OdooRPC数据封装调用文档

Made by zhujaidong May 2016

一.移动(Mobile)RPC调用总览

- IP:101.200.194.191
- port:6000

RPC程序用thrift框架封装接口，请使用thrift框架来调用接口。

二.接口说明

所有有返回值的接口返回的都是json化的对象，比如json.dumps（{'uid':uid}）
第一个接口列出了所有可能的返回值，下面的接口就只列出成功时的返回值，如果有不一样的会列出
1.移动登录验证接口

- interface：isLoginSuccess
- param_1："password"
- param_2: "username"
- return：

```
    - {'uid':uid}(成功)
    - {"faultcode": "Wrong username or password"}（失败,用户名或密码错误）
    - {"faultcode": "database " + DB + " does not exist"}（数据库错误）
    - {"faultcode": "AccessDenied"}
```

2.获取设备信息接口

- interface： getAssets
- param_1: "uid" isLoginSuccess的返回值
- param_2: "password
- return：

```
    - [{u'install_building_id': [1,u'\u4e09\u53f7\u9505\u7089\u7a7a\u5730\u533a\u57df'],
     u'id':3, u'name': u'\u6c28\u6c34\u8f93\u9001\u6cf5B'},...](成功)
```

3.获取属性信息

- interface： getAttributes
- param_1: "uid"
- param_2: "password"
- return：

```
    - [{u'asset_id': [2, u'\u6c28\u6c34\u8f93\u9001\u6cf5A'], u'id': 6,
      u'name':u'\u8f93\u9001\u6cf5\u8fd0\u884c\u72b6\u6001'},...](成功)
```

4.根据时间获取警告事件

- interface： getErrorAndEventByTime
- param_1: "uid"
- param_2: "password"
- param_3: "starttime"
- param_4: "endtime"
- return：

```
    - {u'faultcode': u'Warning and Event does not exist'}(不存在警告和事件)
```

5.根据时间获取维修记录

- interface: getRepairInfoByTime

- param_1: "uid"
- param_2: "password"
- param_3: "starttime"
- param_4: "endtime"
- return：

```
    - [{u'asset_id': [2, u'\u6c28\u6c34\u8f93\u9001\u6cf5A'], u'state':
      u'finished', u'finished_time': u'2016-05-24 11:12:52', u'id': 11, u'name': u'123'}]（获取维修记录）
    - {"faultcode": "Repair Info does not exist"}（维修记录不存在）
```

6.通过oid获取设备信息

- interface：getAssetByOid
- param_1: "oid" 整形
- param_2: "uid"
- param_3: "password"
- return：

```
    - {u'opc_server': False, u'code': u'Best-002', u'name':
      u'\u7a00\u91ca\u6c34\u6db2\u4f4d\u8ba1', u'factory_number': False,
      u'specification': False, u'oid': u'123', u'opc_client': False, u'control_points':
      [], u'buying_price': 0.0, u'brand': False, u'attributes': [46, 47], u'id': 11}（成功获取）
```

7.通过assetid获取采集点信息

- interface: getCollectPointsByAssetId
- param*1: "asset*id" 整形
- param_2: "uid"
- param_3: "password"
- return：

```
    - {'faultcode': u'Asset ID does not exist!'}（设备不存在）
```

8.通过oid获取采集点信息

- interface: getCollectPointsOid
- param_1: "oid"
- param_2: "uid"
- param_3: "password"
- return：

```
    - {"asset_collectpoints": [{"attribute_id": 108, "attribute_code": "CC003",
      "attribute_name": "\u5bb9\u5668\u6db2\u4f4d\u503c",
      "name": "\u7a00\u91ca\u6c34\u6db2\u4f4d\u503c"}, {"attribute_id": 109,
      "attribute_code": "CC004", "attribute_name":
      "\u7a00\u91ca\u6c34\u4f4e\u6db2\u4f4d\u8b66\u62a5", "name":
      "\u7a00\u91ca\u6c34\u4f4e\u6db2\u4f4d\u8b66\u62a5"}]}（成功）
    - {"faultcode": "Collect Point does not exist!"}(采集点不存在)
    - {"faultcode": "Oid or Attributes does not exist!"}（oid或属性不存在）
```

9.通过oid获取控制点信息

- interface: getControlPointByOid
- param_1: "oid"
- param_2: "uid"
- param_3: "password"
- return：

```
    - {"faultcode": "Control Point does not exist!"}（控制点不存在）
    - {"faultcode": "Oid or Control Points does not exist!"}（oid或者控制点不存在）
```

