

自治区检查检验互通共享平台

接口文档 V1.4

（检验、检查结果数据上报）

内蒙古自治区卫生健康委员会

2026 年 3 月

修订说明

版本号	修订日期	修订说明
V1.0	2026-01-14	1.1 与 1.2 入参 datas 下增加 cityCode 与 cityName
V1.1	2026-01-26	删除原 1.2 质控后数据上报接口，1.2 更改为检验数据作废接口 1.1 新增入参 labMoney
V1.2	2026-01-28	1.1 新增入参 criticalValue 危急值
V1.3	2026-02-06	1.1 新增入参 recognFlag 互认标识
	2026-02-27	细化了对 datasign 的说明,拼接后的报告 id 的 key 为 reportIds。即对拼接后的 reportsIds 进行签名。
v1.4	2026-03-13	1、更新了加密算法的工具类代码和用例参考； 2、增加检查类数据上报和作废

1 检验数据

1.1 检验数据上报接口

1.1.1 接口内容说明

接口地址	upload/labScopeReport/insertSourceLabReport				
参数类型	参数	参数名称	必填	类型	说明
输入	requestId	请求标识	Y	String	36 位随机数，保证每次请求唯一
	appId	对接方编码	Y	String	对接方编码
	once	时间戳	Y	String	时间戳
	encrypt	内容是否加密	Y	Boolean	要求统一加密传输设置为 true 加密算法采用国密算法 sm2
	sign	签验串	Y	String	对 requestId、appId、once 字段 JSON 进行签名，签名算法参考附录 1-1
	content	入参对象	Y	content	“输入”中的 content 对象见以下定义
	dataSign	数据签名	Y	String	为保证双方对接数据一致性，需对入参对象中的报告 id 依次字符串拼接后进行签名，报告 id 拼接后的 key 为 reportIds 签名算法参考附录 1-1
输出	status	服务状态码	Y	String	服务状态码，参考附录 2-1
	code	返回码	Y	Integer	服务返回码，参考附录 2-2
	message	异常描述	Y	String	描述信息

“输入”中的 content 对象说明

参数	说明	必填	类型	备注
content	JSONObject 对象，包含以下字段			
reportTime	上报时间，格式：	Y	String	

	yyyy-MM-dd HH:mm:ss			
itemNum	上报数据条数	Y	Integer	
itemType	数据类型	Y	String	参考附录 2-10
datas	jsonArray 数组，内容为上报的检验数据，包含以下字段			
cityCode	盟市编码	Y	string	参考值域规范-盟市编码
cityName	盟市名称	Y	string	参考值域规范-盟市编码
orgCode	机构编码	Y	String	参考值域规范-机构编码；盟市互认平台使用-市级卫健委编码
orgName	机构名称	Y	String	参考值域规范-机构编码；盟市互认平台使用-市级卫健委编码
name	患者姓名	Y	String	
idCardType	证件类型	Y	String	参考附录 2-4
idCardNo	证件号码	Y	String	
gender	性别	Y	String	参考附录 2-3
age	年龄	Y	String	
medicalRecordNumber	病历号	Y	String	
wardNumber	病区	N	String	
bedNumber	床号	N	String	
patientType	病人类型	N	String	参考附录 2-5
diagnosisCode	诊断编码	N	String	
diagnosisName	诊断名称	N	String	
reportId	报告 id	Y	String	院内唯一标识
reportFormNo	报告单号	Y	String	
reportFormTitle	报告单名称	Y	String	
reportTime	报告时间，格式：yyyy-MM-dd HH:mm:ss	Y	String	
applicationDept	申请科室	Y	String	参考值域规范-科室分类与代码
applicant	申请医生	Y	String	
labCode	项目编码	Y	String	
labName	项目名称	Y	String	
labMoney	项目费用	Y	Integer	精确至个位数

