IRP Proxy SDK

技术文档 (Java版)

版本号 1.0

PKU

DOA Group

2020.9

目录

1.介绍

IRP Proxy SDK是PKU_DOA框架中的Handle System功能组件，该组件提供了对IRP（Identifier/Resolution Protocol）协议所规定的Handle System Protocol系统进行基本的Handle解析、操作以及前缀管理等功能。IRP Proxy SDK是基于CNRI的Handle Client lib库对相关的功能接口进行了封装，该lib库为CNRI机构的Handle协议实现，我们在该lib库的基础上对DOA架构中的Handle System功能组件进行了开发，并提供对DONA下辖国内外多个MPA相关Handle系统的接入与兼容，包括CNRI(HDL)、中科院(DataPid)、中数等。用户(开发者)可以直接基于IRP Proxy SDK对自己所申请的Handle标识进行解析与管理，也可以通过DOA架构的Client端实现对DO相关的标识管理操作。

有关PKU_DOA框架的详细信息可以访问www.xxxxxx.com获取更多内容

更多问题请联系xxxxxx@pku.edu.cn

2.安装与部署

IRP Proxy SDK是一个支持Handle标识管理与操作的lib库，该组件目前可以对全球DONA下辖所有MPA(Multi-Primary Administrators)负责的Handle标识进行解析，并对基于CNRI提供的IRP协议原生LHS认证方式所管理的Handle进行创建、修改以及删除等操作。

IRP Proxy SDK使用方式十分简单，当前只提供Java版本，用户只需将下载好的doa_irp_proxy.jar加入到目标项目的本地lib库中，就可以使用IRP Proxy SDK对Handle系统进行访问。

IRP Proxy SDK源码下载地址

3.接口说明

IRP Proxy SDK提供了统一的Handle解析及操作调用接口，所有API接口均是基于CNRI的IRP协议lib库进行实现，接口的具体使用方法说明详见下文各小节，返回结果均为Json类型的String字符串，其中不同返回码对应的意义见下表1。

表1 返回码对应的方法调用结果

返回码值	表示结果	返回码值	表示结果
-1	SDK调用异常	400	未授权或管理员无效
1	调用成功	401	请求的Handle操作没有权限
2	一般错误	402	需要进行LHS管理员认证
3	LHS服务器未响应	403	LHS管理员认证失败
4	IRP协议错误（消息格式有误）	404	提供的证书或认证信息无效
5	请求操作码不支持	405	认证超时
6	请求所需LHS转发次数过多	406	认证时发生内部错误
7	LHS服务器只支持解析操作	500	会话超时
100	未找见所请求的Handle标识	501	无法建立会话
101	需创建的Handle标识已经存在	502	会话Key无效
102	请求中的Handle标识无效	503	会话交互需要RSAKEY
200	未找见所请求的Handle Value	504	无效的会议建立信息请求
201	需创建的Handle Value已经存在	505	重复建立会话请求拒绝
202	请求中的Handle Value格式有误		
300	SiteInfo（LHS服务器）信息失效		
301	目标Handle不属于所请求的LHS服务器		
302	需要进行Handle请求转发		
303	需要进行命名授权（前缀）代理		

下面为listHandlesUnderNA接口成功调用后的返回结果格式:

```
{
    "NamingAuthority": "20.500.12750",
    "values": ["20.500.12750/TEST.98765", "20.500.12750/TEST.HELLODOA",
"20.500.12750/TEST_HANDLE", "20.500.12750/TEST_PKU_11"],
    "responseCode": 1
}
```

3.1 解析Handle

解析Handle接口提供对DONA下辖所有Handle System系统的Handle标识进行解析，其中handle为必填参数，handleType以及handleIndex为选填参数，表示只返回满足条件的HandeValue，如果不指定，则默认返回请求的Handle标识下所有handleValue数据,方法名称、请求参数以及调用方式如下：

String resolveHandle(String handle)

String resolveHandle(String handle, String[] handleType, int[] handleIndex)

- ***handle*** *待解析的Handle 标识*
- ***handleType*** *返回指定type 的HandleValue 数据，可一次指定多个*
- ***handleIndex*** *返回指定index 的HandleValue 数据，可一次指定多个*

```
//带参数handleTypey以及handleIndex的调用
String result1 = HandleOperator.resolveHandle("20.500.12750",new String[]
{"HS_PUBKEY"}, new int[]{1});
//只使用Handle标识进行调用
String result2 = HandleOperator.resolveHandle("21.86116.6/sciencedb.4");
```

注意：同时指定handleType参数与handleIndex参数时表示返回满足二者条件的并集而非交集，如上述调用示例的第一个表示返回Handle标识20.500.12750下 type 为 HS_PUBKEY 以及 index 为1的所有Handle Value，而并非返回type 为 HS_PUBKEY 且 index 为1的Handle Value。

3.2 创建Handle

创建Handle接口支持在Handle System系统中新建一个Handle标识，当前只支持和基于CNRI的IRP协议原生LHS认证方式配置部署的Handle Server进行交互，这种情况下SDK在创建一个新的Handle标识时会默认生成一条 HS_ADMIN 的HandleValue来存储当前Handle标识的管理员权限信息，表明管理员对当前新增Handle标识所拥有的权限，**系统默认该HandleValue的 index 值为300，管理员为认证私钥中的管理员Handle信息（一般为前缀对应的admin）,对应的权限默认全部开放**，具体的权限信息说明见IRP协议RFC3651 3.2.1小节,该方法名称、请求参数以及调用方式如下：

