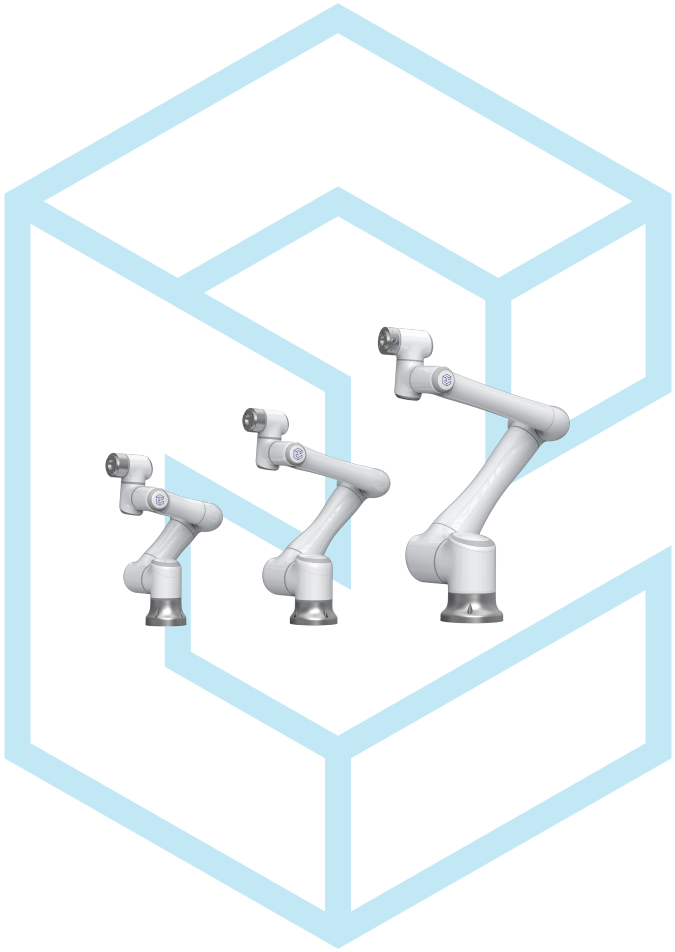
ELITE ROBOT
艾利特机器人

ELITE ROBOT EC系列

编程手册



SDK-Socket 手册

苏州艾利特机器人有限公司

2021-12-24

版本：6.0.22

目录

1 简介 **1**

2 控制接口 **2**

2.1 Python 数据处理示例 3

2.2 接口服务 5

2.2.1 伺服服务 (ServoService) 5

2.2.1.1 获取机械臂伺服状态 5

2.2.1.2 设置机械臂伺服状态 5

2.2.1.3 同步伺服编码器数据 6

2.2.1.4 清除报警 6

2.2.1.5 获取同步状态 7

2.2.2 参数服务 (ParamService) 7

2.2.2.1 获取机器人状态 7

2.2.2.2 获取机器人模式 8

2.2.2.3 获取机器人输出端关节位置信息 8

2.2.2.4 获取机器人当前位姿信息 8

2.2.2.5 获取机器人马达速度 9

2.2.2.6 获取机器人当前坐标系 10

2.2.2.7 获取机器人循环模式 11

2.2.2.8 获取机器人当前作业运行行号 11

2.2.2.9 获取机器人当前编码器值列表 12

2.2.2.10 获取机器人当前工具号 13

2.2.2.11 切换机器人当前工具号 13

2.2.2.12 获取机器人当前用户坐标号 14

2.2.2.13 切换机器人当前用户坐标号 14

2.2.2.14 获取机器人当前力矩信息 15

2.2.2.15 获取轨迹运动当前运行点位序号 15

2.2.2.16 指定坐标系 16

2.2.2.17 拖动示教开关 17

2.2.2.18 设置机械臂负载和重心 17

2.2.2.19 设置机械臂工具中心 18

2.2.2.20 获取碰撞状态 19

2.2.2.21 获取用户坐标系数据 20

2.2.2.22 指定循环模式 20

2.2.2.23 设置用户坐标系数据 21
2.2.2.24 获取工具坐标系数据 22
2.2.2.25 获取工具负载质量 23
2.2.2.26 获取工具质心 24
2.2.2.27 获取机器人类型 24
2.2.2.28 获取机器人 DH 参数 25
2.2.2.29 设置碰撞使能 25
2.2.2.30 设置碰撞灵敏度 26
2.2.2.31 获取自动生成的加密字符串 26
2.2.2.32 设置安全参数 27
2.2.2.33 获取机器人运行速度 29
2.2.2.34 清除碰撞状态 29
2.2.2.35 获取远程模式下机器人当前工具号 30
2.2.2.36 设置远程模式下机器人当前工具号 30
2.2.2.37 获取直角坐标系下的法兰盘中心位姿 31
2.2.2.38 获取用户坐标系下的法兰盘中心位姿 31
2.2.2.39 获取机器人子类型 32
2.2.2.40 获取安全参数使能状态 32
2.2.2.41 获取安全功率 33
2.2.2.42 获取安全动量 33
2.2.2.43 获取安全工具力 34
2.2.2.44 获取安全肘部力 34
2.2.2.45 获取速度百分比 35
2.2.2.46 获取拖动最大启动速度 35
2.2.2.47 获取最大力矩误差百分比 36
2.2.2.48 设置末端按钮状态 36
2.2.2.49 获取末端按钮状态 37
2.2.2.50 获取碰撞检测使能状态 37
2.2.2.51 获取碰撞灵敏度 38
2.2.2.52 获取当前 tcp 在当前用户坐标系下的位姿 38
2.2.2.53 获取机械臂本体异常情况 39
2.2.2.54 获取关节运动速度 39
2.2.2.55 获取 tcp 加速度 40
2.2.2.56 获取关节加速度 40
2.2.2.57 获取 tcp 运动速度 41
2.2.2.58 获取机器人的紧急停止状态 41
2.2.2.59 获取模拟量输出电压值 42

2.2.2.60 获取工具负载和质心 42
2.2.2.61 获取机器人输入端关节位置信息 43
2.2.3 运动服务 (MovementService) 43
2.2.3.1 关节运动 43
2.2.3.2 直线运动 46
2.2.3.3 圆弧运动 48
2.2.3.4 旋转运动 49
2.2.3.5 添加路点信息 2.0 51
2.2.3.6 清除路点信息 2.0 53
2.2.3.7 轨迹运动 2.0 53
2.2.3.8 jog 运动 54
2.2.3.9 停止机器人运行 55
2.2.3.10 机器人自动运行 56
2.2.3.11 机器人暂停 56
2.2.3.12 检查 jbi 文件是否存在 57
2.2.3.13 运行 jbi 文件 57
2.2.3.14 获取 jbi 文件运行状态 58
2.2.3.15 设置机器人运行速度 59
2.2.3.16 关节匀速运动 60
2.2.3.17 停止关节匀速运动 61
2.2.3.18 直线匀速运动 62
2.2.3.19 停止直线匀速运动 62
2.2.3.20 指定坐标系下直线运动 63
2.2.4 运动学服务 (KinematicsService) 64
2.2.4.1 逆解函数 64
2.2.4.2 正解函数 65
2.2.4.3 基坐标到用户坐标位姿转化 66
2.2.4.4 用户坐标到基坐标位姿转化 67
2.2.4.5 位姿相乘 67
2.2.4.6 位姿求逆 68
2.2.5 IO 服务 (IOService) 69
2.2.5.1 获取输入 IO 状态 69
2.2.5.2 获取输出 IO 状态 69
2.2.5.3 设置输出 IO 状态 70
2.2.5.4 获取虚拟输入 IO 状态 70
2.2.5.5 获取虚拟输出 IO 状态 71
2.2.5.6 设置虚拟输出 IO 状态 71

2.2.5.7 读取多个 M 虚拟 IO . . . 72
2.2.5.8 获取模拟量输入 . . . 72
2.2.5.9 设置模拟量输出 . . . 73
2.2.6 变量服务 (VarService) . . . 73
2.2.6.1 获取系统 B 变量值 . . . 73
2.2.6.2 设置系统 B 变量值 . . . 74
2.2.6.3 获取系统 I 变量值 . . . 74
2.2.6.4 设置系统 I 变量值 . . . 75
2.2.6.5 获取系统 D 变量值 . . . 75
2.2.6.6 设置系统 D 变量值 . . . 76
2.2.6.7 获取系统 P 变量是否启用 . . . 76
2.2.6.8 获取 P 变量的值 . . . 77
2.2.6.9 设置 P 变量的值 . . . 77
2.2.6.10 获取 V 变量的值 . . . 78
2.2.6.11 设置 V 变量的值 . . . 78
2.2.6.12 保存变量数据 . . . 79
2.2.7 透传服务 (TransparentTransmissionService) . . . 80
2.2.7.1 初始化透传服务 . . . 80
2.2.7.2 设置当前透传伺服目标关节点 . . . 80
2.2.7.3 获取当前机器人是否处于透传状态 . . . 81
2.2.7.4 添加透传伺服目标关节点信息到缓存中 . . . 81
2.2.7.5 清空透传缓存 . . . 83
2.2.7.6 Example 1 . . . 83
2.2.7.7 Example 2 . . . 86
2.2.8 系统服务 (SystemService) . . . 90
2.2.8.1 获取控制器软件版本号 . . . 90
2.2.8.2 获取伺服版本号 . . . 90
2.2.9 时间戳服务 (TrajectoryService) . . . 91
2.2.9.1 初始化运动 . . . 91
2.2.9.2 添加运动点位 . . . 92
2.2.9.3 停止添加点位 . . . 93
2.2.9.4 检查执行状态 . . . 94
2.2.9.5 开始带时间戳运动 . . . 95
2.2.9.6 暂停运动 . . . 96
2.2.9.7 恢复运动 . . . 97
2.2.9.8 停止运动 . . . 97
2.2.9.9 清空缓存 . . . 98

2.2.9.10 Example 1 98
2.2.9.11 Example 2 101
2.3 Examples 104
2.3.1 Example 1 104
2.3.2 Example 2 107
2.3.3 Example 3 109

3 监控接口 112
3.1 监控接口数据说明列表 112
3.2 Example 114

4 日志接口 120
4.1 Example 120

5 原始日志接口 121
5.1 Example 121

第 1 章 简介

提醒



本手册适用软件版本为：2.19.2。

艾利特机器人为支持用户进行二次开发而开放了机器人控制器端口，如**表** 1-1 所示。

表 1-1 . 控制器端口

端口号	名称	功能
8055	控制接口	接收指定格式的 json 字符串
8056	监控接口	输出机器人信息
8058	日志接口	输出解析后的日志信息文件
8059	原始日志接口	输出原始的日志信息文件

用户可通过 socket 通讯连接对应的控制器端口，来进行一些操作从而实现对应的功能。

第 2 章 控制接口

用户可通过 socket 通讯向控制器控制端口发送指定格式的 json 字符串来实现相关功能，如下所示。

```
发送
  {"jsonrpc":"2.0","method":"方法名称","params":参数,"id":id}

接收

  正常
    {"jsonrpc":"2.0","result":结果,"id":id}

  出错
    {"jsonrpc":"2.0","error":{"code":错误代码,"message":"出错信 息"},"
       id":id}
```

提醒



该功能适用于 2.13.0 及以上版本。

发送 json 字符串时的 id 和接收结果时的 id 一致，如下所示。

```
发送
    {"jsonrpc":"2.0","method":"cmd_set_payload","params":{"cog"
       :[1,2,3],"tool_num":1,"m":12},"id":1}

    {"jsonrpc":"2.0","method":"checkJbiExist","params":{"filename":"
       123123"},"id":1}

    {"jsonrpc":"2.0","method":"getRobotState","params":[],"id":1}


  接收
    正常
      {"jsonrpc":"2.0","result":"false","id":1}

    出错
```

```
14      {"jsonrpc":"2.0","error":{"code":-32601,"message":"Method not
            found."},"id":1}
```

提示



发送和接收都以 \n 结尾

目前 json 协议常见返回异常有两种：

JRPC_METHOD_NOT_FOUND -32601，JRPC_INTERNAL_ERROR -32693。

- 32601 为未找到对应接口，需要检查接口名称是否正确或确认当前版本是否支持该接口。
- 32693 为接口内部定义的异常，未找到相应参数，参数超出范围，不满足执行条件等均报此类异常。此类错误只需根据错误信息检查参数及其范围还有执行条件是否满足即可。

2.1 Python 数据处理示例

本章节示例，均采用 Python 语言。用户可根据本节示例进行代码的修改。

提醒



Python 语言版本需为 Python3。

```
1  import socket
2  import json
3  import time
4
5  def connectETController(ip,port=8055):
6    sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
7    try:
8      sock.connect((ip,port))
9      return (True,sock)
10   except Exception as e:
11     sock.close()
12     return (False,)
13
14 def disconnectETController(sock):
15   if(sock):
```

```
16      sock.close()
17      sock=None
18    else:
19      sock=None
20
21  def sendCMD(sock,cmd,params=None,id=1):
22    if(not params):
23      params=[]
24    else:
25      params=json.dumps(params)
26    sendStr="{{\"method\":\"{0}\",\"params\":{1},\"jsonrpc\":\"2.0\",\"id
        \":{2}}}".format(cmd,params,id)+"\n"
27    try:
28      sock.sendall(bytes(sendStr,"utf-8"))
29      ret=sock.recv(1024)
30      jdata=json.loads(str(ret,"utf-8"))
31      if("result" in jdata.keys()):
32        return (True,json.loads(jdata["result"]),jdata["id"])
33      elif("error" in jdata.keys()):
34        return (False,jdata["error"],jdata["id"])
35      else:
36        return (False,None,None)
37    except Exception as e:
38      return (False,None,None)
39
40  if __name__ == "__main__":
41    # 机器人IP地址
42    robot_ip="192.168.1.200"
43    conSuc,sock=connectETController(robot_ip)
44    if(conSuc):
45      # 获取机器人状态
46      suc, result, id = sendCMD(sock, "getRobotState")
47      # 打印结果
48      print(result)
```

2.2 接口服务

2.2.1 伺服服务 (ServoService)

2.2.1.1 获取机械臂伺服状态

```
{"jsonrpc":"2.0","method":"getServoStatus","id":id}
```

功能： 获取机械臂伺服状态

参数： 无

返回： 启用 true，未启用 false

示例：

```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if (conSuc):
    # 获取机械臂伺服状态
    suc, result ,id=sendCMD(sock,"getServoStatus")
```

2.2.1.2 设置机械臂伺服状态

```
{"jsonrpc":"2.0","method":"set_servo_status","params":{"status":status},"
   id":id}
```

功能： 设置伺服使能状态

参数： status：伺服开关，范围：int[0,1]，1 为开，0 为关

返回： 成功 true，失败 false

示例：

```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if (conSuc):
    # 获取机械臂伺服状态
    suc, result ,id=sendCMD(sock, "getServoStatus")
    if ( result  == 0):
      # 设置机械臂伺服状态ON
      suc, result ,id=sendCMD(sock,"set_servo_status",{" status ":1})
      time.sleep(1)
```

注意： 本命令只支持在 remote 模式下使用。

2.2.1.3 同步伺服编码器数据

```
{"jsonrpc":"2.0","method":"syncMotorStatus","id":id}
```

功能： 同步伺服编码器数据

参数： 无

返回： 成功 true，失败 false

示例：

```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if (conSuc):
    # 获取同步状态
    suc, result, id = sendCMD(sock, "getMotorStatus")
    if (result == 0):
      # 同步伺服编码器数据
      suc, result, id = sendCMD(sock, "syncMotorStatus")
      time.sleep(0.5)
```

注意： 本命令只支持在 remote 模式下使用。

2.2.1.4 清除报警

```
{"jsonrpc":"2.0","method":"clearAlarm","id":id}
```

功能： 清除报警

参数： 无

返回： 成功 true，失败 false

示例：

```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if (conSuc):
    # 清除报警
    ret, result, id = sendCMD(sock, "clearAlarm")
```

注意： 本命令只支持在 remote 模式下使用。

2.2.1.5 获取同步状态

```
{"jsonrpc":"2.0","method":"getMotorStatus","id":id}
```

功能： 获取同步状态

参数： 无

返回： 同步 true，未同步 false

示例：

```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if (conSuc):
    # 获取同步状态
    suc， result , id = sendCMD(sock, "getMotorStatus")
```

2.2.2 参数服务 (ParamService)

2.2.2.1 获取机器人状态

```
{"jsonrpc":"2.0","method":"getRobotState","id":id}
```

功能： 获取机器人状态

参数： 无

返回： 停止状态 0，暂停状态 1，急停状态 2，运行状态 3，报警状态 4，碰撞状态 5

示例：

```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if (conSuc):
    # 获取机器人状态
    suc， result , id = sendCMD(sock, "getRobotState")
```

注意： 本指令获取的急停状态只会短暂存在，很快会被报警覆盖。如果需要获取急停状态，请参考第 2.2.2.58 小节。

2.2.2.2 获取机器人模式

```
{"jsonrpc":"2.0","method":"getRobotMode","id":id}
```

功能： 获取机器人模式

参数： 无

返回： 示教模式 0，自动模式 1，远程模式 2

示例：

```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if (conSuc):
    # 获取机器人模式
    suc， result , id = sendCMD(sock, "getRobotMode")
```

2.2.2.3 获取机器人输出端关节位置信息

```
{"jsonrpc":"2.0","method":"getRobotPos","id":id} 或 {"jsonrpc":"2.0","
   method":"get_joint_pos","id":id}
```

功能： 获取机器人输出端关节位置信息

参数： 无

返回： 机器人当前的位置信息 double pos[8]

示例：

```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if (conSuc):
    # 获取机器人当前位置信息
    suc， result , id = sendCMD(sock, "get_joint_pos")
```

注意： 指令 getRobotPos 不推荐使用

2.2.2.4 获取机器人当前位姿信息

```
{"jsonrpc":"2.0","method":"getRobotPose","params":{"coordinate_num":
   coordinate_num,"tool_num":tool_num,"unit_type": unit_type},"id":id} 或
    {"jsonrpc":"2.0","method":"getTcpPose","params":{"coordinate_num":
   coordinate_num,"tool_num":tool_num,"unit_type":unit_type},"id":id} 或
   {"jsonrpc":"2.0","method":"get_tcp_pose","params":{"coordinate_num":
   coordinate_num,"tool_num":tool_num,"unit_type": unit_type},"id":id}
```

功能： 获取机器人当前位姿信息

参数： coordinate_num: 坐标号；int[-1,7], -1: 基坐标系，0~7: 对应用户坐标系
tool_num: 工具号：int[-1,7]，-1：当前工具号，0~7: 对应工具号
unit_type：int[0,1]，可选参数, 返回 pose 的 rx,ry,rz 的单位类型，0：角度，1：弧度，不写默认为弧度值。

返回： 机器人当前位姿信息 double pose[6]

示例：
```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if (conSuc):
    # 获取机器人当前位姿信息
    suc, result , id=sendCMD(sock,"get_tcp_pose",{"coordinate_num": 0,"tool_num": 0})
```

注意： 若同时使用参数 coordinate_num 和参数 tool_num，返回对应工具号、对应用户坐标系下的用户坐标；若单独使用 coordinate_num 或 tool_num，则返回基坐标系下的机器人位姿。
指令 getRobotPose 和 getTcpPose 不推荐使用。

2.2.2.5 获取机器人马达速度

```
{"jsonrpc":"2.0","method":"getMotorSpeed","id":id} 或 {"jsonrpc":"2.0","
   method":"get_motor_speed","id":id}
```

功能： 获取机器人马达速度

参数： 无

返回： 机器人马达速度 double speed[8]

示例：

```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  point = []
  point.append([0.0065,-103.9938,102.2076,-88.2138,
  90.0000,0.0013,0.0000,0.0000])
  point.append([-16.2806,-82.4996,81.9848,-89.4851,
  90.0000,-16.2858,0.0000,0.0000])
  point.append([3.7679, -71.7544, 68.7276, -86.9732,
  90.0000, 3.7627, 0.0000, 0.0000])
  point.append([12.8237,-87.3028,87.2361,-89.9333
  90.0000,12.8185,0.0000,0.0000])
  if(conSuc):
    for i in range(0, 4, 1):
      # 关节运动
      suc, result ,id=sendCMD(sock,"moveByJoint",{"targetPos":point[i],"speed":30})
      while(True):
        # 获取机器人马达速度
        suc, result ,id =sendCMD(sock,"get_motor_speed")
        print( result )
        # 获取机器人状态
        suc, result ,id=sendCMD(sock,"getRobotState")
        if( result == 0):
          break
```