laboratoryName	实验室名称	Y	String	
executiveDept	执行科室	Y	String	参考值域规范-科室分类与代码
executor	检验医生	Y	String	
labInstrument	检验仪器	Y	String	
auditor	审核医生	Y	String	
specimenCategory	标本类别	Y	String	参考附录 2-6
specimenNo	标本号	Y	String	
barCode	条码号	Y	String	
detectionNo	检测号	Y	String	
address	实验室地址	Y	String	
phone	实验室联系电话	Y	String	
acquisitionTime	采集时间，格式： yyyy-MM-dd HH:mm:ss	Y	String	
acceptTime	接收时间，格式： yyyy-MM-dd HH:mm:ss	Y	String	
createTime	添加时间，格式： yyyy-MM-dd HH:mm:ss	Y	String	
criticalValue	危急值	N	String	
details	josnArray 数组，内容为检验项目明细，包含以下蓝色字体字段	Y	String	
labDetailCode	细项编码	Y	String	参考值域规范-基本医疗服务项目
multiRowValue	多行参考值	Y	String	
labDetailName	细项名称	Y	String	参考值域规范-基本医疗服务项目
abbreviation	医学专业缩写，例如血小板 PLT	Y	String	
testMethod	检验方法	Y	String	参考附录 2-7
result	检验结果	Y	String	
unit	参考值和检验结果的单位	Y	String	
resultNormal	结果与参考值的比较结果	Y	String	参考附录 2-8
recognFlag	互认标识	Y	String	0：非互认项目 1：是互认项目

1.1.2 入参示例

{

```
"requestId": "f9b0e360854c44e3bef36422807699a0",
"appId": "3701020020000",
"once": "234324234234232",
"sign": "85B6EB77B1109867A00B0C2CC2845FF0AC5363F93C0C",
"dataSign": "85B6EB77B1109867A00B0C2CC2845FF0AC5363F93C0C",
"encrypt": true,
"content": {
  "itemNum": "上报数据条数",
  "itemType": "数据类型 ",
  "reportTime": "报送时间 ",
  "datas": [{
    "cityCode": "盟市编码",
    "cityName": "盟市名称",
    "orgCode": "机构编码",
    "orgName": "机构名称",
    "name": "姓名",
    "idCardType": "01",
    "idCardNo": "370102196704243326",
    "gender": "1",
    "age": "16",
    "medicalRecordNumber": "病历号",
    "wardNumber": "病区",
    "bedNumber": "床号",
    "patientType": "1",
    "diagnosisCode": "诊断编码",
    "diagnosisName": "诊断名称",
    "reportId": "报告 id",
    "reportFormNo": "检验报告单号",
    "reportFormTitle": "报告单名称",
    "reportTime": "2025-07-10 08:30:00",
```

```
"applicationDept": "申请科室内科",
"applicant": "申请医生",
"labCode": "项目编码",
"labName": "项目名称",
"labMoney": "项目费用",
"laboratoryName": "实验室名称",
"executiveDept": "执行科室",
"executor": "检验张三",
"auditor": "审核医生",
"specimenCategory": "1",
"specimenNo": "标本号",
"barCode": "条码号",
"detectionNo": "检测号",
"address": "实验室地址",
"phone": "实验室联系电话",
"acquisitionTime": "2025-07-10 08:30:00",
"acceptTime": "2025-07-10 08:30:00",
"createTime": "2025-07-10 08:30:00",
"criticalValue": "危急值"
"details": [{
    "multiRowValue": "",
    "labDetailCode": "细项编码",
    "labDetailName": "细项名称",
    "abbreviation": "医学专业缩写",
    "testMethod": "检验方法",
    "result": "检验结果",
    "unit": "参考值和检验结果的单位",
    "resultNormal": "1",
    "recognFlag": "0"
}]备注： “content 包含内容应为加密，此处为展现结构明文展示”
```

```
    ]]  
  }  
}
```

1.1.3 出参示例

1. 上报接口调用成功后返回样例。

```
{  
  "status": "SUCCESS",  
  "code": 200,  
  "message": "",  
}
```

