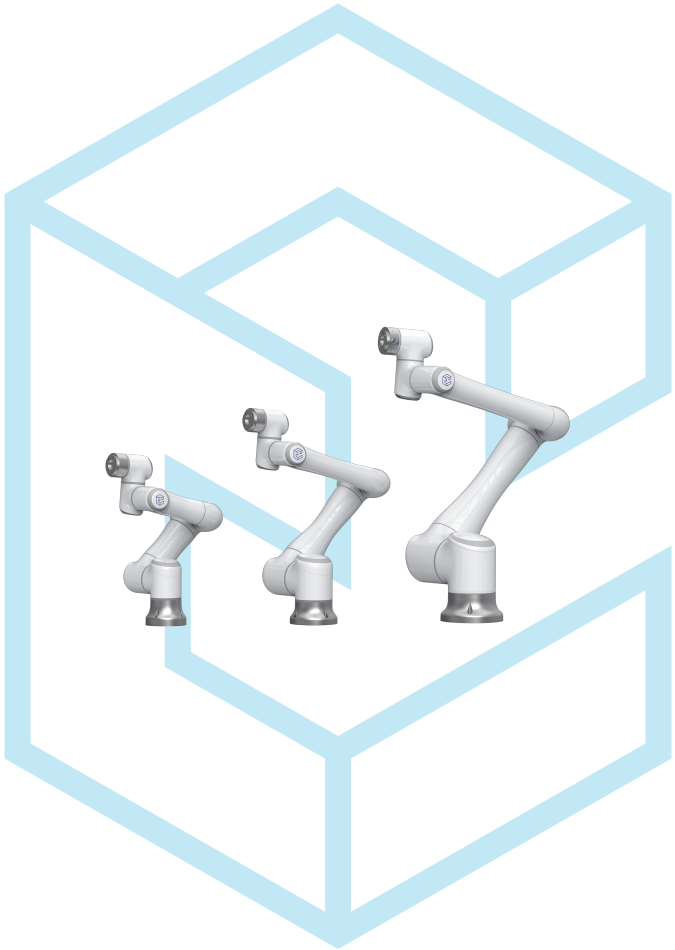
ELITE ROBOT EC Series Programming Manual



SDK-Socket Script Manual

2021-08-06

Version: 6.0.14

Contents

1 Introduction **1**

2 Control Interface **2**

2.1 Python data processing example 3

2.2 Interface service 4

2.2.1 Servo Service(ServoService) 4

2.2.1.1 Get the servo status of the robotic arm 4

2.2.1.2 Set servo status of robotic arm 5

2.2.1.3 Synchronous Servo Encoder Data 5

2.2.1.4 Clear alarm 6

2.2.1.5 Get synchronization status 6

2.2.2 Parameter Service(ParamService) 7

2.2.2.1 Get robot status 7

2.2.2.2 Get Robot Mode 7

2.2.2.3 Get information about the current position of the robot 8

2.2.2.4 Get the current pose information of the robot 8

2.2.2.5 Get robot motor speed 9

2.2.2.6 Get the current coordinate system of the robot 10

2.2.2.7 Get robot cycle mode 11

2.2.2.8 Get the current job running line number of the robot 11

2.2.2.9 Get the current encoder value list of the robot 12

2.2.2.10 Get the current tool number of the robot 13

2.2.2.11 Switch the current tool number of the robot 13

2.2.2.12 Get the current user coordinate number of the robot 14

2.2.2.13 Switch the current user coordinate number of the robot 14

2.2.2.14 Get the current torque information of the robot 15

2.2.2.15 Get the current running point number of the trajectory 15

2.2.2.16 Get analog input 16
2.2.2.17 Set analog output 17
2.2.2.18 Specify coordinate system 18
2.2.2.19 Drag the teaching switch 18
2.2.2.20 Set robot arm load and center of gravity 19
2.2.2.21 Set Robotic Arm Tool Center 19
2.2.2.22 Get collision status 20
2.2.2.23 Get user coordinate system data 20
2.2.2.24 Specify cycle mode 21
2.2.2.25 Set user coordinate system data 22
2.2.2.26 Get tool coordinate system data 23
2.2.2.27 Get tool load quality 24
2.2.2.28 Get tool centroid 24
2.2.2.29 Get robot type 25
2.2.2.30 Get robot DH parameters 25
2.2.2.31 Set collision enable 26
2.2.2.32 Set collision sensitivity 26
2.2.2.33 Set security parameters 27
2.2.2.34 Get robot running speed 28
2.2.2.35 Clear collision status 29
2.2.2.36 Get the current tool number of the robot in remote mode 29
2.2.2.37 Set the current tool number of the robot in remote mode 30
2.2.2.38 Get the center pose of flange in Cartesian coordinate system 31
2.2.2.39 Get the center pose of the flange in the user coordinate system . . . 31
2.2.2.40 Set the value of the P variable 32
2.2.2.41 Get robot subtype 32
2.2.2.42 Get security parameter enable status 33
2.2.2.43 Get safe power 33
2.2.2.44 Get Safe Momentum 33
2.2.2.45 Access to safety tools 34