注意： 指令 getMotorSpeed 不推荐使用。

2.2.2.6 获取机器人当前坐标系

```
{"jsonrpc":"2.0","method":"getCurrentCoord","id":id}
```

功能： 获取机器人当前坐标系

参数： 无

返回： 关节 0，直角 1，工具 2，用户 3，圆柱 4

示例：
```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if (conSuc):
    # 获取机器人当前坐标系
    suc， result ， id = sendCMD(sock, "getCurrentCoord")
```

2.2.2.7 获取机器人循环模式

```
{"jsonrpc":"2.0","method":"getCycleMode","id":id}
```

功能： 获取机器人循环模式

参数： 无

返回： 单步 0，单循环 1，连续循环 2

示例：
```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if (conSuc):
    # 获取机器人循环模式
    suc， result ， id = sendCMD(sock, "getCycleMode")
```

2.2.2.8 获取机器人当前作业运行行号

```
{"jsonrpc":"2.0","method":"getCurrentJobLine","id":id}
```

功能： 获取机器人当前作业运行行号

参数： 无

返回： jbi 行号

示例：

```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if (conSuc):
    # 获取机械臂伺服状态
    suc, result ,id=sendCMD(sock, "getServoStatus")
    if ( result  == 0):
      # 设置机械臂伺服状态ON
      suc, result ,id=sendCMD(sock,"set_servo_status",{" status ":1})
      time.sleep (1)
    # 检查jbi文件是否存在
    suc, result ,id=sendCMD(sock,"checkJbiExist",{"filename": jbi_filename })
    if (suc  and  result ==1):
      # 运行jbi文件
      suc, result ,id=sendCMD(sock,"runJbi",{"filename": jbi_filename })
      if (suc  and  result ):
        checkRunning=3
        while(checkRunning==3):
          # 获取jbi文件运行状态
          suc, result ,id=sendCMD(sock,"getJbiState")
          checkRunning=result["runState"]
          # 获取机器人当前作业运行行号
          # 该行号需要将点位信息的行数算进去，并不是示教器程序的行号
          suc, result ,id=sendCMD(sock,"getCurrentJobLine")
          print ( result )
          time.sleep (0.1)
```

2.2.2.9 获取机器人当前编码器值列表

```
{"jsonrpc":"2.0","method":"getCurrentEncode","id":id}
```

功能： 获取机器人当前编码器值列表

参数： 无

返回： 机器人当前编码器值列表 double encode[8]

示例：
```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if (conSuc):
    # 获取机器人当前编码器值列表
    suc， result ， id = sendCMD(sock, "getCurrentEncode")
```

2.2.2.10 获取机器人当前工具号

```
{"jsonrpc":"2.0","method":"getToolNumber","id":id}
```

功能： 获取机器人示教模式下的当前工具号

参数： 无

返回： 机器人当前工具号，范围：int[0,7]

示例：
```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if (conSuc):
    # 获取机器人当前工具号
    suc， result ， id = sendCMD(sock, "getToolNumber")
```

2.2.2.11 切换机器人当前工具号

```
{"jsonrpc":"2.0","method":"setToolNumber","params":{"tool_num":tool_num}"
   id":id}
```

功能： 切换机器人当前工具号

参数： tool_num：工具号，范围：int[0,7]

返回： 成功 true，失败 false

示例：

```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if (conSuc):
    # 切换机器人当前工具号
    suc, result ,id=sendCMD(sock,"setToolNumber",{"tool_num":7})
    time.sleep (0.5)
    # 获取机器人当前工具号
    suc， result ， id = sendCMD(sock, "getToolNumber")
```

注意： 本命令仅可切换示教模式下的当前工具号。

2.2.2.12 获取机器人当前用户坐标号

```
{"jsonrpc":"2.0","method":"getUserNumber","id":id}
```

功能： 获取机器人当前用户坐标号

参数： 无

返回： 机器人当前用户坐标号，范围：int[0,7]

示例：

```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if (conSuc):
    # 获取机器人当前用户坐标号
    suc， result ， id = sendCMD(sock, "getUserNumber")
```

2.2.2.13 切换机器人当前用户坐标号

```
{"jsonrpc":"2.0","method":"setUserNumber","params":{"user_num":user_num}"
   id":id}
```

功能： 切换机器人当前用户坐标号

参数： user_num：用户坐标号，范围：int[0,7]

返回： 成功 true，失败 false

示例：
```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if(conSuc)::
    # 切换机器人用户坐标号
    suc, result ,id=sendCMD(sock,"setUserNumber",{"user_num":7})
    time.sleep(0.5)
    # 获取机器人当前用户坐标号
    suc， result ， id = sendCMD(sock, "getUserNumber")
```

注意： 本命令只支持在 remote 模式下使用。

2.2.2.14 获取机器人当前力矩信息

```
{"jsonrpc":"2.0","method":"getRobotTorques","id":id} 或{"jsonrpc":"2.0","
   method":"get_motor_torque","id":id}
```

功能： 获取机器人当前力矩信息

参数： 无

返回： 机器人当前力矩信息 double torques[8]，关节额定力矩千分比，单位 ‰。

示例：
```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if(conSuc):
    # 获取机器人当前力矩信息
    suc， result ， id = sendCMD(sock, "get_motor_torque")
```

注意： 指令 getRobotTorques 不推荐使用。

2.2.2.15 获取轨迹运动当前运行点位序号

```
{"jsonrpc":"2.0","method":"getPathPointIndex","id":id}
```

功能： 获取机器人当前运行点位序号

参数： 无

返回： 存储当前运行点位序号,-1 为非路点运动

示例：

```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  C000 = [0.0065,-103.9938,102.2076,-88.2138,
   90.0000,0.0013,0.0000,0.0000]
  C001 = [-16.2806,-82.4996,81.9848,-89.4851,
   90.0000,-16.2858,0.0000,0.0000]
  C002 = [3.7679, -71.7544, 68.7276, -86.9732,
  90.0000, 3.7627, 0.0000, 0.0000]
  if (conSuc):
    # 清除路点信息2.0
    suc, result , id = sendCMD(sock, "clearPathPoint")
    if ( result == True):
      # 添加路点信息2.0
      suc, result , id = sendCMD(sock, "addPathPoint", {"wayPoint": C000,"moveType": 0, "speed":
            50, "smooth": 7})
      suc, result , id = sendCMD(sock, "addPathPoint", {"wayPoint": C001,"moveType":0, "speed":
          50, "smooth": 7})
      suc, result , id = sendCMD(sock, "addPathPoint", {"wayPoint": C002,"moveType": 0, "speed":
            50, "smooth": 7})
      # 轨迹运动2.0
      suc, result , id = sendCMD(sock, "moveByPath")
      while(True):
        # 获取 trackfile 文件运行的行号(与示教器显示行号一致)
        suc, result , id = sendCMD(sock, "getPathPointIndex")
        print ( result )
        # 获取机器人状态
        suc, result , id = sendCMD(sock, "getRobotState")
        if ( result == 0):
          break
```

2.2.2.16 指定坐标系

```
{"jsonrpc":"2.0","method":"setCurrentCoord","params":{"coord_mode":
   coord_mode,},"id":id}
```

功能： 指定坐标系

参数： coord_mode：坐标系，范围 int[0，4]。关节：0，直角：1，工具：2，用户：3，圆柱：4

返回： 成功 true，失败 false

示例：
```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if (conSuc):
    for i in range(0, 5, 1)
      # 指定坐标系
      suc, result ,id=sendCMD(sock,"setCurrentCoord",{"coord_mode":i})
      time.sleep (0.5)
      # 获取机器人当前坐标
      suc, result ,id=sendCMD(sock,"getCurrentCoord")
      print ( result )
```

注意： 本命令只支持在 remote 模式下使用。

2.2.2.17 拖动示教开关

```
{"jsonrpc":"2.0","method":"drag_teach_switch","params":{"switch":switch
   },"id":id}
```

功能： 拖动示教开关

参数： switch：开关，范围：int[0,1]，0 为关，1 为开

返回： 成功 true，失败 false

示例：
```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if (conSuc):
    # 拖动示教开关
    suc, result ,id=sendCMD(sock,"drag_teach_switch",{"switch":1})
```

注意： 本命令只支持在 remote 模式下使用。

2.2.2.18 设置机械臂负载和重心

```
{"jsonrpc":"2.0","method":"cmd_set_payload","params":{"tool_num":tool_num
   ,"m":m,"point":point},"id":id} 或 {"jsonrpc":"2.0","method":"
   cmd_set_payload","params":{"tool_num":tool_num,"m":m,"cog":cog},"id":
   id}
```

功能： 设置机械臂负载和重心

参数： tool_num：工具号，范围：int[0,7]
m：负载重量，单位 Kg，double, 范围：EC63：[0,3.6]，EC66：[0,7.2]，EC612：[0,14.4]
point 或 cog：重心，x,y,z, 单位毫米，范围：double[-5000,5000]

返回： 成功 true，失败 false

示例：

```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if (conSuc):
    # 设置机械臂负载和重心
    suc, result , id=sendCMD(sock,"cmd_set_payload",{"tool_num":0,"m":5,"cog":[10,20,30]})
```

注意： 2.16.2 及以上版本版本支持工具号的选择，tool_num 为何值，设置的就是几号工具的负载和重心。
参数 point 不推荐使用。
本命令只支持在 remote 模式下使用。

2.2.2.19 设置机械臂工具中心

```
{"jsonrpc":"2.0","method":"cmd_set_tcp","params":{"point":point,"tool_num
   ":tool_num,"unit_type":unit_type},"id":id}
```

功能： 设置机械臂工具中心

参数： tool_num：工具号，范围：int[0,7]
point：工具中心，前三项单位毫米，范围：double[-500,500]，后三项单位：弧度，范围：double[-π,π] 或角度，范围：double[-180,180]
unit_type：可选参数，int[0,1]，设置工具中心的 rx,ry,rz 的单位类型，0：角度，1：弧度，不写默认为弧度值。

返回： 成功 true，失败 false

示例：
```
if __name__ == "__main__":
# 机器人IP地址
robot_ip = "192.168.1.202"
conSuc, sock = connectETController(robot_ip)
if (conSuc):
  # 设置机械臂工具中心
  suc, result , id = sendCMD(sock, "cmd_set_tcp", {"point": [10, 0, 0, 30, 0, 0], "tool_num":
      1, "unit_type":0})
  print(suc, result , id)
else:
  print("连接失败")
disconnectETController(sock)
```

注意： 本命令只支持在 remote 模式下使用。

2.2.2.20 获取碰撞状态

```
{"jsonrpc":"2.0","method":"getCollisionState","id":id}
```

功能： 获取碰撞状态

参数： 无

返回： 发生碰撞:1，未发生碰撞:0

示例：
```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if(conSuc):
    # 获取碰撞状态
    suc, result , id = sendCMD(sock, "getCollisionState")
```

注意： 本命令只支持在 remote 模式下使用。

2.2.2.21 获取用户坐标系数据

```
{"jsonrpc":"2.0","method":"getUserFrame","params":{"user_num": user_num,"
   unit_type":unit_type},"id":id}
```

功能： 获取用户坐标系数据

参数： user_num: 用户坐标号，范围：int[0,7]
unit_type: int[0,1]，可选参数, 返回 pose 的 rx,ry,rz 的单位类型，范围 int [0,1]
默认：弧度，0：角度，1：弧度

返回： 用户坐标系数据 double pose[6]

示例：

```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if (conSuc):
    for i in range(8):
      suc, result , id = sendCMD(sock, "getUserFrame", {"user_num": i,"unit_type":1}))
      print ("用户坐标号= ",i)
      print ("suc = ", suc, "", "id = ", id)
      if (suc):
        print (" result =", result )
      else :
        print ("err_msg=", result ["message"])
```

注意： unit_type 参数仅适用于 v2.15.2 及以上版本。

2.2.2.22 指定循环模式

```
{"jsonrpc":"2.0","method":"setCycleMode","params":{"cycle_mode":
   cycle_mode},"id":id}
```

功能： 指定循环模式

参数： cycle_mode：循环模式，范围：int[0,2] 单步：0，单循环：1，连续循环：2

返回： 成功 true, 失败 false

示例：

```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if (conSuc):
    #设置循环模式为单循环
    ret , result , id = sendCMD(sock, "setCycleMode",{"cycle_mode":1})
    if ret :
      print (" result =", result )
    else :
      print ("err_msg=", result ["message"])
```

注意： 本命令只支持在 remote 模式下使用。

2.2.2.23 设置用户坐标系数据

```
{"jsonrpc":"2.0","method":"setUserFrame","params":{"user_num":user_num,"
   user_frame":user_frame,"unit_type":unit_type},"id":id}
```

功能： 设置用户坐标系数据

参数： user_num: 用户号，范围 int [0,7]
user_frame: 用户坐标系数据，double user_frame[6]，范围：[-1e+9,1e+9]，x,y,z 单位：毫米，rx,ry,rz 单位：度/rad
unit_type: 用户坐标系的 rx,ry,rz 的单位类型，int[0,1], 可选参数,rx,ry,rz 的单位类型，0：角度，1：弧度，不写默认为弧度值。

返回： 成功 true, 失败 false

示例：
```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if (conSuc):
    user_frame=[499.011212,570.517817,
    247.082805,-3.141593,-0.000000,-0.773067]
    ret , result , id=sendCMD(sock,"setUserFrame",{"user_num":0,"user_frame":user_frame,"unit_type"
        :1})
    if ret :
      print (" result =", result )
    else :
      print ("err_msg=", result ["message"])
```

注意： 本命令只支持在 remote 模式下使用。
unit_type 参数仅适用于 v2.15.2 及以上版本。

2.2.2.24 获取工具坐标系数据

```
{"jsonrpc":"2.0","method":"getTcpPos","params":{"tool_num":tool_num,"
   unit_type":unit_type},"id":id}
```

功能： 获取工具坐标系数据

参数： tool_num: 工具坐标号，范围 int [0,7]
unit_type：int[0,1]，返回坐标系的 rx,ry,rz 的单位类型，0：返回角度，1：返回弧度，可选参数，不写默认返回角度。

返回： 工具坐标系数据 double pose[6]

示例：
```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if (conSuc):
    ret , result , id = sendCMD(sock, "getTcpPos", {"tool_num": 0})
    if ret :
      print (" result =", result )
    else :
      print ("err_msg=", result ["message"])
```

2.2.2.25 获取工具负载质量

```
{"jsonrpc":"2.0","method ":"getPayload","params":{"tool_num":tool_num},"
   id":id}
```

功能： 获取工具负载质量

参数： tool_num: 工具坐标号，范围 int [0,7]

返回： 工具负载质量，double m

示例：
```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if (conSuc):
    ret , result , id = sendCMD(sock, "getPayload", {"tool_num": 0})
    if ret :
      print (" result =", result )
    else :
      print ("err_msg=", result ["message"])
```

注意： 本指令不推荐使用。

2.2.2.26 获取工具质心

```
{"jsonrpc":"2.0","method":"getCentreMass","params":{"tool_num":tool_num
   },"id":id}
```

功能： 获取工具质心

参数： tool_num: 工具坐标号，范围 int [0,7]

返回： 工具负载质心，double cog

示例：
```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if (conSuc):
    ret , result , id = sendCMD(sock, "getCentreMass", {"tool_num": 0})
    if ret :
      print (" result =", result )
    else :
      print ("err_msg=", result ["message"])
```

注意： 本指令不推荐使用。

2.2.2.27 获取机器人类型

```
{"jsonrpc":"2.0","method":"getRobotType","id":id}
```

功能： 获取机器人类型

参数： 无

返回： 机器人类型 int 62(六轴协作机器人)、60（垂直多关节串联机器人）、41（四轴旋转关节机器人）、40（码垛机器人）、43（SCARA 机器人）、30（Delta 并联机器人）

示例：
```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if (conSuc):
    ret , result , id = sendCMD(sock, "getRobotType")
    if ret :
      print (" result =", result )
    else :
      print ("err_msg=", result ["message"])
```

2.2.2.28 获取机器人 DH 参数

```
{"jsonrpc":"2.0","method":"getDH","params":{"index":index},"id":id}
```

功能： 获取机器人 DH 参数

参数： index: 范围 int [0,11]，对应连杆参数 d1~d12

返回： DH 参数

示例：

```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if (conSuc):
    # 获取连杆参数d1的值
    ret , result , id = sendCMD(sock, "getDH", {"index":0})
    if ret :
      print (" result =", result )
    else :
      print ("err_msg=", result ["message"])
```

2.2.2.29 设置碰撞使能

```
{"jsonrpc":"2.0","method":"setCollisionEnable","params":{"enable":
    enable},"id":id}
```

功能： 设置碰撞使能

参数： enable: int[0,1]，1: 打开碰撞开关，0：关闭碰撞开关

返回： 成功 True, 失败 False

示例：

```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if (conSuc):
   # 打开碰撞开关
    ret , result , id = sendCMD(sock, "setCollisionEnable", {"enable": 1})
    if ret :
      print (" result =", result )
    else :
      print ("err_msg=", result ["message"])
```

注意： 本命令不推荐使用。

2.2.2.30 设置碰撞灵敏度

```
{"jsonrpc":"2.0","method":"setCollisionSensitivity","params":{"value":
   value}, "id":id}
```

功能： 设置碰撞灵敏度

参数： value：灵敏度范围 int [10,100]

返回： 成功 True, 失败 False

示例：

```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if (conSuc):
    # 设置碰撞灵敏度为50%
    ret , result , id = sendCMD(sock, " setCollisionSensitivity ", {"value": 50})
    if ret :
      print (" result =", result )
    else :
      print ("err_msg=", result ["message"])
```

注意： 本命令只支持在 remote 模式下使用。
本命令不推荐使用。

2.2.2.31 获取自动生成的加密字符串

```
{"jsonrpc":"2.0","method":"get_remote_sys_password","id":id}
```

功能： 获取自动生成的加密字符串

参数： 无

返回： 自动生成的加密字符串

示例：

```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip = "192.168.1.200"
  conSuc, sock = connectETController(robot_ip)
  if (conSuc):
    ret, str, id = sendCMD(sock, "get_remote_sys_password")
    print(str)
  else:
    print("连接失败")
  disconnectETController(sock)
```

注意： 本命令不推荐使用。

2.2.2.32 设置安全参数

```
{"jsonrpc":"2.0","method":"setSafetyParams","params":{"password":password
  ,"enable":enable,"mode":mode,"power":power,"momentum":momentum,"
  tool_force":tool_force,"elbow_force": elbow_force,"speed":speed,"
  collision_enable":collision_enable,"collision_sensitivity":
  collision_sensitivity},"id":id}
```

功能： 设置安全参数

参数： password：界面未设置远程模式密码时：密码默认为字符串“123456”；界面设置远程模式用户密码后：首先需通过“get_remote_sys_password”，获取系统生成的加密字符串，与设置的远程模式的用户密码组合，计算其 MD5 值作为 json 安全参数设置所需的密码

enable: 安全限制参数使能，int[0,1]，1：使能，0：未使能

mode: 模式，int[0,1]，0：正常模式，1：缩减模式

power: 功率，范围：double [80,1500]，单位：W

momentum: 动量，范围：double [5,90]，单位：kg·m/s

tool_force: 工具力，范围：double [100,400]，单位：N

elbow_force: 肘部力:double [100,400]，单位：N

speed: 速度百分比，double [0-100]，单位：%

collision_enable：int[0,1]，可选参数，设置碰撞检测开关，0：关闭碰撞检测开关，1：打开碰撞检测开关

collision_sensitivity：可选参数，设置碰撞检测灵敏度，范围 int[10,100], 单位：%

返回： 成功 True, 失败 False

示例：

```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip = "192.168.1.202"
  conSuc, sock = connectETController(robot_ip)
  if (conSuc):
    ret , str1 , id = sendCMD(sock, "get_remote_sys_password")
    word = hashlib.md5()
    str2 = "123456"
    word.update(str1.encode("utf8"))
    word.update(str2.encode("utf8"))
    password = word.hexdigest()
    print(password)
    ret , result , id = sendCMD(sock, "setSafetyParams", {"password": password, "enable": 1, "
        mode": 1, "power": 400, "momentum": 90, "tool_force": 400, "elbow_force": 400, "speed":
          0.5, " collision_enable ": 0, " collision_sensitivity ": 10})
    print( result )
  else:
    print ("连接失败")
  disconnectETController(sock)
```

