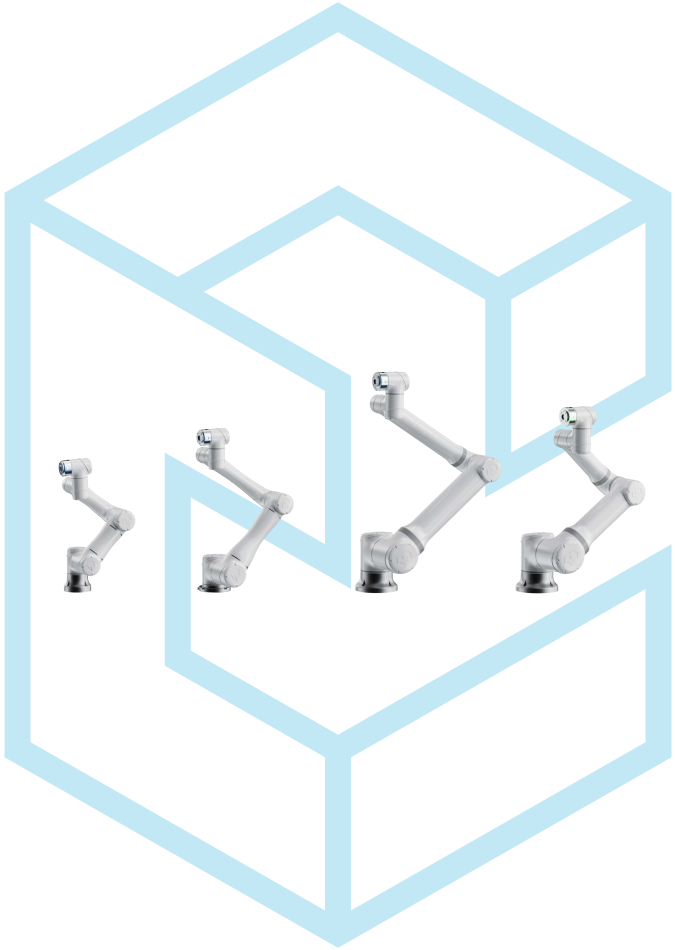
ELITE ROBOTS
艾利特机器人

ELITE ROBOTS EC系列
编程手册



SDK-Socket手册

艾利特智能机器人股份有限公司

2025-12-31

版本：Ver3.22.2

目录

第 1 章 简介

提醒

本手册适用软件版本为：Ver3.22.2。

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}
```

```
12
13      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。

```python
import socket
import json
import time

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,)
```

```
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 2.2.2.61 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":"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[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):
    # 获取机器人输出端关节位置信息
    suc， result ， id = sendCMD(sock, "get_joint_pos")
```

注意： 指令 getRobotPos 不再维护，会逐渐废弃，建议使用 get_joint_pos 来获取机器人输出端关节位置信息。

2.2.2.4 获取机器人当前位姿信息

```
{"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[-1,15], -1: 基坐标系，0~15: 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；若都不使用或只使用参数 tool_num，则返回基坐标系下的机器人位姿，参数 coordinate_num 不可单独使用。

指令 getRobotPose 和 getTcpPose 不再维护，会逐渐废弃，建议使用 get_tcp_pose 来获取机器人当前位姿信息。

2.2.2.5 获取机器人电机速度

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

功能： 获取机器人电机速度

参数： 无

返回： 机器人电机速度 double speed[6]

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])
  point.append([-16.2806,-82.4996,81.9848,-89.4851,
  90.0000,-16.2858])
  point.append([3.7679, -71.7544, 68.7276, -86.9732,
  90.0000, 3.7627])
  point.append([12.8237,-87.3028,87.2361,-89.9333,
  90.0000,12.8185])
  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):
        # 获取机器人电机速度
        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 不再维护，会逐渐废弃，建议使用 get_motor_speed 来获取机器人马达速度。

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，基座 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[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):
    # 获取机器人当前编码器值列表
    suc， result， id = sendCMD(sock, "getCurrentEncode")
```

2.2.2.10 获取机器人示教模式下的当前工具号

```
{"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):
    # 获取机器人示教模式下的当前工具号
    suc， result， id = sendCMD(sock, "getToolNumber")
```

2.2.2.11 切换机器人示教模式下的当前工具号

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

功能： 切换机器人示教模式下的当前工具号

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):
    # 切换机器人示教模式下的当前工具号
    suc, result ,id=sendCMD(sock,"setToolNumber",{"tool_num":7})
    time.sleep (0.5)
    # 获取机器人示教模式下的当前工具号
    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。
本命令只支持在 remote 模式下使用。

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[0,15]

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[0,15]

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":"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[6]，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： getRobotTorques 不再维护，会逐渐废弃，建议使用 get_motor_torque 来获取机器人当前力矩信息。

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： 获取轨迹运动当前运行点位序号

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]
  C001 = [-16.2806,-82.4996,81.9848,-89.4851,
   90.0000,-16.2858]
  C002 = [3.7679, -71.7544, 68.7276, -86.9732,
  90.0000, 3.7627]
  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, "circular_radius":20})
      suc, result , id = sendCMD(sock, "addPathPoint", {"wayPoint": C001,"moveType":0, "speed":
          50, "circular_radius":20})
      suc, result , id = sendCMD(sock, "addPathPoint", {"wayPoint": C002,"moveType": 0, "speed":
            50, "circular_radius":0})
      # 轨迹运动2.0
      suc, result , id = sendCMD(sock, "moveByPath")
      while(True):
        # 获取轨迹运动当前运行点位序号
        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，基座：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 获取机器人拖动使能状态

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

功能： 获取机器人拖动使能状态

参数： 无

返回： 拖动使能处于打开状态true，拖动使能处于关闭状态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):
    # 获取机器人拖动使能状态
    suc, result ,id=sendCMD(sock,"get_drag_state")
```

2.2.2.19 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 不再维护，逐渐废弃。
本命令只支持在 remote 模式下使用。

2.2.2.20 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。

2.2.2.21 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 )
```

2.2.2.22 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。

2.2.2.23 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。

2.2.2.24 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。

2.2.2.25 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"])
```

2.2.2.26 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： 本指令不再维护，逐渐废弃。

2.2.2.27 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： 本指令不再维护，逐渐废弃。

2.2.2.28 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"])
```

2.2.2.29 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"])
```

2.2.2.30 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： 本命令不再维护，逐渐废弃。

2.2.2.31 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。
本命令不再维护，逐渐废弃。

2.2.2.32 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)
```

2.2.2.33 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：

```python
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： 本命令支持在 remote 模式和自动模式下使用。

2.2.2.34 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"])
```

2.2.2.35 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"])
```

注意： 本命令只支持在 remote 模式下使用。

2.2.2.36 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。

2.2.2.37 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。

2.2.2.38 获取基座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： 获取基座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： 基座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):
        # 获取基座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。

2.2.2.39 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。

2.2.2.40 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。

2.2.2.41 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。

2.2.2.42 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,"getRobotSafetyPower")
        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。

2.2.2.44 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。

2.2.2.45 获取当前tcp坐标系下的“外部”力和力矩

```
{"jsonrpc":"2.0","method":"get_tcp_force","params":{"ref_tcp":ref_tcp},"id":id}
```

功能： 获取当前 tcp 的“外部”力和力矩

参数： ref_tcp：参考坐标系，int[0,1]，可选参数, 0代表基坐标系，1代表tcp坐标系，不写默认为tcp坐标系。

返回： 当前 tcp的“外部”力和力矩double force[6]，前三位为外部力，后三位为力矩

示例：

```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="172.16.11.248"
  conSuc,sock=connectETController(robot_ip)
  if(conSuc):
```

```
    # 获取当前tcp在基坐标系下的力和力矩
    suc， result ， id = sendCMD(sock, "get_tcp_force",{"ref_tcp": 0})
    print ( suc, result, id)
else :
    print ("连接失败")
disconnectETController (sock)
```

注： 力和力矩的默认方向为tcp。

2.2.2.46 获取当前关节力矩

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

功能： 获取当前关节力矩

参数： 无

返回：当前关节力矩double torques[6]

示例：

```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip = "172.16.11.248"
  conSuc, sock = connectETController( robot_ip )
  if (conSuc):
    # 获取机器人当前关节力矩
    suc， result ， id = sendCMD(sock, "get_joint_torques")
    print ( suc, result, id )
  else :
    print ("连接失败")
  disconnectETController (sock)
```

注： 关节力矩为电机力矩减去“自身驱动所需力矩”，反应的是“外部”力矩。

2.2.2.47 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。

2.2.2.48 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。

2.2.2.49 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。

2.2.2.50 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:
        # 获取机器的最大力矩误差百分比
        suc, result , id =sendCMD(sock,"getRobotTorqueErrorMaxPercents")
        print ( result )
```

注意： 本命令适用于 v2.16.2 及以上版本。

2.2.2.51 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:
        # 设置末端按钮状态
        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。

2.2.2.52 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,"checkFlangeButton",{"button_num":0})
        print （suc， result ， id）
```

注意： 本命令适用于 v2.16.2 及以上版本。

2.2.2.53 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)
```

2.2.2.54 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)
```

2.2.2.55 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)
    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)
```

2.2.2.56 获取机器人报警信息序列号

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

功能： 获取机器人报警信息序列号

参数： 无

返回： 成功返回最近 5 条机器人报警信息的序列号，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)
```

2.2.2.57 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[6], 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)
```

2.2.2.58 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)
```

2.2.2.59 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[6]，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)
```

2.2.2.60 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)
```

2.2.2.61 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 )
```

2.2.2.62 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)
```

2.2.2.63 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[6]: 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)
```

2.2.2.64 获取机器人伺服编码器精确状态

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

功能： 获取机器人伺服编码器精确状态

参数： 无

返回： 1：精确，0：非精确

示例：

```
if __name__ == "__main__":
    # 机器人IP地址
    robot_ip = "192.168.1.202"
    conSuc, sock = connectETController(robot_ip)
    if (conSuc):
        # 获取机器人伺服编码器精确状态
        suc, result , id = sendCMD(sock, " get_servo_precise_position_status ")
        print ( result )
    else :
        print ("连接失败")
    disconnectETController (sock)
```

2.2.2.65 获取机器人伺服报警状态

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

功能： 获取机器人伺服报警状态

参数： 无

返回： 1：处于伺服报警状态，0：不处于伺服报警状态

示例：

```
if __name__ == "__main__":
    # 机器人IP地址
    robot_ip = "172.16.11.248"
    conSuc, sock = connectETController(robot_ip)
    if (conSuc):
        # 获取机器人伺服报警状态
        suc, result , id = sendCMD(sock, "get_servo_alarm_state")
        print ( result )
    else :
        print ("连接失败")
    disconnectETController (sock)
```

2.2.2.66 清除预约队列

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

功能： 清除预约队列

参数： 无

返回： 成功True，失败False

示例：
```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip = "172.16.11.248"
  conSuc, sock = connectETController(robot_ip)
  if (conSuc):
    # 清除预约队列
    suc, result , id = sendCMD(sock, "book_program_clear")
    print ( result )
  else :
    print ("连接失败")
  disconnectETController (sock)
```

注意： 本命令只有在remote模式下才会生效，且机器人必须处于停止或报警（即静止）情况下，才能清空预约队列。

2.2.2.67 获取真实的末端位姿数据

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

功能： 获取基坐标系或者指定用户坐标系下的真实末端位姿数据

参数： tool_num：工具坐标编号，可选，int[0,7]，无传参则表示当前工具号，否则为指定工具号

user_num：用户坐标编号，可选，int[0,15]， 无传参则表示获取的位姿是在基坐标 下，否则为指定用户坐标系下的位姿

返回： 机器人位姿信息 double pose [6]

示例：
```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="172.16.11.248"
  conSuc,sock=connectETController(robot_ip)
  if (conSuc):
    # 获取1号用户坐标系下1号工具的真实末端位姿数据
    suc, result ,id=sendCMD(sock,"get_actual_tcp",{"tool_num":1," user_num ":1})
    print ( result )
  else :
    print ("连接失败")
  disconnectETController (sock)
```

2.2.2.68 获取目标插补末端位姿数据

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

功能： 获取基坐标系或者指定用户坐标系下的目标插补末端位姿数据

参数： tool_num：工具坐标编号，可选，int[0,7]，无传参则表示当前工具号，否则为指定工具号

user_num：用户坐标编号，可选，int[0,15]， 无传参则表示获取的位姿是在基坐标 下，否则为指定用户坐标系下的位姿

返回： 机器人位姿信息 double pose[6]

示例：

```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="172.16.11.248"
  conSuc,sock=connectETController(robot_ip)
  if (conSuc):
    # 获取1号用户坐标系下1号工具的目标插补末端位姿数据
    suc, result ,id=sendCMD(sock,"get_target_tcp",{"tool_num":1,"user_num":1})
    print ( result )
  else :
    print ("连接失败")
  disconnectETController (sock)
```

2.2.2.69 获取真实的关节数据

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

功能： 获取当前真实的关节数据

参数： 无

返回： 机器人关节信息 double joint[6]

示例：

```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="172.16.11.248"
  conSuc,sock=connectETController(robot_ip)
  if (conSuc):
    # 获取当前真实的关节数据
    suc, result ,id=sendCMD(sock,"get_actual_joint")
    print ( result )
  else :
```

```
    print ("连接失败")
disconnectETController (sock)
```

2.2.2.70 获取目标插补关节数据

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

功能： 获取当前的目标插补关节数据

参数： 无

返回： 机器人关节信息 double joint[6]

示例：

```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="172.16.11.248"
  conSuc,sock=connectETController( robot_ip )
  if (conSuc):
    # 获取当前目标插补关节数据
    suc, result , id=sendCMD(sock,"get_target_joint")
    print ( result )
  else :
    print ("连接失败")
  disconnectETController (sock)
```

2.2.2.71 获取线性插补位姿数据

```
{"jsonrpc":"2.0","method":"get_interp_pose","params":{"data1":data1,
   "data2":data2, "ratio":ratio},"id":id}
```

功能： 获取指定的两个位姿之间的线性插补位姿数据

参数： data1：位姿数据，必选，double pose[6]，前三位代表位置，单位为毫米，范围是[-1e+09~1e+09]，后三位代表姿态，单位为弧度，范围是[-π,π]

data2：位姿数据，必选，double pose[6]，前三位代表位置，单位为毫米，范围是[-1e+09~1e+09]，后三位代表姿态，单位为弧度，范围是[-π,π]

ratio：浮点型数据，必选，代表比例值，范围是[0,1]，当数值=0时，机器人返回第一个位姿，当数值=1时，机器人返回第二个位姿

返回： 机器人位姿信息 double pose [6]

示例：

```
if __name__ == "__main__":
  # 机器人IP地址
```

```
robot_ip="172.16.11.248"
conSuc,sock=connectETController( robot_ip )
point1=[371.533, 101.636, 3.038, 0, -0.174, 2.861]
point2=[346.312, -256.945, -91.131, -0.014, 0.521, 1.903]
if (conSuc):
    # 获取指定的两个位姿之间的线性插补位姿数据
    suc, result , id=sendCMD(sock,"get_interp_pose",{"data1":point1,"data2":point2,"ratio":0.5})
    print ( result )
else :
   print ("连接失败")
disconnectETController (sock)
```

2.2.2.72 获取关节温度

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

功能： 获取关节温度

参数： 无

返回： double joint_temp[6]

示例：

```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="172.16.11.248"
  conSuc,sock=connectETController( robot_ip )
  if (conSuc):
      suc, result , id=sendCMD(sock,"get_joint_temp")
      print (suc, result, id )
  else :
      print ("连接失败")
  disconnectETController (sock)
```

2.2.2.73 获取目标位姿对应的全部逆解结果

```
{"jsonrpc":"2.0","method":"inverseKinematicAll","params":{"targetPose":
   targetPose},"id":id}
```

功能： 获取目标位姿对应的全部逆解结果。

参数： targetPose：目标位姿，double pose[6]，前三位代表位置，单位为毫米，范围是[-1e+09~1e+09]，后三位代表姿态，单位为弧度，范围是[-π,π]