2. 上报接口调用失败样例

调用失败的原因，请参考附录 1-2.

```
{  
  "status": "FAILURE",  
  "code": 10001,  
  "message": "签验失败",  
}
```

1.2 检验数据作废接口

1.2.1 接口内容说明

接口地址	upload/labScopeReport/cancelLabsReport				
参数类型	参数	参数名称	必填	类型	说明
输入	requestId	请求标识	Y	String	36 位随机数，保证每次请求唯一
	appId	对接方编码	Y	String	对接方编码
	once	时间戳	Y	String	时间戳
	encrypt	内容是否加密	Y	Boolean	要求统一加密传输 设置为 true 加密算法采用国密算

					法 sm2
	sign	签验串	Y	String	对 requestId、appId、once 字段 JSON 进行签名，签名算法参考附录 1-1
	content	入参对象	Y	Content	“输入”中的 content 对象见以下定义
	dataSign	数据签名	Y	String	为保证双方对接数据一致性，需对入参对象中的报告 id 依次字符串拼接后进行签名，报告 id 拼接后的 key 为 reportIds 签名算法参考附录 1-1
输出	status	服务状态码	Y	String	服务状态码，参考附录 2-1
	code	服务返回码	Y	Integer	服务返回码，参考附录 2-2
	message	异常描述	Y	String	描述信息

“输入”中的 content 对象说明

参数	说明	必填	类型	备注
content	JSONObject 对象，包含以下字段			
reportTime	上报时间，格式：yyyy-MM-dd HH:mm:ss	Y	String	
itemNum	作废数据条数	Y	Integer	
itemType	数据类型	Y	String	参考附录 2-10
datas	jsonArray 对象，内容为作废的检验数据，包含以下字段			
orgCode	机构编码	Y	String	参考值域规范-机构编码
reportId	检验报告 id	Y	String	院内唯一标识
labCode	项目编码	Y	String	

1.2.2 入参示例

```
{
  "requestId": "f9b0e360854c44e3bef36422807699a0",
  "appId": "3701020020000",
  "once": "234324234234232",
```

```
"sign": "85B6EB77B1109867A00B0C2CC2845FF0AC5363F93C0C",
"dataSign": "85B6EB77B1109867A00B0C2CC2845FF0AC5363F93C0C",
"encrypt": false,
"content": {
    "itemNum": "作废数据条数",
    "itemType": "数据类型 ",
    "datas": [{
        "orgCode": "机构编码",
        "reportId": "检验报告 id",
        "labCode": "检验类别编码",
    }]
}
```

1.2.3 出参示例

1 作废接口调用成功后返回样例

```
{
    "status": "SUCCESS",
    "code": 200,
    "message": "",
}
```

2 失败样例

调用失败的原因，请参考附录 1-2.

```
{
    "status": "FAILURE",
    "code": 10001,
    "message": "签验失败",
}
```

2 检查数据

2.1 检查数据上传接口

2.1.1 接口内容说明

接口地址	upload/examScopeReport/insertExamReport				
参数类型	参数	参数名称	必填	类型	说明
输入	requestId	请求标识	Y	String	36 位随机数，保证每次请求唯一
	appId	对接机构编码	Y	String	机构编码
	once	时间戳	Y	String	时间戳
	encrypt	内容是否加密	Y	Boolean	要求统一加密传输设置为 true
	sign	签验串	Y	String	对 appId、once 字段 JSON 进行签名，签名算法参考附录 1-1
	dataSign	数据签名	Y	String	为保证双方对接数据一致性，需对入参对象中的报告 id 依次字符串拼接后进行签名，报告 id 拼接后的 key 为 reportIds 签名算法参考附录 1-1
	content	入参对象	Y	content	“输入”中的 content 对象见以下定义
输出	status	服务状态码	Y	String	服务状态码，参考附录 2-1
	code	服务返回码	Y	Integer	服务返回码，参考附录 2-2
	message	异常描述	Y	String	描述信息