注意： 本命令只支持在 remote 模式下使用。

2.2.2.33 获取机器人运行速度

```
{"jsonrpc":"2.0","method":"getSpeed","id":id}
```

功能： 获取机器人自动速度

参数： 无

返回： 自动速度 double

示例：

```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if (conSuc):
    ret , result , id = sendCMD(sock, "getSpeed")
    if ret :
      print (" result =", result )
    else :
      print ("err_msg=", result ["message"])
```

2.2.2.34 清除碰撞状态

```
{"jsonrpc":"2.0","method":"resetCollisionState","id":id}
```

功能： 清除碰撞状态

参数： 无

返回： 成功 True，失败 False

示例：

```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if (conSuc):
    ret , result , id = sendCMD(sock, " resetCollisionState ")
    if ret :
      print (" result =", result )
    else :
      print ("err_msg=", result ["message"])
```

2.2.2.35 获取远程模式下机器人当前工具号

```
{"jsonrpc":"2.0","method":"getAutoRunToolNumber","id":id}
```

功能： 获取远程模式下机器人当前工具号

参数： 无

返回： 远程模式下机器人当前工具号, 范围:int[0,7]
注：自动模式下的工具号与远程模式下的工具号一致。

示例：
```
if __name__ == "__main__":
    ip = "192.168.1.202"
    conSuc, sock = connectETController(ip)
    # print(conSuc)
    if conSuc:
        ret , result , id = sendCMD(sock, "getAutoRunToolNumber")
        if ret :
            print ("result = ", result )
        else :
            print ("err_msg = ", result ["message"])
```

注意： 本命令适用于 v2.14.4 及以上版本。

2.2.2.36 设置远程模式下机器人当前工具号

```
{"jsonrpc":"2.0","method":"setAutoRunToolNumber","params":{"tool_num":
    tool_num},"id":id}
```

功能： 设置远程模式下机器人当前工具号

参数： tool_num: 工具号，范围：int[0,7]

返回： 成功 True, 失败 False

示例：
```
if __name__ == "__main__":
    ip = "192.168.1.202"
    conSuc, sock = connectETController(ip)
    if conSuc:
        ret , result ,id=sendCMD(sock,"setAutoRunToolNumber",{"tool_num": 0})
        if ret :
            print ("result = ", result )
        else :
            print ("err_msg = ", result ["message"])
```

注意： 本命令只支持在 remote 模式下使用。
本命令适用于 v2.14.4 及以上版本。

2.2.2.37 获取直角坐标系下的法兰盘中心位姿

```
{"jsonrpc":"2.0","method":"get_base_flange_pose","params":{"unit_type":
   unit_type},"id":id}
```

功能： 获取直角坐标系下的法兰盘中心位姿

参数： unit_type：int[0,1]，返回位姿的 rx,ry,rz 的单位类型，0：返回角度，1：返回弧度，可选参数，不写默认返回弧度值

返回： 直角坐标系下的法兰盘中心位姿，Double[6]

示例：

```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="192.168.1.202"
  conSuc,sock=connectETController(robot_ip)
  if (conSuc):
    # 获取直角坐标系下的法兰盘中心位姿
    ret , result , id = sendCMD(sock,"get_base_flange_pose",{"unit_type": 0})
    print (" result =", result )
  else :
    print ("连接失败")
  disconnectETController (sock)
```

注意： 本命令适用于 v2.14.4 及以上版本。

2.2.2.38 获取用户坐标系下的法兰盘中心位姿

```
{"jsonrpc":"2.0","method":"get_user_flange_pose","params":{"unit_type":
   unit_type},"id":id}
```

功能： 获取用户坐标系下的法兰盘中心位姿

参数： unit_type：int[0,1]，返回位姿的 rx,ry,rz 的单位类型，0：返回角度，1：返回弧度，可选参数，不写默认返回弧度值

返回： 用户坐标系下的法兰盘中心位姿，Double[6]

示例：
```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="192.168.1.202"
  conSuc,sock=connectETController(robot_ip)
  if (conSuc):
    # 获取用户坐标系下的法兰盘中心位姿
    ret , result , id  = sendCMD(sock,"get_user_flange_pose",{"unit_type": 0})
    print (" result =", result )
  else :
    print ("连接失败")
  disconnectETController (sock)
```

注意： 本命令适用于 v2.14.4 及以上版本。

2.2.2.39 获取机器人子类型

```
{"jsonrpc":"2.0","method":"getRobotSubtype","id":id}
```

功能： 获取机器人子类型

参数： 无

返回： 机器人子类型 int

示例：
```
if __name__ == "__main__":
  ip  = "192.168.1.202"
  conSuc,sock=connectETController(ip)
  if  conSuc:
    # 获取机器人子类型
    suc, result , id=sendCMD(sock,"getRobotSubtype")
    print ( result )
```

注意： 本命令适用于 v2.16.2 及以上版本。

2.2.2.40 获取安全参数使能状态

```
{"jsonrpc":"2.0","method":"getRobotSafetyParamsEnabled","id":id}
```

功能： 获取安全参数使能状态

参数： 无

返回： 关闭 0，打开 1

示例：
```
if __name__ == "__main__":
    ip = "192.168.1.202"
    conSuc, sock = connectETController(ip)
    if conSuc:
        # 获取安全参数使能状态
        suc, result ,id=sendCMD(sock,"getRobotSafetyParamsEnabled")
        print ( result )
```

注意： 本命令适用于 v2.16.2 及以上版本。

2.2.2.41 获取安全功率

```
{"jsonrpc":"2.0","method":"getRobotSafetyPower","id":id}
```

功能： 获取安全功率

参数： 无

返回： 正常模式和缩减模式下的功率值 double

示例：
```
if __name__ == "__main__":
    ip = "192.168.1.202"
    conSuc, sock = connectETController(ip)
    if conSuc:
        # 获取机器人安全功率
        suc, result ,id=sendCMD(sock,"getRobotSafeyPower")
        print ( result )
```

注意： 本命令适用于 v2.16.2 及以上版本。

2.2.2.42 获取安全动量

```
{"jsonrpc":"2.0","method":"getRobotSafetyMomentum","id":id}
```

功能： 获取安全动量

参数： 无

返回： 正常模式和缩减模式下的动量值 double

示例：
```
if __name__ == "__main__":
    ip = "192.168.1.202"
    conSuc, sock = connectETController(ip)
    if conSuc:
        # 获取机器人安全动量
        suc, result , id =sendCMD(sock,"getRobotSafetyMomentum")
        print ( result )
```

注意： 本命令适用于 v2.16.2 及以上版本。

2.2.2.43 获取安全工具力

```
{"jsonrpc":"2.0","method":"getRobotSafetyToolForce","id":id}
```

功能： 获取安全工具力

参数： 无

返回： 正常模式和缩减模式下的工具力 double

示例：
```
if __name__ == "__main__":
    ip = "192.168.1.202"
    conSuc, sock = connectETController(ip)
    if conSuc:
        # 获取机器人安全工具力
        suc, result , id =sendCMD(sock,"getRobotSafetyToolForce")
        print ( result )
```

注意： 本命令适用于 v2.16.2 及以上版本。

2.2.2.44 获取安全肘部力

```
{"jsonrpc":"2.0","method":"getRobotSafetyElbowForce","id":id}
```

功能： 获取安全肘部力

参数： 无

返回： 正常模式和缩减模式下的肘部力 double

示例：
```
if __name__ == "__main__":
    ip = "192.168.1.202"
    conSuc, sock = connectETController(ip)
    if conSuc:
        # 获取机器人安全肘部力
        suc, result , id =sendCMD(sock,"getRobotSafetyElbowForce")
        print ( result )
```

注意： 本命令适用于 v2.16.2 及以上版本。

2.2.2.45 获取速度百分比

```
{"jsonrpc":"2.0","method":"getRobotSpeedPercentage","id":id}
```

功能： 获取机器人的速度百分比

参数： 无

返回： 正常模式和缩减模式下的速度百分比 double

示例：
```
if __name__ == "__main__":
    ip = "192.168.1.202"
    conSuc, sock = connectETController(ip)
    if conSuc:
        # 获取机器人速度百分比
        suc, result , id =sendCMD(sock,"getRobotSpeedPercentage")
        print ( result )
```

注意： 本命令适用于 v2.16.2 及以上版本。

2.2.2.46 获取拖动最大启动速度

```
{"jsonrpc":"2.0","method":"getRobotDragStartupMaxSpeed","id":id}
```

功能： 获取拖动最大启动速度

参数： 无

返回： 机器人拖动过程中的拖动最大启动速度 double

示例：
```
if __name__ == "__main__":
    ip = "192.168.1.202"
    conSuc, sock = connectETController(ip)
    if conSuc:
        # 获取拖动最大启动速度
        suc, result , id =sendCMD(sock,"getRobotDragStartupMaxSpeed")
        print ( result )
```

注意： 本命令适用于 v2.16.2 及以上版本。

2.2.2.47 获取最大力矩误差百分比

```
{"jsonrpc":"2.0","method":"getRobotTorqueErrorMaxPercents","id":id}
```

功能： 获取机器的最大力矩误差百分比

参数： 无

返回： 机器人力控参数中的最大力矩误差百分比 double

示例：
```
if __name__ == "__main__":
    ip = "192.168.1.202"
    conSuc, sock = connectETController(ip)
    if conSuc:
        # 获取机器人拖动最大启动速度
        suc, result , id =sendCMD(sock,"getRobotTorqueErrorMaxPercents")
        print ( result )
```

注意： 本命令适用于 v2.16.2 及以上版本。

2.2.2.48 设置末端按钮状态

```
{"jsonrpc":"2.0","method":"setFlangeButton","params":{"button_num":
   button_num,"state":state},"id":id}
```

功能： 设置末端按钮状态

参数： button_num：按钮，int[0,1]，0：蓝色按钮，1：绿色按钮
state：状态，int[0,2]，0：禁用，1：拖动，2：记点

返回： 成功 true，失败 false

示例：
```
if __name__ == "__main__":
    ip = "192.168.1.202"
    conSuc, sock = connectETController(ip)
    if conSuc:
        # 获取机器人拖动最大启动速度
        suc, result ,id =sendCMD(sock,"setFlangeButton",{"button_num":0,"state":1})
        print ( result )
```

注意： 本命令只支持在 remote 模式下使用。
本命令适用于 v2.16.2 及以上版本。

2.2.2.49 获取末端按钮状态

```
{"jsonrpc":"2.0","method":"checkFlangeButton","params":{"button_num":
    button_num},"id":id}
```

功能： 获取末端按钮状态

参数： button_num：按钮，int[0,1]，0：蓝色按钮，1：绿色按钮

返回： 禁用 0，拖动 1，记点 2

示例：
```
if __name__ == "__main__":
    ip = "192.168.1.202"
    conSuc, sock = connectETController(ip)
    if conSuc:
        # 获取末端按钮状态
        suc, result ,id=sendCMD(sock,"checkFlangeButtonFlangeButton,{"button_num":0})
        print （suc， result ， id）
```

注意： 本命令适用于 v2.16.2 及以上版本。

2.2.2.50 获取碰撞检测使能状态

```
{"jsonrpc":"2.0","method":"get_collision_enable_status","id":id}
```

功能： 获取碰撞检测使能状态

参数： 无

返回： 0：未使能，1：使能

示例：
```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="192.168.1.202"
  conSuc,sock=connectETController(robot_ip)
  if (conSuc):
    # 获取碰撞检测使能状态
    suc， result， id = sendCMD(sock, " get_collision_enable_status ")
    print ( result )
  else :
    print ("连接失败")
  disconnectETController (sock)
```

2.2.2.51 获取碰撞灵敏度

```
{"jsonrpc": "2.0", "method": "getCollisionSensitivity", "id": id}
```

功能： 获取碰撞灵敏度

参数： 无

返回： 碰撞灵敏度 int

示例：
```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip = "192.168.1.202"
  conSuc, sock = connectETController(robot_ip)
  if (conSuc):
    # 获取碰撞灵敏
    suc， result， id = sendCMD(sock, " getCollisionSensitivity ")
    print ( result )
  else :
    print ("连接失败")
  disconnectETController (sock)
```

2.2.2.52 获取当前 tcp 在当前用户坐标系下的位姿

```
{"jsonrpc":"2.0","method":"getTcpPoseInUser","params":{"unit_type":
   unit_type},"id":id}
```

功能： 获取当前 tcp 在当前用户坐标系下的位姿

参数： unit_type：int[0,1]，可选参数, 返回 pose 的 rx,ry,rz 的单位类型，0：角度，1：弧度，不写默认为弧度值。

返回： 当前 tcp 在用户坐标系下的位姿

示例：

```
if __name__ == "__main__":
    # 机器人IP地址
    robot_ip="192.168.1.202"
    conSuc,sock=connectETController(robot_ip)
    res = [100, 0, 0, 0, 0, 0, 0, 0]
    if (conSuc):
        # 获取当前tcp在当前用户坐标系下的位姿信息
        suc, result , id = sendCMD(sock, "getTcpPoseInUser",{"unit_type": 0})
        print ( result )
    else :
        print ("连接失败")
    disconnectETController (sock)
```

2.2.2.53 获取机械臂本体异常情况

```
{"jsonrpc":"2.0","method":"getAlarmNum","id":id}
```

功能： 获取机械臂本体异常情况

参数： 无

返回： 成功返回最近 5 条机器人报警编号的字符串，失败返回 false

示例：

```
if __name__ == "__main__":
    # 机器人IP地址
    robot_ip = "192.168.1.202"
    conSuc, sock = connectETController(robot_ip)
    if (conSuc):
        # 获取机械臂本体异常情况
        suc, result , id = sendCMD(sock, "getAlarmNum")
        print ( result )
    else :
        print ("连接失败")
    disconnectETController (sock)
```

2.2.2.54 获取关节运动速度

```
{"jsonrpc":"2.0","method":"get_joint_speed","id":id}
```

功能： 获取关节运动速度

参数： 无

返回： 关节运动速度 double speed[8], 单位：度/s

示例：

```
if __name__ == "__main__":
    # 机器人IP地址
    robot_ip = "192.168.1.202"
    conSuc, sock = connectETController(robot_ip)
    if (conSuc):
        while 1:
            suc, result , id = sendCMD(sock, "get_joint_speed")
            print(suc, result , id)
            time.sleep(0.001)
        else:
            print("连接失败")
    disconnectETController(sock)
```

2.2.2.55 获取 tcp 加速度

```
{"jsonrpc":"2.0","method":"get_tcp_acc","id":id}
```

功能： 获取 tcp 加速度

参数： 无

返回： tcp 运动加速度 double tcp_acc, 单位 mm/s^2

示例：

```
if __name__ == "__main__":
    # 机器人IP地址
    robot_ip = "192.168.1.202"
    conSuc, sock = connectETController(robot_ip)
    if (conSuc):
        while 1:
            suc, result , id = sendCMD(sock, "get_tcp_acc")
            print(suc, result , id)
            time.sleep(0.01)
    else:
        print("连接失败")
    disconnectETController(sock)
```

2.2.2.56 获取关节加速度

```
{"jsonrpc":"2.0","method":"get_joint_acc","id":id}
```

功能： 获取关节加速度

参数： 无

返回： 关节运动加速度 double joint_acc[8]，单位：度/s^2

示例：

```
if __name__ == "__main__":
    # 机器人IP地址
    robot_ip = "192.168.1.202"
    conSuc, sock = connectETController(robot_ip)
    if (conSuc):
        while 1:
            suc, result , id = sendCMD(sock, "get_joint_acc")
            print(suc, result , id)
            time.sleep(0.01)
    else:
        print("连接失败")
    disconnectETController(sock)
```

2.2.2.57 获取 tcp 运动速度

```
{"jsonrpc": "2.0", "method": "get_tcp_speed", "id": id}
```

功能： 获取 tcp 运动速度

参数： 无

返回： 当前 tcp 运动速度 double tcp_speed，单位：毫米/秒

示例：

```
if __name__ == "__main__":
    # 机器人IP地址
    robot_ip = "192.168.1.202"
    conSuc, sock = connectETController(robot_ip)
    if (conSuc):
        # 获取tcp运动速度
        suc, result , id = sendCMD(sock, "get_tcp_speed")
        print( result )
    else:
        print("连接失败")
    disconnectETController(sock)
```

2.2.2.58 获取机器人的紧急停止状态

```
{"jsonrpc":"2.0","method":"get_estop_status","id":id}
```

功能： 获取机器人的紧急停止状态

参数： 无

返回： int[0,1]，机器人是否处于紧急停止状态，1: 紧急停止，0：非紧急停止

示例：
```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip = "192.168.1.202"
  conSuc, sock = connectETController(robot_ip)
  if (conSuc):
    suc, result , id = sendCMD(sock,"get_estop_status")
    print ( result )
```

2.2.2.59 获取模拟量输出电压值

```
{"jsonrpc":"2.0","method":"get_analog_output","params":{"addr":addr},"id
   ":id}
```

功能： 获取模拟量输出电压值

参数： addr：模拟量输出地址，范围：int [0,4]

返回： 模拟量输出电压值，double

示例：
```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="192.168.1.202"
  conSuc,sock=connectETController(robot_ip)
  if (conSuc):
    for i in range(0,5):
      # 获取模拟量输出电压值
      suc, result , id = sendCMD(sock,"get_analog_output",{"addr":i})
      print ( result )
```

2.2.2.60 获取工具负载和质心

```
{"jsonrpc":"2.0","method":"get_tool_payload","params":{"tool_num":
   tool_num},"id":id}
```

功能： 获取工具负载和质心

参数： tool_num：工具号, 范围 int[0,7]

返回： m: 工具负载质量,double

tool_cog : 工具负载质心，double cog[3]

示例：

```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="192.168.1.202"
  conSuc,sock=connectETController(robot_ip)
  if (conSuc):
    for i in range(0,8):
      # 获取工具负载和质心
        suc, result , id = sendCMD(sock, "get_tool_payload",{"tool_num": i})
        print( result )
  else:
    print ("连接失败")
  disconnectETController(sock)
```

2.2.2.61 获取机器人输入端关节位置信息

```
{"jsonrpc":"2.0","method":"get_motor_pos","id":id}
```

功能： 获取机器人输入端关节位置信息

参数： 无

返回： double pos[8]: 机器人输入端关节位置信息

示例：

```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip = "192.168.1.202"
  conSuc, sock = connectETController(robot_ip)
  if (conSuc):
    # 获取机器人输入端关节位置信息
    suc, result , id = sendCMD(sock, "get_motor_pos")
    print( result )
  else:
    print ("连接失败")
  disconnectETController(sock)
```

2.2.3 运动服务 (MovementService)

2.2.3.1 关节运动

```
{"jsonrpc":"2.0","method":"moveByJoint","params":{"targetPos":targetPos,"
   speed":speed,"acc":acc,"dec":dec,"cond_type":cond_type,"cond_num:":
   cond_num:,"cond_value":cond_value},"id":id}
```

功能： 关节运动

参数： targetpos：目标关节点 double pos[8]，范围为 [-360,360]
speed：运行速度，范围：double[0.01,100]
cond_type：IO 类型, 0 为数字量输入 X，1 为数字量输出 Y, 范围 int[0,1]
cond_num: IO 地址，范围 int[0,63]
cond_value: IO 状态，范围 int[0,1], 实际 IO 状态与该值一致时，立即放弃本次未完成的运动，执行下一条指令。
acc：加速度百分比，范围：int [1,100], 可选参数, 不写默认值为 0。
dec：减速度百分比，范围：int [1,100], 可选参数, 不写默认值为 0