2.2.2.46 Get safe elbow force 34
2.2.2.47 Get speed percentage 35
2.2.2.48 Get the maximum starting speed of dragging 35
2.2.2.49 Get the maximum torque error percentage 36
2.2.2.50 Set end button state 36
2.2.2.51 Get end button status 37
2.2.3 Movement Service(MovementService) 37
2.2.3.1 Joint Movement 37
2.2.3.2 Linear motion 40
2.2.3.3 Circular Movement 42
2.2.3.4 Rotation Movement 44
2.2.3.5 Add waypoint information 2.0 46
2.2.3.6 Clear waypoint information 2.0 48
2.2.3.7 Track Movement 2.0 48
2.2.3.8 Jog Movement 49
2.2.3.9 Stop robot operation 50
2.2.3.10 Robot runs automatically 51
2.2.3.11 Robot stop 51
2.2.3.12 Check if the jbi file exists 52
2.2.3.13 Run jbi file 52
2.2.3.14 Get jbi file running status 53
2.2.3.15 Set robot running speed 54
2.2.3.16 Joint motion at uniform speed 55
2.2.3.17 Linear uniform motion 56
2.2.3.18 Linear motion under specified coordinate system 56
2.2.4 Kinematics Service(KinematicsService) 57
2.2.4.1 Inverse solution function 57
2.2.4.2 Positive solution function 58
2.2.4.3 Base coordinate to user coordinate pose conversion 58
2.2.4.4 User coordinate to base coordinate pose conversion 59

2.2.4.5 Inverse solution function 2.0, with reference point position 60
2.2.4.6 Position multiplication 60
2.2.4.7 Pose inversion 61
2.2.5 IO Service(IOService) 61
2.2.5.1 Get input IO status 61
2.2.5.2 Get output IO status 62
2.2.5.3 Set output IO status 62
2.2.5.4 Get virtual input IO status 63
2.2.5.5 Get virtual output IO status 63
2.2.5.6 Set virtual output IO status 64
2.2.5.7 Read multiple M virtual IO 64
2.2.6 Variable service(VarService) 65
2.2.6.1 Get system B variable value 65
2.2.6.2 Set system B variable value 66
2.2.6.3 Get system I variable value 66
2.2.6.4 Set system I variable value 67
2.2.6.5 Get system D variable value 67
2.2.6.6 Set system D variable value 68
2.2.6.7 Get whether the system P variable is enabled 68
2.2.6.8 Get the value of P variable 69
2.2.6.9 Set the scope of the V variable 69
2.2.6.10 Get the value of V variable 70
2.2.6.11 Set the value of the V variable 70
2.2.6.12 Save variable data 71
2.2.7 Transparent Transmission Service 71
2.2.7.1 Initialize transparent transmission service 71
2.2.7.2 Set the current transparent transmission servo target joint point . . 72
2.2.7.3 Add transparent transmission of servo target joint point information to the cache 73
2.2.7.4 Clear the transparent transmission cache 75

2.2.7.5 Get whether the current robot is in a transparent transmission state 75
2.2.7.6 Example . . . 76
2.2.8 System Service(SystemService) . . . 78
2.2.8.1 Get the controller software version number . . . 78
2.2.8.2 Get servo version number . . . 79
2.3 Examples . . . 79
2.3.1 Example 1 . . . 79
2.3.2 Example 2 . . . 82
2.3.3 Example 3 . . . 84

3 Monitor Interface 87
3.1 Data description list of monitor interface . . . 87
3.2 Example . . . 89

4 Log Interface 94
4.1 Example . . . 94

5 Raw Log Interface 95
5.1 Example . . . 95

Chapter 1 Introduction

Elite Robot has opened robot controller ports to support users in secondary development, as shown in **Table** 1-1 .

Table 1-1 . Controller port

Port Number	Name	Function
8055	Control Interface	receive json string in specified format
8056	Monitor Interface	output robot information

The user can connect to the corresponding controller port through socket communication to perform some operations to realize the corresponding function.

Chapter 2 Control Interface

The user can send a json string in the specified format to the control interface of the controller through socket communication to achieve related functions, as shown below.

```
send

{"jsonrpc":"2.0","method":"method name","params":parameters,"id":id}

receive

  Normal
    {"jsonrpc":"2.0","result":result,"id":id}
  Error
    {"jsonrpc":"2.0","error":{"code":error code,"message":"error
       message"},"id":id}
```

CAUTION

This function is applicable to 2.13.0 and above versions.