“输入”中的 content 对象的含义

参数	说明	必填	类型	备注
content	JSONObject 对象，包含以下字段			
reportTime	上报时间，格式: yyyy-MM-dd HH:mm:ss	Y	String	
itemNum	上报数据条数	Y	Integer	

itemType	数据类型	Y	String	参考附录 2-10
datas	jsonArray 数组, 包含以下字段			
cityCode	盟市编码	Y	string	参考值域规范-盟市编码
cityName	盟市名称	Y	string	参考值域规范-盟市编码
orgCode	医疗机构编码	Y	String	参考值域规范-机构编码; 盟市互认平台使用-市级卫健委编码
orgName	机构名称	Y	String	参考值域规范-机构编码; 盟市互认平台使用-市级卫健委编码
name	患者姓名	Y	String	
idCardType	证件类型	Y	String	参考附录 2-4
idCardNo	证件号码	Y	String	
gender	性别	Y	String	参考附录 2-3
age	年龄	Y	String	
medicalRecordNumber	病历号	Y	String	
wardNumber	病区	N	String	
bedNumber	床号	N	String	
patientType	病人类型	Y	String	参考附录 2-5
diagnosisCode	诊断编码	Y	String	
diagnosisName	诊断名称	Y	String	
reportId	报告 id	Y	String	院内唯一标识
reportFormNo	影像号(全球影像唯一标识, studyUid)	Y	String	
reportFormTitle	报告单名称	Y	String	
applicationDept	申请科室	Y	String	
applicant	申请医生	Y	String	
laboratoryName	实验室名称	Y	String	
executiveDept	执行科室	Y	String	
checkNo	检查号	Y	String	
examMode	项目模态	Y	String	参考附录 2-9
examCode	项目编码	Y	String	参考值域规范-基本医疗服务项目
examName	项目名称	Y	String	参考值域规范-基本医疗服务项目
examMoney	检查价格	Y	String	
checkTime	检查时间, 格式: yyyy-MM-dd	Y	String	

	HH:mm:ss			
executor	检查医生	Y	String	
reportorName	报告医生	Y	String	
reportTime	报告时间，格式： yyyy-MM-dd HH:mm:ss	Y	String	
auditor	审核医生	Y	String	
auditTime	审核时间，格式 yyyy-MM-dd HH:mm:ss	Y	String	
createTime	创建时间	Y	String	
criticalValue	危急值	N	String	
details	josnArray 数组， 内容为检验项目 明细，包含以下 蓝色字体字段			
examMethod	检查方法	Y	String	
imgAddress	影像地址	N	String	
pacsImage	影像所见	Y	String	
parts	部位	Y	String	
result	检查结果	Y	String	
conclusion	检查结论（详细）	Y	String	

2.1.2 入参示例

```
{
  "requestId": "f9b0e360854c44e3bef36422807699a0",
  "appId": "3701020020000",
  "once": "234324234234232",
  "sign": "85B6EB77B1109867A00B0C2CC2845FF0AC5363F93C0CCCA350",
  "dataSign": "85B6EB77B1109867A00B0C2CC2845FF0AC5363F93C0C",
  "encrypt": false,
  "content": {
    "itemNum": "上报数据条数",
```

```
"itemType": "数据类型 ",
"reportTime": "上报时间 ",
"datas": [{
  "cityCode": "盟市编码",
  "cityName": "盟市名称",
  "orgCode": "机构编码",
  "orgName": "机构名称",
  "name": "user1",
  "idCardType": "01",
  "idCardNo": "370102196704243326",
  "gender": "1",
  "age": "16",
  "medicalRecordNumber": "0102",
  "wardNumber": "01-01",
  "bedNumber": "01-02",
  "patientType": "1",
  "diagnosisCode": "dia01",
  "diagnosisName": "诊断 01",
  "reportId": "20230713-01",
  "reportFormNo": "2023071201",
  "reportFormTitle": "370102196704243326",
  "applicationDept": "内科",
  "applicant": "applicantDoctor01",
  "laboratoryName": "370102196704243326",
  "executiveDept": "370102196704243326",
  "checkNo": "1",
  "examMode": "XT",
  "checkTime": "2023-08-01 08:30:00",
  "examCode": "00100",
  "examName": "",
```

```
"examMoney": "检查价格",
"executor": "03",
"reportorName": "1",
"reportTime": "2023-08-01 08:30:00",
"auditor": "1",
"auditTime": "2023-08-01 08:30:00",
"createTime": "2023-08-01 08:30:00",
"criticalValue": "危急值"
"details": [{
    "pacsImage": "1",
    "result": "1",
    "conclusion": "12",
    "parts": "01",
    "examMethod": "02",
    "imgAddress": "url",

  ]}]
}
```