返回： 成功 true，失败 false

示例：

```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="192.168.1.200"
  conSuc, sock = connectETController(robot_ip)
  point = []
  point.append([0.0065,-103.9938,102.2076,-88.2138,
   90.0000,0.0013,0.0000,0.0000])
  point.append([-16.2806,-82.4996,81.9848,-89.4851,
   90.0000,-16.2858,0.0000,0.0000])
  point.append([3.7679, -71.7544, 68.7276, -86.9732,
  90.0000, 3.7627, 0.0000, 0.0000])
  point.append([12.8237,-87.3028,87.2361,-89.9333,
   90.0000,12.8185,0.0000,0.0000])
  if(conSuc):
    # 获取机械臂伺服状态
    suc, result, id = sendCMD(sock, "getServoStatus")
    if(result == 0):
      # 设置机械臂伺服状态ON
      suc, result, id = sendCMD(sock,"set_servo_status",{"status":1})
      time.sleep(1)
      for i in range(4):
        # 关节运动
        suc, result ,id=sendCMD(sock,"moveByJoint",{"targetPos":point[i],"speed":30,"acc":10,"dec
            ":10,"cond_type":0,"cond_num":7,"cond_value":1})
        while(True):
          # 获取机器人状态
          suc, result, id = sendCMD(sock, "getRobotState")
          if (result == 0):
            break
```

2.2.3.2 直线运动

```
{"jsonrpc":"2.0","method":"moveByLine","params":{"targetPos":targetPos,"
   speed_type":speed_type,"speed":speed,"acc":acc,"dec":dec,"cond_type":
   cond_type,"cond_num:":cond_num:,"cond_value ":cond_value},"id":id}
```

功能： 直线运动

参数： targetpos：目标关节点 double pos[8]，范围为 [-360,360]
speed: 运行速度。double，类型为直线速度范围：1-3000；为旋转角速度，范围：1-300；为绝对直线速度，范围：直线最小速度参数值-直线最大速度参数值；为绝对旋转角速度，范围：旋转角最小速度参数值-旋转角最大速度参数值
speed_type: 速度类型，0 为 V(直线速度)，1 为 VR(旋转角速度)，2 为 AV(绝对直线速度)，3 为 AVR(绝对旋转角速度)。可选。
cond_type：IO 类型, 0 为数字量输入 X，1 为数字量输出 Y, 范围 int[0,1]
cond_num: IO 地址，范围 int[0,63]
cond_value: IO 状态，范围 int[0,1], 实际 IO 状态与该值一致时，立即放弃本次未完成的运动，执行下一条指令
acc：加速度百分比，范围：int [1,100], 可选参数, 不写默认值为 0。
dec：减速度百分比，范围：int [1,100], 可选参数, 不写默认值为 0

返回： 成功 true，失败 false

示例：

```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="192.168.1.205"
  conSuc, sock = connectETController(robot_ip)
  point = []
  point.append([0.0065,-103.9938,102.2076,-88.2138,
   90.0000,0.0013,0.0000,0.0000])
  point.append([-16.2806,-82.4996,81.9848,-89.4851,
   90.0000,-16.2858,0.0000,0.0000])
  point.append([3.7679, -71.7544, 68.7276, -86.9732,
  90.0000, 3.7627, 0.0000, 0.0000])
  point.append([12.8237,-87.3028,87.2361,-89.9333,
   90.0000,12.8185,0.0000,0.0000])
  if(conSuc):
    # 设置机械臂伺服状态ON
    suc, result, id = sendCMD(sock,"set_servo_status",{"status":1})
    time.sleep(1)
    for i in range(4):
      # 直线运动
      suc, result ,id=sendCMD(sock,"moveByLine",{"targetPos":point[i],"speed_type":0,"speed":200,
          "cond_type":0,"cond_num":7,"cond_value":1})
      while(True):
        # 获取机器人状态
        suc, result, id = sendCMD(sock, "getRobotState")
        if (result == 0):
          break
```

注意： 无 speed_type 参数时，speed 表示为绝对直线速度。

2.2.3.3 圆弧运动

```
{"jsonrpc":"2.0","method":"moveByArc","params":{"midPos":midPos,"
   targetPos":targetPos,"speed_type":speed_type,"speed":speed"cond_type":
   cond_type,"cond_num:":cond_num:,"cond_value":cond_value},"id":id}
```

功能： 圆弧运动

参数： middlepos：中间关节点

targetpos：目标关节点 double pos[8]，范围为 [-360,360]

speed: 运行速度。double，类型为直线速度范围：[1,3000]；为旋转角速度，范围：[1,300]；为绝对直线速度，范围：直线最小速度参数值-直线最大速度参数值；为绝对旋转角速度，范围：旋转角最小速度参数值-旋转角最大速度参数值

speed_type: 速度类型，int[0,3]，0 为 V(直线速度)，1 为 VR(旋转角速度)，2 为 AV(绝对直线速度)，3 为 AVR(绝对旋转角速度)。可选。

cond_type：IO 类型, 0 为数字量输入 X，1 为数字量输出 Y, 范围 int[0,1]

cond_num: IO 地址，范围 int[0,63]

cond_value: IO 状态，范围 int[0,1], 实际 IO 状态与该值一致时，立即放弃本次未完成的运动，执行下一条指令。

acc：加速度百分比，范围：int [1,100], 可选参数, 不写默认值为 0。

dec：减速度百分比，范围：int [1,100], 可选参数, 不写默认值为 0。

返回： 成功 true，失败 false

示例：

```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  P000 = [0.0065,-103.9938,102.2076,-88.2138,
   90.0000,0.0013,0.0000,0.0000]
  P001 = [-16.2806,-82.4996,81.9848,-89.4851,
   90.0000,-16.2858,0.0000,0.0000]
  if (conSuc):
    # 获取机械臂伺服状态
    suc, result ,id=sendCMD(sock,"getServoStatus")
    if ( result  == 0):
      # 设置机械臂伺服状态ON
      suc, result ,id=sendCMD(sock,"set_servo_status",{" status ":1})
      time.sleep(1)
    # 圆弧运动
    suc, result ,id=sendCMD(sock,"moveByArc",{"midPos":P000,"targetPos":P001,"speed_type":0,"
        speed":20,"cond_type":0,"cond_num":7,"cond_value":1})
```

注意： 无 speed_type 参数时，speed 表示为绝对直线速度。

2.2.3.4 旋转运动

```
{"jsonrpc":"2.0","method":"moveByRotate","params":{"targetPos":targetPos
   ,"speed_type":speed_type,"speed":speed"cond_type":cond_type,"cond_num
   :":cond_num:,"cond_value":cond_value},"id":id}
```

功能： 旋转运动

参数： targetpos：目标关节点 double pos[8]，范围为 [-360,360]
speed: 运行速度。double，类型为直线速度范围：[1,3000]；为旋转角速度，范围：[1,300]；为绝对直线速度，范围：直线最小速度参数值-直线最大速度参数值；为绝对旋转角速度，范围：旋转角最小速度参数值-旋转角最大速度参数值
speed_type: 速度类型，int[0,3]，0 为 V(直线速度)，1 为 VR(旋转角速度)，2 为 AV(绝对直线速度)，3 为 AVR(绝对旋转角速度)。可选。
cond_type：IO 类型, 0 为数字量输入 X，1 为数字量输出 Y, 范围 int[0,1]
cond_num: IO 地址，范围 int[0,63]
cond_value: IO 状态，范围 int[0,1], 实际 IO 状态与该值一致时，立即放弃本次未完成的运动，执行下一条指令。
acc：加速度百分比，范围：int [1,100], 可选参数, 不写默认值为 0。
dec：减速度百分比，范围：int [1,100], 可选参数, 不写默认值为 0。

返回： 成功 true，失败 false

示例：

```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  P000 = [0.0065,-103.9938,102.2076,-88.2138,
   90.0000,0.0013,0.0000,0.0000]
  if(conSuc):
    # 获取机械臂伺服状态
    suc, result ,id=sendCMD(sock,"getServoStatus")
    if( result  == 0):
      # 设置机械臂伺服状态ON
      suc, result ,id=sendCMD(sock,"set_servo_status",{" status ":1})
      time.sleep(1)
    # 旋转运动
    suc, result ,id=sendCMD(sock,"moveByRotate",{"targetPos":P000,"speed_type":0,"speed":20,"
        cond_type":0,"cond_num":7,"cond_value":1})
```

注意： 无 speed_type 参数时，speed 表示为绝对旋转角速度。
本命令不推荐使用。

提醒



以上命令只支持在 remote 模式下使用。
cond_type、cond_num 和 cond_value 为可选参数，适用于 2.14.4 及以上版本。
speed_type、acc、dec，适用于 2.16.2 及以上版本，均为可选参数。
执行以上命令前，请确保机器人处于停止状态。如果机器人正在运行，先发 stop 命令，等待机器人停止。

2.2.3.5 添加路点信息 2.0

```
{"jsonrpc":"2.0","method":"addPathPoint","params":{"wayPoint":wayPoint,"
   moveType":moveType,"speed_type":speed_type,"speed":speed,"smooth":
   smooth"cond_type":cond_type,"cond_num:":cond_num:,"cond_value":
   cond_value},"id":id}
```

功能： 添加路点信息 2.0

参数： waypoint：目标位置

moveType：0 关节运动，1 直线运动，2 绕工具尖端点旋转运动，3 圆弧运动

speed: 运行速度。double，关节运动时，关节速度范围：[1,100]。直线、旋转、圆弧运动时，类型为直线速度范围：[1,3000]；为旋转角速度，范围：[1,300]；为绝对直线速度，范围：直线最小速度参数值-直线最大速度参数值；为绝对旋转角速度，范围：旋转角最小速度参数值-旋转角最大速度参数值

无 speed_type 参数时，表示为：运动速度，关节运动速度范围 [1,100], 直线及圆弧速度范围 [1,3000], 旋转运动速度范围 [1,300]

speed_type: 速度类型，int[0,3]，0 为 V(直线速度)，1 为 VR(旋转角速度)，2 为 AV(绝对直线速度)，3 为 AVR(绝对旋转角速度)。可选。

smooth：平滑度，范围：int[0,7]

cond_type：IO 类型, 可选参数，0 为数字量输入 X，1 为数字量输出 Y, 范围 int[0,1]

cond_num: IO 地址，可选参数，范围 int[0,63]

cond_value: IO 状态，可选参数，范围 int[0,1], 实际 IO 状态与该值一致时，立即放弃本次未完成的运动，执行下一条指令

返回： 成功 true，失败 false

示例：

```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  C000 = [0.0065,-103.9938,102.2076,-88.2138,
   90.0000,0.0013,0.0000,0.0000]
  C001 = [-16.2806,-82.4996,81.9848,-89.4851,
   90.0000,-16.2858,0.0000,0.0000]
  C002 = [3.7679, -71.7544, 68.7276, -86.9732,
  90.0000, 3.7627, 0.0000, 0.0000]
  C003 = [12.8237,-87.3028,87.2361,-89.9333,
   90.0000,12.8185,0.0000,0.0000]
  # 清除路点信息2.0
  suc, result , id = sendCMD(sock, "clearPathPoint")
  if( result == True):
    # 添加路点信息2.0
    suc, result ,id=sendCMD(sock,"addPathPoint",{"wayPoint":C000,"moveType":0,"speed":50,"smooth
        ":7})
    suc, result ,id=sendCMD(sock,"addPathPoint",{"wayPoint":C001,"moveType":1,"speed_type":0,"
        speed":50,"smooth":7})
    suc, result ,id=sendCMD(sock,"addPathPoint",{"wayPoint":C002,"moveType":1,"speed_type":0,"
        speed":50,"smooth":7})
    suc, result ,id=sendCMD(sock,"addPathPoint",{"wayPoint":C003,"moveType":1,"speed_type":0,"
        speed":50,"smooth":7})
```

注意： 本命令只支持在 remote 模式下使用。

若运动类型为关节运动，则 speed_type 参数无效，不推荐使用。

2.2.3.6 清除路点信息 2.0

```
{"jsonrpc":"2.0","method":"clearPathPoint","id":id}
```

功能： 清除路点信息 2.0

参数： 无

返回： 成功 true，失败 false

示例：

```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if (conSuc):
    # 清除路点信息2.0
    suc， result ， id = sendCMD(sock, "clearPathPoint")
```

注意： 本命令只支持在 remote 模式下使用。

2.2.3.7 轨迹运动 2.0

```
{"jsonrpc":"2.0","method":"moveByPath","id":id}
```

功能： 轨迹运动 2.0

参数： 无

返回： 失败：-1, 成功：路点总个数

示例：

```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  C000 = [0.0065,-103.9938,102.2076,-88.2138,
   90.0000,0.0013,0.0000,0.0000]
  C001 = [-16.2806,-82.4996,81.9848,-89.4851,
   90.0000,-16.2858,0.0000,0.0000]
  C002 = [3.7679, -71.7544, 68.7276, -86.9732,
  90.0000, 3.7627, 0.0000, 0.0000]
   if (conSuc):
     # 清除路点信息2.0
     suc, result , id = sendCMD(sock, "clearPathPoint")
     if ( result == True):
       # 添加路点信息2.0
       suc, result , id = sendCMD(sock, "addPathPoint", {"wayPoint": C000,"moveType": 0, "speed":
            50, "smooth": 7})
       suc, result , id = sendCMD(sock, "addPathPoint", {"wayPoint": C001,"moveType":0, "speed":
           50, "smooth": 7})
       suc, result , id = sendCMD(sock, "addPathPoint", {"wayPoint": C002,"moveType": 0, "speed":
            50, "smooth": 7})
       # 轨迹运动2.0
       suc, result , id = sendCMD(sock, "moveByPath")
       while(True):
         # 获取 trackfile 文件运行的行号(与示教器显示行号一致)
         suc, result , id = sendCMD(sock, "getPathPointIndex")
         print ( result )
         # 获取机器人状态
         suc, result , id = sendCMD(sock, "getRobotState")
         if ( result == 0):
           break
```

注意： 本命令只支持在 remote 模式下使用。
执行此命令前，请确保机器人处于停止状态。如果机器人正在运行，先发 stop 命令，等待机器人停止。

2.2.3.8 jog 运动

```
{"jsonrpc":"2.0","method":"jog","params":{"index":index,"speed":speed},"
   id":id}
```

功能： jog 运动

参数： index：轴方向或者坐标系方向编号，范围：int[0,11]
speed: 手动速度百分比，范围 double [0.05,100] (可选参数，非必填)

返回： 成功 true，失败 false

示例：

```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if (conSuc):
    # 获取机械臂伺服状态
    suc, result , id = sendCMD(sock, "getServoStatus")
    if( result == 0):
      # 设置机械臂伺服状态ON
      suc, result ,id=sendCMD(sock,"set_servo_status",{" status ":1})
      time.sleep (1)
    # 指定坐标系
    suc, result ,id=sendCMD(sock, "setCurrentCoord", {"coord_mode":1})
    for i in range (0, 10, 1):
      # x轴负方向jog运动
      suc, result , id = sendCMD(sock, "jog", {"index" :0,"speed":10})
      print (suc, result , id)
      time.sleep (0.1)
    suc, result ,id = sendCMD(sock, "stop")
```

注意： 停止发送 jog 命令之后，机器人并不会立刻停止，而是需要通过下文的" 停止机器人运行" 命令来使机器人立刻停止。
本命令只支持在 remote 模式下使用。
超过 1 秒未接收到下一条 jog 运动指令，停止接收 jog 指令，机器人 jog 运动停止

2.2.3.9 停止机器人运行

```
{"jsonrpc":"2.0","method":"stop","id":id}
```

功能： 停止机器人运行

参数： 无

返回： 成功 true，失败 false

示例：
```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if(conSuc):
    # 机器人停止
    suc， result， id = sendCMD(sock, "stop")
```

注意： 本命令只支持在 remote 模式下使用。

2.2.3.10 机器人自动运行

```
{"jsonrpc":"2.0","method":"run","id":id}
```

功能： 机器人自动运行

参数： 无

返回： 成功 true，失败 false

示例：
```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if(conSuc):
    # 机器人暂停
    suc， result， id = sendCMD(sock, "pause")
    time.sleep(1)
    # 机器人启动
    suc， result， id = sendCMD(sock, "run")
```

注意： 本命令只支持在 remote 模式下使用。

2.2.3.11 机器人暂停

```
{"jsonrpc":"2.0","method":"pause","id":id}
```

功能： 机器人暂停

参数： 无

返回： 成功 true，失败 false

示例：
```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if (conSuc):
    # 机器人暂停
    suc， result ， id = sendCMD(sock, "pause") time.sleep(1)
```

注意： 本命令只支持在 remote 模式下使用。

2.2.3.12 检查 jbi 文件是否存在

```
{"jsonrpc":"2.0","method":"checkJbiExist","params":{"filename":filename
   },"id":id}
```

功能： 检查 jbi 文件是否存在

参数： filename：待检查文件名，字符串

返回： 0: 不存在，1: 存在

示例：
```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  jbi_filename ="test"
  if (conSuc):
    # 检查jbi文件是否存在
    suc, result ,id=sendCMD(sock,"checkJbiExist",{"filename": jbi_filename })
```

2.2.3.13 运行 jbi 文件

```
{"jsonrpc":"2.0","method":"runJbi","params":{"filename":filename},"id":id
   }
```

功能： 运行 jbi 文件

参数： filename：待运行文件名，字符串

返回： 成功 true，失败 false

示例：

```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  jbi_filename ="test"
  if (conSuc):
    # 检查jbi文件是否存在
    suc, result ,id=sendCMD(sock,"checkJbiExist",{"filename": jbi_filename })
    if (suc and result ==1):
      # 运行jbi文件
      suc, result ,id=sendCMD(sock,"runJbi",{"filename": jbi_filename })
```

注意： 本命令只支持在 remote 模式下使用。
执行此命令前，请确保机器人处于停止状态。如果机器人正在运行，先发 stop 命令，等待机器人停止。

2.2.3.14 获取 jbi 文件运行状态

```
{"jsonrpc":"2.0","method":"getJbiState","id":id}
```

功能： 获取 jbi 文件运行状态

参数： 无

返回： jbiName: 文件名

runState:0 停止状态,1 暂停状态,2 急停状态,3 运行状态,4 错误状态

示例：

```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  jbi_filename ="test"
  if (conSuc):
    # 检查jbi文件是否存在
    suc, result ,id=sendCMD(sock,"checkJbiExist",{"filename": jbi_filename })
    if (suc and result ==1):
      # 运行jbi文件
      suc, result ,id=sendCMD(sock,"runJbi",{"filename": jbi_filename })
      if (suc and result ):
        checkRunning=3
        while(checkRunning==3):
          # 获取jbi文件运行状态
          suc, result ,id=sendCMD(sock,"getJbiState")
          checkRunning=result["runState"]
          time.sleep (0.1)
```

2.2.3.15 设置机器人运行速度

```
{"jsonrpc":"2.0","method":"setSpeed","params":{"value":value},"id":id}
```

功能： 设置机器人运行速度

参数： value：速度，范围：double [0.05,100]

返回： 成功 true，失败 false

示例：

```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if (conSuc):
    # 设置机器人运行速度为30%
    suc, result , id = sendCMD(sock, "setSpeed", {"value": 30})
  else:
    print ("连接失败")
  disconnectETController(sock)
```

注意： 本命令适用于 v2.13.1 及以上版本。
本命令只支持在 remote 模式下使用。

2.2.3.16 关节匀速运动

```
{"jsonrpc":"2.0","method":"moveBySpeedj","params":{"vj":vj,"acc":acc,"t":
   t}," id":id}
```

功能： 关节匀速运动

参数： vj：double 型，8 个关节的速度值，八位数单位：度/秒
acc：关节加速度，int，范围：大于 0，单位：度/s^2
t：SPEEDJ 执行时间，double，范围：大于 0，单位,：秒

返回： 成功 true，失败 false

示例：
```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  speed_j    =[1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0]
  if(conSuc):
    # 关节匀速运动
    suc, result ,id=sendCMD(sock,"moveBySpeedj",{"vj":speed_j,"acc":20,"t":5})
    print(suc, result , id)
```

