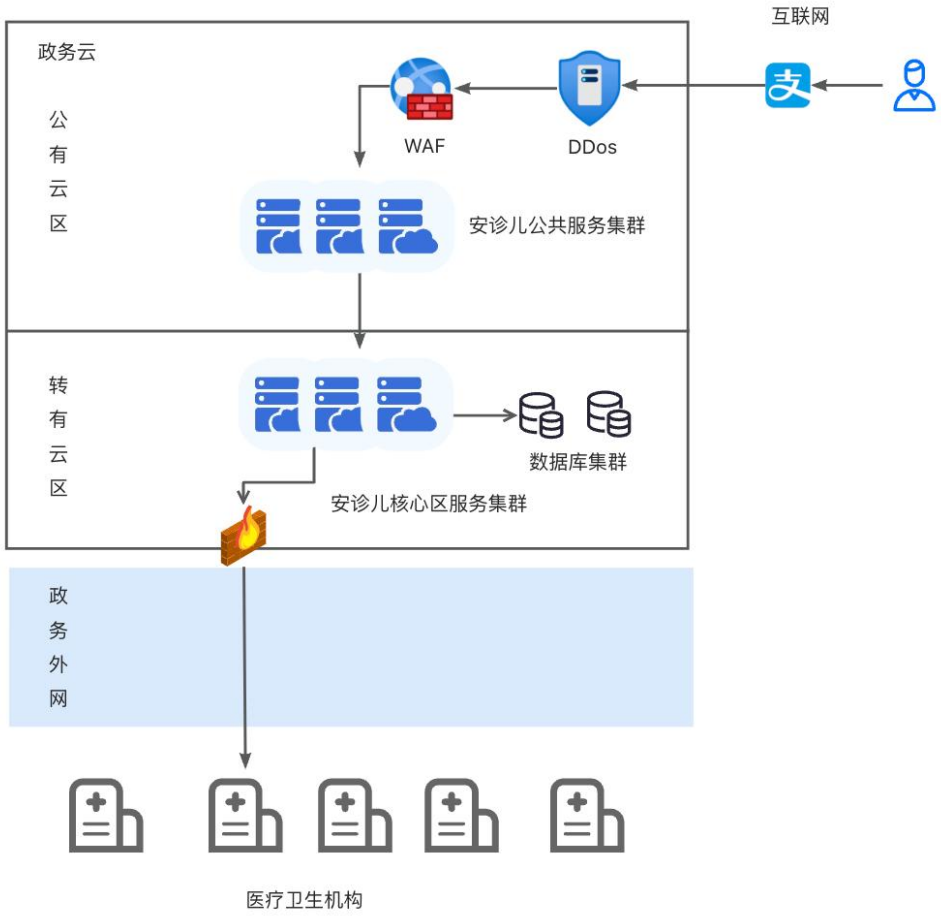
本部分规定了浙江省健康数据高铁安诊儿专线接口传输的方式、技术规范以及数据元的属性与描述规则。

本部分适用于各级各类医疗机构信息系统或区域卫生信息平台和安诊儿服务平台的数据传输。

2 平台介绍

将 “以病人为中心”贯穿于诊前、诊中、诊后医疗服务各环节，依托大模型、大数据、人工智能等新技术，智能生成全程就医实时路径，实现患者一站式智慧服务，帮助患者简化就医流程，提升患者就医体验。