2.1.3 出参示例

3. 1 上报接口调用成功后返回样例。

```
{
  "status": "SUCCESS",
  "code": 200,
  "message": "",
}
```

4. 上报接口调用失败样例

调用失败的原因，请参考附录 2-2.

```

{
    "status": "FAILURE",
    "code": 10001,
    "message": "签验失败",
}

```

2.2 检查数据作废撤销接口

2.2.1 接口内容说明

接口地址	upload/examScopeReport/cancelExamScopeReport				
参数类型	参数	参数名称	必填	类型	说明
输入	requestId	请求标识	Y	String	36 位随机数，保证每次请求唯一
	appId	对接机构编码	Y	String	机构编码
	once	时间戳	Y	String	时间戳
	encrypt	内容是否加密	Y	Boolean	要求统一加密传输设置为 true
	sign	签验串	Y	String	对 requestId、appId、once 字段 JSON 进行签名，签名算法参考附录 1-1
	content	入参对象	Y	Content	“输入”中的 content 对象见以下定义
	dataSign	数据签名	Y	String	为保证双方对接数据一致性，需对入参对象中的报告 id 依次字符串拼接后进行签名，报告 id 拼接后的 key 为 reportIds 签名算法参考附录 1-1
输出	status	服务状态码	Y	String	服务状态码，参考附录 2-1
	code	服务返回码	Y	Integer	服务返回码，参考附录 2-2
	message	描述	Y	String	描述信息

	requestId	请求标识	Y	String	36 位随机数，保证每次请求唯一
	appId	对接机构编码	Y	String	机构编码
	once	时间戳	Y	String	时间戳
	encrypt	内容是否加密	Y	Boolean	是否对 content 参数加密，true：加密；false：不加密（注意都是小写，出现大写不认）
	sign	签验串	Y	String	对 requestId、appId、once 字段 JSON 进行签名，签名算法参考附录 2-1
	content	结果	Y		“输出”中的 content 对象见以下定义

“输入”中的 content 参数说明

参数	说明	必填	类型	备注
content	JSONObject 对象，包含以下字段			
reportTime	上报时间，格式：yyyy-MM-dd HH:mm:ss	Y	String	
itemNum	作废数据条数	Y	Integer	
itemType	数据类型	Y	String	参考附录 2-10
datas	JSONArray 数组，内容为作废的数据集合，包含以下字段			
orgCode	医疗机构编码	Y	String	
reportId	检查报告 id	Y	String	院内唯一标识
examCode	检查编码	Y	String	参考值域规范-基本医疗服务项目

2.2.2 入参示例：

```
{
  "requestId": "f9b0e360854c44e3bef36422807699a0",
  "appId": "3701020020000",
  "once": "234324234234232",
```

```

"sign": "85B6EB77B1109867A00B0C2CC2845FF0AC536",
"encrypt": false,
"dataSign": "85B6EB77B1109867A00B0C2CC2845FF0AC5363F93C0C",
"content": {
    "itemNum": "作废数据条数",
    "itemType": "数据类型 ",//检查 or 检验
    "reportTime": "报送时间 ",
    "datas": [{
        "reportId": "检查报告唯一标识",
        "examCode": "检查项目编码",
        "orgCode": "机构编码",
    }]
}
}