注意： 本命令只支持在 remote 模式下使用。
moveBySpeedj 运动过程中，发送多条 moveBySpeedj 或 moveBySpeedl 指令，或发送 stopj 指令时，机器人在执行完 moveBySpeedj 指令后，不减速，继续执行运动过程中用户发送的最后一条指令。

2.2.3.17 停止关节匀速运动

```
{"jsonrpc":"2.0","method":"stopj","params":{"acc":acc},"id":id}
```

功能： 停止关节匀速运动

参数： acc：int，关节加速度，以此加速度停止运动，单位：度/s^2，范围：大于 0

返回： 成功 true，失败 false

示例：
```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if(conSuc):
    suc, result ,id=sendCMD(sock,"moveBySpeedj",{"vj" :[20,0,0,0,0,0,0,0], "acc" :50, "t" :2})
    print(suc, result ,id)
    time.sleep(1)
    suc, result ,id=sendCMD(sock,"stopj",{"acc":10})
    print( result )
```

2.2.3.18 直线匀速运动

```
{"jsonrpc":"2.0","method":"moveBySpeedl","params":{"v":v,"acc":acc,"arot
    ":arot,"t":t},"id":id}
```

功能： 直线匀速运动

参数： v：double 型，沿 6 个方向运动的速度值，单位：前三个为毫米/秒，后三个为度/秒
acc：位移加速度，int，范围：大于 0，单位：mm/s^2
arot：可选参数，int，姿态加速度，范围：大于 0，单位：度/s^2
t：SPEEDJ 执行时间，double，范围：大于 0，单位：秒

返回： 成功 true，失败 false

示例：
```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  speed_l  =[1.0,1.0,1.0,1.0,1.0,1.0]
  if (conSuc):
    # 直线匀速运动
    suc, result , id=sendCMD(sock,"moveBySpeedl",{"v":speed_l,"acc":100,"arot":10,"t" :3.0})
    print (suc, result , id)
```

注意： 本命令只支持在 remote 模式下使用。
moveBySpeedl 运动过程中，发送多条 moveBySpeedj 或 moveBySpeedl 指令，或发送 stopl 指令时，机器人在执行完 moveBySpeedl 指令后，不减速，继续执行运动过程中用户发送的最后一条指令。

2.2.3.19 停止直线匀速运动

```
{"jsonrpc":"2.0","method":"stopl","params":{"acc":acc},"id":id}
```

功能： 停止直线匀速运动

参数： acc：int，加速度，以此加速度停止运动，单位：mm/s^2 或度/s^2，范围：大于 0

返回： 成功 true，失败 false

示例：

```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if (conSuc):
    suc, result ,id=sendCMD(sock,"moveBySpeedl",{"v":[20,0,0,0,0,0],"acc":50,"arot":0,"t":2})
    print (suc, result ,id)
    time.sleep(1)
    suc, result ,id=sendCMD(sock,"stopl",{"acc":10})
    print ( result )
```

2.2.3.20 指定坐标系下直线运动

```
{"jsonrpc":"2.0","method":"moveByLineCoord","params":{"targetUserPose":
  targetUserPose,"speed_type":speed_type,"speed":speed,"acc":acc,"dec":
  dec,"user_coord":user_coord,"cond_type":cond_type,"cond_num:":cond_num
  :,"cond_value":cond_value,"unit_type":unit_type},"id":id}
```

功能： 指定坐标系下直线运动

参数： targetUserPose：指定用户坐标系下的位姿，其中 rx,ry,rz 为弧度，范围：double[-π,π] 或角度，范围：double[-180,180]
speed: 运行速度。double，类型为直线速度范围：[1,3000]；为旋转角速度，范围：[1,300]；为绝对直线速度，范围：直线最小速度参数值-直线最大速度参数值；为绝对旋转角速度，范围：旋转角最小速度参数值-旋转角最大速度参数值
speed_type: 速度类型，可选参数，int [0,3]，0 为 V(直线速度)，1 为 VR(旋转角速度)，2 为 AV(绝对直线速度)，3 为 AVR(绝对旋转角速度)。
user_coord: 用户坐标系数据，double[6], 其中 rx,ry,rz 为弧度，范围：double[-π,π] 或角度，范围：double[-180,180]，不写当前坐标系。
cond_type：IO 类型，可选参数，0 为数字量输入 X，1 为数字量输出 Y, 范围 int[0,1]
cond_num: IO 地址，可选参数，范围 int[0,63]
cond_value: IO 状态，可选参数，范围 int[0,1], 实际 IO 状态与该值一致时，立即放弃本次未完成的运动，执行下一条指令。
acc：加速度百分比，范围：int [1,100], 可选参数, 不写默认值为 0。
dec：减速度百分比，范围：int [1,100], 可选参数, 不写默认值为 0。
unit_type：用户坐标和用户坐标系的 rx,ry,rz 的单位类型，int [0,1]，0：角度，1：弧度，可选参数，不写默认为弧度值。

返回： 成功 true，失败 false

示例：

```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  point =[211,126,343,-2.58,-0.013,-1.813]
  if (conSuc):
    # 指定坐标系下直线运动
    suc, result , id=sendCMD(sock,"moveByLineCoord",{"targetUserPose": point,"user_coord"
          :[0,0,0,0,0,0], "speed_type":1,"speed":30,"unit_type":1})
    print (suc, result , id)
```

注意： 本命令适用于 v2.16.2 及以上版本。
本命令只支持在 remote 模式下使用。
无 speed_type 参数时，speed 表示为绝对直线速度。

2.2.4 运动学服务 (KinematicsService)

2.2.4.1 逆解函数

```
{"jsonrpc":"2.0","method":"inverseKinematic","params":{"targetPose":
   targetPose,"referencePos":referencePos,"unit_type":unit_type},"id":id}
```

功能： 逆解函数，带参考点位置逆解，根据位姿信息得到对应的机械臂关节角信息

参数： targetPose：目标位姿信息，rx,ry,rz 的范围：弧度为 [-π,π]，角度为 [-180,180]
referencePos：逆解参考点关节角信息 double pos[8]，范围为 [-360,360]，可选参数
unit_type：输入位姿的 rx,ry,rz 的单位类型，int [0,1]，0：角度，1：弧度，可选参数，不写默认为弧度值

返回： 关节坐标

示例：
```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="192.168.1.200"
  # 参考点
  P000 = [0, -90, 90, -90, 90, 0, 0, 0]
  conSuc,sock=connectETController(robot_ip)
  if (conSuc):
    # 获取机器人当前位姿信息
    suc, result , id = sendCMD(sock, "get_tcp_pose")
    # 逆解函数2.0,带参考点位置逆解
    suc, result ,id=sendCMD(sock,"inverseKinematic",{"targetPose": result ,"referencePos":P000})
```

2.2.4.2 正解函数

```
{"jsonrpc":"2.0","method":"positiveKinematic","params":{"targetPos":
   targetPos,"unit_type":unit_type},"id":id}
```

功能： 正解函数，根据机械臂关节角信息得到对应的位姿信息

参数： targetpos：目标关节角度信息 double pos[8]，范围为 [-360,360]
unit_type：返回位姿的 rx,ry,rz 的单位类型，int [0,1]，0：返回角度，1：返回弧度，可选参数，不写默认为弧度值

返回： 获取的响应位姿信息:double pose[6]

示例：
```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if (conSuc):
    # 获取机器人当前位置信息
    suc, result , id = sendCMD(sock, "get_joint_pos")
    # 正解函数
    suc, result ,id=sendCMD(sock,"positiveKinematic",{"targetPos": result ,"unit_type":1})
```

2.2.4.3 基坐标到用户坐标位姿转化

```
{"jsonrpc":"2.0","method":"convertPoseFromCartToUser","params":{"
   targetPose":targetPose,"userNo":userNo,"unit_type":unit_type},"id":id}
```

功能： 基坐标到用户坐标位姿转化函数，当前用户坐标系下，根据基坐标的位姿信息得到对应用户坐标系下的位姿信息

参数： targetPose：基坐标系下的位姿信息，double pose[6]，rx,ry,rz 的范围：弧度为 [-π,π]，角度为 [-180,180]
userNo：用户坐标号，范围：int[0,7]
unit_type：返回位姿的 rx,ry,rz 的单位类型，int [0,1]，0：返回角度，1：返回弧度，可选参数，不写默认为弧度值

返回： 用户标系下的位姿信息:double user_pose[6]

示例：
```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if (conSuc):
    # 获取机器人当前位姿信息
    suc, result , id = sendCMD(sock, "get_tcp_pose")
    # 基坐标到用户坐标位姿转化
    suc, result ,id=sendCMD(sock,"convertPoseFromCartToUser",{"targetPose":result, "userNo":0,"
        unit_type":1})
```

2.2.4.4 用户坐标到基坐标位姿转化

```
{"jsonrpc":"2.0","method":"convertPoseFromUserToCart","params":{"
   targetPose":targetPose,"userNo":userNo,"unit_type":unit_type},"id":id}
```

功能： 用户坐标到基坐标位姿转化，当前用户坐标系下，根据用户坐标的位姿信息得到对应基坐标系下的位姿信息

参数： targetPose：用户坐标系下的位姿信息，double pose[6]，rx,ry,rz 的范围：弧度为 [-π,π]，角度为 [-180,180]

userNo：用户坐标号，范围：int[0,7]

unit_type：输入位姿和返回位姿的 rx,ry,rz 的单位类型，int [0,1]，0：角度，1：弧度，可选参数，不写默认为弧度值

返回： 基坐标系下的位姿信息:double base_pose[6]

示例：

```python
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if (conSuc):
    # 获取机器人当前位姿信息
    suc, result , id = sendCMD(sock, "get_tcp_pose")
    # 用户坐标到基坐标位姿转化
    suc, result ,id=sendCMD(sock,"convertPoseFromUserToCart",{"targetPose":result, "userNo":0})
```

2.2.4.5 位姿相乘

```
{"jsonrpc":"2.0","method":"poseMul","params":{"pose1":pose1,"pose2":pose2
   ,"unit_type":unit_type},"id":id}
```

功能： 位姿相乘

参数： pose1：位姿信息，double pose[6]，rx,ry,rz 的范围：弧度为 [-π,π]，角度为 [-180,180]
pose2：位姿信息，double pose[6]，rx,ry,rz 的范围为：弧度为 [-π,π]，角度为 [-180,180]
unit_type：输入位姿和返回位姿的 rx,ry,rz 的单位类型，int [0,1]，0：角度，1：弧度，可选参数，不写默认为弧度值

返回： 位姿相乘结果信息:double response_pose[6]

示例：
```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="192.168.1.200"
  # pose1
  V000 = [10, -10, 10, 0, 0, 0]
  conSuc,sock=connectETController(robot_ip)
  if (conSuc):
    # 获取机器人当前位姿信息
    suc, result , id = sendCMD(sock, "get_tcp_pose")
    # 位姿相乘
    suc, result ,id=sendCMD(sock,"poseMul",{"pose1":V000,"pose2":result,"unit_type":1})
```

2.2.4.6 位姿求逆

```
{"jsonrpc":"2.0","method":"poseInv","params":{"pose":pose,"unit_type":
   unit_type},"id":id}
```

功能： 位姿求逆

参数： pose：位姿信息，double pose[6]，rx,ry,rz 的范围：弧度为 [-π,π]，角度为 [-180,180]
unit_type：输入位姿和返回位姿的 rx,ry,rz 的单位类型，int [0,1]，0：角度，1：弧度，可选参数，不写默认为弧度值

返回： 位姿求逆结果信息:double response_pose[6]

示例：
```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if (conSuc):
    # 获取机器人当前位姿信息
    suc, result , id = sendCMD(sock, "get_tcp_pose")
    # 位姿求逆
    suc, result , id = sendCMD(sock, "poseInv", {"pose": result ,"unit_type":1})
```

2.2.5 IO 服务 (IOService)

2.2.5.1 获取输入 IO 状态

```
{"jsonrpc":"2.0","method":"getInput","params":{"addr":addr},"id":id}
```

功能： 获取输入 IO 状态

参数： addr：输入 IO 地址，范围：int[0,19][48,51]

返回： 输入 IO 状态，int[0,1]，0 为关，1 为开

示例：

```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if (conSuc):
    for i in range(0, 64 ,1):
      # 获取输入IO状态
      suc, result , id = sendCMD(sock, "getInput", {"addr":i})
      print ( result )
```

2.2.5.2 获取输出 IO 状态

```
{"jsonrpc":"2.0","method":"getOutput","params":{"addr":addr},"id":id}
```

功能： 获取输出 IO 状态

参数： addr：输出 IO 地址，范围：int[0,19][48,51]

返回： 输出 IO 状态，int[0,1]，0 为关，1 为开

示例：

```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if (conSuc):
    for n in range(0, 20 ,1):
      # 获取输出IO状态
      suc, result , id = sendCMD(sock, "getOutput", {"addr":n})
      print ( result )
```

2.2.5.3 设置输出 IO 状态

```
{"jsonrpc":"2.0","method":"setOutput","params":{"addr":addr,"status":
   status},"id":id}
```

功能： 设置输出 IO 状态

参数： addr：输出 IO 地址，范围：int[0,19][48,49]
status：IO 状态，int[0,1]，0 为关，1 为开

返回： 成功 true，失败 false

示例：

```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if(conSuc):
    for i in range(0, 20 ,1):
      # 设置输出IO状态
      suc, result , id=sendCMD(sock,"setOutput",{"addr":i," status ":1})
      print( result )
```

注意： 本命令只支持在 remote 模式下使用。

2.2.5.4 获取虚拟输入 IO 状态

```
{"jsonrpc":"2.0","method":"getVirtualInput","params":{"addr":addr},"id":
   id}
```

功能： 获取虚拟输入 IO 状态

参数： addr：虚拟 IO 地址，范围：int[0,399]

返回： 输入 IO 状态，int[0,1]，0 为关，1 为开

示例：

```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if(conSuc):
    for i in range(0, 400 ,1):
      # 获取虚拟输入IO状态
      suc, result , id=sendCMD(sock,"getVirtualInput",{"addr":i})
      print( result )
```

2.2.5.5 获取虚拟输出 IO 状态

```
{"jsonrpc":"2.0","method":"getVirtualOutput","params":{"addr":addr},"id":
   id}
```

功能： 获取虚拟输出 IO 状态

参数： addr：虚拟 IO 地址，范围：int [400,1535]

返回： 输出 IO 状态，int[0,1]，0 为关，1 为开

示例：

```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if (conSuc):
    for n in range(528, 800 ,1):
    # 获取虚拟输出IO状态
    suc, result , id=sendCMD(sock,"getVirtualOutput",{"addr":n})
    print ( result )
```

2.2.5.6 设置虚拟输出 IO 状态

```
{"jsonrpc":"2.0","method":"setVirtualOutput","params":{"addr":addr,"
   status":status},"id":id}
```

功能： 设置虚拟输出 IO 状态

参数： addr：输出 IO 地址，范围：int[528,799]
status：输出 IO 状态，int[0,1]，0 为关，1 为开

返回： 成功 true，失败 false

示例：

```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if (conSuc):
    for i in range(528, 800 ,1):
      # 设置虚拟输出IO状态
      suc, result , id=sendCMD(sock,"setVirtualOutput",{"addr":i," status ":1})
```

注意： 本命令只支持在 remote 模式下使用。

2.2.5.7 读取多个 M 虚拟 IO

```
{" jsonrpc ":"2.0" ," method ":" getRegisters ","params":{"addr":addr,"
   len":len},"id":id}
```

功能： 读取多个 M 虚拟 IO

参数： addr: 虚拟 IO 地址范围 int [0,1535]
len: 起始地址开始向后读取长度为 (16*len) 个虚拟 IO 范围 int [1,96]
addr+16*len 的范围为 int[0,1535]

返回： 虚拟 IO 值列表 (每 16 个虚拟 IO 值用一个十进制整数表示，列表长度为 len)

示例：

```python
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if (conSuc):
    # 获取M0~M16的值
    ret , result , id=sendCMD(sock,"getRegisters",{"addr": 0, "len": 1})
    if ret :
      print (" result =", result )
    else :
      print ("err_msg=", result ["message"])
```

2.2.5.8 获取模拟量输入

```
{"jsonrpc":"2.0","method":"getAnalogInput","params":{"addr":addr},"id":id
   }
```

功能： 获取模拟量输入

参数： addr：模拟量地址，范围：int[0,2]

返回： 模拟量值，范围：double[-10,10]

示例：

```python
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if (conSuc):
    for i in range(0, 2, 1):
      # 获取模拟量输入
      suc , result , id = sendCMD(sock, "getAnalogInput", {"addr":i})
```

2.2.5.9 设置模拟量输出

```
{"jsonrpc":"2.0","method":"setAnalogOutput","params":{"addr":addr,"value
    ":value},"id":id}
```

功能： 设置模拟量输出

参数： addr：模拟量地址，范围：int[0,4]
value：模拟量值，addr 为 0-3 时，范围：double[-10,10]；addr 为 4 时，范围：double[0,10]

返回： 成功 true，失败 false

示例：

```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if (conSuc):
    # 设置模拟量输出
    suc, result ,id=sendCMD(sock,"setAnalogOutput",{"addr":0,"value":-10})
    suc, result ,id=sendCMD(sock,"setAnalogOutput",{"addr":1,"value":-3.5})
    suc, result ,id=sendCMD(sock,"setAnalogOutput",{"addr":2,"value":0})
    suc, result ,id=sendCMD(sock,"setAnalogOutput",{"addr":3,"value":0.5})
    suc, result ,id=sendCMD(sock,"setAnalogOutput",{"addr":4,"value":0.5})
```

注意： 本命令只支持在 remote 模式下使用。

2.2.6 变量服务 (VarService)

2.2.6.1 获取系统 B 变量值

```
{"jsonrpc":"2.0","method":"getSysVarB","params":{"addr":addr},"id":id}
```

功能： 获取系统 B 变量值

参数： addr：变量地址，范围：int [0,255]

返回： 变量值，范围：int[0,2147483647]

示例：

```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if (conSuc)
    for n in range(0, 11 ,1):
      # 获取系统B变量值
      suc, result , id = sendCMD(sock, "getSysVarB", {"addr":n})
      print ( result )
```

2.2.6.2 设置系统 B 变量值

```
{"jsonrpc":"2.0","method":"setSysVarB","params":{"addr":addr,"value":
   value},"id":id}
```

功能： 设置系统 B 变量值

参数： addr：变量地址，范围：int [0,255]
value：变量值，范围：int[0,2147483647]

返回： 成功 true，失败 false

示例：

```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if (conSuc)
    for n in range(0, 11 ,1):
      # 设置系统B变量值
      suc, result ,id=sendCMD(sock,"setSysVarB",{"addr":i,"value":100})
```

注意： 本命令只支持在 remote 模式下使用。

2.2.6.3 获取系统 I 变量值

```
{"jsonrpc":"2.0","method":"getSysVarI","params":{"addr":addr},"id":id}
```

功能： 获取系统 I 变量值

参数： addr：变量地址，范围：int [0,255]

返回： 变量值，范围：int[-32767,32767]

示例：

```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if (conSuc)
    for n in range(0, 11 ,1):
      # 获取系统I变量值
      suc, result , id = sendCMD(sock, "getSysVarI", {"addr":n})
      print ( result )
```

2.2.6.4 设置系统 I 变量值

```
{"jsonrpc":"2.0","method":"setSysVarI","params":{"addr":addr,"value":
   value},"id":id}
```

功能： 设置系统 I 变量值

参数： addr：变量地址，范围：int [0,255]
value：变量地址，范围：int[-32767,32767]

返回： 成功 true，失败 false

示例：

```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if (conSuc)
    for n in range(0, 11 ,1):
      # 设置系统I变量值
      suc, result ,id=sendCMD(sock,"setSysVarI",{"addr":i,"value":100})
```

注意： 本命令只支持在 remote 模式下使用。

2.2.6.5 获取系统 D 变量值

```
{"jsonrpc":"2.0","method":"getSysVarD","params":{"addr":addr},"id":id}
```

功能： 获取系统 D 变量值

参数： addr：变量地址，范围：int [0,255]

返回： 变量值，范围：double[-1e+09,1e+09]

示例：

```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if (conSuc)
    for n in range(0, 11 ,1):
      # 获取系统D变量值
      suc, result , id = sendCMD(sock, "getSysVarD", {"addr":n})
      print ( result )
```