3 网络拓扑



安诊儿专线接口包含双向交易，需要实现省平台与医疗卫生机构网络的双向联通。

1、省平台云端接口地址（医院->省平台）：

https://59.202.52.129:21154/api/his/data/transmission（测试环境-政务外网）

https://59.202.52.129:28171/api/his/data/transmission（正式环境-政务外网）

2、医疗卫生机构源端地址（省平台->医院）：

由医院提供（代理至政务外网，不同业务接口建议封装为一个 IP+端口地址，同时做好白名单限制）

测试阶段开放以下 IP 访问： 8.149.246.112 8.149.135.174 8.149.240.76

正式上线开放以下 IP 访问： 59.202.48.80；

备注：各级卫健委负责下属医疗卫生机构网络配置工作。

4 接口说明

安诊儿服务平台接口按照就医事件消息上报和院端业务调用服务场景的不同，分为**消息接口**和**服务接口**两类。

4.1 消息接口

安诊儿服务平台开放接口，医疗卫生机构向安诊儿服务平台上传/更新预约、挂号、停诊、门诊、住院等就医消息的接口。

4.1.1 数据传输方法

消息接口服务方式使用 HTTPS API 接口传输数据，交互双方通过具体的数据请求和应答完成数据传输。

4.1.2 消息服务分类

云陪诊数字人消息接口数据传输服务分类见表 1。

表 1 服务分类

服务名称	交易代码	交易触发点	服务提供者	服务请求者
数据上传服务	1001	数据产生，实时	安诊儿服务平台	卫生信息数据源
数据更新服务	1002	数据变更，实时	安诊儿服务平台	卫生信息数据源
数据查询服务	1003	数据查询，实时	卫生信息数据源	安诊儿服务平台

4.1.3 数据对接流程

4.1.3.1 数据上传

由接入方卫生信息服务向安诊儿服务平台发起数据上传请求，此交易会触发数据的存储行为，安诊儿服务平台向接入方卫生信息服务应答上传和存储的结果信息。

安诊儿服务平台存储数据的执行符合唯一性是否必填，当发生唯一性是否必填冲突时，数据存储将执行异常，安诊儿服务平台应答异常处理结果。

4.1.3.2 数据更新

由接入方卫生信息服务向安诊儿服务平台发起数据更新请求，此交易会触发数据的更新行为，安诊儿服务平台向接入方卫生信息服务应答更新的结果信息。

安诊儿服务平台更新数据的执行依赖已上传和存储的存量数据，当平台不存在要更新的原始数据记录时，安诊儿服务平台将应答异常处理结果。

4.1.3.3 数据查询

由安诊儿服务平台向接入方卫生信息服务发起数据查询请求，此交易会触发接入方卫生信息服务数据查询行为，接入方卫生信息服务向安诊儿服务平台应答查询结果信息。

4.1.4 消息接口定义

数据传输服务接口的定义见表 2。

表 2 消息接口定义

功能说明	根据具体的请求消息，进行数据传输，并以数据流的形式返回传输结果
通讯方式	HTTPS
消息编码	UTF-8
请求方法	POST
数据格式	Content-Type: application/json

4.1.4.1 数据结构

4.1.4.1.1 请求数据结构

请求数据包含请求头 Header（控制参数）和请求体 Body（业务参数）两个部分，Header 请求头参数结构见表 3。

表 3 Header 参数

参数名称	**是否必填**	**数据类型**	**参数说明**
Content-Type	是	String	application/json;charset=UTF-8
tradeCode	是	String	交易代码。见表 1 服务分类
datasetCode	是	String	数据集代码。见《浙江省健康数据高铁安诊儿专线接口规范 第 2 部分：消息接口》2.1 各数据集定义
requestId	是	String	请求唯一编码。由服务请求者定义，最长 32 位
platformCode	条件必选	String	区域平台归属单位的 18 位统一社会信用代码。（单家医疗机构接入时不可传值），通过区域平台接入方式请求时必传。
medOrgCode	是	String	医疗机构的 18 位统一社会信用代码

medHosCode	是	String	医疗机构院区的顺序号代码。多个院区应使用相同的编码体系
timestamp	是	String	请求时间戳（13 位毫秒级，例：1735704000000）
signature	是	String	加签字符串（规则见加签机制）
cryptoVersion	否	String	秘钥版本（不参与签名）

Body 参数结构见表 4。

表 4 Body 参数体

参数名称	是否必填	数据类型	参数说明
requestBiz	是	String	请求业务参数，内容需加密。见各数据集业务请求参数定义

4.1.4.2.1 响应数据结构

响应数据结构见表 5。

表 5 响应参数体

字段名称	**是否必填**	**数据类型**	**字段说明**
success	是	String	响应结果，true.成功 false.失败
errCode	是	String	响应代码
errMsg	是	String	错误信息
responseBiz	是	String	响应业务参数，内容需加密。见各数据集业务响应参数定义