```

2.2.3 出参示例：

1 作废接口调用成功后返回样例

```

{
    "status": "SUCCESS",
    "code": 200,
    "message": "",
}

```

2 失败样例

调用失败的原因，请参考附录 1-2.

```

{
    "status": "FAILURE",
    "code": 10001,
    "message": "签验失败",
}

```

附录 1：签名和加密

1. 签名算法说明

所有接口均采用 MD5, 签名算法 requestId、appId、once 进行签名。

签名工具类见附录 1-3, 签名测试用例请参考附录 1-5。

2. 加密算法说明

检查检验数据属于隐私数据, 必须加密传输, 故 encrypt 必须设置为 true, 对 content 整体加密后进行传输, 加密算法采用 sm2 国密算法。

3. 签名工具类代码

```
public class SignatureVerificationUtil {
    private final static String SIGN_FIELD = "sign";
    private final static String SIGN_STR SIGN_KEY = "&sign_key=";
    private final static String CIPHERTEXT_FIELD = "ciphertext";
    private final static String DATA_FIELD = "data";
    private final static String NULL = "null";

    /**
     * 验证签名
     *
     * @param json    参数
     * @param secret 签名密钥
     * @return true false
     * @throws NoSuchAlgorithmException
     */
    public static boolean verify(String json, String secret) throws IOException,
        NoSuchAlgorithmException {
        JsonNode obj = JsonUtil.instance().readTree(json);
        String sign = obj.path(SIGN_FIELD).asText();
        String signTmp = sign(json, secret);
        if (signTmp.equalsIgnoreCase(sign)) {
            return true;
        }
    }
}
```

```

        return false;
    }

    public static String sign(String json, String secret) throws IOException,
        NoSuchAlgorithmException {
        JsonNode obj = JsonUtil.instance().readTree(json);
        SortedMap<String, String> map = getParamSimple(obj);
        map.remove(DATA_FIELD);
        StringBuilder sb = new StringBuilder();
        for (String key : map.keySet()) {
            if (SIGN_FIELD.equals(key)) {
                continue;
            }
            if (StringUtils.isBlank(map.get(key)) || NULL.equals(map.get(key))) {
                continue;
            }
            sb.append(key).append("=").append(map.get(key));
        }
        sb.append(SIGN_STR_SIGN_KEY).append(secret);
        return EncryptUtil.getMD5Str(sb.toString());
    }

    private static SortedMap<String, String> getParamSimple(JsonNode obj) {
        SortedMap<String, String> sortedMap = new TreeMap<>();
        for (Iterator<String> it = obj.fieldNames(); it.hasNext(); ) {
            String name = it.next();
            JsonNode node = obj.get(name);
            if (node.isArray()) {
                sortedMap.put(name, node.toString());
            } else {
                sortedMap.put(name, node.asText());
            }
        }
        return sortedMap;
    }
}

```

4. 加密工具类代码

```

/**
 * SM2 加密算法
 * @param pubKeyHexString 公钥
 * @param data            明文数据
 * @return
 */