String createHandle(String handle, List<Map<String, String>> valuesToCreate)

- ***handle*** *创建的Handle 标识名称*

- ***valuesToCreate*** *创建的Handle标识中所存储的HandleValues数据，格式为map类型的List,需指定index,type以及data三个字段*

```
String LHSIPAddress = "47.102.45.68";
String prikeyPath = "F:/Project/admpriv.bin";
String adminHandleID = "0.NA/20.500.12750";
int adminIndex = 300;
//构建HandleValue数据
Map<String, String> handleValue = new HashMap<>();
handleValue.put("index","6000");
handleValue.put("type","test");
handleValue.put("data","test_data");
List<Map<String,String>> handleValues = new ArrayList<>();
handleValues.add(handleValue);

//初始化HandleOperator对象以及LHS认证信息
HandleOperator handleOperator = new
HandleOperator(LHSIPAddress,adminHandleID,adminIndex,prikeyPath);
//调用创建Handle方法
String result = handleOperator.createHandle("20.500.12750/test",handleValues);
```

注意：valuesToCreate参数可以为null，此种情况下将只创建对应的Handle标识以及默认的HS_ADMIN的HandleValue。

3.3 删除Handle

删除Handle接口提供对具有LHS管理权限的Handle标识进行删除操作，该方法会删除对应的Handle标识以及Handle下面的所有HandleValue数据，方法名称、请求参数以及调用方式如下：

String deleteHandle(String handle)

- ***handle*** *需要删除的 Handle 标识名称*

```
String result = handleOperator.deleteHandle("20.500.12750/test");
```

3.4 新增HandleValue

新增HandleValue接口提供对具有LHS管理权限的Handle标识新增HandleValue的操作，SDK提供新增单个或多个HandleValue的接口,方法名称、请求参数以及调用方式如下：

String addHandleValue(String handle, Map<String, String> valueToAdd)

String addHandleValue(String handle, List<Map<String, String>> valuesToAdd)

- ***handle*** *对应的 Handle 标识名称*
- ***valueToAdd*** *新增的HandleValues数据，格式为map*
- ***valuesToAdd*** *新增的HandleValues数据，格式为map类型的List*

```
Map<String, String> handleValue2 = new HashMap<>();
handleValue2.put("index","7000");
handleValue2.put("type","werwerwe");
handleValue2.put("data","werwer");
List<Map<String,String>> handleValues = new ArrayList<>();
handleValues.add(handleValue);
handleValues.add(handleValue2);
//新增单个HandleValue
String result1 =
handleOperator.modifyHandleValue("20.500.12750/test",handleValue);
//新增多个HandleValue
String result2 =
handleOperator.modifyHandleValue("20.500.12750/test",handleValues);
```

3.5 修改HandleValue

修改HandleValue接口提供对具有LHS管理权限的Handle标识下的HandleValue进行更新修改操作，SDK提供修改单个或多个HandleValue的接口,方法名称、请求参数以及调用方式如下：

String modifyHandleValue(String handle, Map<String, String> valueToModify)

String modifyHandleValue(String handle, List<Map<String, String>> valuesToModify)

- ***handle*** *对应的Handle 标识名称*
- ***valueToModify*** *修改的HandleValues数据，格式为map*
- ***valuesToModify*** *修改的HandleValues数据，格式为map类型的List*

```
Map<String, String> handleValue2 = new HashMap<>();
handleValue2.put("index","7000");
handleValue2.put("type","change");
handleValue2.put("data","change_test");
List<Map<String,String>> handleValues = new ArrayList<>();
handleValues.add(handleValue);
handleValues.add(handleValue2);
//修改单个HandleValue
String result1 =
handleOperator.modifyHandleValue("20.500.12750/test",handleValue);
//修改多个HandleValue
String result2 =
handleOperator.modifyHandleValue("20.500.12750/test",handleValues);
```

3.6 移除HandleValue

移除HandleValue接口提供对具有LHS管理权限的Handle标识下指定的HandleValue数据进行移除操作，需保证所删除的HandleValue数据的index在Handle标识下存在，方法名称、请求参数以及调用方式如下：

String removeHandleValue(String handle,int handleValueIndex)

- ***handle*** *待删除的Handlevalue 所在Handle标识名称*
- ***handleValueIndex*** *待删除的HandleValue 对应的index值*

```
String result = handleOperator.removeHandleValue("20.500.12750/test",5000);
```

3.7 NA相关操作

NA操作接口提供对具有LHS管理权限的前缀进行操作，包括返回某个前缀下的所有子前缀以及返回某个前缀下的所有Handle标识，方法名称、请求参数以及调用方式如下：

String listHandlesUnderNA(String prefix)

String listNAs(String prefix)

- ***prefix*** *所请求的前缀名称*

```
String result1 = handleOperator.listHandlesUnderNA("20.500.12750");
String result2 = handleOperator.listNAs("20.500.12750");
```