4.1.5 获取平台密钥接口定义

4.1.5.1.1 获取平台授权密钥

功能说明	根据医院信息获取平台授权的密钥信息（SM2 加密）），医院调用平台业务接口时，需要先调用此接口获取加密秘钥
通讯方式	HTTPS
消息编码	UTF-8
请求方法	POST
数据格式	Content-Type: application/json
平台测试地址	https://host:port/api/crypto/getKMInfo

4.1.5.1.2 请求数据结构

请求数据只包含请求头 Header（控制参数），获取平台秘钥接口不需要请求体 Body 部分，Header 请求头参数结构见表 6。

表 6 Header 参数

参数名称	**是否必填**	**数据类型**	**参数说明**
Content-Type	是	String	application/json;charset=UTF-8
requestId	是	String	请求唯一编码。由服务请求者定义，最长 32 位
platformCode	条件必选	String	区域平台归属单位的 18 位统一社会信用代码。（单家医疗机构接入时不可传值），通过区域平台接入方式请求时必传。
medOrgCode	是	String	医疗机构的 18 位统一社会信用代码
medHosCode	是	String	医疗机构院区的顺序号代码。多个院区应使用相同的编码体系
timestamp	是	String	请求时间戳（13 位毫秒级，例：1735704000000）
signature	是	String	签名信息

4.1.5.1.3 响应数据结构

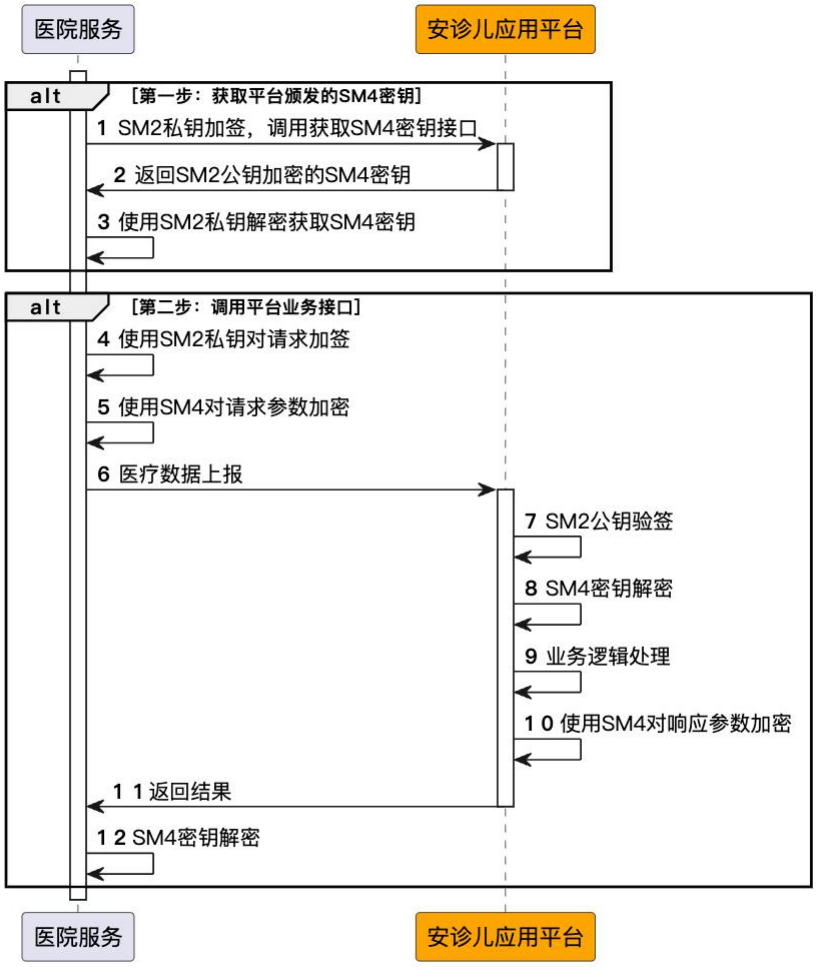
响应数据结构见表 7。

表 7 响应参数体

字段名称	是否必填	数据类型	字段说明
success	是	String	响应结果，true.成功 false.失败
errCode	是	String	响应代码
errMsg	是	String	错误信息
responseBiz	是	String	响应业务参数，内容通过 SM2 公钥加密。响应内容 {"cryptoInfo":"SM4 密 钥 （ 十 六 进 制 ）","cryptoVersion":"密钥版本信息"}