2.2.6.6 设置系统 D 变量值

```
{"jsonrpc":"2.0","method":"setSysVarD","params":{"addr":addr,"value":
   value},"id":id}
```

功能： 设置系统 D 变量值

参数： addr：变量地址，范围：int [0,255]
value：变量值，范围：double[-1e+09,1e+09]

返回： 成功 true，失败 false

示例：
```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if(conSuc)
    for n in range(0, 11 ,1):
      # 设置系统D变量值
      suc, result ,id=sendCMD(sock,"setSysVarD",{"addr":i,"value":100})
```

注意： 本命令只支持在 remote 模式下使用。

2.2.6.7 获取系统 P 变量是否启用

```
{"jsonrpc":"2.0","method":"getSysVarPState","params":{"addr":addr},"id":
   id}
```

功能： 获取系统 P 变量是否启用

参数： addr：变量地址，范围：int [0,255]

返回： 存储 P 变量启用状态，0：未启用，1：已启用

示例：
```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if(conSuc)
    for i in range(0, 101 ,1):
      # 获取系统P变量是否启用
      suc, result , id = sendCMD(sock, "getSysVarPState", {"addr": i})
```

2.2.6.8 获取 P 变量的值

```
{"jsonrpc":"2.0","method":"getSysVarP","params":{"addr":addr},"id":id}
```

功能： 获取系统 P 变量值

参数： addr：变量地址，范围：int [0,255]

返回： 系统 P 变量值 double pos[8]

示例：

```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if (conSuc)
    for i in range(0, 101 ,1):
      # 获取系统P变量是否启用
      suc, result , id = sendCMD(sock, "getSysVarPState", {"addr": i})
        if( result == 1):
          # 获取系统P变量值
          suc, result , id = sendCMD(sock, "getSysVarP", {"addr":i})
          print( result )
```

2.2.6.9 设置 P 变量的值

```
{"jsonrpc":"2.0","method":"setSysVarP","params":{"addr:":addr,"pos:":pos
   },"id":id}
```

功能： 设置系统 P 变量的值

参数： addr: 变量地址，范围 int[0,255]
pos: p 变量的值，double pos[8]，范围 [-360,360]

返回： 成功 True, 失败 False

示例：
```
if __name__ == "__main__":
    ip = "192.168.1.202"
    conSuc, sock = connectETController(ip)
    point = [0, -90, 0, -90, 90, 0, 0, 0]
    if conSuc:
        ret , result , id=sendCMD(sock,"setSysVarP",{"addr":0,"pos":point})
        if ret :
            print ( result )
        else :
            print ("err_msg = ", result ["message"])
```

注意： 本命令只支持在 remote 模式下使用。
本命令适用于 v2.15.2 及以上版本。

2.2.6.10 获取 V 变量的值

```
{"jsonrpc":"2.0","method":"getSysVarV","params":{"addr":addr},"id":id}
```

功能： 获取系统 V 变量值

参数： addr：变量地址，范围：int [0,255]

返回： 系统 V 变量值 double pose[6]

示例：
```
if __name__ == "__main__":
    # 机器人IP地址
    robot_ip="192.168.1.200"
    conSuc,sock=connectETController(robot_ip)
    if (conSuc)
        for i in range(0, 101 ,1):
            # 获取系统V变量值
            suc, result , id = sendCMD(sock, "getSysVarV", {"addr": i})
            print ( result )
```

2.2.6.11 设置 V 变量的值

```
{"jsonrpc":"2.0","method":"setSysVarV","params":{"addr:":addr,"pose:":
    pose},"id":id}
```

功能： 设置系统 V 变量的值

参数： addr: 变量地址，范围 int[0,255]
pose：V 变量的值，double pose[6]，rx,ry,rz 的范围为 [-π,π]

返回： 成功 True, 失败 False

示例：
```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="192.168.1.205"
  conSuc,sock=connectETController(robot_ip)
  pose = [200, 125.5, -50, 0, 0, 0]
  if (conSuc):
    # 设置系统V变量值
    suc, result , id = sendCMD(sock, "setSysVarV", {"addr": 0,"pose":
          [243.5,-219.4,169.578000,3.139376,-0.002601,0.106804]})
    print (suc, result )
  else :
    print ("连接失败")
  disconnectETController (sock)
```

注意： 本命令只支持在 remote 模式下使用。

2.2.6.12 保存变量数据

```
{"jsonrpc":"2.0","method":"save_var_data","id":id}
```

功能： 保存系统变量数据

参数： 无

返回： 成功 True, 失败 False

示例：
```
if __name__ == "__main__":
  ip = "192.168.1.202"
  conSuc, sock = connectETController(ip)
  pos = [200, 125.5, -50, 1.57, -1.57, 3.14]
  if conSuc:
    ret , result , id = sendCMD(sock, "save_var_data")
    if ret :
      print ( result )
    else :
      print ("err_msg = ", result ["message"])
```

注意： 本命令只支持在 remote 模式下使用。
本命令适用于 v2.15.2 及以上版本。

2.2.7 透传服务 (TransparentTransmissionService)

2.2.7.1 初始化透传服务

```
{"jsonrpc":"2.0","method":"transparent_transmission_init","params":{"
   lookahead":lookahead,"t":t,"smoothness":smoothness,"response_enable":
   response_enable},"id":id}
```

功能： 初始化机器人透传服务

参数： lookahead：前瞻时间，单位 ms，范围：int [10,1000]
t：采样时间，单位 ms，范围：int [2,100]
smoothness：增益，单位百分比，范围：double [0,1]。
注：smoothness 当前版本不适用
response_enable：可选参数，不写默认有返回值。int[0,1]，添加点位指令是否返回值，0：无返回值，1：有返回值

返回： 成功 true，失败 false

示例：
```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if (conSuc):
    # 初始化透传服务
    suc, result ,id=sendCMD(sock," transparent_transmission_init ",{"lookahead":400,"t":10,"
        smoothness":0.1,"response_enable": 0})
```

注意： 本命令只支持在 remote 模式下使用。

2.2.7.2 设置当前透传伺服目标关节点

```
{"jsonrpc":"2.0","method":"tt_set_current_servo_joint","params":{"
   targetPos":targetPos},"id":id}
```

功能： 设置当前透传伺服目标关节点

参数： targetpos：目标关节点 double pos[8]，范围为 [-360,360]

返回： 成功 true，失败 false

示例：
```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  # 透传起始点
  P0 = [0, -90, 0, -90, 90, 0, 0, 0]
  if (conSuc):
    # 初始化透传服务
    suc, result ,id=sendCMD(sock," transparent_transmission_init ",{"lookahead":400,"t":10,"
        smoothness":0.1})
    # 设置当前透传目标关节点
    suc, result ,id=sendCMD(sock," tt_set_current_servo_joint ",{" targetPos ": P0})
```

注意： 本命令只支持在 remote 模式下使用。
本命令不推荐使用。

2.2.7.3 获取当前机器人是否处于透传状态

```
{"jsonrpc":"2.0","method":"get_transparent_transmission_state","id":id}
```

功能： 获取当前机器人是否处于透传状态

参数： 无

返回： 当前透传状态。0：非透传状态，1：透传状态

示例：
```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if (conSuc):
    # 获取当前机器人是否处于透传状态
    suc, result ,id =sendCMD(sock," get_transparent_transmission_state ")
```

2.2.7.4 添加透传伺服目标关节点信息到缓存中

```
{"jsonrpc":"2.0","method":"tt_put_servo_joint_to_buf","params":{"
   targetPos":targetPos},"id":id} 或 {"jsonrpc":"2.0","method":"
   tt_put_servo_joint_to_buf","params":{"targetPose":targetPose},"id":
   id}
```

功能： 添加透传伺服目标关节点信息到缓存中

参数： targetpos：目标关节点 double pos[8]，范围为 [-360,360] 或 targetPose：目标位姿点 double pos[6]

返回： 成功 true，失败 false

示例：

```
if __name__ == "__main__":
# 机器人IP地址
    robot_ip="192.168.1.202"
    conSuc,sock=connectETController(ip)
    i = 0
    if (conSuc):
        # 获取当前机器人是否处于透传状态
        suc, result ,id=sendCMD(sock, " get_transparent_transmission_state ")
        print (suc, result , id)
        if ( result == 1):
            # 清空透传缓存
            suc, result , id = sendCMD(sock," tt_clear_servo_joint_buf ")
            time . sleep (0.5)
        # 打开文件
            file_name = 'D :\\ tttest8 . txt '
            fo = open(file_name, "r")
            while 1:
                # 依次读取文件的每一行（点位信息）
                line = fo . readline ()
                if not line : break
                # 去掉每行头尾空白
                line_list = line . strip ()
                line_list = list (map( float , line_list . split (' , ')))
                if (i == 0):
                    # 关节运动到起始点
                    suc, result ,id=sendCMD(sock,"moveByJoint",{"targetPos": line_list ,"speed":30})
                    wait_stop () # 等待机器人停止
                    # 初始化透传服务
                    suc, result ,id=sendCMD(sock," transparent_transmission_init ",{"lookahead":400,"t":10,
                        "smoothness":0.1,"response_enable":1})
                    print (suc, result , id)
                # 添加透传伺服目标关节点信息到缓存中
                suc, result , id = sendCMD(sock, " tt_put_servo_joint_to_buf ",{" targetPos ": line_list })
                time . sleep (0.01)
                i = i + 1
```

注意： 本命令只支持在 remote 模式下使用。
targetPos 和 targetPose 两个参数二选一，一个指令只能发送一个参数。

2.2.7.5 清空透传缓存

```
{"jsonrpc":"2.0","method":"tt_clear_servo_joint_buf","id":id}
```

功能： 清空透传缓存

参数： 无

返回： 成功 true，失败 false

示例：

```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if(conSuc):
    # 获取当前机器人是否处于透传状态
    suc, result ,id =sendCMD(sock," get_transparent_transmission_state ")
    if( result == 1):
      # 清空透传缓存
      suc, result ,id=sendCMD(sock,"tt_clear_servo_joint_buf")
      time.sleep (0.5)
```

注意： 本命令只支持在 remote 模式下使用。

2.2.7.6 Example 1

```
1  import socket
2  import json
3  import time
4  import random
5
6  def connectETController(ip,port=8055):
7      sock = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
8      try:
9          sock.connect((ip,port))
10         return (True,sock)
11     except Exception as e:
12         sock.close()
13         return (False,None)
14
15 def disconnectETController(sock):
16     if(sock):
17         sock.close()
```

```
18          sock=None
19      else:
20          sock=None
21
22  def sendCMD(sock,cmd,params=None,id=1):
23      if(not params):
24          params=[]
25      else:
26          params=json.dumps(params)
27      sendStr="{{\"method\":\"{0}\",\"params\":{1},\"jsonrpc\":\"2.0\",\"
           id\":{2}}}".format(cmd,params,id)+"\n"
28      try:
29          sock.sendall(bytes(sendStr,"utf-8"))
30          ret =sock.recv(1024)
31          jdata=json.loads(str(ret,"utf-8"))
32          if("result" in jdata.keys()):
33              return (True,json.loads(jdata["result"]),jdata["id"])
34          elif("error" in jdata.keys()):
35              return (False,jdata["error"],jdata["id"])
36          else:
37              return (False,None,None)
38      except Exception as e:
39          return (False,None,None)
40
41  def wait_stop():
42      while True:
43          time.sleep(0.01)
44          ret1, result1, id1 = sendCMD(sock, "getRobotState")
45          if (ret1):
46              if result1 == 0 or result1 == 4:
47                  break
48          else:
49              print("getRobotState failed")
50              break
51
52  if __name__ == "__main__":
53      # 机器人IP地址
54      robot_ip="192.168.1.202"
55      conSuc,sock=connectETController(robot_ip)
```

```
    print(conSuc)
    if(conSuc):
    # 获取机器人状态
        suc, result, id = sendCMD(sock, "getRobotState")
        if(result == 4):
            # 清除报警
            suc, result, id = sendCMD(sock, "clearAlarm")
            time.sleep(0.5)
        # 获取同步状态
        suc, result, id = sendCMD(sock, "getMotorStatus")
        if(result == 0):
            # 同步伺服编码器数据
            suc, result, id = sendCMD(sock, "syncMotorStatus")
            time.sleep(0.5)
        # 获取机械臂伺服状态
        suc, result, id = sendCMD(sock, "getServoStatus")
        if (result == 0):
            # 设置机械臂伺服状态ON
            suc, result, id = sendCMD(sock, "set_servo_status",{"status
                ":1})
            time.sleep(1)
      # 获取当前机器人是否处于透传状态
        suc, result, id = sendCMD(sock, "
           get_transparent_transmission_state")
        print(suc, result, id)
        if(result == 1):
            # 清空透传缓存
            suc, result, id = sendCMD(sock,"tt_clear_servo_joint_buf")
            time.sleep(0.5)
      # 打开文件
        file_name = 'D:\\tttest8.txt'
        fo = open(file_name, "r")
        while 1:
            # 依次读取文件的每一行（点位信息）
            line = fo.readline()
            if not line : break
            # 去掉每行头尾空白
            line_list = line.strip()
            line_list = list(map(float, line_list.split(',')))
```

```
93              print(i,line_list)
94              if (i == 0):
95                  # 关节运动到起始点
96                  suc, result, id = sendCMD(sock, "moveByJoint", {"
                        targetPos": line_list, "speed": 30})
97                  wait_stop() # 等待机器人停止
98                  # 初始化透传服务
99                  suc, result, id = sendCMD(sock, "
                        transparent_transmission_init", {"lookahead": 400, "
                        t": 10, "smoothness": 0.1, "response_enable": 1})
100                 print(suc, result, id)
101             # 添加透传伺服目标关节点信息到缓存中
102             suc, result, id = sendCMD(sock, "tt_put_servo_joint_to_buf"
                    ,{"targetPos": line_list})
103             time.sleep(0.01)
104             i = i + 1
105         # 关闭文件
106         fo.close()
107         # 清空透传缓存
108         suc, result, id = sendCMD(sock, "tt_clear_servo_joint_buf")
109         print("clear_ret = ", suc)
110   else:
111         print("连接失败")
112     disconnectETController(sock)
```

2.2.7.7 Example 2

```
1  import socket
2  import json
3  import time
4
5  def connectETController(ip,port=8055):
6      sock = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
7      try:
8          sock.connect((ip,port))
9          return (True,sock)
10     except Exception as e:
11         sock.close()
12         return (False,None)
```

```
13
14 def disconnectETController(sock):
15     if(sock):
16         sock.close()
17         sock=None
18     else:
19         sock=None
20
21 def sendCMD(sock,cmd,params=None,id=1):
22     if(not params):
23         params=[]
24     else:
25         params=json.dumps(params)
26     sendStr="{{\"method\":\"{0}\",\"params\":{1},\"jsonrpc\":\"2.0\",\"
       id\":{2}}}".format(cmd,params,id)+"\n"
27     try:
28         sock.sendall(bytes(sendStr,"utf-8"))
29         # print(sock.recv)
30         ret =sock.recv(1024)
31         jdata=json.loads(str(ret,"utf-8"))
32         if("result" in jdata.keys()):
33             return (True,json.loads(jdata["result"]),jdata["id"])
34         elif("error" in jdata.keys()):
35             return (False,jdata["error"],jdata["id"])
36         else:
37             return (False,None,None)
38     except Exception as e:
39         return (False,None,None)
40
41 def send_Point(sock,cmd,params=None,id=1):
42     if(not params):
43         params=[]
44     else:
45         params=json.dumps(params)
46     sendStr="{{\"method\":\"{0}\",\"params\":{1},\"jsonrpc\":\"2.0\",\"
       id\":{2}}}".format(cmd,params,id)+"\n"
47     sock.sendall(bytes(sendStr,"utf-8"))
48
49 def wait_stop():
```

```
50      while True:
51          time.sleep(0.01)
52          ret1, result1, id1 = sendCMD(sock, "getRobotState")  #
                getRobotstate
53          if (ret1):
54              if result1 == 0 or result1 == 4:
55                  break
56          else:
57              print("getRobotState failed")
58              break
59
60
61  if __name__ == "__main__":
62      # 机器人IP地址
63      robot_ip="192.168.1.202"
64      conSuc,sock=connectETController(robot_ip)
65      P0 = [0, -90, 0, -90, 90, 0, 0, 0]
66      point = []
67      i = 0
68      if(conSuc):
69          # 获取机器人状态
70          suc, result, id = sendCMD(sock, "getRobotState")
71          if(result == 4):
72              # 清除报警
73              suc, result, id = sendCMD(sock, "clearAlarm")
74              time.sleep(0.5)
75          # 获取同步状态
76          suc, result, id = sendCMD(sock, "getMotorStatus")
77          if(result == 0):
78              # 同步伺服编码器数据
79              suc, result, id = sendCMD(sock, "syncMotorStatus")
80              time.sleep(0.5)
81          # 获取机械臂伺服状态
82          suc, result, id = sendCMD(sock, "getServoStatus")
83          if (result == 0):
84              # 设置机械臂伺服状态ON
85              suc, result, id = sendCMD(sock, "set_servo_status",{"status
                    ":1})
86              time.sleep(1)
```

```
87          # 获取当前机器人是否处于透传状态
88          suc, result, id = sendCMD(sock, "
               get_transparent_transmission_state")
89          print(result)
90          if(result == 1):
91              # 清空透传缓存
92              suc, result, id = sendCMD(sock,"tt_clear_servo_joint_buf")
93              time.sleep(0.5)
94          # 打开文件
95          file_name = 'D:\\tttest8.txt'
96          fo = open(file_name, "r")
97          while 1:
98              # 依次读取文件的每一行（点位信息）
99              line = fo.readline()
100             if not line : break
101             # 去掉每行头尾空白
102             line_list = line.strip()
103             line_list = list(map(float, line_list.split(',')))
104
105             if (i == 0):
106                 # 关节运动到起始点
107                 suc, result, id = sendCMD(sock, "moveByJoint", {"
                       targetPos": line_list, "speed": 30})
108                 print(result)
109                 wait_stop() # 等待机器人停止
110                 print(1)
111                 # 初始化透传服务
112                 suc, result, id = sendCMD(sock, "
                       transparent_transmission_init", {"lookahead": 400, "
                       t": 10, "smoothness": 0.1, "response_enable":0})
113                 print(result)
114             # 添加透传伺服目标关节点信息到缓存中
115             send_Point(sock, "tt_put_servo_joint_to_buf",{"targetPos":
                   line_list})
116             print(result)
117             time.sleep(0.01)
118             i = i + 1
119         # 关闭文件
120         fo.close()
```

```
121         # 清空透传缓存
122         suc, result, id = sendCMD(sock, "tt_clear_servo_joint_buf")
123         print("clear_ret = ", suc)
124     else:
125         print("连接失败")
126     disconnectETController(sock)
```

2.2.8 系统服务 (SystemService)

2.2.8.1 获取控制器软件版本号

```
{"jsonrpc":"2.0","method":"getSoftVersion","id":id}
```

功能： 获取控制器软件版本号

参数： 无

返回： 控制器软件版本号

示例：

```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if(conSuc):
    # 获取控制器软件版本号
    suc, result, id = sendCMD(sock, "getSoftVersion")
    print(result)
```

2.2.8.2 获取伺服版本号

```
{" jsonrpc ":"2.0" ," method ":" getJointVersion ","params":{"axis":axis
   },"id":id}
```

功能： 获取伺服版本号

参数： axis: 范围 int [0,7]，对应轴号 1~8

返回： 伺服版本号

示例：

```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if (conSuc):
    # 获取1轴伺服版本号
    ret , result , id = sendCMD(sock, "getJointVersion", {"axis":0})
    if ret :
      print (" result =", result )
    else :
      print ("err_msg=", result ["message"])
```

注意： 该功能适用的伺服版本为 11 及以上版本。

2.2.9 时间戳服务 (TrajectoryService)

2.2.9.1 初始化运动

```
{"jsonrpc":"2.0","method":"start_push_pos","params":{"path_lenth":
   path_lenth,"pos_type":pos_type,"ref_joint_pos":ref_joint_pos,"
   ref_frame":ref_frame,"ref_flag":ref_flag},"id":id}
```