返回：关节坐标 double pos[6]，范围[-π,π]

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")
    # 逆解函数2.0，位姿逆解所有的关节角度值
    suc, result ,id=sendCMD(sock,"inverseKinematicAll",{"targetPose": result})
```

2.2.2.74 设置系统时间

```
{"jsonrpc":"2.0","method":"set_system_time","params":{"time":[year, month,
   day, hour, minute, second]},"id":id}
```

功能： 设置系统时间

参数： time：可设置1970年-2037年的任意时间，int[6]，分别为年、月、日、小时、分钟和秒

返回：成功TRUE，失败 FALSE

示例：

```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip = "192.168.1.202"
  conSuc, sock = connectETController(ip)
  if (conSuc):
    # 设置系统时间
    suc, result ,id=sendCMD(sock,"set_system_time",{"time": [2023, 4,25,13,5,20]})
    print (suc， result ， id)
```

注意： 本命令适用于 v3.10.2 及以上版本。

2.2.2.75 设置末端指示灯控制模式

```
{"jsonrpc":"2.0","method":"set_ending_pilot_lamp_ctrl_mode","params":
   "mode":mode},"id":id}
```

功能： 设置末端指示灯控制模式

参数： mode：0，常亮模式，1，自定义模式

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):
    # 设置末端指示灯控制模式为自定义模式
    ret, result ,id=sendCMD(sock,"set_ending_pilot_lamp_ctrl_mode", {"mode":1})
```

注意： 本命令适用于 v3.10.2 及以上版本。

2.2.2.76 获取当前机器人末端指示灯控制模式

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

功能： 获取当前机器人末端指示灯控制模式

参数： 无

返回： 0：常亮模式，1：自定义模式

示例：

```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip = "192.168.1.200"
  conSuc, sock = connectETController(ip)
  if (conSuc):
    # 获取末端指示灯控制模式
    ret, result ,id=sendCMD(sock,"get_ending_pilot_lamp_ctrl_mode")
    print (ret, result , id)
```

注意： 本命令适用于 v3.10.2 及以上版本。

2.2.2.77 设置焊机参数

```
{"jsonrpc":"2.0","method":"set_welding_machine_parameters","params":
   {"prepareAspirationTime":prepareAspirationTime,"
   delayAspirationTime":delayAspirationTime,"arcCheckTime":arcCheckTime,
   "arcConfirmTime":arcConfirmTime,"arcExhaustedCheckTime":
   arcExhaustedCheckTime},"id":id}
```

功能： 设置焊机参数

参数： 可选参数：

prepareAspirationTime：预备送气时间，范围：double[0,10000]，单位：s
delayAspirationTime：延迟送气时间，范围：double[0,10000]，单位：s
arcCheckTime：弧检测时间，范围：double[0,10000]，单位：s
arcConfirmTime：弧检测确认时间，范围：double [0,10000]，单位：s
arcExhaustedCheckTime：弧耗尽检测时间，范围：double [0,10000]，单位：s
inchingWireFeedingTime：点动送丝时间，范围：double [0,1]，单位：s
weldSwitchSignal：焊接启动信号（数字量输出索引号），范围：int [0,19]
gasSwitchSignal：手动送气信号（数字量输出索引号），范围：int [0-19]
wireFeedSwitchSignal：手动送丝信号（数字量输出索引号），范围：int[0,19]
wireWithdrawalSwitchSignal：手动退丝信号（数字量输出索引号），范围： int[1,19]
arcSignal：起弧成功反馈信号（数字量输入索引号），范围：int[4,19]
machineAlarmSignal：焊机报警信号（数字量输入索引号），范围：int[4,19]
analogControlCurrentSignal：焊接电流（模拟量输入索引号），范围：int[1,4]
analogControlVoltageSignal：焊接电压（模拟量输入索引号），范围：int[1,4]

返回： 成功 True, 失败 False

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):
    ret , result , id = sendCMD(sock, "set_welding_machine_parameters", {"prepareAspirationTime":
        1, "delayAspirationTime": 1, "arcCheckTime": 1, "arcConfirmTime": 1})
```

注意： 本命令适用于 v3.12.2 及以上版本。

2.2.2.78 获取焊机参数

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

功能： 获取焊机参数

参数： 无

返回： 焊机参数列表

prepareAspirationTime：预备送气时间，单位：s
delayAspirationTime：延迟送气时间，单位：s
arcCheckTime：弧检测时间，单位：s
arcConfirmTime：弧检测确认时间，单位：s
arcExhaustedCheckTime：弧耗尽检测时间，单位：s
inchingWireFeedingTime：点动送丝时间，单位：s
weldSwitchSignal：焊接启动信号（数字量输出索引号）
gasSwitchSignal：手动送气信号（数字量输出索引号）
wireFeedSwitchSignal：手动送丝信号（数字量输出索引号）
wireWithdrawalSwitchSignal：手动退丝信号（数字量输出索引号）
arcSignal：起弧成功反馈信号（数字量输入索引号）
machineAlarmSignal：焊机报警信号（数字量输入索引号）
analogControlCurrentSignal：焊接电流（模拟量输入索引号）
analogControlVoltageSignal：焊接电压（模拟量输入索引号）

示例：

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

注意： 本命令适用于 v3.12.2 及以上版本。

2.2.2.79 设置工艺号对应的焊接参数

```
{"jsonrpc":"2.0","method":"set_welding_parameters","params":{"techNum":
   techNum,"note":note, "startCurrent":startCurrent,"arcCurrent":arcCurrent,"
   antiStickCurrent":antiStickCurrent,"endCurrent":endCurrent,"
   startVoltage":startVoltage,"startVoltage":startVoltage,"
   antiStickVoltage":antiStickVoltage},"id":id}
```

功能： 设置工艺号对应的焊接参数

参数： techNum：工艺号，范围：int[1,8]
可选参数，范围：电流最小值为曲线参数中的控制电流1的值，电流最大值为控制电流2的值，电压最小值为控制电压1 的值，电压最大值为控制电压2的值
note：描述，string类型，最大长度为31
startCurrent：焊接电流，double，单位：A
arcCurrent：起弧电流，double，单位：A
antiStickCurrent：防粘丝电流，double，单位：A
endCurrent：收弧电流，double，单位：A
startVoltage：焊接电压，double，单位：V
arcVoltage：起弧电压，double，单位：V
antiStickVoltage：防粘丝电压，double，单位：V
endVoltage：收弧电压，double，单位：V
antiStickTime：防粘丝时间，double，单位：s
startTime：起弧时间，double，单位：s
endTime：收弧时间，double，单位：s

返回： 成功 True, 失败 False

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):
        ret , result , id = sendCMD(sock, "set_welding_parameters", {"techNum": 2, "note": "test", "
            startCurrent": 1, "arcCurrent": 1, "antiStickCurrent": 1, "endCurrent": 1, "startVoltage": 1,
            "arcVoltage": 1, "antiStickVoltage": 1})
```

注意： 本命令适用于 v3.12.2 及以上版本。

2.2.2.80 获取工艺号对应的焊接参数

```
{"jsonrpc":"2.0","method":"get_welding_parameters","params":{"techNum":
   techNum},"id":id}
```

功能： 获取工艺号对应的焊接参数

参数： techNum：工艺号，范围：int[1,8]

返回： 焊接参数列表

note：描述，string类型，最大长度为31

startCurrent：焊接电流，double，单位：A

arcCurrent：起弧电流，double，单位：A

antiStickCurrent：防粘丝电流，double，单位：A

endCurrent：收弧电流，double，单位：A

startVoltage：焊接电压，double，单位：V

arcVoltage：起弧电压，double，单位：V

antiStickVoltage：防粘丝电压，double，单位：V

endVoltage：收弧电压，double，单位：V

antiStickTime：防粘丝时间，double，单位：s

startTime：起弧时间，double，单位：s

endTime：收弧时间，double，单位：s

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,”get_welding_parameters”, {”techNum”:2})
```

注意： 本命令适用于 v3.12.2 及以上版本。

2.2.2.81 设置焊接曲线参数

```
{"jsonrpc":"2.0","method":"set_welding_curve_parameters","params":{"
   outputCurrent1":outputCurrent1,"outputCurrent2":outputCurrent2,"
   outputVoltage1":outputVoltage1,"outputVoltage2":outputVoltage2,"
   correspondCurrent1":correspondCurrent1,"correspondCurrent2":
   correspondCurrent2},"id":id}
```

功能： 设置焊接曲线参数

参数： 可选参数：

outputCurrent1：控制电流1，范围：double[0,10]，单位：A
outputCurrent2：控制电流2，范围：double[0,10]，单位：A
outputVoltage1：控制电压1，范围：double[0,10]，单位：V
outputVoltage2：控制电压2，范围：double[0,10]，单位：V
correspondCurrent1：焊接电流1，范围：double[0,10]，单位：A
correspondCurrent2：焊接电流2，范围：double[0,10]，单位：A
correspondVoltage1：焊接电压1，范围：double[0,10]，单位：V
correspondVoltage2：焊接电压2，范围：double[0,10]，单位：V

返回： 成功 True, 失败 False

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):
    ret , result , id = sendCMD(sock, "set_welding_curve_parameters", {"outputCurrent1": 1, "
        outputCurrent2": 1, "outputVoltage1": 1, "outputVoltage2": 1, "correspondCurrent1": 1,
         "correspondCurrent2": 1, "correspondVoltage1": 1, "correspondVoltage2": 1})
```

注意： 本命令适用于 v3.12.2 及以上版本。

2.2.2.82 获取焊接曲线参数

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

功能： 获取焊接曲线参数

参数： 无

返回： 焊接曲线参数列表

outputCurrent1：控制电流1，单位：A

outputCurrent2：控制电流2，单位：A

outputVoltage1：控制电压1，单位：V

outputVoltage2：控制电压2，单位：V

correspondCurrent1：焊接电流1，单位：A

correspondCurrent2：焊接电流2，单位：A

correspondVoltage1：焊接电压1，单位：V

correspondVoltage2：焊接电压2，单位：V

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,"get_welding_curve_parameters")
    print ( result )
```

注意： 本命令适用于 v3.12.2 及以上版本。

2.2.2.83 设置工艺号对应的摆焊参数

```
{"jsonrpc":"2.0","method":"set_pendulum_welding_parameters","params":
   {"techNum":techNum,"note":note,"type":type,"frequency":frequency,"amplitude
   ":amplitude,"stopTime1":stopTime1,"stopTime2":stopTime2},"id":id}
```

功能： 设置工艺号对应的摆接参数

参数： techNum：工艺号，范围：int[1,16]

可选参数：

note：描述，string类型，最大长度为31

type：摆焊类型，0：sin 摆，1：三角摆

frequency：频率，范围：double[0,20]

amplitude：振幅，范围：double[0,50]

stopTime1：停止点1的停止时间，范围：double[0,10000]

stopTime2：停止点2的停止时间，范围：double[0,10000]

返回： 成功 True, 失败 False

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):
     ret , result , id = sendCMD(sock, "set_pendulum_welding_parameters", {"techNum": 2, "note":
          "test", "type": 0, "frequency": 10, "amplitude": 10, "stopTime1": 10, "stopTime2":10})
```

注意： 本命令适用于 v3.12.2 及以上版本。

2.2.2.84 获取工艺号对应的摆焊参数

```
{"jsonrpc":"2.0","method":"get_pendulum_welding_parameters","params":{
   "techNum":techNum},"id":id}
```

功能： 获取工艺号对应的摆接参数

参数： techNum：工艺号，范围：int[1,16]

返回： 摆焊参数列表
note：描述，string类型，最大长度为31
type：摆焊类型，0：sin 摆，1：三角摆
frequency：频率，范围：double[0,20]
amplitude：振幅，范围：double[0,50]
stopTime1：停止点1的停止时间，范围：double[0,10000]
stopTime2：停止点2的停止时间，范围：double[0,10000]

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, "get_pendulum_welding_parameters", {"techNum": 2})
```

注意： 本命令适用于 v3.12.2 及以上版本。

2.2.2.85 设置焊机使能

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

功能： 设置焊机使能

参数： enable：0：关闭，1：打开

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):
     ret, result ,id=sendCMD(sock,"set_welding_machine_enable", {"enable":0})
```

注意： 本命令适用于 v3.12.2 及以上版本。

2.2.2.86 获取焊机使能状态

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

功能： 获取焊机使能状态

参数： 无

返回： 0：关闭，1：打开

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_welding_machine_enable_status")
     print ( result )
```

注意： 本命令适用于 v3.12.2 及以上版本。

2.2.2.87 设置机器人上下电状态

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

功能： 设置机器人上下电

参数： status: 0：下电，1：上电

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, "set_robot_power_status", {"status": 0})
```

注意： 1. 本命令适用于 v3.12.2 及以上版本；
2. 本命令仅适用于EC系列MINI控制柜。

2.2.2.88 获取机器人上下电状态

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

功能： 获取机器人上下电状态

参数： 无

返回： 成功：status：0：下电状态，1：上电中状态，2：上电状态；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, "get_robot_power_status")
```

注意： 1. 本命令适用于 v3.12.2 及以上版本；
2. 本命令仅适用于EC系列MINI控制柜。

2.2.2.89 设置刹车电压类型

```
{"jsonrpc":"2.0","method":"set_brake_voltage_type","params":{"type":
   type},"id":id}
```

功能： 设置刹车电压类型

参数： type: 0：59V，1：51V

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, "set_brake_voltage_type", {"type": 0})
```

注意： 仅 MINI 柜可用。
本命令适用于 v3.14.2 及以上版本。

2.2.2.90 获取刹车电压类型

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

功能： 获取刹车电压类型

参数： 无

返回： 刹车电压类型，0：59V，1：51V

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, "get_brake_voltage_type")
```

注意： 仅 MINI 柜可用。
本命令适用于 v3.14.2 及以上版本。

2.2.2.91 设置力控拖动功能

```
{"jsonrpc":"2.0","method":"set_force_control_drag","params":{"status":
   status,"mode":mode,"free_axes":free_axes},"id":id}
```

功能： 设置力控拖动功能

参数： status: 启用状态，0：关闭力控拖动，1：打开力控拖动

mode: 特征，0：基座，1：TCP（可选参数）

free_axes：拖动时，X，Y，Z，Rx，Ry，Rz 各方向自由度，int 数组为 6，0：禁用，1：启用（可选参数）

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, "set_force_control_drag", {"status": 0, "mode": 0, "free_axes":
     [1, 1, 1, 1, 1, 1]})
```

注意： 仅在打开拖动时，设置 mode 和 free_axes 有效，如不输入mode和free_axes，则默认使用之前的配置，且该配置断电不保存。

本命令适用于 v3.14.2 及以上版本。

2.2.2.92 获取力控拖动功能

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

功能： 获取力控拖动功能

参数： 无

返回： 力控拖动状态列表，status：力控拖动打开状态，0：关闭，1：打开，mode：特征，0：基座，1：TCP，free_axes：拖动时，X，Y，Z，Rx，Ry，Rz 各方向自由度，int 数组大小为 6，0：禁用，1：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):
     ret , result , id = sendCMD(sock, "get_force_control_drag_status")
```

注意： 本命令适用于 v3.14.2 及以上版本。

2.2.2.93 设置非正交坐标系

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

功能： 设置非正交坐标系

参数： user_num：用户坐标号，int类型，范围[8,15]

matrix：非正交坐标系矩阵，double matrix[4][4]，其中矩阵的第4行只能为[0,0,0,1]

返回： 成功true，失败false

示例：

```
if __name__ == ”__main__”:
  # 机器人IP地址
  robot_ip=”192.168.1.200
  conSuc,sock=connectETController(robot_ip)
  matrix=[[177.805, -0.877039, 0.444788, 465.55041], [0925846, -0.352284, -0.338461,
  -394.034827], [0.377902, 0.32647, 0.829221, -165.554674], [0,0,0,1]]
        if (conSuc:)
   suc,result,id=sendCMD(sock,"set_nonorthogonal_user_frame",{"user_num:9,"matrix":matrix})
```

2.2.2.94 获取非正交坐标系矩阵

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

功能： 获取非正交坐标系矩阵

参数： user_num：用户坐标号，int类型，范围[8,15]

返回： 非正交坐标系矩阵，double matrix[4][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:)
            for i in range(8,16):
                      suc, result, id =sendCMD(sock,"get_nonorthogonal_user_frame",{"user_num": i})
                      print("用户坐标号=",i)
                      print("suc=", suc, "", "id=", id)
                      if (suc):
                              print("result=", result)
                      else:
                               print("err_msg=", result["message"])
```