4.1.6 数据加密加签处理

表 8 加解密流程时序图



4.1.6.1 加解密流程

——加密时，使用加密算法对业务参数明文进行加密得到加密结果。
——解密时，使用解密算法对加密结果进行解密得到业务参数明文。

4.1.6.2 加解密方法

安诊儿服务平台数据传输服务的加解密方法使用 SM4 算法，加解密密钥由安诊儿的密钥管理平台统一维护，并设置 7 天有效期（过期后将自动刷新密钥）。

获取密钥后对请求参数体信息加密，示例：

平台颁发 SM4 密钥	726be1646879423e26eb7977c4d80c48

Body 参数明文	{
	"requestBiz":{
	"cfbh":"2022082066316802",
	"loadTime":"20220101120809"}
	}

业务参数 SM4 加密密文	12ab40eaa060b1e47bd2c83b6987cc4cae4cc2da9f3750ba7f33d0077a9273c5145a527174a3b76e7e17f22c544a6aa19727ff06c67ab3dbebaa485bb8b6ed7e9ecd9355f06865c2001206ae51ebf7fa

最终请求体 Body	{
	"requestBiz":"12ab40eaa060b1e47bd2c83b6987cc4cae4cc2da9f3750ba7f33d0077a9273c5145a527174a3b76e7e17f22c544a6aa19727ff06c67ab3dbebaa485bb8b6ed7e9ecd9355f06865c2001206ae51ebf7fa"
	}

4.1.6.4 签名流程

4.1.6.4.1 签名规则

为了请求过程的安全性，业务各方之间的请求需要实现加签，以保证数据的安全性。请求方实现对请求参数数据的加签，服务方根据 Header 请求参数 signature，对签名进行验签，签名不合法的请求将会被拒绝，同时对 timestamp 时间戳进行校验，超出 10 秒范围内的请求将会被拒绝。报文的签名及验证使用 SM2 算法，具体处理机制要求如下:

1）对 Header 所有非空业务参数数(cryptoVersion、signature 参数不参与签名)按照 key 的 ascii 顺序进行升序排序，然后拼接成一串待加签字符串，格式为：key=value&key=value。

示例：

平台接入：

datasetCode=HDSD.YYJL&medHosCode=med_hos_code_001&medOrgCode=med_org_code_001&platformCode=platform_code_001&requestId=63d78369-a947-475a-8fab-65267a0595c0&timestamp=1739978872979&tradeCode=1001

机构接入：

datasetCode=HDSD.YYJL&medHosCode=med_hos_code_001&medOrgCode=med_org_code_001&requestId=605d6672-854a-44f4-ae98-1102e97bcfc8&timestamp=1739978238539&tradeCode=1001

2）生成签名：对拼接的字符串使用 SM2 算法通过私钥生成加签字符串 signature

4.1.6.4.2 签名示例

SM2 公钥	04b674b6edfd08f754410b21cc3c8b522f318a1f1b27403bc63a290b7e1790d03d967d06c46e69357793967ff42a34d77ed5d7bb40d31b7f5ab0b0840017cff114

SM2 私钥	a99d6978656355b570a19cc707b2ea68ed033c3fa436df9c0a15e84a06069c7b
加签前参数内容	datasetCode=HDSD.YYJL&medHosCode=med_hos_code_001&medOrgCode=med_org_code_001&requestId=1408c07c-bfbb-4135-a516-9232ac0a3bbf&tradeCode=1001
加签后字符串(签名)	304502206ea399a815ccd5dcd70486f97fa3fa4faa84a0ec47e2dcb9d6786820f3cc520202210080a96be8c0c8d762a2a7ff0ee7153471f2292a19e77a98095d0916fb19da17b8