功能： 初始化运动

参数： path_lenth：传送的点位个数，int，范围：大于等于 3

pos_type：点位类型，int[0,1]，0：关节，1：位姿

ref_joint_pos：参考点，double pos[8]，如果输入的是位姿点位，这个参考点为第一个点的逆解参考点

ref_frame：坐标系，double pose[6]，如果是基于基座坐标系，则全为 0；如果输入坐标是位姿点位，这个参数是点位的坐标系。

ret_flag：int[0,1]，添加点位指令是否返回值，0：无返回值，1：有返回值

返回： 成功 true，失败 false

示例：

```
if __name__ == "__main__":
    # 机器人IP地址
    robot_ip = "192.168.1.202"
    conSuc, sock = connectETController(robot_ip)
    pos = [0, -90, 0, -90, 90, 0, 0, 0]
    frame = [0, 0, 0, 0, 0, 0]
    if (conSuc):
    suc, result , id=sendCMD(sock,"start_push_pos",{"path_lenth":10,"pos_type":0," ref_joint_pos ":
        pos,"ref_frame":frame," ref_flag ":1})
    print ( result )
```

2.2.9.2 添加运动点位

```
{"jsonrpc":"2.0","method":"push_pos","params":{"timestamp":timestamp,"pos
    ":pos},"id":id}
```

功能： 添加运动点位

参数： timestamp：double，点位的时间戳（在点位序列的第几个时间点），单位：s，范围：大于等于 0，且递增

pos：double pos[8]/double pose[6]，点位数据

返回： 成功 true，失败 false

示例：

```
iif __name__ == "__main__":
    # 机器人IP地址
    robot_ip = "192.168.1.202"
    conSuc, sock = connectETController(robot_ip)
    pos = [0, -90, 0, -90, 90, 0, 0, 0]
    frame = [0, 0, 0, 0, 0, 0]
    res = 0
    if (conSuc):
        ret , result , id = sendCMD(sock, "start_push_pos", {"path_lenth":10, "pos_type": 0, "
            ref_joint_pos ": pos, "ref_frame": frame, " ret_flag ": 1})
        print ( result )
        ret , joint , id = sendCMD(sock, "get_joint_pos")
        time . sleep (0.2)
        for i in range (0, 10):
            ret , result , id = sendCMD(sock, "push_pos", {"timestamp": res , "pos": joint })
            print ( result , i )
            joint [0] += 0.01
            res = res + 0.002
```

注意： 传输的第一个点位的时间戳必须为 0。

2.2.9.3 停止添加点位

```
{"jsonrpc":"2.0","method":"stop_push_pos","id":id}
```

功能： 停止添加时间戳点位

参数： 无

返回： 成功 true，失败 false

示例：

```
if __name__ == "__main__":
    # 机器人IP地址
    robot_ip = "192.168.1.202"
    conSuc, sock = connectETController(robot_ip)
    pos = [0, -90, 0, -90, 90, 0, 0, 0]
    frame = [0, 0, 0, 0, 0, 0]
    res = 0
    if (conSuc):
        ret , result , id = sendCMD(sock, "start_push_pos", {"path_lenth":10, "pos_type": 0, "
            ref_joint_pos ": pos, "ref_frame": frame, " ret_flag ": 1})
        print ( result )
        ret , joint , id = sendCMD(sock, "get_joint_pos")
        time.sleep (0.2)
        for i in range(0, 10):
            ret , result , id = sendCMD(sock, "push_pos", {"timestamp": res , "pos": joint })
            print ( result ,i)
            joint [0] += 0.01
            res = res + 0.002
        ret , result , id = sendCMD(sock, "stop_push_pos")
        print ( result )
```

注意： stop_push_pos 和 push_pos 是对应关系，只有对应的 push_pos 发送的所有点位正确，才会返回 True，其他情况，均返回 False。

2.2.9.4 检查执行状态

```
{"jsonrpc":"2.0","method":"check_trajectory","id":id}
```

功能： 检查执行状态

参数： 无

返回： int[-3,0]，0：传送点位和时间戳正确，-1：点位长度不相符，-2：点位格式错误，-3：时间戳不规范。

示例：

```
if __name__ == "__main__":
    # 机器人IP地址
    robot_ip = "192.168.1.202"
    conSuc, sock = connectETController(robot_ip)
    pos = [0, -90, 0, -90, 90, 0, 0, 0]
    frame = [0, 0, 0, 0, 0, 0]
    res = 0
    if (conSuc):
        ret , result , id = sendCMD(sock, "start_push_pos", {"path_lenth":10, "pos_type": 0, "
            ref_joint_pos ": pos, "ref_frame": frame, " ret_flag ": 1})
        print ( result )
        ret , joint , id = sendCMD(sock, "get_joint_pos")
        time.sleep (0.2)
        for i in range(0, 10):
            ret , result , id = sendCMD(sock, "push_pos", {"timestamp": res , "pos": joint })
            print ( result , i)
            joint [0] += 0.01
            res = res + 0.002
        ret , result , id = sendCMD(sock, "stop_push_pos")
        print ( result )
        ret , result , id = sendCMD(sock, "check_trajectory")
        print ( result )
```

2.2.9.5 开始带时间戳运动

```
{"jsonrpc":"2.0","method":"start_trajectory","params":{"speed_percent":
   speed_percent},"id":id}
```

功能： 开始带时间戳运动

参数： speed_percent：double，轨迹速度百分比，即以原始速度乘百分比的速度运动，单位：%，范围：大于等于 0.1

返回： 成功 true，失败 false

示例：

```
if __name__ == "__main__":
    # 机器人IP地址
    robot_ip = "192.168.1.202"
    conSuc, sock = connectETController(robot_ip)
    pos = [0, -90, 0, -90, 90, 0, 0, 0]
    frame = [0, 0, 0, 0, 0, 0]
    res = 0
    if (conSuc):
        ret, result, id = sendCMD(sock, "start_push_pos", {"path_lenth":10, "pos_type": 0, "
            ref_joint_pos ": pos, "ref_frame": frame, " ret_flag ": 1})
        print(result)
        ret, joint, id = sendCMD(sock, "get_joint_pos")
        time.sleep(0.2)
        for i in range(0, 10):
            ret, result, id = sendCMD(sock, "push_pos", {"timestamp": res, "pos": joint})
            print(result, i)
            joint[0] += 0.01
            res = res + 0.002
        ret, result, id = sendCMD(sock, "stop_push_pos")
        print(result)
        ret, result, id = sendCMD(sock, "check_trajectory")
        print(result)
        ret, result, id = sendCMD(sock, " start_trajectory ", {"speed_percent": 50})
        print(result)
```

注意： 只要没有执行 flush_trajectory, 或者 start_push_pos，那么当前轨迹可以循环运行，而不用重复传输

2.2.9.6 暂停运动

```
{"jsonrpc":"2.0","method":"pause_trajectory","id":id}
```

功能： 暂停运动

参数： 无

返回： 成功 true，失败 false

示例：

```
if __name__ == "__main__":
    # 机器人IP地址
    robot_ip = "192.168.1.202"
    conSuc, sock = connectETController(robot_ip)
    if (conSuc):
        ret , result , id=sendCMD(sock, "pause_trajectory")
        print ( result )
```

2.2.9.7 恢复运动

```
{"jsonrpc":"2.0","method":"resume_trajectory","id":id}
```

功能： 恢复运动

参数： 无

返回： 成功 true，失败 false

示例：

```
if __name__ == "__main__":
    # 机器人IP地址
    robot_ip = "192.168.1.202"
    conSuc, sock = connectETController(robot_ip)
    if (conSuc):
        ret , result , id=sendCMD(sock, "resume_trajectory")
        print ( result )
```

2.2.9.8 停止运动

```
{"jsonrpc":"2.0","method":"stop_trajectory","id":id}
```

功能： 停止运动

参数： 无

返回： 成功 true，失败 false

示例：

```
if __name__ == "__main__":
    # 机器人IP地址
    robot_ip = "192.168.1.202"
    conSuc, sock = connectETController(robot_ip)
    if (conSuc):
        ret , result , id=sendCMD(sock, "stop_trajectory")
        print ( result )
```

2.2.9.9 清空缓存

```
{"jsonrpc":"2.0","method":"flush_trajectory","id":id}
```

功能： 清空缓存

参数： 无

返回： 成功 true，失败 false

示例：

```
if __name__ == "__main__":
    # 机器人IP地址
    robot_ip = "192.168.1.202"
    conSuc, sock = connectETController(robot_ip)
    if (conSuc):
        ret , result , id=sendCMD(sock, "flush_trajectory")
        print ( result )
```

2.2.9.10 Example 1

```
1  import socket
2  import json
3  import time
4
5  def connectETController(ip,port=8055):
6      sock = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
7      try:
8          sock.connect((ip,port))
9          return (True,sock)
10     except Exception as e:
```

```
11          sock.close()
12          return (False,None)
13
14  def disconnectETController(sock):
15      if(sock):
16          sock.close()
17          sock=None
18      else:
19          sock=None
20
21  def sendCMD(sock,cmd,params=None,id=1):
22      if(not params):
23          params=[]
24      else:
25          params=json.dumps(params)
26      sendStr="{{\"method\":\"{0}\",\"params\":{1},\"jsonrpc\":\"2.0\",\"
           id\":{2}}}".format(cmd,params,id)+"\n"
27      try:
28          sock.sendall(bytes(sendStr,"utf-8"))
29          # print(sock.recv)
30          ret =sock.recv(1024)
31          jdata=json.loads(str(ret,"utf-8"))
32          if("result" in jdata.keys()):
33              return (True,json.loads(jdata["result"]),jdata["id"])
34          elif("error" in jdata.keys()):
35              return (False,jdata["error"],jdata["id"])
36          else:
37              return (False,None,None)
38      except Exception as e:
39          return (False,None,None)
40
41  def wait_stop():
42      while True:
43          time.sleep(0.01)
44          ret1, result1, id1 = sendCMD(sock, "getRobotState")
45          if (ret1):
46              if result1 == 0 or result1 == 4:
47                  break
48          else:
```

```
49              print("getRobotState failed")
50              break
51
52  if __name__ == "__main__":
53      ip = "192.168.1.200"
54      conSuc, sock = connectETController(ip)
55      start_pos = [0, -90, 0, -90, 90, 0, 0, 0]
56      ref_pos = [0, 0, 0, 0, 0, 0]
57      res = 0
58      if conSuc:
59          suc, result, id = sendCMD(sock, "getRobotState")
60          if (result == 4):
61              # 清除报警
62              suc, result, id = sendCMD(sock, "clearAlarm")
63              time.sleep(0.5)
64          # 获取同步状态
65          suc, result, id = sendCMD(sock,"getMotorStatus")
66          if result != True:
67              # 同步伺服编码器数据
68              suc, result, id = sendCMD(sock,"syncMotorStatus")
69              time.sleep(0.5)
70          time.sleep(0.5)
71          # 获取机械臂伺服状态
72          suc, result, id = sendCMD(sock, "getServoStatus")
73          if result == 0:
74              # 设置机械臂伺服状态ON
75              ret, result, id = sendCMD(sock, "set_servo_status", {"
                    status": 1})
76          suc, result, id = sendCMD(sock, "moveByJoint", {"targetPos":
                start_pos, "speed": 50})
77          wait_stop()
78          suc, result, id = sendCMD(sock, "start_push_pos", {"path_lenth"
                : 10, "pos_type": 0, "ref_joint_pos": start_pos, "ref_frame"
                : ref_pos, "ret_flag": 1})
79          print(result)
80          time.sleep(0.2)
81          for i in range(0, 10):
82              suc, result, id = sendCMD(sock, "push_pos", {"timestamp":
                    res, "pos": start_pos})
```

```
83              start_pos[0] += 0.02
84              res = res + 0.002
85          time.sleep(1)
86          suc, result, id = sendCMD(sock, "stop_push_pos")
87          print(result)
88          suc, result, id = sendCMD(sock, "check_trajectory")
89          print(result)
90          suc, result, id = sendCMD(sock, "start_trajectory", {"
               speed_percent": 50})
91          print(result)
92          time.sleep(5)
93          suc, result, id = sendCMD(sock, "flush_trajectory")
94          print(result)
95      else:
96          print("连接失败")
97      disconnectETController(sock)
```

2.2.9.11 Example 2

```
1  import socket
2  import json
3  import time
4
5  def connectETController(ip,port=8055):
6      sock = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
7      try:
8          sock.connect((ip,port))
9          return (True,sock)
10     except Exception as e:
11         sock.close()
12         return (False,None)
13
14 def disconnectETController(sock):
15     if(sock):
16         sock.close()
17         sock=None
18     else:
19         sock=None
20
```

```
21 def sendCMD(sock,cmd,params=None,id=1):
22     if(not params):
23         params=[]
24     else:
25         params=json.dumps(params)
26     sendStr="{{\"method\":\"{0}\",\"params\":{1},\"jsonrpc\":\"2.0\",\"
       id\":{2}}}".format(cmd,params,id)+"\n"
27     try:
28         sock.sendall(bytes(sendStr,"utf-8"))
29         # print(sock.recv)
30         ret =sock.recv(1024)
31         jdata=json.loads(str(ret,"utf-8"))
32         if("result" in jdata.keys()):
33             return (True,json.loads(jdata["result"]),jdata["id"])
34         elif("error" in jdata.keys()):
35             return (False,jdata["error"],jdata["id"])
36         else:
37             return (False,None,None)
38     except Exception as e:
39         return (False,None,None)
40
41 def send_Point(sock,cmd,params=None,id=1):
42     if(not params):
43         params=[]
44     else:
45         params=json.dumps(params)
46     sendStr="{{\"method\":\"{0}\",\"params\":{1},\"jsonrpc\":\"2.0\",\"
       id\":{2}}}".format(cmd,params,id)+"\n"
47     sock.sendall(bytes(sendStr,"utf-8"))
48
49 def wait_stop():
50     while True:
51         time.sleep(0.01)
52         ret1, result1, id1 = sendCMD(sock, "getRobotState")
53         if (ret1):
54             if result1 == 0 or result1 == 4:
55                 break
56         else:
57             print("getRobotState failed")
```

```
58              break
59
60  if __name__ == "__main__":
61      ip = "192.168.1.200"
62      conSuc, sock = connectETController(ip)
63      start_pos = [0, -90, 0, -90, 90, 0, 0, 0]
64      ref_pos = [0, 0, 0, 0, 0, 0]
65      res = 0
66      if conSuc:
67          suc, result, id = sendCMD(sock, "getRobotState")
68          if (result == 4):
69              # 清除报警
70              suc, result, id = sendCMD(sock, "clearAlarm")
71              time.sleep(0.5)
72          # 获取同步状态
73          suc, result, id = sendCMD(sock,"getMotorStatus")
74          if result != True:
75              # 同步伺服编码器数据
76              suc, result, id = sendCMD(sock,"syncMotorStatus")
77              time.sleep(0.5)
78          time.sleep(0.5)
79          # 获取机械臂伺服状态
80          suc, result, id = sendCMD(sock, "getServoStatus")
81          if result == 0:
82              # 设置机械臂伺服状态ON
83              ret, result, id = sendCMD(sock, "set_servo_status", {"
                    status": 1})
84          suc, result, id = sendCMD(sock, "moveByJoint", {"targetPos":
                start_pos, "speed": 50})
85          wait_stop()
86          suc, result, id = sendCMD(sock, "start_push_pos", {"path_lenth"
                : 10, "pos_type": 0, "ref_joint_pos": start_pos, "ref_frame"
                : ref_pos, "ret_flag": 0})
87          print(result)
88          time.sleep(0.2)
89          for i in range(0, 10):
90              send_Point(sock, "push_pos", {"timestamp": res, "pos":
                    start_pos})
91              start_pos[0] += 0.02
```

```
92              res = res + 0.002
93          time.sleep(1)
94          suc, result, id = sendCMD(sock, "stop_push_pos")
95          print(result)
96          suc, result, id = sendCMD(sock, "check_trajectory")
97          print(result)
98          suc, result, id = sendCMD(sock, "start_trajectory", {"
               speed_percent": 50})
99          print(result)
100         time.sleep(5)
101         suc, result, id = sendCMD(sock, "flush_trajectory")
102         print(result)
103     else:
104         print("连接失败")
105     disconnectETController(sock)
```

2.3 Examples

2.3.1 Example 1



```
1  import socket
2  import json
3  import time
4  import random
5
6  def connectETController(ip,port=8055):
7      sock = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
8      try:
9          sock.connect((ip,port))
10         return (True,sock)
11     except Exception as e:
12         sock.close()
13         return (False,None)
14
15 def disconnectETController(sock):
16     if(sock):
17         sock.close()
18         sock=None
19     else:
```

```
20          sock=None
21
22  def sendCMD(sock,cmd,params=None,id=1):
23      if(not params):
24          params=[]
25      else:
26          params=json.dumps(params)
27      sendStr="{{\"method\":\"{0}\",\"params\":{1},\"jsonrpc\":\"2.0\",\"
            id\":{2}}}".format(cmd,params,id)+"\n"
28      try:
29          sock.sendall(bytes(sendStr,"utf-8"))
30          ret =sock.recv(1024)
31          jdata=json.loads(str(ret,"utf-8"))
32          if("result" in jdata.keys()):
33              return (True,json.loads(jdata["result"]),jdata["id"])
34          elif("error" in jdata.keys()):
35              return (False,jdata["error"],jdata["id"])
36          else:
37              return (False,None,None)
38      except Exception as e:
39          return (False,None,None)
40
41  if __name__ == "__main__":
42      # 机器人IP地址
43      robot_ip="192.168.1.202"
44      conSuc,sock=connectETController(robot_ip)
45      print(conSuc)
46      if(conSuc):
47          # 清除警报
48          ret, result, id = sendCMD(sock, "clearAlarm")
49          print("清除报警")
50          print("ret = ", ret, " ", "id = ", id)
51          if (ret == True):
52              print("result = ", result)
53              time.sleep(1)
54          else:
55              print("err_msg = ", result["message"])
56          # 获取同步状态
57          ret, result, id = sendCMD(sock, "getMotorStatus")
```

```
        print("获取同步状态")
        print("ret = ", ret, " ", "id = ", id)
        if (ret == True):
            print("result = ", result)
            if(result != 1):
                # 同步
                ret1, result1, id = sendCMD(sock, "syncMotorStatus")
                print("同步")
                print("ret = ", ret1, " ", "id = ", id)
                if (ret1 == True):
                    print("result = ", result1)
                    time.sleep(0.5)
                else:
                    print("err_msg = ", result1["message"])
        else:
            print("err_msg = ", result["message"])


        # 打开伺服
        ret, result, id = sendCMD(sock, "set_servo_status", {"status"
            :1})
        print("打开伺服")
        print("ret = ", ret, " ", "id = ", id)
        if (ret == True):
            print("result = ", result)
            time.sleep(1)
        else:
            print("err_msg = ", result["message"])
        # 获取伺服状态
        ret, result, id = sendCMD(sock, "getServoStatus")
        print("ret = ", ret, " ", "id = ", id)
        if(ret == True):
            print("result = ", result)
        else:
            print("err_msg = ", result["message"])
    else:
        print("连接失败")
    disconnectETController(sock)
```