2.2.2.95 计算非正交坐标系矩阵

```
{"jsonrpc":"2.0","method":"calculate_nonorthogonal_user_frame","params":{"
   rorg":rorg,"rxx":rxx,"rxy":rxy},"id":id}
```

功能： 计算非正交用户坐标系矩阵

参数： rorg：坐标原点的关节位置，类型double pos[6]

rxx：x轴上的点的关节位置，类型double pos[6]

rxy：y轴上的点的关节位置，类型double pos[6]

返回： 成功返回非正交坐标系矩阵，double matrix[4][4]，失败返回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):
        rorg = [-51.816374, -70.756973, 95.618502, -114.936728, 90.041281, 14.863426]
        rxx = [-41.445506, -84.775568, 103.743502, -109.049383, 90.026620, 25.234182]
        rxy = [-69.160899, -81.584969, 100.127166, -108.601852, 90.060957, -2.481481]
        suc, result, id = sendCMD(sock, "calculate_nonorthogonal_user_frame", {"rorg": rorg, "rxx":
        rxx, "rxy": rxy})
```

2.2.2.96 计算空间中两点的直线距离

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

功能：计算空间中两点的直线距离

参数：pose1：第一个点的位姿数据，用数组 double pose1[6]表示；

pose2：第二个点的位姿数据，用数组double pose2[6]表示。

前3位表示位置（mm），后3位表示姿态（rad）。

返回：两点间的距离（distance），单位是毫米（mm）。

示例：

```
if __name__ == "__main__":
    # 机器人IP地址
    robot_ip= "172.16.11.248"
    conSuc, sock = connectETController(robot_ip)

    if (conSuc):
        pose1 = [-3.29, -85.84, 64.45, -1.936792, -1.844988, 0.180118]

        pose2 = [19.31, -86.97, 84.76, -1.798736, -1.55631, -0.540354]

        suc, result, id = sendCMD(sock, "get_point_distance", {"pose1": pose1, "pose2": pose2}

        print (suc, result, id)
    else:

        print("连接失败")

    disconnectETController(sock)
```

2.2.2.97 计算位姿变化量

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

功能： 计算位姿变化量

参数： pose1：第一个点的位姿数据，用数组double pose1[6]表示；

pose2：第二个点的位姿数据，用数组double pose2[6]表示。

前3位表示位置（mm），后3位表示姿态（rad）。

返回： double_pose_return_data[6]：位姿变化的结果，

前3位表示位置（mm），后3位表示姿态（rad）

示例：

```
if __name__ == "__main__":
    # 机器人IP地址
    robot_ip= "172.16.11.248"
    conSuc, sock = connectETController(robot_ip)
    if (conSuc):
        pose1 = [-3.29, -85.84, 64.45, -1.936792, -1.844988, 0.180118]

        pose2 = [19.31, -86.97, 84.76, -1.798736, -1.55631, -0.540354]

        suc, result, id = sendCMD(sock, "pose_sub", {"pose1": pose1, "pose2": pose2}

        print (suc, result, id)
    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/target_pose":
   targetPos/target_pose,"speed":speed,"acc":acc,"dec":dec,
    "cond_type":cond_type,"cond_num":cond_num,"cond_value":cond_value,"
    cond_judgment":cond_judgment,"ref_pos":ref_pos},
    "id":id}
```

功能： 关节运动

参数： targetPos/target_pose：必选参数，两者选一，targetPos代表关节角度，double targetPos[6]

单位为角度，范围为[-360,360]，目标点位姿，double target_pose[6]，前三位代表位置，单位x、y、z为毫米，范围为[-1e+09~1e+09]；后三位代表姿态，单位

Rx、Ry、Rz为弧度，范围为[-π,π]

ref_pos：逆解参考点关节角度，double pos[6]，范围为[-360,360]；可选参数，不填则使用当前位置作为参考点关节角度

speed：运行速度，范围：double[0.01,100]

cond_type：可选参数，0代表数字量输入 X，1代表数字量输出Y，2代表自定义输入，范围int[0,2]

cond_num： I/O 地址，可选参数，范围：int[0,63]

cond_value： I/O 状态，可选参数，范围：int[0,1], 实际 I/O 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

cond_judgment：条件判断，可选参数，字符串类型，为自定义判断语句，当满足条件时，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，可选参数，范围：int[1,100], 不写默认值为 80

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

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 = []
    point.append([0.0065,-103.9938,102.2076,-88.2138,
```

```
 90.0000,0.0013])
point.append([-16.2806,-82.4996,81.9848,-89.4851,
 90.0000,-16.2858])
point.append([3.7679, -71.7544, 68.7276, -86.9732,
90.0000, 3.7627])
point.append([12.8237,-87.3028,87.2361,-89.9333,
 90.0000,12.8185])
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
```

备注： 当cond_type=0和1时，cond_num和cond_value参数有效，当cond_type=2时，cond_judgment参数有效。请参考JBI指令中UNTIL条件指令的语法格式输入自定义条件。

2.2.3.2 直线运动

```
{"jsonrpc":"2.0","method":"moveByLine","params":{"targetPos/target_pose":targetPos/
  target_pose,"speed_type":speed_type,"speed":speed,"acc":acc,"dec":dec,"cond_type"
  :cond_type,"cond_num":cond_num,"cond_value":cond_value},"cond_judgment":cond_judg
  ment,"ref_pos":ref_pos},"id":id}
```

功能： 直线运动

参数： targetPos/target_pose：必选参数，两者选一，targetPos代表关节角度，double targetPos[6]，范围为[-360,360]，目标点位姿，double target_pose[6]，前三位代表位置，单位x、y、z为毫米，范围为[-1e+09~1e+09]，后三位代表姿态，单位Rx 、Ry、Rz为弧度，范围为[-π,π]

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

ref_pos：逆解参考点关节角度，double pos[6]，范围为[-360,360]；可选参数，不填则使用当前位置作为参考点关节角度

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：可选参数，0代表数字量输入X，1代表数字量输出Y，2代表自定义输入，范围：int[0,2]

cond_num：I/O地址，可选参数，范围：int[0,63]

cond_value：I/O状态，可选参数，范围：int[0,1], 实际 I/O 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

cond_judgment：条件判断，可选参数，字符串类型，为自定义条件判断语句，当满足条件时，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80

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

返回：成功 true，失败 false

示例：

```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="192.168.1.205"
  conSuc, sock = connectETController(robot_ip)
  point = []
  point.append([0.0065,-103.9938,102.2076,-88.2138,
  90.0000,0.0013])
  point.append([-16.2806,-82.4996,81.9848,-89.4851,
  90.0000,-16.2858])
  point.append([3.7679, -71.7544, 68.7276, -86.9732,
  90.0000, 3.7627])
  point.append([12.8237,-87.3028,87.2361,-89.9333,
  90.0000,12.8185])
  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
```

备注：无speed_type参数时，speed表示为绝对直线速度。当cond_type=0和1时，cond_num和cond_value参数有效，当cond_type=2时，cond_judgment参数有效。请参考JBI指令中UNTIL条件指令的语法格式输入自定义条件。

2.2.3.3 圆弧运动

```
{"jsonrpc":"2.0","method":"moveByArc","params":{"midPos/mid_pose":midPos/mid_pose,"
    targetPos":targetPos,"speed_type":speed_type,"speed":speed,"acc":acc,"dec":dec,"
    cond_type":cond_type,"cond_num":cond_num,"cond_value":
    cond_value,"cond_judgment":cond_judgment,"ref_pos":ref_pos},"id":id}
```

功能：圆弧运动

参数： midPos/mid_pose：必选参数，两者选一，midPos代表中间点关节角度，单位为角度，double midPos[6]，范围为[-360,360]；mid_Pose代表中间点位姿，double mid_Pose[6]，前三位代表位置，单位x 、y 、z为毫米，范围为[-1e+09~1e+09]，后三位代表姿态，单位Rx 、Ry、Rz为弧度，范围为[-π,π]

ref_pos：逆解参考点关节角度，double pos[6]，范围为[-360,360]；可选参数，不填则使用当前位置作为参考点关节角度

targetPos/targ et_pose：可选参数，两者选一，targetPos代表关节角度，d ouble targ etPos [6]单位为角度，范围为[-360,360]；targ et_pose位姿，d ouble pose[6]，前三位代表位置，单位x 、y 、z为毫米，范围：[-1e+09~1e+09]，后三位代表姿态，单位x 、y、z、为毫米，范围为[-π,π]

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：可选参数，0代表数字量输入X，1代表数字量输出Y，2代表自定义输入，范围int[0,2]

cond_num：I/O 地址，可选参数，范围：int[0,63]

cond_value：I/O 状态，可选参数，范围：int[0,1], 实际 I/O 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

cond_judgment：条件判断，可选参数，字符串类型，为自定义条件判断语句，当满足条件时，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 80

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

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)
  P000 = [0.0065,−103.9938,102.2076,−88.2138,
   90.0000,0.0013]
```

```
P001 = [-16.2806,-82.4996,81.9848,-89.4851,
 90.0000,-16.2858]
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})
```

备注： 无speed_type参数时，speed表示为绝对直线速度。当cond_type=0和1时，cond_num和cond_value参数有效，当cond_type=2时，cond_judgment参数有效。请参考JBI指令中UNTIL条件指令的语法格式输入自定义条件。

2.2.3.4 旋转运动

```
{"jsonrpc":"2.0","method":"moveByRotate","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： 旋转运动

参数： targetpos：目标关节点 double pos[6]，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：可选参数，0代表数字量输入X，1代表数字量输出Y，2代表自定义输入，范围：int[0,2]
cond_num: I/O 地址，可选参数，范围：int[0,63]
cond_value: I/O 状态，可选参数，范围：int[0,1], 实际 I/O 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80
dec：减速度百分比，范围：int[1,100], 可选参数, 不写默认值为acc的值

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 )
  P000 = [0.0065,-103.9938,102.2076,-88.2138,
```

```
90.0000,0.0013]
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。
本命令不再维护，逐渐废弃。

提醒

以上命令只支持在 remote 模式下使用。
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/wayPose":wayPoint/
    wayPose,"moveType":moveType,"speed_type":speed_type,"speed":speed,"acc":acc,"
    dec":dec,"smooth":smooth,"cond_type":cond_type,"cond_num":cond_num,"
    cond_value":cond_value,"cond_judgment":cond_judgment,"trigger_cond_type":
    trigger_cond_type,"trigger_cond_value":trigger_cond_value,"trigger_io_type":
    trigger_io_type,"trigger_io_addr":trigger_io_addr,"trigger_io_status":
    trigger_io_status},"ref_pos":ref_pos,"FPT":FPT},"id":id}
    或"jsonrpc":"2.0","method":addPathPoint,"params":{"wayPoint/wayPose":wayPoint
    wayPose,"moveType":moveType,"speed_type":speed_type,
    speed":speed,"acc":acc,"dec":dec,"circular_radius":circular_radius,"
    "cond_type":cond_type,"cond_num":cond_num,"cond_value":cond_value,"
    cond_judgment":cond_judgment,"trigger_cond_type":trigger_cond_type,"
    trigger_cond_value":trigger_cond_value,"trigger_io_type":trigger_io_type,
    "trigger_io_addr":trigger_io_addr,"trigger_io_status":trigger_io_status
    "ref_pos":ref_pos,"FPT":FPT},"id":id}
```

功能： 添加路点信息 2.0

参数： wayPoint/wayPose：必选参数，两者选一，wayPoint代表目标点关节角度，double wayPoint[6]单位为角度，范围为[-360,360]；wayPose代表目标点末端位姿，d ouble way_Pose[6]的前三位代表位置，单位x 、y 、z为毫米，范围：[-1e+09~1e+09]，后三位代表姿态，单位Rx 、Ry、Rz为弧度，范围为[-π,π]

ref_Pose：逆解参考点关节角度，double pos[6]，范围为[-360,360]；可选参数，不填该

参数，若已有已添加的路点则参考点使用上个添加的路点，否则使用当前位置作为参考点关节角度

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]，可选参数，最后一个点位的平滑度必须为 0。不再维护，逐渐废弃。

circular_radius：交融半径，范围：doube[0, 2608]，单位毫米，可选参数，不写默认为 0。最后一个点位的交融半径必须为 0。

cond_type：可选参数，0 代表数字量输入 X，1代表为数字量输出 Y，2代表自定义输入，范围：int[0,2]

cond_num：I/O 地址，可选参数，范围：int[0,63]

cond_value：I/O 状态，可选参数，范围：int[0,1], 实际 I/O 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

cond_judgment：条件判断，可选参数，字符串类型，为自定义条件判断语句，当满足条件时，立即放弃本次未完成的运动，执行下一条指令

trigger_cond_type：trigger条件类型，0：不使用trigger；1：STARTTIME；2：ENDTIME；3：STARTLEN；4：ENDLEN，执行mov时范围为[0,2]

trigger_cond_value：对应trigger条件类型1-4，分别表示运动开始后的时间，单位为秒；离运动停止所剩余的时间，单位为秒；到运动起点的距离，单位为毫米；到运动终点的距离，单位为毫米

trigger_io_type：I/O类型，0：数字量输出；1：虚拟量输出

trigger_io_addr：I/O地址，int类型，数字量输出范围为[0,19]U[48,49]，虚拟量输出范围为[528,911]

trigger_io_status：状态，0：关；1：开

acc：加速度百分比，范围：int[1,100]，可选参数，不写默认值为 80

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

FPT：确定一段圆弧的终点；0：关；1：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)
C000 = [0.0065,-103.9938,102.2076,-88.2138,
 90.0000,0.0013]
C001 = [-16.2806,-82.4996,81.9848,-89.4851,
 90.0000,-16.2858]
# 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,"
      circular_radius":50})
  suc, result ,id=sendCMD(sock,"addPathPoint",{"wayPoint":C001,"moveType":1,"speed_type":0,"
      speed":50,"circular_radius":0})
```

注意：当cond_type=0和1时，cond_num和cond_value参数有效，当cond_type=2时，cond_judgment参数有效。请参考JBI指令中UNTIL条件指令的语法格式输入自定义条件。不写trigger参数则默认不使用trigger，使用trigger时默认不使用UNTIL。

提醒

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。
参数 circular_radius 与参数 smooth 二选一使用，推荐使用参数 circular_radius。

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]
  C001 = [-16.2806,-82.4996,81.9848,-89.4851,
   90.0000,-16.2858]
  C002 = [3.7679, -71.7544, 68.7276, -86.9732,
  90.0000, 3.7627]
  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, "circular_radius":20})
      suc, result , id = sendCMD(sock, "addPathPoint", {"wayPoint": C001,"moveType":0, "speed":
            50, "circular_radius":20})
      suc, result , id = sendCMD(sock, "addPathPoint", {"wayPoint": C002,"moveType": 0, "speed":
            50, "circular_radius":0})
      # 轨迹运动2.0
      suc, result , id = sendCMD(sock, "moveByPath")
      while(True):
        # 获取轨迹运动当前运行点位序号
        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，6个关节的速度值，六位数单位：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]
  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 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], "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 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,"arot":arot},"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，范围：大于 0
arot：可选参数，int，姿态加速度，范围：大于 0，单位：度/s^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):
    suc, result ,id=sendCMD(sock,"moveBySpeedl",{"v":[20,0,0,0,0,0],"acc":50,"arot":10,"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：可选参数，0 代表数字量输入 X，1 代表数字量输出 Y，2代表自定义输入，范围 int[0,2]

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

cond_judgment：条件判断，可选参数，字符串类型，为自定义条件判断语句，当满足条件时，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 80

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

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。当cond_type=0和1时，cond_num和cond_value参数有效，当cond_type=2时，cond_judgment参数有效。请参考JBI指令中UNTIL条件指令的语法格式输入自定义条件。