The id when sending the json string is the same as the id when receiving the result, as shown below.

```
Send
    {"jsonrpc":"2.0","method":"cmd_set_payload","params":{"point"
       :[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}


  Receive
    Normal
      {"jsonrpc":"2.0","result":"false","id":1}

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

NOTICE

Both sending and receiving end with \n

Currently, there are two common exceptions returned by the json protocol:

JRPC_METHOD_NOT_FOUND -32601, JRPC_INTERNAL_ERROR -32693.

- 32601 means that the corresponding interface is not found. You need to check whether the interface name is correct or whether the current version supports the interface.
- 32693 is an exception defined inside the interface. Corresponding parameters are not found, parameters are out of range, and execution conditions are not met. Such exceptions are reported. Such errors only need to check whether the parameters and their ranges and execution conditions are satisfied according to the error information.

2.1 Python data processing example

The examples in this chapter are all in Python. Users can modify the code according to the examples in this section.

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

```
22    if(not params):
23      params=[]
24    else:
25      params=json.dumps(params)
26    sendStr="{{\"method\":\"{0}\",\"params\":{1},\"jsonrpc\":\"2.0\",\"id
        \":{2}}}".format(cmd,params,id)+"\n"
27    try:
28      sock.sendall(bytes(sendStr,"utf-8"))
29      ret=sock.recv(1024)
30      jdata=json.loads(str(ret,"utf-8"))
31      if("result" in jdata.keys()):
32        return (True,json.loads(jdata["result"]),jdata["id"])
33      elif("error" in jdata.keys()):
34        return (False,jdata["error"],jdata["id"])
35      else:
36        return (False,None,None)
37    except Exception as e:
38      return (False,None,None)
39
40  if __name__ == "__main__":
41    # Robot IP address
42    robot_ip="192.168.1.200"
43    conSuc,sock=connectETController(robot_ip)
44    if(conSuc):
45      # Get robot status
46      suc, result, id = sendCMD(sock, "getRobotState")
47      # Print results
48      print(result)
```

2.2 Interface service

2.2.1 Servo Service(ServoService)

2.2.1.1 Get the servo status of the robotic arm

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

Function: Get the servo status of the robotic arm

Parameter: None

Return: Enable true, not enable false

Example:

```
if __name__ == "__main__":
  # Robot IP address
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if(conSuc):
    # Get the servo status of the robotic arm
    suc,result,id=sendCMD(sock,""getServoStatus)
```

2.2.1.2 Set servo status of robotic arm

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

Function: Set servo enable state

Parameter: status: servo switch, range: int[0,1], 1 is on, 0 is off

Return: True for success, false for failure

Example:

```
if __name__ == "__main__":
  # Robot IP address
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if(conSuc):
    # Get the servo status of the robotic arm
    suc,result,id=sendCMD(sock, "getServoStatus")
    if(result == 0):
      # Set the servo status of the robot arm to ON
      suc,result,id=sendCMD(sock,"set_servo_status",{"status":1})
      time.sleep(1)
```

Note: This command is only supported in remote mode.

2.2.1.3 Synchronous Servo Encoder Data

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

Function: Synchronize servo encoder data

Parameter: None

Return: True for success, false for failure

Example:

```
if __name__ == "__main__":
  # Robot IP address
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if(conSuc):
    # Get synchronization status
    suc, result, id = sendCMD(sock, "getMotorStatus")
    if(result == 0):
      # Synchronize servo encoder data
      suc, result, id = sendCMD(sock, "syncMotorStatus")
      time.sleep(0.5)
```

Note: This command is only supported in remote mode.

2.2.1.4 Clear alarm

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

Function: Clear alarm

Parameter: None

Return: True for success, false for failure

Example:

```
if __name__ == "__main__":
  # Robot IP address
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if(conSuc):
    # Clear alarm
    ret, result, id = sendCMD(sock, "clearAlarm")
```

Note: This command is only supported in remote mode.

2.2.1.5 Get synchronization status

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

Function: Get synchronization status

Parameter: None

Return: true for unsynchronized, false for synchronization

Example:

```
if __name__ == "__main__":
  # Robot IP address
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if(conSuc):
    # Get synchronization status
    suc, result, id = sendCMD(sock, "getMotorStatus")
```

2.2.2 Parameter Service(ParamService)

2.2.2.1 Get robot status

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

Function: Get robot status

Parameter: None

Return: Stop state 0, pause state 1, emergency stop state 2, running state 3, error state 4, collision state 5

Example:

```
if __name__ == "__main__":
  # Robot IP address
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if(conSuc):
    # Get robot status
    suc, result, id = sendCMD(sock, "getRobotState")
```

2.2.2.2 Get Robot Mode

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

Function: Get Robot Mode

Parameter: None

Return: Teaching mode 0, operating mode 1, remote mode 2

Example:

```
if __name__ == "__main__":
  # Robot IP address
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if(conSuc):
    # Get Robot Mode
    suc, result, id = sendCMD(sock, "getRobotMode")
```

2.2.2.3 Get information about the current position of the robot

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

Function: Get information about the current position of the robot

Parameter:

Return: The current position information of the robot double pos[8]

Example:

```
if __name__ == "__main__":
  # Robot IP address
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if(conSuc):
    # Get information about the current position of the robot
    suc, result, id = sendCMD(sock, "getRobotPos")
```

2.2.2.4 Get the current pose information of the robot

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

Function: Get the current pose information of the robot

Parameter: coordinate_num: coordinate number; int[-1,7], -1: base coordinate, 0~7: corresponding user coordinate
tool_num: tool number: int[-1,7], -1: current tool number, 0~7: corresponding tool number

Return: The current pose information of the robot double pose[6]

Example:

```
if __name__ == "__main__":
  # Robot IP address
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if(conSuc):
    # Get the current pose information of the robot
    suc,result,id=sendCMD(sock,"getRobotPose",{"coordinate_num": 0,
        "tool_num": 0})
```

Note: The parameters coordinate_num and tool_num are only applicable to v2.16.2 version and above. Without parameters, the robot pose under the base frame will be returned by default.

2.2.2.5 Get robot motor speed

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

Function: Get robot motor speed

Parameter: None

Return: Robot motor speed double speed[8]

Example:

```
if __name__ == "__main__":
  # Robot IP address
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  point = []
  point.append([0.0065,-103.9938,102.2076,-88.2138,
  90.0000,0.0013,0.0000,0.0000])
  point.append([-16.2806,-82.4996,81.9848,-89.4851,
  90.0000,-16.2858,0.0000,0.0000])
  point.append([3.7679, -71.7544, 68.7276, -86.9732,
  90.0000, 3.7627, 0.0000, 0.0000])
  point.append([12.8237,-87.3028,87.2361,-89.9333,
  90.0000,12.8185,0.0000,0.0000])
  if(conSuc):
    for i in range(0, 4, 1):
      # Joint movement
      suc,result,id=sendCMD(sock,"moveByJoint",{"targetPos":point[i
          ],"speed":30})
      while(True):
        # Get robot motor speed
        suc,result,id =sendCMD(sock,"getMotorSpeed")
        print(result)
        # Get robot status
        suc,result,id=sendCMD(sock,"getRobotState")
        if(result == 0):
          break
```

2.2.2.6 Get the current coordinate system of the robot

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

Function: Get the current coordinate system of the robot

Parameter: None

Return: Joint 0, Right Angle 1, Tool 2, User 3, Cylinder 4

Example:

```
if __name__ == "__main__":
  # Robot IP address
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if(conSuc):
    # Get the current coordinate system of the robot
    suc, result, id = sendCMD(sock, "getCurrentCoord")
```

2.2.2.7 Get robot cycle mode

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

Function: Get robot cycle mode

Parameter: None

Return: Single step 0, single loop 1, continuous loop 2

Example:

```
if __name__ == "__main__":
  # Robot IP address
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if(conSuc):
    # Get robot cycle mode
    suc, result, id = sendCMD(sock, "getCycleMode")
```

2.2.2.8 Get the current job running line number of the robot

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

Function: Get the current job running line number of the robot

Parameter: None

Return: JBI line number

Example:

```
if __name__ == "__main__":
  # Robot IP address
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if(conSuc):
    # Get the servo status of the robotic arm
    suc,result,id=sendCMD(sock, "getServoStatus")
    if(result == 0):
      # Set the servo status of the robot arm to ON
      suc,result,id=sendCMD(sock,"set_servo_status",{"status":1})
      time.sleep(1)
    # Check if the jbi file exists
    suc,result,id=sendCMD(sock,"checkJbiExist",{"filename":
        jbi_filename})
    if(suc and result==1):
      # Run jbi file
      suc,result,id=sendCMD(sock,"runJbi",{"filename":jbi_filename
          })
      if(suc and result):
        checkRunning=3
        while(checkRunning==3):
          # Get jbi file running status
          suc,result,id=sendCMD(sock,"getJbiState")
          checkRunning=result["runState"]
          # Get the current job running line number of the robot
          # The line number needs to count the line number of the
              point information, not the line number of the teach
              pendant program
          suc,result,id=sendCMD(sock,"getCurrentJobLine")
          print(result)
          time.sleep(0.1)
```

2.2.2.9 Get the current encoder value list of the robot

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

Function: Get the current encoder value list of the robot

Parameter:

Return: The current encoder value list of the robot double encode[8]

Example:

```
if __name__ == "__main__":
  # Robot IP address
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if(conSuc):
    # Get the current encoder value list of the robot
    suc, result, id = sendCMD(sock, "getCurrentEncode")
```

2.2.2.10 Get the current tool number of the robot

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

Function: Get the current tool number of the robot

Parameter: None

Return: The current tool number of the robot, range: 0~7

Example:

```
if __name__ == "__main__":
  # Robot IP address
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if(conSuc):
    # Get the current tool number of the robot
    suc, result, id = sendCMD(sock, "getToolNumber")
```

2.2.2.11 Switch the current tool number of the robot

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

Function: Switch the current tool number of the robot

Parameter: tool_num: Tool number, range: int[0,7]

Return: True for success, false for failure

Example:

```
if __name__ == "__main__":
  # Robot IP address
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if(conSuc):
    # Switch the current tool number of the robot
    suc,result,id=sendCMD(sock,"setToolNumber",{"tool_num":7})
    time.sleep(0.5)
    # Get the current tool number of the robot
    suc, result, id = sendCMD(sock, "getToolNumber")
```

Note: This command can only switch the current tool number in teaching mode.

2.2.2.12 Get the current user coordinate number of the robot

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

Function: Get the current user coordinate numberr of the robot

Parameter: None

Return: Coordinate number of the current user of the robot, range: 0~7

Example:

```
if __name__ == "__main__":
  # Robot IP address
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if(conSuc):
    # Get the current user coordinate number of the robot
    suc, result, id = sendCMD(sock, "getUserNumber")
```

2.2.2.13 Switch the current user coordinate number of the robot

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

Function: Switch the current user coordinate number of the robot

Parameter: user_num: user coordinate number, range: int[0,7]

Return: True for success, false for failure

Example:

```
if __name__ == "__main__":
  # Robot IP address
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if(conSuc)::
    # Switch the current user coordinate number of the robot
    suc,result,id=sendCMD(sock,"setUserNumber",{"user_num":7})
    time.sleep(0.5)
    # Get the current user coordinate number of the robot
    suc, result, id = sendCMD(sock, "getUserNumber")
```

Note: This command is only supported in remote mode.

2.2.2.14 Get the current torque information of the robot

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

Function: Get the current torque information of the robot

Parameter: None

Return: Robot current torque information double torques[8]m, The ratio per thousand of the ratedtorque of the joint, unit ‰.

Example:

```
if __name__ == "__main__":
  # Robot IP address
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if(conSuc):
    # Get the current torque information of the robot
    suc, result, id = sendCMD(sock, "getRobotTorques")
```

2.2.2.15 Get the current running point number of the trajectory

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

Function: Get the current running point number of the robot

Parameter: None

Return: Store the serial number of the current running point, -1 is non-waypoint movement

Example:

```
if __name__ == "__main__":
  # Robot IP address
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  C000 = [0.0065,-103.9938,102.2076,-88.2138,
  90.0000,0.0013,0.0000,0.0000]
  C001 = [-16.2806,-82.4996,81.9848,-89.4851,
  90.0000, -16.2858, 0.0000, 0.0000]
  C002 = [3.7679, -71.7544, 68.7276, -86.9732,
  90.0000, 3.7627, 0.0000, 0.0000]
  if(conSuc):
    # Clear waypoint information 2.0
    suc, result, id = sendCMD(sock, "clearPathPoint")
    if(result == True):
      # Add waypoint information 2.0
      suc, result, id = sendCMD(sock, "addPathPoint", {"wayPoint":
          C000,"moveType": 0, "speed": 50, "smooth": 7})
      suc, result, id = sendCMD(sock, "addPathPoint", {"wayPoint":
          C001,"moveType":0, "speed": 50, "smooth": 7})
      suc, result, id = sendCMD(sock, "addPathPoint", {"wayPoint":
          C002,"moveType": 0, "speed": 50, "smooth": 7})
      # Trajectory movement 2.0
      suc, result, id = sendCMD(sock, "moveByPath")
      while(True):
        # Get the line number of trackfile file running (consistent
            with the line number displayed by the teach pendant)
        suc, result, id = sendCMD(sock, "getPathPointIndex")
        print(result)
        # Get robot status
        suc, result, id = sendCMD(sock, "getRobotState")
        if(result == 0):
          break
```

2.2.2.16 Get analog input

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

Function: Get analog input

Parameter: addr: analog address, range: 0~2

Return: Analog value, range: -10~10

Example:

```
if __name__ == "__main__":
  # Robot IP address
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if(conSuc):
    for i in range(0, 2, 1):
      # Get analog input
      suc , result , id = sendCMD(sock, "getAnalogInput", {"addr":i
          })
```

2.2.2.17 Set analog output

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

Function: Set analog output

Parameter: addr: analog address, range: 0~4
value: analog value, range: double[-10,10]

Return: True for success, false for failure

Example:

```
if __name__ == "__main__":
  # Robot IP address
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if(conSuc):
    # Set analog output
    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})
```

Note: This command is only supported in remote mode.

2.2.2.18 Specify coordinate system

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

Function: Specify coordinate system

Parameter: coord_mode: coordinate system,range int[0,4]. joint: 0, right angle: 1, tool: 2, user: 3, cylinder: 4

Return: True for success, false for failure

Example:

```
if __name__ == "__main__":
  # Robot IP address
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if(conSuc):
    for i in range(0, 5, 1)
      # Specify coordinate system
      suc,result,id=sendCMD(sock,"setCurrentCoord",{"coord_mode":i
          })
      time.sleep(0.5)
      # Specify coordinate system
      suc,result,id=sendCMD(sock,"getCurrentCoord")
      print(result)
```

Note: This command is only supported in remote mode.

2.2.2.19 Drag the teaching switch

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

Function: Drag the teaching switch

Parameter: switch: switch, range: int[0,1],0 is off, 1 is on

Return: True for success, false for failure

Example:

```
if __name__ == "__main__":
  # Robot IP address
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if(conSuc):
    # Drag the teaching switch
    suc,result,id=sendCMD(sock,"drag_teach_switch",{"switch":1})
```

Note: This command is only supported in remote mode.

2.2.2.20 Set robot arm load and center of gravity

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

Function: Set robot arm load and center of gravity

Parameter: tool_num: Tool number, range: int[0,7]
m: load weight, unit Kg, range: EC63 0~3.6 EC66 0~7.2 EC612 0~14.4
point: center of gravity, x, y, z, unit mm, range: double[-5000,5000]

Return: True for success, false for failure

Example:

```python
if __name__ == "__main__":
  # Robot IP address
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if(conSuc):
    # Set robot arm load and center of gravity
    suc, result, id = sendCMD(sock, "cmd_set_payload",{"tool_num"
        :0, "m":5, "point":[10, 20, 30]})
```

Note: Version 2.16.2 and above support the selection of tool number. The value of tool_num is set as the load and center of gravity of the tool.
This command is only supported in remote mode.

2.2.2.21 Set Robotic Arm Tool Center

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

Function: Set Robotic Arm Tool Center

Parameter: tool_num: tool number, range: int[0,7]
point: tool center, the first three items are in millimeters, the range: double[-500,500], the last three items are in radians, the range: double[-π,π]

Return: True for success, false for failure

Example:
```
if __name__ == "__main__":
  # Robot IP address
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if(conSuc):
      # Set Robotic Arm Tool Center
      suc,result,id=sendCMD(sock,"cmd_set_tcp",{"point"
          :[100,100,100,0,0,0],"tool_num":0})
```

Note: This command is only supported in remote mode.

2.2.2.22 Get collision status

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

Function: Get collision status

Parameter: None

Return: Collision: 1, No collision: 0

Example:
```
if __name__ == "__main__":
  # Robot IP address
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if(conSuc):
    # Get collision status
    suc, result, id = sendCMD(sock, "getCollisionState)
```

Note: This command is only supported in remote mode.

2.2.2.23 Get user coordinate system data

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

Function: Get user coordinate system data

Parameter: user_num: User coordinate number, range: int[0,7]
unit_type: optional parameter, returns the unit type of pose rx, ry, rz, range int [0,1]
Default: radians, 0: angle, 1: radians

Return: User coordinate system data double pose[6]

Example:

```
if __name__ == "__main__":
  # Robot IP address
  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("User coordinate number =",i)
      print("suc=",suc,"","id =",id)
      if (suc):
        print("result=", result)
      else:
        print("err_msg=",result["message"])
```

Note: The unit_type parameter is only applicable to v2.15.2 and above.

2.2.2.24 Specify cycle mode

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

Function: Specify cycle mode

Parameter: cycle_mode: cycle mode, range: int[0,2] single step: 0, single cycle: 1, continuous cycle: 2

Return: Success true, failure false

Example:

```
if __name__ == "__main__":
  # Robot IP address
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if(conSuc):
    #Set the loop mode to single loop
    ret, result, id = sendCMD(sock, "setCycleMode",{"cycle_mode"
        :1})
    if ret:
      print("result=", result)
    else:
      print("err_msg=", result["message"])
```

Note: This command is only supported in remote mode.

2.2.2.25 Set user coordinate system data

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

Function: Set user coordinate system data

Parameter: user_num: user number, range int [0~7]
user_frame: user coordinate system data double user_frame[6]
unit_type: optional parameter, rx, ry, rz unit type, range int [0,1]
Default: radians, 0: angle, 1: radians

Return: True for success, false for failure

Example:

```
if __name__ == "__main__":
  # Robot IP address
  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"])
```

Note: This command is only supported in remote mode.
The unit_type parameter is only applicable to v2.15.2 and above.

2.2.2.26 Get tool coordinate system data

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

Function: Get tool coordinate system data

Parameter: tool_num: tool coordinate number, range int [0~7]

Return: Tool coordinate system data double pose[6], rx,ry,rz are angles

Example:

```
if __name__ == "__main__":
  # Robot IP address
  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"])
```

2.2.2.27 Get tool load quality

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

Function: Get tool load quality

Parameter: tool_num: tool coordinate number, range int [0~7]

Return: Tool load quality, double m

Example:

```
if __name__ == "__main__":
  # Robot IP address
  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"])
```

2.2.2.28 Get tool centroid

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

Function: Get tool centroid

Parameter: tool_num: tool coordinate number, range int [0~7]

Return: Tool load quality, double centre_of_mass[3]

Example:

```
if __name__ == "__main__":
  # Robot IP address
  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"])
```

2.2.2.29 Get robot type

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

Function: Get robot type

Parameter: no

Return: Robot type int 62 (six-axis collaborative robot), 60 (vertical multi-joint series robot), 41 (four-axis rotary joint robot), 40 (palletizing robot), 43 (SCARA robot), 30 (Delta parallel robot)

Example:

```
if __name__ == "__main__":
  # Robot IP address
  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"])
```

2.2.2.30 Get robot DH parameters

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

Function: Get robot DH parameters

Parameter: index: range int [0,11], corresponding to connecting rod parameter d1~d12

Return: DH parameters

Example:

```
if __name__ == "__main__":
  # Robot IP address
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if(conSuc):
    # Get the value of connecting rod parameter d1
    ret, result, id = sendCMD(sock, "getDH", {"index":0})
    if ret:
      print("result=", result)
    else:
      print("err_msg=", result["message"])
```

2.2.2.31 Set collision enable

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

Function: Set collision enable

Parameter: enable: 1: turn on the collision switch, 0: turn off the collision switch

Return: True for success, False for failure

Example:

```
if __name__ == "__main__":
  # Robot IP address
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if(conSuc):
    # Turn on the collision switch
    ret, result, id = sendCMD(sock, "setCollisionEnable", {"enable"
        : 1})
    if ret:
      print("result=", result)
    else:
      print("err_msg=", result["message"])
```

Note: This command is only supported in remote mode.

2.2.2.32 Set collision sensitivity

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

Function: Set collision sensitivity

Parameter: value: sensitivity range int [10,100]

Return: True for success, False for failure

Example:

```
if __name__ == "__main__":
  # Robot IP address
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if(conSuc):
    # Set collision sensitivity to 50%
    ret, result, id = sendCMD(sock, "setCollisionSensitivity", {"
        value": 50})
    if ret:
      print("result=", result)
    else:
      print("err_msg=", result["message"])
```

Note: This command is only supported in remote mode.

2.2.2.33 Set security parameters

```
{"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},"id
   ":id}
```

Function: Set security parameters

Parameter: password: Temporary plain code, the value is 123456
enable: Safety restriction parameter enable, 1: enable, 0: disable
mode: mode, 0: normal mode, 1: reduced mode
power: power, range: double [80,1500]
momentum: momentum, range: double [5,90]
tool_force: Tool force, range: double [100,400]
elbow_force: elbow force: double [100,400]
speed: speed percentage, double [0-100]

Return: True for success, False for failure

Example:

```
if __name__ == "__main__":
  # Robot IP address
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if(conSuc):
    ret,result,id=sendCMD(sock,"setSafetyParams",{"enable":1,"mode"
        :1,"power":80,"momentum":5,"tool_force":100,"elbow_force"
        :100,"speed":20})
    if ret:
      print("result=", result)
    else:
      print("err_msg=", result["message"])
```

Note: This command is only supported in remote mode.
The password parameter is only applicable to v2.16.2 and above.

2.2.2.34 Get robot running speed

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

Function: Get the robot's automatic speed

Parameter: no

Return: Automatic speed double

Example:

```
if __name__ == "__main__":
  # Robot IP address
  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 Clear collision status

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

Function: Clear collision status

Parameter: no

Return: True for success, False for failure

Example:

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

2.2.2.36 Get the current tool number of the robot in remote mode

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

Function: Get the current tool number of the robot in remote mode

Parameter: None

Return: The current tool number of the robot in remote mode, range: 0~7
Note: The tool number in automatic mode is the same as the tool number in remote mode.

Example:

```
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"])
```

Note: This command is applicable to v2.14.4 and above.

2.2.2.37 Set the current tool number of the robot in remote mode

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

Function: Get the current tool number of the robot in remote mode

Parameter: tool_num: tool number, range 0~7

Return: True for success, False for failure

Example:

```
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"])
```

Note: This command is only supported in remote mode.
This command is applicable to v2.14.4 and above.

2.2.2.38 Get the center pose of flange in Cartesian coordinate system

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

Function: Get the center pose of the flange in the Cartesian coordinate system

Parameter: None

Return: The center pose of the flange in the Cartesian coordinate system, Double[6]

Example:

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

Note: This command is applicable to v2.14.4 and above.

2.2.2.39 Get the center pose of the flange in the user coordinate system

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

Function: Get the center pose of the flange in the Cartesian coordinate system

Parameter: None

Return: Center pose of flange in user coordinate system, Double[6]

Example:

```
if __name__ == "__main__":
  ip = "192.168.1.205"
  conSuc, sock = connectETController(ip))
  if conSuc:
    ret, result, id = sendCMD(sock, "get_user_flange_pose")
    if ret:
      print("result = ", result)
    else:
      print("err_msg = ", result["message"])
```

Note: This command is applicable to v2.14.4 and above.

2.2.2.40 Set the value of the P variable

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

Function: Set the value of the system P variable

Parameter: addr: variable address, range int[0,255]
pos: the value of the p variable, double pos[8], range [-360,360]

Return: True for success, False for failure

Example:

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

Note: This command is only supported in remote mode.
This command is applicable to v2.15.2 and above.