4.1.6.5 加解密、签名代码示例（Java）

4.1.6.5.1 Java 代码示例：

```
import cn.hutool.core.util.HexUtil;

import cn.hutool.crypto.SmUtil;

import cn.hutool.crypto.asymmetric.KeyType;

import cn.hutool.crypto.asymmetric.SM2;

import com.alibaba.fastjson.JSON;

import com.alibaba.fastjson.JSONObject;

import com.example.demo.utils.HttpUtils;

import org.apache.commons.collections4.MapUtils;

import org.junit.jupiter.api.Test;




import java.io.Serializable;

import java.util.*;



/**
```

```
 * his 数据集上传安诊儿专线平台流程-代码示例

 */

public class DataTransmissionDemo {



    // sm2 公私钥

    String SM2_PUBLIC_KEY =
"04b674b6edfd08f754410b21cc3c8b522f318a1f1b27403bc63a290b7e1790d03d967d06c46e69357793967ff42a34
d77ed5d7bb40d31b7f5ab0b0840017cff114";

    String SM2_PRIVATE_KEY =
"a99d6978656355b570a19cc707b2ea68ed033c3fa436df9c0a15e84a06069c7b";



    // 医疗机构统一社会信用代码

    String medOrgCode = "medOrgCode_001";

    // 院区编码

    String medHosCode = "medHosCode_001";

    // 区域平台归属单位的 18 位统一社会信用代码

    String platformCode = "platformCode_001";



    // 数据集编码 预约记录
```

```
String YYJL_DATASET_CODE = "HDSD.YYJL";

// 数据集编码 挂号记录

String GHJL_DATASET_CODE = "HDSD.GHJL";

// 数据集编码 门诊就诊记录

String MZJZJL_DATASET_CODE = "HDSD.MZJZJL";



// 交易编码 上传

String INSERT_TRADE_CODE = "1001";

// 交易编码 更新

String UPDATE_TRADE_CODE = "1002";



/**

 * his 上传数据集流程-代码示例

 */

@Test

public void hisServerDatasetUpload() {



    System.out.println("----------  ↓请求获取 SM4 数字信封↓ ----------");
```

```
// Header 参数

HashMap<String, String> header = new HashMap<>();

header.put("requestId", UUID.randomUUID().toString());

header.put("medOrgCode", medOrgCode);

header.put("medHosCode", medHosCode);

// 平台提供，接入时按需填入

header.put("platformCode", platformCode);

header.put("timestamp", String.valueOf(System.currentTimeMillis()));



// 预签名字符串

String preSignature = buildPreSignature(header);

System.out.println("预签名字符串：" + preSignature);

// 签名 （sm2 私钥,预签名内容）

String signature = sm2Sign(SM2_PRIVATE_KEY, preSignature);

header.put("signature", signature);

System.out.println("signature：" + signature);
```

```
        String keyReqResult = HttpUtils.post("获取 SM4 密钥信息", "https://ip:port/api/crypto/getKMInfo",
null, header);

        //使用 SM2 解密密钥密文

        String decryptResult = SmUtil.sm2(SM2_PRIVATE_KEY, null).decryptStr(keyReqResult,
KeyType.PrivateKey);

        SecretKeyResponse response = JSONObject.parseObject(decryptResult, SecretKeyResponse.class);

        // SM4 密钥

        String sm4Key = response.getCryptoInfo();

        //  秘钥版本号

        String cryptoVersion = response.getCryptoVersion();

        System.out.println("----------  ↑请求获取 SM4 数字信封↑ ----------");



        System.out.println("----------  ↓请求安诊儿平台上传数据集↓ ----------");



        //  请求对象 json

        JSONObject businessParam = new JSONObject();

        businessParam.put("tyshxydm", medOrgCode);

        businessParam.put("yqdm", medHosCode);

        businessParam.put("sfzjhm", "sfzjhm");
```

```
businessParam.put("...", "...省略");

// 业务请求体 json 字符串

String jsonString = businessParam.toJSONString();

String bizRequestEncrypt = sm4Encrypt(jsonString, sm4Key);

System.out.println("sm4 加密后字符串：" + bizRequestEncrypt);

JSONObject bizRequest = new JSONObject();

bizRequest.put("bizRequest", bizRequestEncrypt);

String bizRequestJSONString = bizRequest.toJSONString();




// 数据上传头部参数

HashMap<String, String> datasetUploadHeader = new HashMap<>();

// 交易编码：查询

datasetUploadHeader.put("tradeCode", INSERT_TRADE_CODE);

// 数据集编码

datasetUploadHeader.put("datasetCode", YYJL_DATASET_CODE);

// 请求唯一标识 随机数

datasetUploadHeader.put("requestId", UUID.randomUUID().toString());
```

```
// 医疗机构统一社会信用代码

datasetUploadHeader.put("medOrgCode", medOrgCode);

// 院区编码

datasetUploadHeader.put("medHosCode", medHosCode);