2.2.3.21 编码器零位校准

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

功能： 编码器零位校准

参数： 无

返回： 成功 true，失败 false

示例：

```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip = "192.168.0.202"
  conSuc, sock = connectETController(robot_ip)
  if (conSuc):
    suc, result, id = sendCMD(sock, "getServoStatus")
    print(suc, result, id)
    time.sleep(0.5)
    if result == 0:
      # 设置机械臂伺服状态
      ret, result, id=sendCMD(sock, "set_servo_status", {"status": 1})
      print( result )
    time.sleep(1)
    # 编码器零位校准
    suc, result, id=sendCMD(sock, "calibrate_encoder_zero_position")
    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。

2.2.3.22 获取暂停状态下打开拖动时记录的点位及jbi信息

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

功能： 获取暂停状态下打开拖动时记录的点位及jbi信息

参数： 无

返回： 暂停信息对象，包含以下关键字

record_state：记录状态：0：未记录，1：记录（记录状态为1时，其余数据才有效）

pause_pos：暂停时的关节角度，大小为6的double数组

pause_jbi_name：暂停时的jbi名称，string

pause_line：暂停时的行号

示例：

```
if __name__ == '__main__':
    # 机器人ip地址
    robot_ip = "192.168.1.200"
    conSuc, sock = connectETController(robot_ip)
    if (conSuc):
      suc, result, id = sendCMD(sock, "get_pause_info_before_dragging")
```

备注：本命令适用于v3.16.2及以上版本。

2.2.3.23 运动到拖动前记录的暂停点

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

功能： 运动到拖动前记录的暂停点

参数： speed: 运行速度百分比，double 类型，范围[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):
        suc, result, id = sendCMD(sock, "move_to_pause_pos_befor_dragging", {"spend": 100})
```

备注：本命令适用于v3.16.2及以上版本。

2.2.3.24 继续运行拖动前暂停的jbi

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

功能： 继续运行拖动前暂停的jbi

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):
    suc, result , id = sendCMD(sock, "continue_to_run_jbi_paused_before_dragging")
```

备注：本命令适用于v3.16.2及以上版本。

2.2.3.25 清除记录的暂停点位信息和jbi信息

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

功能： 清除记录的暂停点位信息和jbi信息

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):
    suc, result , id = sendCMD(sock,"clear_pause_info_before_drigging")
```

备注：本命令适用于v3.16.2及以上版本。

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[6]，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 double pos[6]

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]
  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})
```

注意： 如需自行选解，即获取目标位姿对应的全部逆解结果，请参见第2.2.2.73章。

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[6]，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[0,15]
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： 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[0,15]
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", {"coordinate_num":1, "tool_num":7})
    # 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})
```

2.2.4.7 位姿转化为齐次矩阵

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

功能：位姿转化为齐次矩阵

参数： pose：位姿，double pose[6]，前3位表示位置（mm），后3位表示姿态（rad）

返回： 如执行失败，无返回值；如执行成功，返回相应的4*4齐次矩阵 double matrix[4][4]

示例：

```
if __name__ == "__main__":
    # 机器人IP地址
    robot_ip = "172.16.11.24"
    conSuc, sock = connectETController(robot_ip)
    if (conSuc):
        pose = [19.31, -86.97, 84.76, -1.798736, -1.55631, -0.540354]
        suc， result， id = sendCMD(sock, "pose_to_matrix", {"pose": pose})
        print(suc, result, id)
    else:
        print("连接失败")
    disconnectETController(sock)
```

2.2.4.8 齐次矩阵转化为位姿

```
{"jsonrpc":"2.0","method":"matrix_to_pose","params":{"matrix":matrix},
"id":id}
```

功能：齐次矩阵转化为位姿

参数： matrix：要转换位姿的矩阵，double matrix[4][4]

返回： 如执行失败，无返回值；

如执行成功，返回位姿大小为6的数组 double pose[6]，
前3位表示位置（mm），后3位表示姿态（rad）

示例：

```
if __name__ == "__main__":
    # 机器人IP地址
    robot_ip = "172.16.11.248"
    conSuc, sock = connectETController(robot_ip)
    if (conSuc):
        matrix = [[0.012421976426268886, 0.7190098094654518, 0.6948889036343292, 19.31],
                  [-0.00745207934501453, -0.6948566542030964, 0.719109655492898, -86.97],
                  [0.999895075002923, -0.014111130434288294, -0.0032733748281666055, 84.76],
                  [0, 0, 0, 1]]
        suc, result, id = sendCMD(sock, "matrix_to_pose", {"matrix": matrix})
        print(suc, result, id)
    else:
        print("连接成功")
    disconnectETController(sock)
```

2.2.4.9 位置和旋转矢量转化为齐次矩阵

```
{"jsonrpc":"2.0","method":"pos_rot_vector_to_matrix","params":{"
    pose_rot_vector":pose_rot_vector},"id":id}
```

功能：将位置和旋转矢量转化为齐次矩阵

参数：pose_rot_vector：位置和旋转矢量，用数组double pose[6]表示，
前3位表示位置，后3位表示旋转矢量

返回：如执行失败，则无返回值；

如执行成功，则返回二维数组齐次矩阵 double matrix[4][4]

示例：

```
if __name__ == "__main__":
    # 机器人IP地址
    robot_ip = "172.16.11.248"
    conSuc, sock = connectETController(robot_ip)
    if (conSuc):
        pos = [19.31, -86.97, 84.76, -1.798736, -1.55631, -0.540354]
        suc, result, id = sendCMD(sock, "pose_rot_vector_to_matrix", {"pose_rot_vector": pos})
        print(suc, result, id)
    else:
        print("连接失败")
    disconnectETController(sock)
```

2.2.4.10 齐次矩阵转化为位置和旋转矢量

```
{"jsonrpc":"2.0","method":"matrix_to_pos_rot_vector","params":{"
    matrix":matrix},"id":id}
```

功能： 将齐次矩阵转化为位置和旋转矢量

参数： matrix：要转化为位置和旋转矢量的齐次矩阵，用double matrix[4][4]表示

返回： 如执行失败，则无返回值；

如执行成功，则返回位置和旋转矢量的数组 double PoseAngleAxis[6]

前3位表示位置，后3位表示旋转矢量

示例：

```
if __name__ == "__main__":
    # 机器人IP地址
    robot_ip = "172.16.11.248"
    conSuc, sock = connectETController( robot_ip )
    if (conSuc):
        matrix = [[5.4892175804788224e-08, -0.9999412440210669, -0.01084013402124294,543.146],
                  [-0.9999999999487177, 5.4892175804788224e-08, -1.0127284944469461e-05, 116.884],
                  [1.0127284944469461e-05, 0.01084013402124294, -0.9999412439697842, 382.408],
                  [0, 0, 0, 1]]
        suc, result, id = sendCMD(sock, "matrix_to_pose_rot_vector", {"matrix": matrix})
        print(suc, result, id)
    else:
        print("连接失败")
    disconnectETController(sock)
```

2.2.4.11 位置和四元数转化为齐次矩阵

```
{"jsonrpc":"2.0","method":"quaternion_to_matrix","params":{"
    quter_pose":quter_pose},"id":id}
```

功能： 将位置和四元数转化为齐次矩阵

参数： quter_pose：位置和四元数，用double pose[7]表示

前3位表示位置，后4位表示四元数

返回： 如执行失败，则无返回值；

如执行成功，则返回二维数组齐次矩阵 double matrix[4][4]

示例：

```
if __name__ == "__main__":
    # 机器人IP地址
    robot_ip = "172.16.11.248"
    conSuc, sock = connectETController(robot_ip)
    if (conSuc):
        pos = [0, 0, 0, 1, 0, 0, 0]
        suc, result, id = sendCMD(sock, "quaternion_to_matrix", {"quter_pose": pos}))
        print(suc, result, id)
    else:
        print("连接失败")
    disconnectETController(sock)
```

2.2.4.12 齐次矩阵转化为位置和四元数

```
{"jsonrpc": "2.0","method": "matrix_to_quaternion","params":{"
    matrix": matrix},"id": id}
```

功能： 将齐次矩阵转化为位置和四元数

参数： matrix：要转化为位置和四元数的齐次矩阵，用double matrix[4][4]表示

返回： 如执行失败，则无返回值；

如执行成功，则返回位置和四元数数组 double pos[7]

前3位表示位置（mm），后4位表示四元数

示例：

```
if __name__ == "__main__":
    # 机器人IP地址
    robot_ip = "172.16.11.248"
    conSuc, sock = connectETController(robot_ip)
    if (conSuc):
        matrix = [[1.0, 0.0, 0.0, 0.0],[0.0, 1.0, 0.0, 0.0],[0.0, 0.0, 1.0, 0.0],[0.0, 0.0, 0.0, 1.0]]
        suc, result, id = sendCMD(sock, "matrix_to_quaternion", {"matrix": matrix}))
        print(suc, result, id)
    else:
        print("连接失败")
    disconnectETController(sock)
```

2.2.4.13 矩阵相乘

```
{"jsonrpc":"2.0","method":"matrix_mul","params":{"matrix1": matrix1,
"matrix2": matrix2},"id":id}
```

功能： 矩阵相乘

参数： matrix1：相乘矩阵1，必须是一个数组 double matrix[row1][col1]表示；

matrix2：相乘矩阵2，必须是一个数组 double matrix[row2][col2]表示。

矩阵的尺寸限制为：row ∈ [2, 6], col ∈ [2, 6]。两个矩阵相乘的条件是：第一个矩阵的列数须等于第二个矩阵的行数（col1=row2）。

返回： 如执行失败，则无返回值；

如执行成功，则返回二维数组齐次矩阵 double matrix[row1][col2]

示例：

```
if __name__ == "__main__":
    # 机器人IP地址
    robot_ip = "172.16.11.248"
    conSuc, sock = connectETController(robot_ip)
    if (conSuc):
        matrix1 = [[0.57154154, 0.72164121, 0.89500456, 0.88861432],
                   [0.39348646, 0.82168934, 0.68840172, 0.78207635],
                   [0.92924593, 0.52347333, 0.08046729, 0.31668283],
                   [0.68091908, 0.69661777, 0.34409425, 0.21902325]]
        matrix2 = [[0.70788248, 0.8651924, 0.44899899, 0.49616621],
                   [0.22551149, 0.88302156, 0.50982376, 0.92935707],
                   [0.58448953, 0.51924245, 0.69608474, 0.54666356],
                   [0.37036362, 0.84448838, 0.91020503, 0.97064273]]
        suc, result, id = sendCMD(sock, "matrix_mul", {"matrix1": matrix1, "matrix2": matrix2})
        print(suc, result, id)
    else:
        print("连接失败")
    disconnectETController(sock)
```

2.2.4.14 方阵求逆

```
{"jsonrpc": "2.0","method": "matriX_inVerse","params":{"matriX": matriX
},"id": id}
```

功能： 方阵求逆

参数： matrix：求逆的方阵，double matrix[m][m]表示m阶方阵，m的范围为[2, 6]

返回： 如执行失败，则无返回值；

如执行成功，则返回求逆后的方阵 double matrix[m][m]

示例：

```
if __name__ == "__main__":
    # 机器人IP地址
    robot_ip = "172.16.11.248"
    conSuc, sock = connectETController( robot_ip )
    if (conSuc):
        matrix = [[0.57154154, 0.72164121, 0.89500456, 0.88861432],
                  [0.39348646, 0.82168934, 0.68840172, 0.78207635],
                  [0.92924593, 0.52347333, 0.08046729, 0.31668283],
                  [0.68091908, 0.69661777, 0.34409425, 0.21902325]]
        suc , result,  id  = sendCMD(sock, "matrix_inVerse", {"matrix": matrix}))
        print(suc, result, id)
    else:
        print("连接失败")
    disconnectETController(sock)
```

2.2.5 I/O 服务 (I/OService)

2.2.5.1 获取输入 I/O 状态

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

功能： 获取输入 I/O 状态

参数： addr：输入 I/O 地址，范围：int[0,19]∪[46,51]，若为MINI控制柜，范围为[0,11]∪[46,51]返回： 输入 I/O 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, 19 ,1):
            # 获取输入I/O状态
            suc, result , id = sendCMD(sock, "getInput", {"addr":i})
            print ( result )
```

2.2.5.2 获取输出 I/O 状态

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

功能： 获取输出 I/O 状态

参数： addr：输出 I/O 地址，范围：int[0,19]∪[48,51]，若为MINI控制柜，范围为[0,7]∪[48,51]

返回： 输出 I/O 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I/O状态
            suc, result , id = sendCMD(sock, "getOutput", {"addr":n})
            print ( result )
```

2.2.5.3 设置输出 I/O 状态

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

功能： 设置输出 I/O 状态

参数： addr：输出 I/O 地址，范围：int[0,19]∪[48,49]，若为MINI控制柜，范围为[0,7]∪[48,51]

status：I/O 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I/O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 I/O 状态

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

功能： 获取虚拟输入 I/O 状态

参数： addr：虚拟 I/O 地址，范围：int[0,399]

返回： 输入 I/O 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I/O状态
      suc, result ,id=sendCMD(sock,"getVirtualInput",{"addr":i})
      print ( result )
```

2.2.5.5 获取虚拟输出 I/O 状态

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

功能： 获取虚拟输出 I/O 状态

参数： addr：虚拟 I/O 地址，范围：int [400,1535]

返回： 输出 I/O 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I/O状态
    suc, result , id=sendCMD(sock,"getVirtualOutput",{"addr":n})
    print ( result )
```

2.2.5.6 设置虚拟输出 I/O 状态

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

功能： 设置虚拟输出 I/O 状态

参数： addr：输出 I/O 地址，范围：int[528,799]

status：输出 I/O 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I/O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 I/O

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

功能： 读取多个 M 虚拟 I/O

参数： addr: 虚拟 I/O 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 I/O 范围 int [1,96]
addr+16*len 的范围为 int[0,1535]

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

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：

```
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 获取模拟量输出

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

功能：获取模拟量输出

参数：addr：模拟量输出地址，范围：int[0,4]

返回：模拟量值，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):
            # 获取模拟量输出值
            suc, result, id = sendCMD(sock,"get_analog_output",{"addr":i})
            print(result)
```

2.2.5.10 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。

2.2.5.11 设置末端扫描按钮模式

```
{"jsonrpc":"2.0","method":"set_scan_button_mode","params":{"mode":mode},
   "id":id}
```

功能：设置末端扫描按钮模式

参数：mode：扫描按钮模式，0：常规模式，1：扫描按钮模式，2：末端摇杆按钮输入模式

返回：ret：-1：失败，0：成功

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(0,1):
      # 设置末端扫描按钮模式
      suc, result, id = sendCMD(sock,"set_scan_button_mode",{"mode":i})
      print(result)
```

2.2.5.12 获取末端扫描按钮模式

```
{"jsonrpc":"2.0","method":"get_scan_button_mode","params":{"":},
   "id":id}
```

功能：获取末端扫描按钮模式

参数：无

返回：mode：扫描按钮模式，0：常规模式，1：扫描按钮模式，2：末端摇杆按钮输入模式

示例：
```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="192.168.1.202"
  conSuc,sock=connectETController(robot_ip)
  if (conSuc):
    # 获取末端扫描按钮模式
    suc, result, id = sendCMD(sock,"get_scan_button_mode",{"":})
    print(result)
```

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[6]，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]
  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]。
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[6]，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]
  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。本命令不再维护，逐渐废弃。

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[6]，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"
```

```
    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(',')))
            print(i,line_list)
            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
        # 关闭文件
        fo.close()
        # 清空透传缓存
        suc, result, id = sendCMD(sock, "tt_clear_servo_joint_buf")
        print("clear_ret = ", suc)
  else:
        print("连接失败")
    disconnectETController(sock)
```

2.2.7.7 Example 2

```
import socket
import json
import time

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()
```

```
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 
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})
```

```
            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(result)
        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})
                print(result)
                wait_stop() # 等待机器人停止
                print(1)
                # 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})
                print(result)
            # 添加透传伺服目标关节点信息到缓存中
            send_Point(sock, "tt_put_servo_joint_to_buf",{"targetPos":
                line_list})
            print(result)
            time.sleep(0.01)
            i = i + 1
        # 关闭文件
```

```
        fo.close()
        # 清空透传缓存
        suc, result, id = sendCMD(sock, "tt_clear_servo_joint_buf")
        print("clear_ret = ", suc)
    else:
        print("连接失败")
    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。

2.2.8.3 设置时区