10.提供给3D图形的接口，用于根据oid查询信息

- interface： packMessage
- param_1: "oid"

- param_2: "uid"
- param_3: "password"
- return：

```
    - {"asset_oid": "51386", "asset_brand": false, "asset_collectpoints":
      [{"attribute_name": "\u8f93\u9001\u6cf5\u8fd0\u884c\u72b6\u6001",
      "attribute_code": "BP001", "attribute_id": 6, "name":
      "\u6cf5\u5de5\u4f5c\u72b6\u6001"}, {"attribute_name":
      "\u8f93\u9001\u6cf5\u7efc\u5408\u6545\u969c", "attribute_code":
      "BP002", "attribute_id": 7, "name":
      "\u6cf5\u8fd0\u884c\u7efc\u5408\u6545\u969c\u72b6\u6001"},
      {"attribute_name": "\u8f93\u9001\u6cf5\u53d8\u9891\u53cd\u9988",
      "attribute_code": "BP003", "attribute_id": 8, "name":
      "\u6cf5\u5b9e\u65f6\u9891\u7387"}, {"attribute_name":
      "\u8f93\u9001\u6cf5\u53d8\u9891\u8d8b\u52bf", "attribute_code": "BP004",
      "attribute_id": 9, "name": "\u6cf5\u53d8\u9891\u53d8\u5316\u8d8b\u52bf"},
      {"attribute_name": "\u8f93\u9001\u6cf5\u5de5\u4f5c\u7535\u6d41\u53cd\u9988", "attribute_code":
      "BP005", "attribute_id": 10, "name": "\u6cf5\u5de5\u4f5c\u7535\u6d41\u53cd\u9988"}],
      "asset_opc_client": false, "asset_specification": false,
      "asset_code": "BST-001", "asset_factory_number": false, "asset_name": "\u6c28\u6c34\u8f93\u9001\u6cf5A",
      "faultcode": "Control Point does not exist!", "asset_buying_price": 0.0,
      "asset_opc_server": false}（成功）
```

11.获取全部控制系统

- interface: getSystemControlInfo
- param_1: "uid"
- param_2: "password"
- return：

```
    - {"systemcontrol": [{"remark": "\u6c28\u6c34\u8f93\u9001\uff0c\u6c28\u6c34\u6d53
      \u5ea6\u8c03\u914d\u63a7\u5236\u3002", "id": 1, "name":
      "SNCR\u63a7\u5236\u7cfb\u7edf"}, {"remark": "\u5229\u7528\u50ac\u5316\u5242
      \u628a\u6c2e\u6c27\u5316\u7269\u4e0e\u6c28\u6c34\u53cd\u5e94\u3002", "id": 2,
      "name": "SCR\u8131\u9500\u7cfb\u7edf"}]}（成功）
    - {"faultcode": "System Control does not exist"}(控制系统不存在)
```

12.获取某个控制系统内的所有设备信息

- interface: getAsset2SystemControl
- param_1: "uid"
- param_2: "password"
- param_3: "systemcontrol_id"
- return：

```
    - {"asset2system": [{"oid": "51386", "system_control_id": 1},
      {"oid": "51335", "system_control_id": 1},
      {"oid": "22475", "system_control_id": 1},
      {"oid": "22431", "system_control_id": 1},
      {"oid": "22387", "system_control_id": 1},
      {"oid": "22320", "system_control_id": 1}}（成功）
```

13.命令反馈接口

- interface: insert_command_feedback
- param_1: "db"
- param_2: "uid"
- param_3: "password"
- param_4: "action_id"
- param_5: "asset_id"
- param_6: "name"
- param_7: "asset_id_2"
- param_8: "action_code_name"
- param_9: "item_id"

- return: 无返回值

14.维修工作流的控制和附件上传

- interface: asset_repair
- param_1: "uid"
- param_2: "password"
- param_3: "name"
- param_4: "atta_name"
- param_5: "atta_description"
- param_6: "file_list"
- param_7: "kind"
- param_8: "flag"
- param_9: "repair_time"
- param_10: "fault_reason"
- param_11: "repair_method"
- param_12: "remark"
- param_13: "line"