// 区域平台归属单位的 18 位统一社会信用代码 非区域平台接入时不可填入

datasetUploadHeader.put("platformCode", platformCode);

// 请求时间戳（13 位毫秒级，例：1735704000000）

datasetUploadHeader.put("timestamp", String.valueOf(System.currentTimeMillis()));



// 预签名

String datasetUploadPreSignature = buildPreSignature(header);

System.out.println("预签名字符串：" + datasetUploadPreSignature);

// 签名 （sm2 私钥,预签名内容）

String datasetUploadSignature = sm2Sign(SM2_PRIVATE_KEY, datasetUploadPreSignature);

datasetUploadHeader.put("signature", datasetUploadSignature);

// cryptoVersion 不参与签名

datasetUploadHeader.put("cryptoVersion", cryptoVersion);
```

```
        //调用平台数据服务

        String dataTransmissionResult = HttpUtils.post("调用平台数据服务",
"https://ip:port/api/his/data/transmission", bizRequestJSONString, datasetUploadHeader);



        JSONObject bizResponse = JSON.parseObject(dataTransmissionResult);

        String dataTransmissionResultBizResponse = bizResponse.getString("bizResponse");



        // 解密

        String transmissionResult = sm4Decrypt(dataTransmissionResultBizResponse, sm4Key);

        System.out.println("sm4 解密后字符串：" + transmissionResult);

    }



    /**

     * 返回密钥信息

     */

    public static class SecretKeyResponse implements Serializable {
```

```
private String cryptoInfo;

private String cryptoVersion;



public String getCryptoInfo() {

    return cryptoInfo;

}



public void setCryptoInfo(String cryptoInfo) {

    this.cryptoInfo = cryptoInfo;

}



public String getCryptoVersion() {

    return cryptoVersion;

}



public void setCryptoVersion(String cryptoVersion) {

    this.cryptoVersion = cryptoVersion;

}
```

```
}



/**
 * SM2 私钥签名
 *
 * @param privateKey  私钥
 * @param content       待签名内容
 * @return  签名值
 */
public static String sm2Sign(String privateKey, String content) {
        SM2 sm2 = new SM2(privateKey, null);
        return sm2.signHexFromHex(HexUtil.encodeHexStr(content));
}



/**
 * SM2 公钥验签
 *
 * @param publicKey  公钥
```

```
     * @param content      原始内容

     * @param sign         签名

     * @return  验签结果

     */

    public static boolean sm2Verify(String publicKey, String content, String sign) {

        SM2 sm2 = new SM2(null, publicKey);

        return sm2.verifyHex(HexUtil.encodeHexStr(content), sign);

    }



    /**

     * SM4 加密

     *

     * @param content  原文

     * @param key         SM4 秘钥

     * @return

     */

    public static String sm4Encrypt(String content, String key) {

        byte[] sm4KeyByte = HexUtil.decodeHex(key);
```

```
        String encryptBody = SmUtil.sm4(sm4KeyByte).encryptHex(content);

        return encryptBody;

    }



    /**

     * SM4 解密

     *

     * @param content SM4 加密字符串

     * @param key         SM4 秘钥

     * @return

     */

    public static String sm4Decrypt(String content, String key) {

        byte[] sm4KeyByte = HexUtil.decodeHex(key);

        String decryptBody = SmUtil.sm4(sm4KeyByte).decryptStr(content);

        return decryptBody;

    }



    /**
```

```
     * 生成预签名字符串
     *
     * @param header
     * @return
     */
    public static String buildPreSignature(HashMap<String, String> header) {
        if (MapUtils.isEmpty(header)) {
            return null;
        }
        List<String> keys = new ArrayList<>(header.keySet());
        Collections.sort(keys); // 按 ASCII 升序排序

        StringBuilder sb = new StringBuilder();
        for (String key : keys) {
            Object value = header.get(key);
            // 处理 null 和不同类型值
            String valueStr = (value != null) ? value.toString() : "";
```

```
                if (sb.length() > 0) {

                    sb.append("&");

                }

                sb.append(key).append("=").append(valueStr);

            }

            return sb.toString();

        }

}
```