```
{"jsonrpc":"2.0","method":"set_timezone","params":{"timezone":
  timezone},"id":id}
```

功能： 设置时区

参数： timezone: 时区，类型int，范围[-12,12]，如该参数为8时，表示UTC+8，即东八区

返回：成功true，失败false

注意： 该设置重启生效

示例：

```
if __name__ == "__main__":
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController( robot_ip )
  if (conSuc):
    # 设置时区
    suc ,result,id = sendCMD(sock, "set_timezone", {"timezone": 8})
    print ( result )
```

2.2.8.4 同步时钟

```
{"jsonrpc":"2.0","method":"set_ntpServer","params":{"time_interval":
  time_interval, "ip_addr":ip_addr},"id":id}
```

功能：启用ntpClient，同步时钟

参数：time_interval: 同步时间间隔，类型int，范围[15,2147483647]，单位：秒，可选参数，不填默认为15，即15秒同步一次

ip_addr：ntp服务端IP地址，类型string

返回：成功true，失败false

示例：

```
if __name__ == "__main__":
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController( robot_ip )
```

```
if (conSuc):
  ntp_ip = "172.16.101.238"
  # 同步时钟
  ret,result,id  = sendCMD(sock, "set_ntpServer", {"time_interval": 15, "ip_addr": ntp_ip})
  print ( result )
```

注意：目前通过此方式同步的时钟不进行断电保存，即断电重启后时钟会恢复为之前手手动设置的时钟

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,"ret_flag":ret_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[6]，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]
    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,"ret_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 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]
    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]
    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]
    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]
    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]
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):
```

```
            suc, result, id = sendCMD(sock, "push_pos", {"timestamp":
                res, "pos": start_pos})
            start_pos[0] += 0.02
            res = res + 0.002
        time.sleep(1)
        suc, result, id = sendCMD(sock, "stop_push_pos")
        print(result)
        suc, result, id = sendCMD(sock, "check_trajectory")
        print(result)
        suc, result, id = sendCMD(sock, "start_trajectory", {"
            speed_percent": 50})
        print(result)
        time.sleep(5)
        suc, result, id = sendCMD(sock, "flush_trajectory")
        print(result)
    else:
        print("连接失败")
    disconnectETController(sock)
```

2.2.9.11 Example 2

```
import socket
import json
import time

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:
```

```
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]
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)
```

2.2.10 Profinet 服务 (ProfinetService)

2.2.10.1 获取 profinet int 型输入寄存器的值

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

功能： 获取 profinet int 型输入寄存器的值

参数： addr: 寄存器起始地址，范围 int[0,31]
length: 寄存器个数，范围 int[1,32]
注：addr 与 length 的和应小于等于 32

返回： 寄存器值列表 int[length]

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profinet int型输入寄存器的值
    suc, result , id = sendCMD(sock, "get_profinet_int_input_registers",{"addr":0,"length":1})
    print ( result )
  else :
    print ("连接失败")
  disconnectETController (sock)
```

2.2.10.2 获取 profinet int 型输出寄存器的值

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

功能： 获取 profinet int 型输出寄存器的值

参数： addr: 寄存器起始地址，范围 int[0,31]
length: 寄存器个数，范围 int[1,32]
注：addr 与 length 的和应小于等于 32

返回： 寄存器值列表 int[length]

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profinet int型输出寄存器的值
    suc, result , id = sendCMD(sock, "get_profinet_int_output_registers",{"addr":1,"length":2})
    print ( result )
  else :
    print ("连接失败")
  disconnectETController (sock)
```

2.2.10.3 获取 profinet float 型输入寄存器的值

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

功能： 获取 profinet float 型输入寄存器的值

参数： addr: 寄存器起始地址，范围 int[0,31]
length: 寄存器个数，范围 int[1,32]
注：addr 与 length 的和应小于等于 32

返回： 寄存器值列表 int[length]

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profinet float 型输入寄存器的值
    suc, result , id = sendCMD(sock, "get_profinet_float_input_registers",{"addr":0,"length":1})
    print ( result )
  else :
    print ("连接失败")
  disconnectETController (sock)
```

2.2.10.4 获取 profinet float 型输出寄存器的值

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

功能： 获取 profinet float 型输出寄存器的值

参数： addr: 寄存器起始地址，范围 int[0,31]
length: 寄存器个数，范围 int[1,32]
注：addr 与 length 的和应小于等于 32

返回： 寄存器值列表 int[length]

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profinet float 型输出寄存器的值
    suc, result , id = sendCMD(sock, "get_profinet_float_output_registers",{"addr":0,"length"
        :1})
    print ( result )
  else :
    print ("连接失败")
  disconnectETController (sock)
```

2.2.10.5 设置 profinet int 型输出寄存器的值

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

功能： 设置 profinet int 型输出寄存器的值

参数： addr: 寄存器起始地址，范围 int[0,31]
length: 寄存器个数，范围 int[1,32]
注：addr 与 length 的和应小于等于 32
value: 寄存器值列表, 类型 int[length], 元素范围 [-2147483648,2147483647]

返回： 成功 True，失败 False

示例：
```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip = "192.168.0.202"
  conSuc, sock = connectETController(robot_ip)
  if (conSuc):
    # 设置profinet int型输出寄存器
    suc, result , id = sendCMD(sock, "set_profinet_int_output_registers", {"addr": 1, "length":
        2, "value": [1,1]})
    print ( result )
  else :
    print ("连接失败")
```

注意： 本命令只支持在 remote 模式下使用。

2.2.10.6 设置 profinet float 型输出寄存器的值

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

功能： 设置 profinet float 型输出寄存器的值

参数： addr: 寄存器起始地址，范围 int[0,31]
length: 寄存器个数，范围 int[1,32]
注：addr 与 length 的和应小于等于 32
value: 寄存器值列表, 类型 double[length], 元素范围 [-3.40E+38,3.40E+38]

返回： 成功 True，失败 False

示例：
```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip = "192.168.0.202"
  conSuc, sock = connectETController(robot_ip)
  if (conSuc):
    # 设置 profinet float 型输出寄存器
    suc, result , id = sendCMD(sock, "set_profinet_float_output_registers", {"addr": 0, "length"
        : 2, "value": [1,1]})
    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。

2.2.11 反向驱动服务 (BackdriveService)

2.2.11.1 获取伺服抱闸打开情况

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

功能： 获取伺服抱闸打开情况

参数： 无

返回： 伺服抱闸打开情况数组

示例：
```
if __name__ == "__main__":
    # 机器人IP地址
    robot_ip = "172.16.11.240"
    conSuc, sock = connectETController(robot_ip)
    if (conSuc):
        suc, result , id = sendCMD(sock, "get_servo_brake_off_status")
        print(suc, result , id)
    else:
        print("连接失败")
    disconnectETController(sock)
```

2.2.11.2 获取是否处于反向驱动模式

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

功能： 获取是否处于反向驱动模式

参数： 无

返回： 1：反向驱动模式，0：不处于反向驱动模式

示例：
```
if __name__ == "__main__":
    # 机器人IP地址
    robot_ip = "172.16.11.240"
    conSuc, sock = connectETController(robot_ip)
    if (conSuc):
        suc, result , id = sendCMD(sock, "get_backdrive_status")
        print(suc, result , id)
    else:
        print("连接失败")
    disconnectETController(sock)
```

2.2.11.3 进入反向驱动模式

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

功能： 进入反向驱动模式

参数： 无

返回： 成功 True，失败 False

示例：
```
if __name__ == "__main__":
    # 机器人IP地址
    robot_ip = "172.16.11.240"
    conSuc, sock = connectETController(robot_ip)
    if (conSuc):
        suc, result , id = sendCMD(sock, "enter_backdrive")
        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，并且机器人必须处于重置状态。

2.2.11.4 退出反向驱动模式

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

功能： 退出反向驱动模式

参数： 无

返回： 成功 True，失败 False

示例：
```
if __name__ == "__main__":
    # 机器人IP地址
    robot_ip = "172.16.11.240"
    conSuc, sock = connectETController(robot_ip)
    if (conSuc):
        suc, result , id = sendCMD(sock, "exit_backdrive")
        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，并且机器人必须处于反向驱动模式。

2.2.11.5 重置控制器状态

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

功能： 重置控制器状态

参数： 无

返回： 成功 True，失败 False

示例：
```
if __name__ == "__main__":
    # 机器人IP地址
    robot_ip = "172.16.11.240"
    conSuc, sock = connectETController(robot_ip)
    if (conSuc):
        suc, result , id = sendCMD(sock, "reset_robot_status")
        print (suc, result , id)
    else :
        print ("连接失败")
    disconnectETController (sock)
```

注意： 机器人必须处于停止或错误（不包括急停报警）状态。
本命令只支持在remote模式下使用。

2.2.12 Ethernet/IP

2.2.12.1 获取Ethernet/IP int型输入寄存器的值

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

功能： 获取Ethernet/IP int型输入寄存器的值

参数： addr：寄存器起始地址，范围int[0,31]

length：寄存器个数，范围int[1,32]

返回： 寄存器值列表int[length]

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Ethernet/IP int型输入寄存器的值
      suc, result ,id=sendCMD(sock,"get_eip_int_input_registers",{"addr":0,"length":1})
    print ( result )
  else :
    print ("连接失败")
  disconnectETController (sock)
```

备注： addr与length的和应小于等于32，本命令适用于v3.5.2及以上版本。

2.2.12.2 获取Ethernet/IP int型输出寄存器的值

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

功能： 获取Ethernet/IP int型输出寄存器的值

参数： addr：寄存器起始地址，范围int[0,31]

length：寄存器个数，范围int[1,32]

返回： 寄存器值列表int[length]

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Ethernet/IP int型输出寄存器的值
      suc, result ,id=sendCMD(sock,"get_eip_int_output_registers",{"addr":1,"length":2})
    print ( result )
  else :
    print ("连接失败")
  disconnectETController (sock)
```

备注： addr与length的和应小于等于32，本命令适用于v3.5.2及以上版本。

2.2.12.3 获取Ethernet/IP float型输入寄存器的值

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

功能： 获取Ethernet/IP float型输入寄存器的值

参数： addr：寄存器起始地址，范围int[0,31]
length：寄存器个数，范围int[1,32]

返回： 寄存器值列表float[length]

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Ethernet/IP float型输入寄存器的值
      suc, result ,id=sendCMD(sock,"get_eip_float_input_registers",{"addr":0,"length":1})
    print ( result )
  else :
    print ("连接失败")
  disconnectETController(sock)
```

备注： addr与length的和应小于等于32，本命令适用于v3.5.2及以上版本。

2.2.12.4 获取Ethernet/IP float型输出寄存器的值

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

功能： 获取Ethernet/IP float型输出寄存器的值

参数： addr：寄存器起始地址，范围int[0,31]

length：寄存器个数，范围int[1,32]

返回： 寄存器值列表float[length]

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Ethernet/IP float型输出寄存器的值
        suc, result , id=sendCMD(sock,"get_eip_float_output_registers",{"addr":0,"length":1})
        print ( result )
    else :
        print ("连接失败")
    disconnectETController(sock)
```

备注： addr与length的和应小于等于32，本命令适用于v3.5.2及以上版本。

2.2.12.5 设置Ethernet/IP int型输出寄存器的值

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

功能： 设置Ethernet/IP int型输出寄存器的值

参数： addr：寄存器起始地址，范围int[0,31]
length：寄存器个数，范围int[1,32]
value：寄存器值列表，类型int[length]，元素范围[-2147483648,2147483647]

返回： 成功True，失败False

示例：

```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="192.168.1.202"
  conSuc,sock=connectETController(robot_ip)
  if (conSuc):
      # 设置Ethernet/IP int型输出寄存器的值
      suc, result ,id=sendCMD(sock,"set_eip_int_output_registers",{"addr":1,"length":2,"value":[1,1]})
    print ( result )
  else :
    print ("连接失败")
  disconnectETController (sock)
```

备注： addr与length的和应小于等于32，本命令只在remote模式下使用，适用于v3.5.2及以上版本。

2.2.12.6 设置Ethernet/IP float型输出寄存器的值

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

功能： 设置Ethernet/IP float型输出寄存器的值

参数： addr：寄存器起始地址，范围int[0,31]
length：寄存器个数，范围int[1,32]
value：寄存器值列表，类型double[length]，元素范围[-3.40E+38,3.40E+38]

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Ethernet/IP float型输出寄存器的值
      suc, result , id=sendCMD(sock,"set_eip_float_output_registers",{"addr":0,"length":2,"value":
      [1.1,1.2]})
    print ( result )
  else :
    print ("连接失败")
  disconnectETController (sock)
```

备注： addr与length的和应小于等于32，本命令只在remote模式下使用，适用于v3.5.2及以上版本。

2.2.13 外部力传感器服务

2.2.13.1 标记力矩数据传输开始

```
{"jsonrpc":"2.0","method":"start_push_force","params":{"return_flag":
   return_flag},"id":id}
```

功能： 标记力矩数据传输开始

参数： return_flag：设置有无返回值，bool类型，True：有返回，False：无返回

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):
      # 标记力矩数据传输开始
      suc, result ,id=sendCMD(sock,"start_push_force",{"return_flag":True})
```

备注： 只有在力控数据源为SDK的时候才能使用。

2.2.13.2 传递力矩数据

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

功能： 传递力矩数据

参数： index：序列号，指示传输的顺序，范围为[0,65535]；
torque_arry*：力矩数据数组，double torques[6]
*注：数据名称包含X方向力、Y方向力、Z方向力、X轴扭矩、Y轴扭矩、Z轴扭矩，当数据类型为double时，则依次表示的是力传感器输出坐标系下的X方向力、Y方向力、Z方向力、X轴力矩、Y轴力矩、Z轴力矩。方向力的单位为kg，扭矩的单位为kgM。若原始数据的单位与坐标系和定义不一致，请对参数进行转换再传入。

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):
        # 标记力矩数据传输开始
        suc, result ,id=sendCMD(sock,"start_push_force",{"return_flag":True})
        if result==True:
            # 传递力矩数据
            suc, result ,id=sendCMD(sock,"push_external_force",{"index":0,"torque_arry":[1,2,3,4,5,6]})
```

备注： 参数index的值应在一次完整的传输过程开始后，从0开始逐一向上递增，达到最大值65535后，从0开始计数，循环往复，通过这种方式标记torque_arry是最新的数据。
使用前需要先调用start_push_force标记外部力矩数据开始传递，且只有在力控数据源为SDK的时候才能使用。

2.2.13.3 结束当前力矩数据的传递

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

功能： 结束当前力矩数据的传递

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):
    # 标记力矩数据传输开始
    suc, result , id=sendCMD(sock,"start_push_force",{"return_flag":True})
    if result==True:
      # 传递力矩数据
      suc, result , id=sendCMD(sock,"push_external_force",{"index":0,"torque_arry":[1,2,3,4,5,6]})
      # 结束传递力矩数据
      suc, result , id=sendCMD(sock,"stop_push_force")
```

备注： 只有在力控数据源为SDK的时候才能使用。

2.2.13.4 获取当前力矩数据来源

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

功能： 获取当前力矩数据来源

参数： 无

返回： int[0,4]，0和1表示力矩数据来自内部，2表示力矩数据来自SDK，3表示力矩数据来自LUA，4表示力矩数据来自末端

示例：

```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if (conSuc):
    # 获取当前力矩数据来源
    suc, result , id=sendCMD(sock,"get_force_ctrl_mode")
    print ( result )
```

2.2.13.5 获取基于外部力传感器的TCP力及力矩信息

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

功能： 获取基于外部力传感器的TCP力及力矩信息

参数： 无

返回： double force[6]，前三位代表力，后三位代表力矩

示例：