```

```

private static String encryptC1C2C3(byte[] data, String pubKeyHexString) {
    // 获取一条 SM2 曲线参数
    X9ECParameters sm2ECParameters = GMNamedCurves.getByName("sm2p256v1");
    // 构造 ECC 算法参数，曲线方程、椭圆曲线 G 点、大整数 N
    ECDomainParameters domainParameters = new
ECDomainParameters(sm2ECParameters.getCurve(), sm2ECParameters.getG(),
sm2ECParameters.getN());
    //提取公钥点
    ECPublicKeyParameters publicKeyParameters = new
ECPublicKeyParameters(sm2ECParameters.getCurve().decodePoint(
Hex.decode(pubKeyHexString)), domainParameters);
    // 公钥前面的 02 或者 03 表示是压缩公钥，04 表示未压缩公钥，04 的时候，可以
    去掉前面的 04
    SM2Engine sm2Engine = new SM2Engine(SM2Engine.Mode.C1C2C3);
    // 设置 sm2 为加密模式
    sm2Engine.init(true, new ParametersWithRandom(publicKeyParameters, new
SecureRandom()));

    byte[] arrayOfBytes = null;
    try {
        arrayOfBytes = sm2Engine.processBlock(data, 0, data.length);
    } catch (Exception e) {
        e.printStackTrace();
        throw new RuntimeException("SM2 加密时出现异常: {} "+ e.getMessage());
    }
    return Hex.toHexString(arrayOfBytes);
}

/**
 * SM2 解密算法
 * @param priKeyHexString 私钥
 * @param cipherData 密文数据
 * @return
 */
private static String decryptC1C2C3(byte[] cipherData, String priKeyHexString) {
    //获取一条 SM2 曲线参数
    X9ECParameters sm2ECParameters = GMNamedCurves.getByName("sm2p256v1");
    //构造 domain 参数
    ECDomainParameters domainParameters = new
ECDomainParameters(sm2ECParameters.getCurve(), sm2ECParameters.getG(),
sm2ECParameters.getN());

```

```

        BigInteger privateKeyD = new BigInteger(priKeyHexString, 16);
        ECPrivateKeyParameters privateKeyParameters = new
ECPrivateKeyParameters(privateKeyD, domainParameters);
        //设置排列模式为 C1C2C3
        SM2Engine sm2Engine = new SM2Engine(SM2Engine.Mode.C1C2C3);
        // 设置 sm2 为解密模式
        sm2Engine.init(false, privateKeyParameters);

        String result = "";
        try {
            byte[] arrayOfBytes = sm2Engine.processBlock(cipherData, 0,
cipherData.length);
            return new String(arrayOfBytes, StandardCharsets.UTF_8);
        } catch (Exception e) {
            e.printStackTrace();
            throw new RuntimeException("SM2 解密时出现异常: {}" + e.getMessage());
        }
    }
}

```

5. 签名、加密测试用例参考代码

```

/**
 * 签名
 */
public void sign3() throws Exception {
    Request request = new Request<>();
    request.setRequestId(UUID.randomUUID().toString());
    request.setAppId("ceshi001");
    request.setOnce(System.currentTimeMillis()+"");
    SignatureVerificationUtil.sign(JsonUtil.toJson(request), "d2ebdd629fd446f2902cdf
a6819afd62605813050f8e4a4f9b84bc8599341dbd");
}

“d2ebdd629fd446f2902cdfa6819afd62605813050f8e4a4f9b84bc8599341dbd”为签名测试用
秘钥。

```

```

/**
 * 验签
 */
public void ver1() throws Exception{
    Request requestnew Request<>();
    request.setRequestId("02ac1c4c-78f0-47b7-9858-95415fe5111c");
}

```

```

request.setAppId("ceshi001");
request.setOnce("1718249126305");
request.setSign("b1652ff232e1b1a4eadb83d0137e8a1c");
SignatureVerificationUtil.verify(JsonUtil.toJson(request), "d2ebdd629fd446f2902c
    dfa6819afd62605813050f8e4a4f9b84bc8599341dbd");
}

```

```

/**
 *加密
 * */
@Test
    public void test2() throws Exception {
        String s = "加解密测试";
        String a =
SM2Base.encryptC1C2C3(s, "04d67cfca03aff1ac54b218c5a6a57b70bd5f9b59515f6ff0216c0
a1894948575ca5e512985f947e922c601fcf49926243afebe188c01c760a861ea6ff62e49532");
    }

```

```

        public static String encryptC1C2C3(String data, String pubKeyHexString) {
            if("").equals(data)) {
                //空字符串加密会导致死锁
                return "";
            }
            return encryptC1C2C3(data.getBytes(StandardCharsets.UTF_8),
pubKeyHexString);
        }