4.1.6.5.2 maven 依赖

```
<dependency>
   <groupId>cn.hutool</groupId>
   <artifactId>hutool-crypto</artifactId>
   <version>5.8.35</version>
</dependency>
```

4.1.7 响应代码

常见响应结果代码见表 10。

表 9 响应结果代码表

响应结果代码	**响应结果含义**	**备注**
1	执行成功	
1000	参数为空	需校对信息后重新上传

1001	参数错误	需校对信息后重新上传
1002	非法参数	需校对信息后重新上传
2000	系统错误	
5001	交易代码不存在	需校对信息后重新上传
5002	数据集代码不存在	需校对信息后重新上传
5003	机构信息不存在	需校对信息后重新上传

4.1.8 请求示例

4.1.8.1 请求参数

请求头

```
Content-Type=application/json;charset=UTF-8

tradeCode=1001

datasetCode=HDSD.YYJL

requestId=7egs53hfe7r43nf3438f3h83fsnj987f

platformCode=123456789098765432

medOrgCode=123456789098765432

medHosCode=01

timestamp=1739979939220

cryptoVersion=1001
```

请求体明文

```
t{
```

```
  "requestBiz":{
    "tyshxydm": "统一社会信用代码",
    "yljgdm": "医疗机构代码",
    "yljgmc": "医疗机构名称",
    "yqdm": "院区代码",
    "yqmc": "院区名称",
    "sfzjlbdm": "身份证件类别代码",
    "sfzjhm": "身份证件号码",
    "cfbh": "处方编号",
    "djfsdm": "代煎方式代码",
    "djdwdm": "代煎单位代码",
    "djdwmc": "代煎单位名称",
    "cfjsrqsj": "20230101000000",
    "scbz": "删除标志"
  }
}
```


请求体加密后


```
{  "requestBiz":"26bc60d3436abc4ef527590a01cad8ddab5bfe1dfe75b335b84425bff2d584ac4ebe93b15d021e450
5c3e58a728762937ba52b6af114135c14980fba93e0698ddc66979e6f86a9e7301cdf612f52a72b13a9c53a4bc1cc557
2f15c3dcf6f612763e5adb2d29412690e1dc2d299d738a504666d86abfac35aaba9ff5714f9636020b1f1e71fc6227677
c4703cbc3afe52f20313eeb6be9a23b4b8891bbcb42dba563c3b57c10a0d418a967e513e0169ab4a9eaa368c6c2e806
bfdbf93209e24662068f8acee9642892cbb0e72bc3660949363f3eb8f4bff7fdc7661de4aae99d0895b9bf6aee25d14d9
3fbb5dd2998607d9acfa4e256d327a005ba556b8f2a5549c4b9ce33402c89d3b955fda4bb6faf558cfc3d5dabc9a48b0
eb37d0ee56f548a9afadcdaeb760ce094eb1c2c89a9d88c075003013879ee7c4503760218f314bbc8f1196aa4c823ac0
```