参数说明：
1.name是维修单号
3.atta_name是附件的名称
4.kind是附件的种类（url，binary），kind从这2者中取值 5.atta_description是附件的描述
6.flag是一个bool值，为True时表示开始维修，False表示完成维修
7.repair_time是维修次数
8.fault_reason、repair_method和remark是故障原因、维修方法和备注
9.line是一个list，[[product_qty,select,record_name,descripe]，]类似于这种，添加维修操作的product_qty，是产品数量，这里传一个字符,比如：'10'，select是'add'和'remove'中的一个，record_name是产品，descripe是描述,传参顺序不能乱，line如果没有请传一个[]列表
10.file_list也是一个列表，里面是由附件名称、内容组成的列表
file_list=[[file1_name,file1_context],[file2_name,file2_context],[file3_name,file3_context]]

15.历史曲线接口

- interface：getCollectData
- param_1 : "starttime" unix timestamp
- param_2: "endtime" unix timestamp
- return：

```
    - [u"{'asset_id': '2', 'collect_point_name':
      '\\xe9\\x87\\x87\\xe9\\x9b\\x86\\xe7\\x82\\xb9\\xe5\\x90\\x8d\\xe7\\xa7\\xb0',
      'attribute_name': '\\xe5\\xb1\\x9e\\xe6\\x80\\xa7\\xe5\\x90\\x8d\\xe7\\xa7\\xb0', 'control_asset_id': '2',
      'warning_state': 'event', 'trigger_repair_so':
      'True', 'value': '20', 'asset_name': '\\xe8\\xae\\xbe\\xe5\\xa4\\x87\\xe5\\x90
      \\x8d\\xe7\\xa7\\xb0', 'upper_limit': '\\xe4\\xb8\\x8a\\xe9\\x99\\x90', 'attribute_id': '2', 'collect_time':
      datetime.datetime(2016, 5, 26, 0, 3, 50, 72283), 'control_asset_name':
      '\\xe6\\x8e\\xa7\\xe5\\x88\\xb6\\xe8\\xae\\xbe\\xe5\\xa4\\x87\\xe5\\x90\\x8d\\xe7\\xa7\\xb0', 'item_id':
      'channel
      .device.group.test6295', 'lower_limit': '\\xe4\\xb8\\x8b\\xe9\\x99\\x90', 'warning_name':
      '\\xe7\\x94\\xb5\\xe6\\xb5\\x81\\xe8\\
       xad\\xa6\\xe5\\x91\\x8a',
      'collect_point_id': '2'}"]
    - 'time error'（输入的时间戳顺序不对）
    - {"faultcode": "connect to redis error"}(连接redis错误)
    - {"faultcode": "CollectData does not exit"}(数据不存在)
```

这边只提供了按时间查询历史数据的接口，如果要实现每个设备的每个属性的历史曲线，可以通过RPC获取的设备信息进行匹配

16.实时曲线

实时曲线是通过websocket获取的
websocket地址：101.200.194.191：8888

三.接口调用示例

```
#! /usr/bin/env python
# -*- coding: utf-8 -*-

import thriftpy
import datetime
import json
import os
import xmlrpclib
from thriftpy.rpc import client_context

main_thrift = thriftpy.load("main.thrift", module_name="main_thrift")

def main():
        with client_context(main_thrift.OdooService, '192.168.108.169', 6000 ,timeout=0) as c:
```

```
            file_path_1 = os.path.abspath("./picture.jpg")
            file_path_2 = os.path.abspath("./test.jpg")
            file_path_3 = os.path.abspath("./README.md")

            with open(file_path_1,"rb") as f1, open(file_path_2,"rb") as f2, open(file_path_3,"rb") as f3:
                file1_context = f1.read().encode("base64")
                file2_context = f2.read().encode("base64")
                file3_context = f3.read().encode("base64")

                file1_name = "picture.jpg"
                file2_name = "test.jpg"
                file3_name = "README.md"

                atta_name = ["123","456","abc"]
                atta_description = ['附件一描述','附件二描述','附件三描述']
                file_list = [[file1_name,file1_context],[file2_name,file2_context],[file3_name,file3_context]]
                atta_info = zip(atta_name, atta_description, file_list)
                url_list = [['URL附件1','www.baidu.com',['www.baidu.com']],['URL附件2','google.com',['google.com']]]

                c.asset_repair(1,'admin','123456',atta_name,atta_description,file_list,'binary',False,0,
```

```
                '设备陈旧','换新设备','该设备需定期保养',[])

if __name__ == '__main__':
        main()
```

- **这是维修工作流的控制和附件上传接口的调用示例，python实现的客户端**
- **测试环境101.200.194.191**

四.参数说明

除了asset_repair接口的参数有其他的类型，其他接口参数都是string类型