```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="172.16.11.223"
  conSuc,sock=connectETController(robot_ip)
  if (conSuc):
     suc, result , id=sendCMD(sock,"get_tcp_force_by_sensor")
     print ( result )
  else
        print ("连接失败")
  disconnectETController (sock)
```

注意： 本命令只支持在外部力传感器处于连接状态时使用。
本命令适用于 v3.9.2 及以上版本。

2.2.13.6 Examples

2.2.13.6.1 Example 1 （数据传输有返回值示例）

```
import socket
import json
import os
import time import sleep

def connectETController(ip="192.168.1.200", port=8055):
    sock1 = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    try:
        sock1.connect((Ip, port))
        return True, sock1
    except Exception as e:
        print('错误明细是',e)
        sock1.close()
        return False, None

def disconnectETController(sock1):
    if sock1:
        sock1.close()
         sock1 = None
     else:
         sock1 = None

def sendCMD(sock1, 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)
        sock1.sendall(bytes(sendStr, "utf-8"))
        ret = sock1.recv(1024)
        jData = json.loads(str(ret, "utf-8"))
        # print("raw data:",jData)
        if "result" in jData.keys():
```

```
            return True, json.loads(jData["result"]), jData["id"]
        elif "error" in jData.keys():
            return False, jData["error"], jData["id"]
        else:
            return False, None, None
    except Exception as e:
        print('错误明细是',e)
        return False, None, None

if __name__ == "__main__":
    robot_ip = "192.168.1.200"
    return_flag = True

    conSuc, sock = connectETController(robot_ip)
    if conSuc:
        ret, result, id = sendCMD(sock, "get_force_ctrl_mode")
        if result != 2:  # sdk mode = 2
            print("please change the mode to sdk.")
            exit()

        ret, result, id = sendCMD(sock, "start_push_force",
           {"return_flag":return_flag})
        if result != True:
            exit()

        # torque data array
        arry = [1.2,2.3,3.2,4.4,5.6,6.9]
        index = 0
        cycle_time = 0.005  #interval(ms)
        host_time = 5       #last time(s)
        loopCnt = host_time / cycle_time
        while loopCnt > 0:
            ret, result, id = sendCMD(sock, "push_external_force",)
               {"index":index,"torque_arry":arry})
            if result != True:
                print("push_external_force failed!", result)
                break

            sleep(cycle_time)
            loopCnt -= 1
```

```
74              arry[0] += 1
75              index += 1
76
77          ret, result, id = sendCMD(sock, "stop_push_force")
78      else:
79          print("连接失败")
80
81      disconnectETController(sock)
```

2.2.13.6.2 Example 2 （数据传输无返回值示例）

```
import socket
import json
import os
from time import sleep

def connectETController(ip="192.168.1.200", port=8055):
    sock1 = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    try:
        sock1.connect((Ip, port))
        return True, sock1
    except Exception as e:
        print('错误明细是',e)
        sock1.close()
        return False, None

def disconnectETController(sock1):
    if sock1:
        sock1.close()
         sock1 = None
     else:
         sock1 = None

def sendCMD(sock1, 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:
        sock1.sendall(bytes(sendStr, "utf-8"))
        ret = sock1.recv(1024)
        jData = json.loads(str(ret, "utf-8"))
        if "result" in jData.keys():
          return True, json.loads(jData["result"]), jData["id"]
        elif "error" in jData.keys():
```

```
            return False, jData["error"], jData["id"]
        else:
            return False, None, None
    except Exception as e:
        print('错误明细是',e)
        return False, None, None

def push_external_force_without_return(index, arry):
    cmd = "push_external_force"
    params = {"index":index, "torque_arry":arry}
    if not params:
        params = []
    else:
        params = json.dumps(params)
    id = 1
    sendStr = "{{\"method\":\"{0}\",\"params\":{1},\"jsonrpc
       \":\"2.0\",\"id\":{2}}}".format(cmd, params, id) + "\n"
    sock1.sendall(bytes(sendStr, "utf-8"))

if __name__ == "__main__":
    robot_ip = "192.168.1.200"
    return_flag = False

    conSuc, sock = connectETController(robot_ip)
    if conSuc:
        ret, result, id = sendCMD(sock, "get_force_ctrl_mode")
        if result != 2:  # 力矩信息传输模式不为2（SDK模式）则退出
            print("please change the mode to sdk.")
            exit()

        ret, result, id = sendCMD(sock, "start_push_force",
           {"return_flag":return_flag})
        if result != True:
            exit()

        # 等待传输的力矩数组
        arry = [1.2,2.3,3.2,4.4,5.6,6.9]
        index = 0
```

```
72          cycle_time = 0.005  # 传输间隔（毫秒）
73          host_time = 5       # 本次传输总时长（秒）
74          loopCnt = host_time / cycle_time
75          while loopCnt > 0:
76              push_external_force_without_return(index, arry)
77              sleep(cycle_time)
78              loopCnt -= 1
79              arry[0] += 1
80              index += 1
81      else:
82          print("连接失败")
83
84      disconnectETController(sock)
85      conSuc, sock = connectETController(robot_ip)
86      if conSuc:
87          ret, result, id = sendCMD(sock, "stop_push_force")
88          disconnectETController(sock)
```

2.2.13.6.3 Example 3（数据解析和传输示例）

```python
import serial
import time
import socket
import json
import struct
from time import sleep

def connectETController(Ip="172.16.11.199", port=8055):
    sock1 = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    try:
        sock1.connect((Ip, port))
        return True, sock1
    except Exception as e:
        print('错误明细是', e)
        sock1.close()
        return False, None


def disconnectETController(sock1):
    if sock1:
        sock1.close()

def sendCMD(sock1, 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:
        sock1.sendall(bytes(sendStr, "utf-8"))
        ret = sock1.recv(1024)
        jData = json.loads(str(ret, "utf-8"))
        if "result" in jData.keys():
            return True, json.loads(jData["result"]), jData["id"]
        elif "error" in jData.keys():
            return False, jData["error"], jData["id"]
        else:
            return False, None, None
    except Exception as e:
        print('错误明细是', e)
        return False, None, None
```

```
42
43  # 全局变量
44  portDev = 'COM4'
45  baud = 460800 # 115200
46  parity = serial.PARITY_NONE # PARITY_ODD/PARITY_EVEN
47  stopbits = serial.STOPBITS_ONE
48  bytesize = serial.EIGHTBITS
49  tmout = 0.02 # unit sec
50
51  CMD_STOP_SEND = '43AA0D0A'
52  CMD_CLEAR_SEND = '47AA0D0A'
53  CMD_SEND_DATA = '49AA0D0A'
54  CMD_ACTIVE_SEND = '48AA0D0A'
55  DATA_HEAD = '48AA'
56  DATA_TAIL = '0D0A'
57
58  if __name__ == "__main__":
59      robot_ip = "172.16.11.200"
60      return_flag = True
61      index = 0
62      torque = [0, 0, 0, 0, 0, 0]
63      tm_begin = 0
64      call_tm = 0
65      err_cnt = 0
66
67      # 打开串口
68      try:
69          ser = serial.Serial(port=portDev, baudrate=baud, parity=parity,
                stopbits=stopbits, bytesize=bytesize, timeout=tmout)
70      except serial.serialutil.SerialException as err:
71          print("OS error: {0}".format(err))
72          exit()
73
74      if not ser.isOpen:
75          print("serial open unsuccessful.")
76          exit()
77
78      # 清空缓冲区
79      ser.flushInput()
80
81      # 连接机器人
82      conSuc, sock = connectETController(robot_ip)
83      if not conSuc:
84          print("Failed to connect to Robot.")
```

```
        exit()

    # 力控数据源式获取模式检查，需在 sdk 模式下
    ret, result, id = sendCMD(sock, "get_force_ctrl_mode")
    if result != 2:   # sdk mode = 2
        print("please change the mode to sdk.")
        exit()

    ret, result, id = sendCMD(sock, "stop_push_force")

    # 配置传感器主动发送数据
    while True:
        ser.flushInput()
        ser.write(bytes.fromhex(CMD_ACTIVE_SEND))
        count = ser.inWaiting() # 获取串口缓冲区数据
        if count > 0:  # 有接收数据表示配置成功
            break
        sleep(0.01)

    # 标记力矩传输开始
    ret, result, id = sendCMD(sock, "start_push_force", {"return_flag" :
        return_flag})
    if result != True:
        exit()

    print("Start to transmission force torque data.")

    # 主循环
    while True:
        # 一帧有效数据为 28 bytes
        if count > 27:
            recvRaw = ser.read(count) # 读出串口数据，为 bytes 类
            beginIdx = recvRaw.find(bytes.fromhex(DATA_HEAD))
            if beginIdx != -1:
                endIdx = recvRaw.find(bytes.fromhex(DATA_TAIL), beginIdx)
                if endIdx != -1 and endIdx - beginIdx == 26:
                    recv = recvRaw[beginIdx:endIdx + 2]
                    recv = list(recv)
                    f1_raw = recv[2:6]
                    f2_raw = recv[6:10]
                    f3_raw = recv[10:14]
                    f4_raw = recv[14:18]
                    f5_raw = recv[18:22]
                    f6_raw = recv[22:26]
```

```
                    f1 = struct.unpack('f', bytes(f1_raw))[0]
                    f2 = struct.unpack('f', bytes(f2_raw))[0]
                    f3 = struct.unpack('f', bytes(f3_raw))[0]
                    f4 = struct.unpack('f', bytes(f4_raw))[0]
                    f5 = struct.unpack('f', bytes(f5_raw))[0]
                    f6 = struct.unpack('f', bytes(f6_raw))[0]
                    torque = [f1, f2, f3, f4, f5, f6]
                    ser.flushInput()
                    err_cnt = 0
                else:
                    sleep(0.001)
            else:
                sleep(0.001)
        else:
            err_cnt += 1
            if err_cnt > 50 :
                print("sensor data receive length(%d) error, Transmission quit!"%(count))
                break


        # 记算完成一次传输的时间，方便调试
        tm_now = time.perf_counter()
        tm_elapse = tm_now - tm_begin
        tm_begin = tm_now
        ret, result, id = sendCMD(sock, "push_external_force", {"index":index, "torque_arry":torque})
        if result != True:
            tm_now = time.perf_counter()
            curr_call_tm = tm_now - tm_begin  # 计算本次接口调用时长
            print("last call time = %f, once loop elapse time = %f, current call time = %f." % (call_tm, tm_elapse, curr_call_tm))
            print("push_external_force failed!", result)
            break

        # 计算上次接口调用时长
        call_tm = time.perf_counter() - tm_begin
        # 必要步骤
        if index >= 65535:
            index = 0
        else:
            index += 1

        sleep(0.001)
```

```
169         count = ser.inWaiting() # 获取串口缓冲区数据
170
171     ret, result, id = sendCMD(sock, "stop_push_force")
172     disconnectETController(sock)
173     ser.close()
```

2.2.14 硬件通讯控制服务

2.2.14.1 打开硬件串口

```
{"jsonrpc":"2.0","method":"open_serial_port","params":{"device_type":
   device_type},"id":id}
```

功能： 打开硬件串口

参数： device_type：数据类型，int[0,1]，0代表RS232通信，1代表RS485通信

返回： 成功True，失败False

示例：

```
if __name__ == "__main__":
    # 机器人IP地址
    robot_ip="172.16.11.248"
    conSuc,sock=connectETController(robot_ip)
    if (conSuc):
        suc, result ,id=sendCMD(sock,"open_serial_port",{"device_type":0})
        print ( suc, result, id )
    else:
        print ("连接失败")
    disconnectETController (sock)
```

备注： 成功打开串口后，会自动配置串口数据，默认波特率为9600，数据长度为8，奇偶校位为‘N’，停止位为1，如需修改，请参考配置串口功能自行设置。

本命令只支持在remote模式下使用。本命令适用于v3.6.2及以上版本。

2.2.14.2 配置硬件串口

```
{"jsonrpc":"2.0","method":"setopt_serial_port","params":{"baud_rate":9600,
   "bits":8,"event":"N", "stop":1},"id":id}
```

功能： 配置硬件串口

参数： baud_rate：波特率，int型，可选。

RS232 通信：波特率有2400，4800，9600，19200，38400，57600，115200，230400，460800，500000，可选择其中一种，不传参时，波特率默认为9600；

RS485 通信：波特率有2400，4800，9600，19200，38400，57600，115200，230400，460800，500000，921600，1000000，2000000，3000000，可选择其中一种，不传参时，波特率默认为9600

bits：数据长度，int型，可选，数据长度有7和8两种，可选择其中一种，不传参时， 数据长度默认为8

event：奇偶校验，字符串数据类型，可选，奇偶校验有“O”、“N”和“E”三种，可选择其中一种，不传参时，奇偶校验默认为“N”

stop：停止位，int型，可选，停止位有1和2两种，可选择其中一种，不传参时，停止位默认为1

返回： 成功True，失败False

示例：

```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="172.16.11.248"
  conSuc,sock=connectETController(robot_ip)
  if (conSuc):
      suc, result , id=sendCMD(sock,"open_serial_port",{"device_type":0})
      print ( suc, result, id )
      suc, result , id=sendCMD(sock,"setopt_serial_port",{"baud_rate":9600,"bits":8, "event":"N",
          "stop":1})
      print ( suc, result, id )
  else:
      print ("连接失败")
  disconnectETController(sock)
```

备注： 只有在打开串口后，才能执行该命令。

本命令只支持在remote模式下使用。本命令适用于v.3.6.2及以上版本。

2.2.14.3 接收数据

```
{"jsonrpc":"2.0","method":"recv_serial_port","params":{"count":5},"id":id}
```

功能： 接收数据

参数： count：接收的数据长度，int[0,256]

hex：接收的数据模式，可选，int[0,1]，0代表ASCII（文本）模式，1代表十六进制模式，可选择其中一种，不传参时，默认为文本模式

timeout：超时时间，可选，int[0,2147483647]，不传参时，超时时间默认为100ms

返回： result：返回结果，成功True，失败False

size：读取到的数据长度

buf：读取到的数据

示例：

```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="172.16.11.248"
  conSuc,sock=connectETController(robot_ip)
  if (conSuc):
        suc, result ,id=sendCMD(sock,"open_serial_port",{"device_type":0})
        print ( suc, result, id )
        suc, recv ,id=sendCMD(sock,"recv_serial_port",{"count":100,"hex":0, "timeout":10})
        print ( suc, recv, id )
        time.sleep(1)
  else:
        print ("连接失败")
  disconnectETController(sock)
```

备注： 根据返回结果来判断读取的数据是否有效，当返回结果为True时，则代表数据有效。
只有在打开串口后，才能执行该命令。
本命令只支持在remote模式下使用。本命令适用于v3.6.2及以上版本。

2.2.14.4 发送数据

```
{"jsonrpc":"2.0","method":"send_serial_port","params":{"send_buf":"hello
   world!"},"id":id}
```

功能： 发送数据

参数： send_buf：需要发送的数据，字符串数据类型，字符串长度最长为256

hex：发送的数据模式，可选，int[0,1]，0代表ASCII（文本）模式，1代表十六进制模式

返回： result：返回结果，成功True，失败False

size：已发送的数据长度

示例：

```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="172.16.11.248"
 conSuc,sock=connectETController(robot_ip)
  if (conSuc):
        suc, result ,id=sendCMD(sock,"open_serial_port",{"device_type":0})
        print ( suc, result, id )
        suc, result ,id=sendCMD(sock,"send_serial_port",{"send_buf":"1024","hex":0})
        print ( suc, result, id )
  else:
        print ("连接失败")
  disconnectETController (sock)
```

备注： 只有在打开串口后，才能执行该命令。
本命令只支持在remote模式下使用。本命令适用于v3.6.2及以上版本。

2.2.14.5 清空硬件缓冲区

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

功能： 清空硬件缓冲区

参数： 无

返回： 成功True，失败False

示例：

```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="172.16.11.248"
  conSuc,sock=connectETController(robot_ip)
  if (conSuc):
      suc, result ,id=sendCMD(sock,"flush_serial_port")
      print ( suc, result, id )
  else:
      print ("连接失败")
  disconnectETController (sock)
```

备注： 本命令只支持在remote模式下使用。本命令适用于v3.6.2及以上版本。

2.2.14.6 关闭硬件串口

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

功能： 关闭硬件串口

参数： 无

返回： 成功True，失败False

示例：