2.3.2 Example 2

```python
import socket
import json
import time


# v1.2

def connectETController(ip, port=8055):
    sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    try:
        sock.connect((ip, port))
        return (True, sock)
    except Exception as e:
        sock.close()
        return (False, None)


def disconnectETController(sock):
    if (sock):
        sock.close()
        sock = None
    else:
        sock = None


def sendCMD(sock, cmd, params=None, id=1):
    if (not params):
        params = []
    else:
        params = json.dumps(params)
    sendStr = "{{\"method\":\"{0}\",\"params\":{1},\"jsonrpc
       \":\"2.0\",\"id\":{2}}}".format(cmd, params, id) + "\n"
    try:
        # print(sendStr)
        sock.sendall(bytes(sendStr, "utf-8"))
        ret = sock.recv(1024)
        jdata = json.loads(str(ret, "utf-8"))
        if ("result" in jdata.keys()):
```

```
38              return (True, json.loads(jdata["result"]), jdata["id"])
39          elif ("error" in jdata.keys()):
40              return (False, jdata["error"], jdata["id"])
41          else:
42              return (False, None, None)
43      except Exception as e:
44          return (False, None, None)
45
46
47  if __name__ == "__main__":
48      ip = "192.168.1.205"
49      conSuc, sock = connectETController(ip)
50      # print(conSuc)
51      if (conSuc):
52          # 获取机器人状态
53          ret, result, id = sendCMD(sock, "getRobotState")
54          print("获取机器人状态")
55          print("ret = ", ret, " ", "id = ", id)
56          if (ret == True):
57              print("result = ", result)
58          else:
59              print("err_msg = ", result["message"])
60          # 获取机器人模式
61          ret, result, id = sendCMD(sock, "getMotorStatus")
62          print("获取机器人模式")
63          print("ret = ", ret, " ", "id = ", id)
64          if (ret == True):
65              print("result = ", result)
66          else:
67              print("err_msg = ", result["message"])
68          # 获取机器人当前位置信息
69          ret, result, id = sendCMD(sock, "get_joint_pos")
70          print("获取机器人当前位置信息")
71          print("ret = ", ret, " ", "id = ", id)
72          if (ret == True):
73              print("result = ", result)
74          else:
75              print("err_msg = ", result["message"])
76          # 获取机器人当前位姿信息
```

```
77          print("获取机器人当前位姿信息")
78          ret, result, id = sendCMD(sock, "get_tcp_pose")
79          print("ret = ", ret, " ", "id = ", id)
80          if (ret == True):
81              print("result = ", result)
82          else:
83              print("err_msg = ", result["message"])
84          # 获取模拟量输入的值
85          ret, result, id=sendCMD(sock, "getAnalogInput", {"addr":1})
86          print("获取模拟量输入的值")
87          print("ret = ", ret, " ", "id = ", id)
88          if(ret == True):
89              print("result = ", result)
90          else:
91              print("err_msg = ", result["message"])
92      else:
93          print("连接失败")
94      disconnectETController(sock)
```

2.3.3 Example 3

```
1  import socket
2  import json
3  import time
4
5
6
7  def connectETController(ip, port=8055):
8      sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
9      try:
10         sock.connect((ip, port))
11         return (True, sock)
12     except Exception as e:
13         sock.close()
14         return (False, None)
15
16
17 def disconnectETController(sock):
18     if (sock):
```

```
19          sock.close()
20          sock = None
21      else:
22          sock = None
23
24
25  def sendCMD(sock, cmd, params=None, id=1):
26      if (not params):
27          params = []
28      else:
29          params = json.dumps(params)
30      sendStr = "{{\"method\":\"{0}\",\"params\":{1},\"jsonrpc
            \":\"2.0\",\"id\":{2}}}".format(cmd, params, id) + "\n"
31      try:
32          # print(sendStr)
33          sock.sendall(bytes(sendStr, "utf-8"))
34          ret = sock.recv(1024)
35          jdata = json.loads(str(ret, "utf-8"))
36          if ("result" in jdata.keys()):
37              return (True, json.loads(jdata["result"]), jdata["id"])
38          elif ("error" in jdata.keys()):
39              return (False, jdata["error"], jdata["id"])
40          else:
41              return (False, None, None)
42      except Exception as e:
43          return (False, None, None)
44
45
46  if __name__ == "__main__":
47      ip = "192.168.1.205"
48      conSuc, sock = connectETController(ip)
49      # print(conSuc)
50      if (conSuc):
51          # 切换当前工具号为0
52          ret, result, id = sendCMD(sock, "setToolNumber", {"tool_num":
                0})
53          print("切换当前工具号为0")
54          print("ret = ", ret, " ", "id = ", id)
55          if (ret == True):
```

```
56              print("result = ", result)
57              time.sleep(3)
58          else:
59              print("err_msg = ", result["message"])
60          # 设置机械臂的负载和重心
61          ret, result, id = sendCMD(sock, "cmd_set_payload", {"tool_num":
                0, "m": 6, "cog": [20.2, 40, 30.5]})
62          print("设置负载和重心")
63          print("ret = ", ret, " ", "id = ", id)
64          if (ret == True):
65              print("result = ", result)
66          else:
67              print("err_msg = ", result["message"])
68          # 设置机械臂工具中心
69          point1 = [1.002, -2.5, 5.0, 0.74, -1.57, 0]
70          ret, result, id = sendCMD(sock, "cmd_set_tcp", {"tool_num": 0,
               "point": point1})  # cmd_set_tcp
71          print("ret = ", ret, " ", "id = ", id)
72          if (ret == True):
73              print("result = ", result)
74          else:
75              print("err_msg = ", result["message"])
76      else:
77          print("连接失败")
78      disconnectETController(sock)
```

第 3 章 监控接口

用户可通过 socket 客户端连接机器人监控接口，获取机器人信息。

警告



在示教器界面，选择“系统 > 系统配置 > 机器人配置”，勾选“通用”下的“远程”后，才可以使用此功能。

提醒



该功能适用于 2.13.1 及以上版本。

3.1 监控接口数据说明列表

名称	类型	字节	说明
Message Size	unsigned int32	4*1	当前数据包长度为 1024，有效字段长度为 607，预留字段长度为 417
timestamp	unsigned int64	8*1	时间戳,1970 年 1 月 1 日至今的毫秒数
autorun_cycelMode	unsigned char	1*1	循环模式，0：单步，1：单循环，2：连续
machinePos	double[AXIS_COUNT]	8*8	关节角度，单位度
machinePose	Double[6]	8*6	直角坐标，前三项单位毫米，后三项单位弧度
machineUserPose	Double[6]	8*6	当前用户坐标，前三项单位毫米，后三项单位弧度
torque	double[AXIS_COUNT]	8*8	关节额定力矩千分比，单位 ‰，

名称	类型	字节	说明
robotState	int32_t	4*1	机器人状态：0：停止，1：暂停，2：急停，3：运行，4：报警
servoReady	int32_t	4*1	抱闸状态：0：未打开，1：已打开。
can_motor_run	int32_t	4*1	同步状态：0：未同步，1：同步
motor_speed	int[AXIS_COUNT]	4*8	电机速度，单位：转/分
robotMode	int32_t	4*1	机器人模式：0：示教模式，1：自动模式，2：远程模式
analog_ioInput	double[ANALOG_IN_NUM]	8*3	模拟量输入口数据，单位 V
analog_ioOutput	double[ANALOG_OUT_NUM]	8*5	模拟量输出口数据，单位 V
digital_ioInput	unsigned int64	8*1	数字量输入口数据
digital_ioOutput	unsigned int64	8*1	数字量输出口数据
collision	unsigned char	1*1	碰撞报警状态，0：非碰撞报警状态，1：碰撞报警状态。
machineFlangePose	Double[6]	8*6	直角坐标系下的法兰盘中心的位姿，前三项单位毫米，后三项单位弧度。
machineUserFlangePose	Double[6]	8*6	用户坐标系下的法兰盘中心的位姿，前三项单位毫米，后三项单位弧度。v2.14.4 新增
emergencyStopState	unsigned char	1*1	是否为急停状态，v2.16.2 新增
tcpSpeed	Double	8*1	tcp 运动速度，单位 mm/s，v2.16.2 新增
jointSpeed	Double[AXIS_COUNT]	8*8	关节运动速度，单位度/s，v2.16.2 新增
tcpAcc	Double	8*1	tcp 加速度，单位：mm/s^2
jointAcc	Double[ANALOG_OUT_NUM]	8*8	关节加速度，单位：度/s
Reserved	unsigned char	1*417	数据包预留长度，长度为 417

3.2 Example

```
import socket
import struct
import collections
import time
import math
HOST = "192.168.1.202"
PORT = 8056

while 1:
      s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
      s.settimeout(8)
      s.connect((HOST,PORT))
      index = 0
      lost = 0

      while True:
      dic = collections.OrderedDict()
      dic['MessageSize'] = 'I'
      dic['TimeStamp'] = 'Q'
      dic['autorun_cycleMode'] = 'B'
      dic['machinePos01'] = 'd'
      dic['machinePos02'] = 'd'
      dic['machinePos03'] = 'd'
      dic['machinePos04'] = 'd'
      dic['machinePos05'] = 'd'
      dic['machinePos06'] = 'd'
      dic['machinePos07'] = 'd'
      dic['machinePos08'] = 'd'
      dic['machinePose01'] = 'd'
      dic['machinePose02'] = 'd'
      dic['machinePose03'] = 'd'
      dic['machinePose04'] = 'd'
      dic['machinePose05'] = 'd'
      dic['machinePose06'] = 'd'
      dic['machineUserPose01'] = 'd'
```

```
36          dic['machineUserPose02'] = 'd'
37          dic['machineUserPose03'] = 'd'
38          dic['machineUserPose04'] = 'd'
39          dic['machineUserPose05'] = 'd'
40          dic['machineUserPose06'] = 'd'
41          dic['torque01'] = 'd'
42          dic['torque02'] = 'd'
43          dic['torque03'] = 'd'
44          dic['torque04'] = 'd'
45          dic['torque05'] = 'd'
46          dic['torque06'] = 'd'
47          dic['torque07'] = 'd'
48          dic['torque08'] = 'd'
49          dic['robotState'] = 'i'
50          dic['servoReady'] = 'i'
51          dic['can_motor_run'] = 'i'
52          dic['motor_speed01'] = 'i'
53          dic['motor_speed02'] = 'i'
54          dic['motor_speed03'] = 'i'
55          dic['motor_speed04'] = 'i'
56          dic['motor_speed05'] = 'i'
57          dic['motor_speed06'] = 'i'
58          dic['motor_speed07'] = 'i'
59          dic['motor_speed08'] = 'i'
60          dic['robotMode'] = 'i'
61          dic['analog_ioInput01'] = 'd'
62          dic['analog_ioInput02'] = 'd'
63          dic['analog_ioInput03'] = 'd'
64          dic['analog_ioOutput01'] = 'd'
65          dic['analog_ioOutput02'] = 'd'
66          dic['analog_ioOutput03'] = 'd'
67          dic['analog_ioOutput04'] = 'd'
68          dic['analog_ioOutput05'] = 'd'
69          dic['digital_ioInput'] = 'Q'
70          dic['digital_ioOutput'] = 'Q'
71          dic['collision'] = 'B'
72          dic['machineFlangePose01'] = 'd'
73          dic['machineFlangePose02'] = 'd'
74          dic['machineFlangePose03'] = 'd'
```

```
75        dic['machineFlangePose04'] = 'd'
76        dic['machineFlangePose05'] = 'd'
77        dic['machineFlangePose06'] = 'd'
78        dic['machineUserFlangePose01'] = 'd'
79        dic['machineUserFlangePose02'] = 'd'
80        dic['machineUserFlangePose03'] = 'd'
81        dic['machineUserFlangePose04'] = 'd'
82        dic['machineUserFlangePose05'] = 'd'
83        dic['machineUserFlangePose06'] = 'd'
84        dic["emergencyStopState"] = "B"
85        dic["tcp_speed"] = "d"
86        dic["joint_speed01"] = "d"
87        dic["joint_speed02"] = "d"
88        dic["joint_speed03"] = "d"
89        dic["joint_speed04"] = "d"
90        dic["joint_speed05"] = "d"
91        dic["joint_speed06"] = "d"
92        dic["joint_speed07"] = "d"
93        dic["joint_speed08"] = "d"
94        dic["tcpacc"] = "d"
95        dic["jointacc01"] = "d"
96        dic["jointacc02"] = "d"
97        dic["jointacc03"] = "d"
98        dic["jointacc04"] = "d"
99        dic["jointacc05"] = "d"
100       dic["jointacc06"] = "d"
101       dic["jointacc07"] = "d"
102       dic["jointacc08"] = "d"
103
104       print("index =", index)
105       data = s.recv (1024)
106       if len(data) != 1024:
107             lost += 1
108             print(str(lost))
109             continue
110
111       names =[]
112       ii=range(len(dic))
113       for key ,i in zip(dic ,ii):
```

```
114             fmtsize = struct. calcsize (dic[key ])
115             data1 , data = data [0: fmtsize], data[fmtsize :]
116             fmt="!" + dic[key]
117             names.append(struct.unpack(fmt ,data1))
118             dic[key] = dic[key], struct.unpack(fmt , data1)
119         output = ""
120         for key in dic.keys ():
121             output += str(key) + ":" + str(dic[key ][1][0]) + ";\n"
122         output = "lost : " + str(lost) + " index : " + str(index) + ";" +
                output + "\n"
123         if dic['MessageSize'] != ('I', (1024,)):
124             s.close()
125             break
126
127         if index %10 == 0:
128             # 打印所有信息
129             print(output)
130             # 打印数据包长度
131             print(dic['MessageSize'])
132             # 打印时间戳
133             timestamp01_value = dic['TimeStamp'][1][0] // 1000
134             timeValue = time.gmtime(int(timestamp01_value))
135             print(time.strftime("%Y-%m-%d %H:%M:%S", timeValue))
136             # 打印关节坐标
137             print(dic['machinePos01'][1][0], dic['machinePos02'][1][0],
138                   dic['machinePos03'][1][0], dic['machinePos04'][1][0],
139                   dic['machinePos05'][1][0], dic['machinePos06'][1][0],
140                   dic['machinePos07'][1][0], dic['machinePos08'][1][0])
141             # 打印直角坐标
142             print(dic['machinePose01'][1][0], dic['machinePose02'
                    ][1][0], dic['machinePose03'][1][0],
143                   dic['machinePose04'][1][0],
144                   dic['machinePose05'][1][0], dic['machinePose06'
                        ][1][0])
145             # 打印用户坐标
146             print(dic['machineUserPose01'][1][0], dic['
                    machineUserPose02'][1][0], dic['machineUserPose03'
                    ][1][0],
147                   dic['machineUserPose04'][1][0],
```

```
148                 dic['machineUserPose05'][1][0], dic['
                        machineUserPose06'][1][0])
149             # 打印关节额定力矩百分比
150             print(dic['torque01'][1][0], dic['torque02'][1][0], dic['
                    torque03'][1][0], dic['torque04'][1][0],
151                 dic['torque05'][1][0],
152                 dic['torque06'][1][0], dic['torque07'][1][0], dic['
                        torque08'][1][0])
153             # 打印机器人状态
154             print(dic['robotState'][1][0])
155             # 打印伺服使能状态
156             print(dic['servoReady'][1][0])
157             # 打印同步状态
158             print(dic['can_motor_run'][1][0])
159             # 打印各轴电机转速
160             print(dic['motor_speed01'][1][0], dic['motor_speed02'
                    ][1][0], dic['motor_speed03'][1][0],
161                 dic['motor_speed04'][1][0], dic['motor_speed05'
                        ][1][0],
162                 dic['motor_speed06'] [1][0], dic['motor_speed07'
                        ][1][0], dic['motor_speed08'][1][0])
163             # 打印机器人模式
164             print(dic['robotMode'][1][0])
165             # 打印模拟量输入口数据
166             print(dic['analog_ioInput01'][1][0], dic['analog_ioInput02'
                    ][1][0], dic['analog_ioInput03'][1][0])
167             # 打印模拟量输出口数据
168             print(dic['analog_ioOutput01'][1][0], dic['
                    analog_ioOutput02'][1][0], dic['analog_ioOutput03'
                    ][1][0],
169                 dic['analog_ioOutput04'][1][0], dic['
                        analog_ioOutput05'][1][0])
170             # 打印数字量输入口数据的二进制形式
171             print(bin(dic['digital_ioInput'][1][0])[2:].zfill(64))
172             # 打印数字量输出口数据的二进制形式
173             print(bin(dic['digital_ioOutput'][1][0])[2:].zfill(64))
174             # 打印碰撞报警状态
175             print(dic["collision"][1][0])
176             # 打印直角坐标系下的法兰盘中心位姿
```

```
177             print(dic['machineFlangePose01'][1][0], dic['
                    machineFlangePose02'][1][0], dic['machineFlangePose03'
                    ][1][0],
178                   dic['machineFlangePose04'][1][0],
179                   dic['machineFlangePose05'][1][0], dic['
                          machineFlangePose06'][1][0])
180             # 打印用户坐标系下的法兰盘中心位姿
181             print(dic['machineUserFlangePose01'][1][0], dic['
                    machineUserFlangePose02'][1][0],
182                   dic['machineUserFlangePose03'][1][0], dic['
                          machineUserFlangePose04'][1][0],
183                   dic['machineUserFlangePose05'][1][0], dic['
                          machineUserFlangePose06'][1][0])
184             # 打印当前是否处于急停状态
185             print(dic["emergencyStopState"][1][0])
186             # 打印tcp运动速度
187             print(dic["tcp_speed"][1][0])
188             # 打印关节运动下各关节运动速度
189             print(dic['joint_speed01'][1][0], dic['joint_speed02'
                    ][1][0],
190                   dic['joint_speed03'][1][0], dic['joint_speed04'
                          ][1][0],
191                   dic['joint_speed05'][1][0], dic['joint_speed06'
                          ][1][0],
192                   dic['joint_speed07'][1][0], dic['joint_speed08'
                          ][1][0])
193             # 打印tcp加速度
194             print(dic["tcpacc"][1][0])
195             # 打印关节运动下各关节加速度
196             print(dic['jointacc01'][1][0], dic['jointacc02'][1][0],
197                   dic['jointacc03'][1][0], dic['jointacc04'][1][0],
198                   dic['jointacc05'][1][0], dic['jointacc06'][1][0],
199                   dic['jointacc07'][1][0], dic['jointacc08'][1][0])
200       index = index +1
201       output = ""
202       dic = {}
203       data = ""
204       time.sleep(0.008)
205       s.close ()
```

第 4 章 日志接口

用户可通过 socket 客户端连接机器人日志接口。

日志类型为：Error,Warning,Info。若输入 Error 类型，则获取 Error 信息；若输入 Warning 类型，则获取 Error 和 Warning 类型的日志信息；若输入 Info，则获取所有类型的日志信息。

连接后，输入 all，输入全部日志；输入数字，如 10，输出最后 10 行日志；输入 exit，退出连接。

4.1 Example

```python
import socket
HOST = "192.168.1.202"
PORT = 8058

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.settimeout(2)
s.connect((HOST,PORT))
# 新建文件或清空文件内容
# file = open(r'D:\205\log\all_err_log.md', 'w').close()
# 获取全部日志信息
str1 = "Type=Info\n"
s.send(str1.encode())
str2 = "all\n"
s.send(str2.encode())
while True:
    try:
        data = s.recv(128000)
        # with open(r'D:\205\log\all_err_log.md', 'a+') as f:
        #     f.write(data.decode())
        print(data.decode())
    except(Exception):
        break
s.close()
```

第 5 章 原始日志接口

用户可通过 socket 客户端连接机器人原始日志接口。

连接后，输入 all，输入全部日志；输入数字，如 10，输出最后 10 行日志；输入 exit，退出连接。

提醒



该功能适用于 2.14.0 及以上版本。

5.1 Example

```python
import socket
HOST = "192.168.1.205"
PORT = 8059

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.settimeout(2)
s.connect((HOST,PORT))
# 新建文件或清空文件内容
file = open(r'D:\205\log\err_log_1.md', 'w').close()

# 获取最近10条日志条目，发送的数字对应获取的日志条目数
str1 = "10\n"
s.send(str1.encode())
while True:
    try:
        data = s.recv(1024)
        with open(r'D:\205\log\err_log_1.md','a+') as f:
            f.write(data.decode())
        print(data.decode())
    except(Exception):
        break
s.close()
```

ELITE ROBOT
艾利特机器人

明天比今天更简单一点

联系我们

商务合作：market@elibot.cn

技术咨询：tech@elibot.cn

苏州公司（生产研发基地）

苏州市工业园区长阳街 259 号中新钟园工业坊 4 号楼 1F（总部）

+86-400-189-9358

+86-0512-83951898

北京分公司

北京市北京经济技术开发区荣华南路 2 号院 6 号楼 1102 室

上海分公司

上海市浦东新区学林路 36 弄 18 号楼

深圳技术服务中心

深圳市宝安区航空路泰华梧桐岛科技创新园 1A 栋 202 室

关注公众号了解更多