```

```

/**
 * 解密
 */
    public void test3() {
        String s =
"046e240b70eb3473ffffd8cf0d150151ad83bf30a26c46ceec2caad883a907734592b135f764d9
13ff45a3fddd15c840bcd16965ab09ea905b91b6652f702be184b89f4d21b9de2288e4f36847eed
352alf570b6a7bc450e780ed55e189c62d8059cb8c60958aad75839ef576a9ed967";
        String a =
SM2Base.decryptC1C2C3(s, "397913bb6c4d559f7b191b24ece101721a94073b959976b46f1aae
ae210cd4b3");
    }
    public static String decryptC1C2C3(String cipherData, String priKeyHexString) {
        if("").equals(cipherData)) {
            //空字符串直接返回
            return "";
        }
    }

```

```
// 使用 BC 库加解密时密文以 04 开头，传入的密文前面没有 04 则补上
//04 表示密文未进行压缩
if (!cipherData.startsWith("04")) {
    cipherData = "04" + cipherData;
}
return decryptC1C2C3(Hex.decode(cipherData), priKeyHexString);
}
```

此为原文“加解密测试”

此为加密后密文

046e240b70eb3473ffffd8cf0d150151ad83bf30a26c46ceec2caad883a907734592b135f764d91
3ff45a3fddd15c840bcd16965ab09ea905b91b6652f702be184b89f4d21b9de2288e4f36847eed3
52a1f570b6a7bc450e780ed55e189c62d8059cb8c60958aad75839ef576a9ed967

若解密后结果与原始明文完全一致，则证明加解密流程正确无误，密钥对匹配且算法实现规范。

解密密钥为“397913bb6c4d559f7b191b24ece101721a94073b959976b46f1aaeae210cd4b3”

附录 2：字典

1. 服务状态码

值（文本）	值含义
SUCCESS	服务成功
FAILURE	服务异常

2. 服务返回码

值（数值）	值含义
200	正常
10001	签验失败
10002	参数解密失败
10003	输入参数 (appId、once、sign) 缺失
10004	输入参数中的 APPID 未授权
10005	参数转 json 失败
500	程序错误，请联系我

3. 性别代码

值(数值)	值含义
1	男性
2	女性
9	未知

4. 证件类型代码

值（文本）	值含义
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
99	其他法定有效证件

5. 病人类型

值（文本）	值含义
1	门诊
2	急诊
3	住院
9	其他

6. 标本类别

值（文本）	值含义
1	血涂片
2	毛刷
3	胸腔积液
4	血液
5	胸水
6	腹水
7	针吸
8	网织红细胞
9	腹穿液
10	乳腺穿刺液
11	囊液
12	大便
13	尿液
14	分泌物
15	腹腔冲洗液
16	前列腺液
17	血清
18	血浆
19	尿
20	血
21	脑脊液
22	精液
23	浆膜腔积液
24	穿刺液
25	引流液
26	痰液

7. 检验方法

值（文本）	值含义
1	比色法
2	阻抗法
3	计算值
4	化学发光法
5	干化学法
6	酶比色法
7	过氧化氢酶清除法
8	选择性清除法
9	凝固法
10	电化学发光法

11	速率散射比浊法
12	色谱法

8. 结果与参考值的比较结果

值（文本）	值含义
0	正常
1	高
-1	低

9. 项目模态

值（文本）	值含义
1	胸透
2	心电
3	牙片
4	超声
5	CT
6	CR
7	DR
8	核磁共振
9	X 光
10	内窥镜
11	ECT
12	彩色多普勒
13	双多普勒
14	PET
15	数字胃肠
16	SPECT
17	雷射表面扫描
18	病理
19	放疗图像
20	其他
21	口腔全景
22	骨密度测定

10. 数据类型

值（文本）	值含义
1	检查
2	检验

11. 互认状态

值（文本）	值含义
0	不互认

1	互认
-1	未操作