```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="172.16.11.248"
  conSuc,sock=connectETController(robot_ip)
  if (conSuc):
      suc, result ,id=sendCMD(sock,"close_serial_port")
      print ( suc, result, id )
  else:
      print ("连接失败")
  disconnectETController (sock)
```

备注： 本命令只支持在remote模式下使用。本命令适用于v3.6.2及以上版本。

2.2.15 TCI 通讯控制服务

2.2.15.1 打开TCI串口

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

功能： 打开TCI串口

参数： 无

返回： 成功True，失败False

示例：

```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="172.16.11.248"
  conSuc,sock=connectETController(robot_ip)
  if (conSuc):
      suc, result ,id=sendCMD(sock,"open_tci")
      print ( suc, result, id )
  else:
      print ("连接失败")
  disconnectETController (sock)
```

备注： 成功打开串口后，会自动配置串口数据，默认波特率为9600，数据长度为8，奇偶校验位为‘N’，停止位为1，如需修改，请参考配置串口功能自行设置。
本命令只支持在remote模式下使用。本命令适用于v3.6.2及以上版本。

2.2.15.2 配置TCI串口

```
{"jsonrpc":"2.0","method":"setopt_tci","params":{"baud_rate":9600, "bits":8,
   "event":"N", "stop":1},"id":id}
```

功能： 配置TCI串口

参数： baud_rate：波特率，int型，可选，波特率有2400, 4800，9600，19200，38400，57600，115200，230400，460800，500000，921600，1000000，2000000，可选择其中一种，不传参时，波特率默认为9600

bits：数据长度，int型，数据长度为8，可选

event：奇偶校验，字符串数据类型，可选，奇偶校验有“O”、“N”和“E”三种，可选择其中一种，不传参时，奇偶校验默认为“N”

stop：停止位，int型，可选，停止位有1和2两种，可选择其中一种，不传参时，停止位默认为1

返回： 成功True，失败False

示例：

```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="172.16.11.248"
  conSuc,sock=connectETController(robot_ip)
  if (conSuc):
      suc, result , id=sendCMD(sock,"open_tci")
      print ( suc, result, id )
      suc, result , id=sendCMD(sock,"setopt_tci",{"baud_rate":9600,"bits":8, "event":"N",
          "stop":1})
      print ( suc, result, id )
  else:
      print ("连接失败")
  disconnectETController (sock)
```

备注： 只有在打开串口后，才能执行该命令。本命令只支持在remote模式下使用。本命令适用于v.3.6.2及以上版本。

2.2.15.3 接收数据

```
{"jsonrpc":"2.0","method":"recv_tci","params":{"count":5},"id":id}
```

功能： 接收数据

参数： count：接收的数据长度，int[0,256]

hex：接收的数据模式，可选，int[0,1]，0代表ASCII（文本）模式，1代表十六进制模式，可选择其中一种，不传参时，默认为文本模式

timeout：超时时间，可选，int[0,2147483647]，不传参时，超时时间默认为100ms

返回： result：返回结果，成功True，失败False

size：读取到的数据长度

buf：读取到的数据

示例：

```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="172.16.11.248"
  conSuc,sock=connectETController(robot_ip)
  if (conSuc):
        suc, result ,id=sendCMD(sock,"open_tci")
        print ( suc, result, id )
        suc, recv, id=sendCMD(sock,"recv_tci",{"count":100,"hex":0, "timeout":10})
        print ( suc, recv, id )
        time.sleep(1)
  else:
        print ("连接失败")
  disconnectETController (sock)
```

备注： 根据返回结果来判断读取的数据是否有效，当返回结果为True时，则代表数据有效。
只有在打开串口后，才能执行该命令。
本命令只支持在remote模式下使用。本命令适用于v3.6.2及以上版本。

2.2.15.4 发送数据

```
{"jsonrpc":"2.0","method":"send_tci","params":{"send_buf":"hello world!"},
   "id":id}
```

功能： 发送数据

参数： send_buf：需要发送的数据，类型为字符串，字符串长度最长为256

hex：发送的数据模式，可选，int[0,1]，0代表ASCII（文本）模式，1代表十六进制模式

返回： result：返回结果，成功True，失败False

size：已发送的数据长度

示例：

```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="172.16.11.248"
 conSuc,sock=connectETController(robot_ip)
  if (conSuc):
        suc, result ,id=sendCMD(sock,"open_tci")
        print ( suc, result, id )
        suc, result ,id=sendCMD(sock,"send_tci",{"send_buf":"1024","hex":0})
        print ( suc, result, id )
  else:
        print ("连接失败")
  disconnectETController (sock)
```

备注： 只有在打开串口后，才能执行该命令。
本命令只支持在remote模式下使用。本命令适用于v3.6.2及以上版本。

2.2.15.5 清空TCI缓冲区

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

功能： 清空TCI缓冲区

参数： 无

返回： 成功True，失败False

示例：

```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="172.16.11.248"
  conSuc,sock=connectETController( robot_ip )
  if (conSuc):
      suc, result , id=sendCMD(sock,"flush_tci")
      print ( suc, result, id )
  else:
      print ("连接失败")
  disconnectETController (sock)
```

备注： 本命令只支持在remote模式下使用。本命令适用于v3.6.2及以上版本。

2.2.15.6 关闭TCI串口

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

功能： 关闭TCI串口

参数： 无

返回： 成功True，失败False

示例：

```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="172.16.11.248"
  conSuc,sock=connectETController(robot_ip)
  if (conSuc):
      suc, result ,id=sendCMD(sock,"close_tci")
      print ( suc, result, id )
  else:
      print ("连接失败")
  disconnectETController(sock)
```

备注： 本命令只支持在remote模式下使用。本命令适用于v3.6.2及以上版本。

2.2.16 力控服务

2.2.16.1 开启力控模式

```
{"jsonrpc":"2.0","method":"start_force_mode","params":{"mode":3,
   "arr_frame":frame,"arr_optional":[6,1,5,1,0,1],"arr_torque":
   [-10,-10,-10,1.3,1.30,1.30],"arr_speed":speed},"id":id}
```

功能： 开启力控

参数： mode：模式，可选，int[0,4]，0-4分别代表固定模式、点模式、运动模式、TCP模式和姿态模式，默认值为0

arr_frame：用户指定的力控坐标系，可选，x，y，z单位为毫米，范围是[-10000, 10000]，Rx，Ry，Rz单位为弧度，范围是[-π,π]，默认值为[0, 0, 0, 0, 0, 0]，此时力控坐标系与基坐标系重合

arr_optional：力控方式（自由掩码），可选，取值范围int[0-6]，0-6分别代表运动控制、力跟踪、固定、浮动、弹簧、浮动&运动、弹簧&运动，默认值为[0, 0, 0, 0, 0, 0]，此时代表运动控制

arr_torque：力控的目标力矩值，可选。double arr_torque[6]，前三位代表力，EC63的力矩范围为[-30,30]N，EC66的力矩范围为[-60,60]N，EC612的力矩范围为[-120,120]N， EC616的力矩范围为[-160,160]N；后三位代表力矩，力矩范围为[-50, 50]N，默认值为[0，0，0，0，0，0]

arr_speed：力控速度限制，可选，单位mm/s和°/s。double arr_speed [6]，前三位代表线速度，范围为[0,200]；后三位代表角速度，范围为[0,11]，默认值为[100, 100, 100, 5.73, 5.73, 5.73]

返回： 成功True，失败False

示例：

```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="172.16.11.248"
  conSuc,sock=connectETController(robot_ip)
  if (conSuc):
      suc, result ,id=sendCMD(sock,"start_force_mode",{"mode":0, "arr_frame":[0,0,0,0,0,0],
          "arr_optional":[1,0,0,0,0,0], "arr_torque":[1,0,0,0,0,0], "arr_speed":[100,100,100,5.73,5.73,5.73
          ]})
      print ( suc, result, id )
  else:
      print ("连接失败")
  disconnectionETController (sock)
```

备注： 使用该指令即可打开力控功能，在力控开启的过程中可动态配置力控相关参数。
当力控处于打开的状态时，执行jog运动，自动关闭力控。
在机器人运动过程中，无法打开力控，也无法进行力控参数的配置。
当力控数据源为未知数据源时，无法进入力控模式。
拖动功能与力控功能互斥，无法同时使用。
目前开发的模式只有固定模式和TCP模式。
目前开发的力控方式只有运动控制和力跟踪。该命令适用于v3.6.2及以上版本。

2.2.16.2 关闭力控模式

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

功能： 关闭力控

参数： 无

返回： 成功True，失败False

示例：
```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="172.16.11.248"
  conSuc,sock=connectETController(robot_ip)
  if (conSuc):
      suc, result ,id=sendCMD(sock,"end_force_mode")
      print ( suc, result, id )
  else:
      print ("连接失败")
  disconnectETController (sock)
```

备注： 当力控处于打开状态时，机器人运动过程中无法关闭力控，发送本指令时，机器人则停止运动。本命令适用于v3.6.2及以上版本。

2.2.16.3 获取力控状态

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

功能：获取力控状态

参数： 无

返回： True表示处于力控模式，False表示处于非力控模式

示例：

```
if __name__ == "__main__":
    # 机器人IP地址
    robot_ip="172.16.11.248"
    conSuc,sock=connectETController(robot_ip)
    if (conSuc):
        suc, result ,id=sendCMD(sock,"get_force_mode_state")
        print ( suc, result, id )
    else:
        print ("连接失败")
    disconnectETController(sock)
```

备注： 本命令适用于v3.6.2及以上版本。

2.2.16.4 力传感器数据清零

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

功能： 力传感器数据清零，记录当前力传感器的数据

参数： 无

返回： 成功true，失败false

示例：

```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="192.168.1.240"
  conSuc,sock=connectETController(robot_ip)
  point = []
  point.append([0, -107.828, 119.349, -109.754, 83.351, -5.531, 0, 0])
  point.append([10, -98.342, 51.251, -92.155, 90, 0, 0, 0])
  point.append([-10.60024752475249, -80.04362623762376, 135.67295792079207,-193.44984567901236,
  123.90856481481481, 24.090277777777775])
  point.append([47.40996287128712, -51.567450495049506, 49.92419554455446, -88.9988425925926,
  79.45138888888889, 160.08603395061726])
  if (conSuc):
    # 获取机械臂伺服状态
    suc, result ,id=sendCMD(sock,"getServoStatus")
    print (suc, result, id )
    if (result = =0):
      # 设置机械臂伺服状态ON
      suc, result ,id=sendCMD(sock,"set_servo_status",{"status":1})
      print (suc, result, id)
    # 关节运动到起始点
    suc, result ,id=sendCMD(sock,"moveByJoint",{"targetPos": point[0],"speed":10,"acc":20,"dec":20})
    print (result):
    time.sleep(1)
    while 1:
      ret, result, id=sendCMD(sock,"getRobotState"):
      if result==0:
        break
    # 打开力控
    suc, result ,id=sendCMD(sock,"start_force_mode",{"mode":0,"arr_frame":[0,0,0,0,0,0],
    "arr_optional":[0,0,1,0,0,0], "arr_torque":[0,0,0,0,0,0],"arr_speed":[100,100,100,5.73,5.73,5.73]})
    print (suc, result, id)
    # 清零
    ret, result ,id=sendCMD(sock,"zero_force_sensor")
    print ( "清零:", result )
    suc, result ,id=sendCMD(sock,"moveByLine",{"targetPos": point[1],"speed":100,"speed_type":0,
    "acc":80,"dec":80})
    print (result):
    while 1:
      ret, result, id=sendCMD(sock,"getRobotState"):
      if result==0:
```

```
            break
    # 关闭力控
    suc, result ,id=sendCMD(sock,"end_force_mode")
    print (suc, result, id)
else :
    print ("连接失败")
disconnectETController (sock)
```

备注：在STARTFORCEMODE后插入本命令将读取并保存当前传感器的数值。在力控过程中，将当前传感器读数减去之前保存的传感器读数。
本命令仅在力控模式下生效。
本命令仅在将力控模式数据源配置为SDK，LUA和末端模式下生效。
本命令仅支持在remote模式下使用。
在力控模式下，当发送end_force_mode指令后，清除当前力传感器数据的功能将会失效。

2.2.16.5 设置力跟踪阻尼参数

```
{"jsonrpc":"2.0","method":"set_force_tracking_damping","params"{"damping":damping},"id":id}
```

功能： 设置力跟踪阻尼参数

参数： damping：double型，范围[0,1]，默认值为0，值越大力跟踪阻尼越大

返回： 成功True，失败False

示例：

```
if __name__ == "__main__":
    # 机器人IP地址
    robot_ip="172.16.11.248"
    conSuc,sock=connectETController( robot_ip )
    if (conSuc):
        suc, result ,id=sendCMD(sock,"set_force_tracking_damping", {"damping": 0.1})
        print ( result )
```

备注：本命令适用于v3.16.2及以上版本。

2.2.16.6 获取力跟踪阻尼参数

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

功能： 获取力跟踪阻尼参数

参数： 无

返回： 力跟踪阻尼参数值

示例：

```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="172.16.11.248"
  conSuc,sock=connectETController(robot_ip)
  if (conSuc):
      suc, result , id=sendCMD(sock,"get_force_tracking_damping")
      print ( result )
```

备注：本命令适用于v3.16.2及以上版本。

2.2.16.7 开启寻力运动

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

功能： 开启寻力运动

参数： 无

返回： 成功True，失败False

示例：

```
if __name__ == "__main__":
  # 机器人IP地址
  robot_ip="172.16.11.223"
  conSuc,sock=connectETController(robot_ip)
  if (conSuc):
      suc, result ,id=sendCMD(sock,"start_force_mode",{"mode":0,"arr_frame":[0,0,0,0,0,0],
      "arr_optional":[0,0,1,0,0,0], "arr_torque":[0,0,0,0,0,0],"arr_speed":[10,10,10,1,1,1]})
      print ( suc, result, id )
      ret, result ,id=sendCMD(sock,"fc_act")
      print ("寻力运动：", result)
      time.sleep(1)
  else:
       print ("连接失败")
  disconnectETController (sock)
```

备注：在发送fc_act指令后，机器人将进行寻力运动，若在此过程中，发送直线运动、关节运动或指定坐标系下直线运动，则会打断当前正在进行的寻力运动，并执行发送的直线运行、关节运动或指定坐标系下直线运动。
若在寻力运动过程中机器人出现报警，则会结束当前正在进行的寻力运动。
当在寻力运动过程中发送暂停或停止指令时，机器人状态为停止状态，且会结束当前正在进行的寻力运动。
本命令仅在机器人停止状态下生效。
本命令仅支持在remote模式下使用。
本命令适用于V3.9.2及以上版本。

2.2.16.8 结束寻力运动

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

功能： 结束寻力运动

参数： 无

返回： 成功True，失败False

示例：