0d0464bc9594250cb8e863e7fa646b394c48170ff909cec070475ecaa0c4ad5318abdfb46d5b66b8e0da72496f567232dcff56423db9ea47f71cf81eba43eed5304b7f7be490b9fc12fbdda84369f9e18a237007cb3e07afe6af2a0b7f5abe6e6ff6e38452966cf110fcc0305da8c09298902916487907a0f2b567a59fd21588423b6266e29b4f82f15de2d845016f76c77b43649382c600edea185a43baf2becb1ea497905742"}

4.1.8.2 响应参数

响应参数密文


```
{
"success":true,
"errCode":"1",
"errMsg":"执行成功",
"responseBiz":"ba805ae8736f3d14c3ffd16f133b56be58cec614aee0e0827492ef66531568f52f69dd917132db36821d924b9b008f4d1874f89f427049c9af5423d11966023e65a1ef5fffdf7a2521acb1fde098a607"
}
```


响应参数解密后


```
{
"success":true,
"errCode":"1",
"errMsg":"执行成功",
"responseBiz":{
  "cfbh":"2022082066316802",
  "loadTime":"20220101120809"
  }
}
```


4.2 服务接口

医疗卫生机构开放标准接口，安诊儿服务平台等省级业务服务平台调用相关接口，实现就医陪诊中涉及的在线取号、排队叫号、费用查询、检查预约、在线缴费、报告查询等服务功能接入。

4.2.1 服务接口分类

云陪诊数字人服务接分类见表 11。

表 10 服务接口分类

服务类别	交易触发点	服务提供方	服务调用方
在线取号	业务交易，实时	医疗卫生机构	浙江省在线取号平台
排队叫号	业务交易，实时	医疗卫生机构	浙江省排队叫号平台
预约挂号	业务交易，实时	医疗卫生机构	浙江省预约挂号平台
费用查询	业务交易，实时	医疗卫生机构	安诊儿服务平台
检查预约	业务交易，实时	医疗卫生机构	安诊儿服务平台
在线缴费	业务交易，实时	医疗卫生机构	安诊儿服务平台
报告查询	业务交易，实时	医疗卫生机构	安诊儿服务平台

4.2.2 接口对接方式

服务调用方为浙江省在线取号平台、浙江省排队叫号平台、浙江省预约挂号平台等区域统建卫生健康便民服务平台，各源端服务提供方与对应服务平台对接，通过安诊儿服务专线接入服务平台。

服务调用方为安诊儿服务平台，则由源端医疗卫生机构提供标准接口，开放安诊儿服务平台调用。

5 值域代码

5.1 CVX-SFZJLBDM 身份证件类别代码

身份证件类别代码规定了公民个人身份证件的代码。

采用 2 位数字顺序代码，从“01”开始编码，按升序排列。见表 2。

表 11 CVX-SFZJLBDM 身份证件类别代码表

值	值含义	说明
01	居民身份证	
02	居民户口簿	
03	护照	
04	军官证	

05	驾驶证	
06	港澳居民来往内地通行证	
07	台湾居民来往内地通行证	
08	临时居民身份证	
09	武装警察身份证件	
10	中华人民共和国港澳居民居住证	
11	中华人民共和国台湾居民居住证	
12	社会保障卡	
98	中华人民共和国外国人永久居留身份证	
99	其他法定有效证件	

5.2 CVX-GHFSDM 挂号方式代码

表 12　CVX-GHFSDM 挂号方式代码表

值	值含义	说明
1	窗口挂号	
2	自助机挂号	
3	电话挂号	
4	网络挂号	包括浙里办、公众号、小程序、生活号、APP、门户网站、挂号平台等
5	诊间挂号	
9	其他	

5.3 CVX_RECIPETYPE 处方类型代码

表 13 CVX_RECIPETYPE 处方方式代码表

值	值含义	说明
1	西药	
2	成药	

3	草药	
4	非药物处方	

5.4 CV0209_01 医疗支付方式代码字典

表 14 CV0209_01 医疗支付方式代码字典

值	值含义	说明
01	社会基本医疗保险	
02	商业医疗保险	
03	大病统筹	
04	新型农村合作医疗	
05	城镇居民基本医疗保险	
06	公费医疗	
07	城镇职工基本医疗保险	
08	贫困救助	
09	全自费	
10	其他社会保险	
99	其他	