```
if __name__ == "__main__":
  # 机器人IP地址
```

```
robot_ip="172.16.11.223"
conSuc,sock=connectETController(robot_ip)
if(conSuc):
    suc, result ,id=sendCMD(sock,"start_force_mode",{"mode":0,"arr_frame":[0,0,0,0,0,0],
    "arr_optional":[0,0,1,0,0,0], "arr_torque":[0,0,0,0,0,0],"arr_speed":[10,10,10,1,1,1]})
    print( suc, result, id )
    ret, result ,id=sendCMD(sock,"fc_act")
    print("寻力运动：", result)
    time.sleep(10)
    ret, result ,id=sendCMD(sock,"fc_deact")
    print("关闭寻力运动：", result)
    suc, result ,id=sendCMD(sock,"end_force_mode")
    print( suc, result, id )
else:
    print("连接失败")
disconnectETController(sock)
```

备注：本命令仅支持在remote模式下使用。
本命令适用于V3.9.2及以上版本。

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
```

```
    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:
        sock.sendall(bytes(sendStr,"utf-8"))
        ret =sock.recv(1024)
        jdata=json.loads(str(ret,"utf-8"))
        if("result" in jdata.keys()):
            return (True,json.loads(jdata["result"]),jdata["id"])
        elif("error" in jdata.keys()):
            return (False,jdata["error"],jdata["id"])
        else:
            return (False,None,None)
    except Exception as e:
        return (False,None,None)

if __name__ == "__main__":
    # 机器人IP地址
    robot_ip="192.168.1.202"
    conSuc,sock=connectETController(robot_ip)
    print(conSuc)
    if(conSuc):
        # 清除警报
        ret, result, id = sendCMD(sock, "clearAlarm")
        print("清除报警")
        print("ret = ", ret, " ", "id = ", id)
        if (ret == True):
            print("result = ", result)
            time.sleep(1)
        else:
            print("err_msg = ", result["message"])
        # 获取同步状态
```

```
57          ret, result, id = sendCMD(sock, "getMotorStatus")
58          print("获取同步状态")
59          print("ret = ", ret, " ", "id = ", id)
60          if (ret == True):
61              print("result = ", result)
62              if(result != 1):
63                  # 同步
64                  ret1, result1, id = sendCMD(sock, "syncMotorStatus")
65                  print("同步")
66                  print("ret = ", ret1, " ", "id = ", id)
67                  if (ret1 == True):
68                      print("result = ", result1)
69                      time.sleep(0.5)
70                  else:
71                      print("err_msg = ", result1["message"])
72          else:
73              print("err_msg = ", result["message"])
74
75
76          # 打开伺服
77          ret, result, id = sendCMD(sock, "set_servo_status", {"status"
                :1})
78          print("打开伺服")
79          print("ret = ", ret, " ", "id = ", id)
80          if (ret == True):
81              print("result = ", result)
82              time.sleep(1)
83          else:
84              print("err_msg = ", result["message"])
85          # 获取伺服状态
86          ret, result, id = sendCMD(sock, "getServoStatus")
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)
```

2.3.2 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()):
```

```
            return (True, json.loads(jdata["result"]), jdata["id"])
        elif ("error" in jdata.keys()):
            return (False, jdata["error"], jdata["id"])
        else:
            return (False, None, None)
    except Exception as e:
        return (False, None, None)


if __name__ == "__main__":
    ip = "192.168.1.205"
    conSuc, sock = connectETController(ip)
    # print(conSuc)
    if (conSuc):
        # 获取机器人状态
        ret, result, id = sendCMD(sock, "getRobotState")
        print("获取机器人状态")
        print("ret = ", ret, " ", "id = ", id)
        if (ret == True):
            print("result = ", result)
        else:
            print("err_msg = ", result["message"])
        # 获取机器人模式
        ret, result, id = sendCMD(sock, "getMotorStatus")
        print("获取机器人模式")
        print("ret = ", ret, " ", "id = ", id)
        if (ret == True):
            print("result = ", result)
        else:
            print("err_msg = ", result["message"])
        # 获取机器人当前位置信息
        ret, result, id = sendCMD(sock, "get_joint_pos")
        print("获取机器人当前位置信息")
        print("ret = ", ret, " ", "id = ", id)
        if (ret == True):
            print("result = ", result)
        else:
            print("err_msg = ", result["message"])
        # 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):
```

```
        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()):
            return (True, json.loads(jdata["result"]), jdata["id"])
        elif ("error" in jdata.keys()):
            return (False, jdata["error"], jdata["id"])
        else:
            return (False, None, None)
    except Exception as e:
        return (False, None, None)


if __name__ == "__main__":
    ip = "192.168.1.205"
    conSuc, sock = connectETController(ip)
    # print(conSuc)
    if (conSuc):
        # 切换当前工具号为0
        ret, result, id = sendCMD(sock, "setToolNumber", {"tool_num":
           0})
        print("切换当前工具号为0")
        print("ret = ", ret, " ", "id = ", id)
        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。

提醒

该功能适用于 2.13.1 及以上版本。

提醒

8056 是tcp长连接，如果获取周期>8ms，有可能缓冲区里面形成堆积。

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 801，预留字段长度为 219，校验字段长度为4
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	基座坐标，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

名称	类型	字节	说明
torque	double[AXIS_COUNT]	8*8	关节额定力矩千分比，单位 ‰
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	伺服状态：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	基座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：$度/s^2$
jointTemperature	Double[6]	8*6	关节温度，单位：℃

名称	类型	字节	说明
jointTorque	Double[6]	8*6	关节输出扭矩，单位：Nm
extJointTorques	Double[6]	8*6	外部关节扭矩估计值，单位：Nm
exTcpForceInTool	Double[6]	8*6	当前工具坐标系下的外部末端力/力矩估计值，前三项单位：N，后三项单位：NM
dragState	unsigned char	1*1	拖动使能状态，1：拖动使能处于打开状态，0：表示拖动使能处于关闭状态
sensor_connected_state	unsigned char	1*1	力控传感器当前状态，1：连接状态，0：断开状态
Reserved	unsigned char	1*219	数据包预留长度，长度219
matchingWord	unsigned int	4*1	校验位，数据包的最后四个字节，值为3967833836（即0xec8056ec）时数据有效，否则数据无效

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'
            dic['machineUserPose02'] = 'd'
            dic['machineUserPose03'] = 'd'
            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'
```

```
            dic['machineUserFlangePose03'] = 'd'
            dic['machineUserFlangePose04'] = 'd'
            dic['machineUserFlangePose05'] = 'd'
            dic['machineUserFlangePose06'] = 'd'
            dic["emergencyStopState"] = "B"
            dic["tcp_speed"] = "d"
            dic["joint_speed01"] = "d"
            dic["joint_speed02"] = "d"
            dic["joint_speed03"] = "d"
            dic["joint_speed04"] = "d"
            dic["joint_speed05"] = "d"
            dic["joint_speed06"] = "d"
            dic["joint_speed07"] = "d"
            dic["joint_speed08"] = "d"
            dic["tcpacc"] = "d"
            dic["jointacc01"] = "d"
            dic["jointacc02"] = "d"
            dic["jointacc03"] = "d"
            dic["jointacc04"] = "d"
            dic["jointacc05"] = "d"
            dic["jointacc06"] = "d"
            dic["jointacc07"] = "d"
            dic["jointacc08"] = "d"
            dic["joint_temperature01"] = "d"
            dic["joint_temperature02"] = "d"
            dic["joint_temperature03"] = "d"
            dic["joint_temperature04"] = "d"
            dic["joint_temperature05"] = "d"
            dic["joint_temperature06"] = "d"
            dic["joint_torque01"] = "d"
            dic["joint_torque02"] = "d"
            dic["joint_torque03"] = "d"
            dic["joint_torque04"] = "d"
            dic["joint_torque05"] = "d"
            dic["joint_torque06"] = "d"
            dic["extjoint_torques01"] = "d"
            dic["extjoint_torques02"] = "d"
            dic["extjoint_torques03"] = "d"
            dic["extjoint_torques04"] = "d"
            dic["extjoint_torques05"] = "d"
            dic["extjoint_torques06"] = "d"
```

```
121             dic["exttcpforceintool01"] = "d"
122             dic["exttcpforceintool02"] = "d"
123             dic["exttcpforceintool03"] = "d"
124             dic["exttcpforceintool04"] = "d"
125             dic["exttcpforceintool05"] = "d"
126             dic["exttcpforceintool06"] = "d"
127             dic["dragState"] = "B"
128             dic["sensor_connected_state"] = "B"
129
130             print("index =", index)
131             data = s.recv (1024)
132             if len(data) != 1024:
133                   lost += 1
134                   print(str(lost))
135                   continue
136
137             names =[]
138             ii=range(len(dic))
139             for key ,i in zip(dic ,ii):
140                   fmtsize = struct. calcsize (dic[key ])
141                   data1 , data = data [0: fmtsize], data[fmtsize :]
142                   fmt="!" + dic[key]
143                   names.append(struct.unpack(fmt ,data1))
144                   dic[key] = dic[key], struct.unpack(fmt , data1)
145             output = ""
146             for key in dic.keys ():
147                   output += str(key) + ":" + str(dic[key ][1][0]) + ";\
                        n"
148             output = "lost : " + str(lost) + " index : " + str(index) +
                  ";" + output + "\n"
149             if dic['MessageSize'] != ('I', (1024,)):
150                   s.close()
151                   break
152             matchingWord = struct.unpack('!I', data[-4:])[0]
153             if matchingWord != 3967833836:
154                   continue
155
156             if index %10 == 0:
157                   # 打印所有信息
158                   print(output)
159                   # 打印数据包长度
```

```
160                 print(dic['MessageSize'])
161                 # 打印时间戳
162                 timestamp01_value = dic['TimeStamp'][1][0] // 1000
163                 timeValue = time.gmtime(int(timestamp01_value))
164                 print(time.strftime("%Y-%m-%d %H:%M:%S", timeValue))
165                 # 打印关节坐标
166                 print(dic['machinePos01'][1][0], dic['machinePos02'
                        ][1][0],
167                       dic['machinePos03'][1][0], dic['machinePos04'
                          ][1][0],
168                       dic['machinePos05'][1][0], dic['machinePos06'
                          ][1][0],
169                       dic['machinePos07'][1][0], dic['machinePos08'
                          ][1][0])
170                 # 打印基座坐标
171                 print(dic['machinePose01'][1][0], dic['machinePose02'
                        ][1][0], dic['machinePose03'][1][0],
172                       dic['machinePose04'][1][0],
173                       dic['machinePose05'][1][0], dic['machinePose06'
                          ][1][0])
174                 # 打印用户坐标
175                 print(dic['machineUserPose01'][1][0], dic['
                        machineUserPose02'][1][0], dic['machineUserPose03'
                        ][1][0],
176                       dic['machineUserPose04'][1][0],
177                       dic['machineUserPose05'][1][0], dic['
                          machineUserPose06'][1][0])
178                 # 打印关节额定力矩百分比
179                 print(dic['torque01'][1][0], dic['torque02'][1][0],
                        dic['torque03'][1][0], dic['torque04'][1][0],
180                       dic['torque05'][1][0],
181                       dic['torque06'][1][0], dic['torque07'][1][0],
                          dic['torque08'][1][0])
182                 # 打印机器人状态
183                 print(dic['robotState'][1][0])
184                 # 打印伺服使能状态
185                 print(dic['servoReady'][1][0])
186                 # 打印同步状态
187                 print(dic['can_motor_run'][1][0])
188                 # 打印各轴电机转速
189                 print(dic['motor_speed01'][1][0], dic['motor_speed02'
```

```
                    ][1][0], dic['motor_speed03'][1][0],
190                     dic['motor_speed04'][1][0], dic['motor_speed05'
                            ][1][0],
191                     dic['motor_speed06'] [1][0], dic['motor_speed07
                            '][1][0], dic['motor_speed08'][1][0])
192                 # 打印机器人模式
193                 print(dic['robotMode'][1][0])
194                 # 打印模拟量输入口数据
195                 print(dic['analog_ioInput01'][1][0], dic['
                        analog_ioInput02'][1][0], dic['analog_ioInput03'
                        ][1][0])
196                 # 打印模拟量输出口数据
197                 print(dic['analog_ioOutput01'][1][0], dic['
                        analog_ioOutput02'][1][0], dic['analog_ioOutput03'
                        ][1][0],
198                     dic['analog_ioOutput04'][1][0], dic['
                            analog_ioOutput05'][1][0])
199                 # 打印数字量输入口数据的二进制形式
200                 print(bin(dic['digital_ioInput'][1][0])[2:].zfill(64)
                        )
201                 # 打印数字量输出口数据的二进制形式
202                 print(bin(dic['digital_ioOutput'][1][0])[2:].zfill
                        (64))
203                 # 打印碰撞报警状态
204                 print(dic["collision"][1][0])
205                 # 打印基座坐标系下的法兰盘中心位姿
206                 print(dic['machineFlangePose01'][1][0], dic['
                        machineFlangePose02'][1][0], dic['
                        machineFlangePose03'][1][0],
207                     dic['machineFlangePose04'][1][0],
208                     dic['machineFlangePose05'][1][0], dic['
                            machineFlangePose06'][1][0])
209                 # 打印用户坐标系下的法兰盘中心位姿
210                 print(dic['machineUserFlangePose01'][1][0], dic['
                        machineUserFlangePose02'][1][0],
211                     dic['machineUserFlangePose03'][1][0], dic['
                            machineUserFlangePose04'][1][0],
212                     dic['machineUserFlangePose05'][1][0], dic['
                            machineUserFlangePose06'][1][0])
213                 # 打印当前是否处于急停状态
```

```
214                 print(dic["emergencyStopState"][1][0])
215                 # 打印tcp运动速度
216                 print(dic["tcp_speed"][1][0])
217                 # 打印关节运动下各关节运动速度
218                 print(dic['joint_speed01'][1][0], dic['joint_speed02'
                        ][1][0],
219                       dic['joint_speed03'][1][0], dic['joint_speed04'
                          ][1][0],
220                       dic['joint_speed05'][1][0], dic['joint_speed06'
                          ][1][0],
221                       dic['joint_speed07'][1][0], dic['joint_speed08'
                          ][1][0])
222                 # 打印tcp加速度
223                 print(dic["tcpacc"][1][0])
224                 # 打印关节运动下各关节加速度
225                 print(dic['jointacc01'][1][0], dic['jointacc02'
                        ][1][0],
226                       dic['jointacc03'][1][0], dic['jointacc04'
                          ][1][0],
227                       dic['jointacc05'][1][0], dic['jointacc06'
                          ][1][0],
228                       dic['jointacc07'][1][0], dic['jointacc08'
                          ][1][0])
229                 # 打印温度
230                 print(dic['joint_temperature01'][1][0], dic['
                        joint_temperature02'][1][0],
231                       dic['joint_temperature03'][1][0], dic['
                          joint_temperature04'][1][0],
232                       dic['joint_temperature05'][1][0], dic['
                          joint_temperature06'][1][0])
233                 # 打印输出扭矩
234                 print(dic['joint_torque01'][1][0], dic['
                        joint_torque02'][1][0],
235                       dic['joint_torque03'][1][0], dic['
                          joint_torque04'][1][0],
236                       dic['joint_torque05'][1][0], dic['
                          joint_torque06'][1][0])
237                 # 打印外部关节扭矩值
238                 print(dic['exjoint_torques01'][1][0], dic['
                        exjoint_torques02'][1][0],
239                       dic['exjoint_torques03'][1][0], dic['
```

```
                        exjoint_torques04'][1][0],
240                     dic['exjoint_torques05'][1][0], dic['
                    exjoint_torques06'][1][0])
241                 # 打印当前工具坐标系下的外部末端力/力矩估计值
242                 print(dic['extcpforceintool01'][1][0], dic['
                    extcpforceintool02'][1][0],
243                     dic['extcpforceintool03'][1][0], dic['
                        extcpforceintool04'][1][0],
244                     dic['extcpforceintool05'][1][0], dic['
                    extcpforceintool06'][1][0])
245                 # 打印拖动使能状态
246                 print(dic["dragState"][1][0])
247                 # 打印力控传感器连接状态
248                 print(dic["sensor_connected_state"][1][0])
249             index = index +1
250             outoput = ""
251             dic = {}
252             data = ""
253             time.sleep (0.008)
254     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 ROBOTS
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：technical@elibot.cn

- **苏州公司（生产基地）**
  苏州市工业园区长阳街 259 号中新钟园工业坊 4 栋
  +86-400-189-9358
- **北京公司**
  北京市经济技术开发区荣华南路 2 号院 6 号楼 1102 室
- **上海公司（研创中心）**
  上海市浦东新区张江科学城学林路 36 弄 18 号
- **深圳公司**
  深圳市宝安区航空路泰华梧桐岛科技创新园 1A 栋 202 室
- **美国公司**
  10521 Research Dr., Ste. 104, 37932, Knoxville, TN (USA)
- **德国公司**
  Münchener Str. 53, 85290, Geisenfeld, Bavaria (Germany)
- **日本公司**
  TOSHIN Hirokoji Honmachi Bldg., 1F, 2-4-3 Sakae, Naka-ku, 460-0008, Nagoya (Japan)
- **墨西哥公司**
  Calzada del pedregal 523, fraccionamiento el pedregal

关注公众号了解更多