2.2.2.41 Get robot subtype

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

Function: Get robot subtype

Parameter: no

Return: Robot subtype int

Example:

```
if __name__ == "__main__":
  ip = "192.168.1.202"
  conSuc,sock=connectETController(ip)
  if conSuc:
    # Get robot subtype
    suc,result,id=sendCMD(sock,"getRobotSubtype")
    print(result)
```

Note: This command is applicable to v2.16.2 and above.

2.2.2.42 Get security parameter enable status

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

Function: Get security parameter enable status

Parameter: no

Return: Close 0, open 1

Example:

```
if __name__ == "__main__":
    ip = "192.168.1.202"
    conSuc, sock = connectETController(ip)
    if conSuc:
        # Get security parameter enable status
        suc,result,id=sendCMD(sock,"getRobotSafetyParamsEnabled")
        print(result)
```

Note: This command is applicable to v2.16.2 and above.

2.2.2.43 Get safe power

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

Function: Get safe power

Parameter: no

Return: The power value in normal mode and reduced mode double

Example:

```
if __name__ == "__main__":
    ip = "192.168.1.202"
    conSuc, sock = connectETController(ip)
    if conSuc:
        # Get robot safety power
        suc,result,id=sendCMD(sock,"getRobotSafeyPower")
        print(result)
```

Note: This command is applicable to v2.16.2 and above.

2.2.2.44 Get Safe Momentum

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

Function: Gain safety momentum

Parameter: no

Return: Momentum value double in normal mode and reduced mode

Example:

```
if __name__ == "__main__":
    ip = "192.168.1.202"
    conSuc, sock = connectETController(ip)
    if conSuc:
        # Get robot safety momentum
        suc,result,id =sendCMD(sock,"getRobotSafetyMomentum")
        print(result)
```

Note: This command is applicable to v2.16.2 and above.

2.2.2.45 Access to safety tools

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

Function: Acquire the power of safety tools

Parameter: no

Return: Tool force double in normal mode and reduced mode

Example:

```
if __name__ == "__main__":
    ip = "192.168.1.202"
    conSuc, sock = connectETController(ip)
    if conSuc:
        # Acquire robot safety tool power
        suc,result,id =sendCMD(sock,"getRobotSafetyToolForce")
        print(result)
```

Note: This command is applicable to v2.16.2 and above.

2.2.2.46 Get safe elbow force

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

Function: Obtain safe elbow force

Parameter: no

Return: Elbow force in normal mode and reduced mode double

Example:

```
if __name__ == "__main__":
  ip = "192.168.1.202"
  conSuc, sock = connectETController(ip)
  if conSuc:
    # Obtain the robot safety elbow force
    suc,result,id =sendCMD(sock,"getRobotSafetyElbowForce")
    print(result)
```

Note: This command is applicable to v2.16.2 and above.

2.2.2.47 Get speed percentage

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

Function: Get the robot's speed percentage

Parameter: no

Return: Speed percentage in normal mode and reduced mode double

Example:

```
if __name__ == "__main__":
  ip = "192.168.1.202"
  conSuc, sock = connectETController(ip)
  if conSuc:
    # Get robot speed percentage
    suc,result,id =sendCMD(sock,"getRobotSpeedPercentage")
    print(result)
```

Note: This command is applicable to v2.16.2 and above.

2.2.2.48 Get the maximum starting speed of dragging

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

Function: Get the maximum start speed of drag

Parameter: no

Return: The maximum starting speed of the robot during dragging double

Example:

```
if __name__ == "__main__":
    ip = "192.168.1.202"
    conSuc, sock = connectETController(ip)
    if conSuc:
        # Get the maximum start speed of the drag
        suc,result,id =sendCMD(sock,"getRobotDragStartupMaxSpeed")
        print(result)
```

Note: This command is applicable to v2.16.2 and above.

2.2.2.49 Get the maximum torque error percentage

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

Function: Get the maximum torque error percentage of the machine

Parameter: no

Return: The percentage of maximum torque error during robot collision double

Example:

```
if __name__ == "__main__":
    ip = "192.168.1.202"
    conSuc, sock = connectETController(ip)
    if conSuc:
        # Get the maximum start speed of robot drag
        suc,result,id =sendCMD(sock,"getRobotTorqueErrorMaxPercents")
        print(result)
```

Note: This command is applicable to v2.16.2 and above.

2.2.2.50 Set end button state

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

Function: Set the end button state

Parameter: button_num: button, 0: blue button, 1: green button
state: state, 0: disable, 1: drag, 2: mark

Return: True for success, false for failure

Example:
```
if __name__ == "__main__":
  ip = "192.168.1.202"
  conSuc, sock = connectETController(ip)
  if conSuc:
    # Get the maximum start speed of robot drag
    suc,result,id =sendCMD(sock,"setFlangeButton",{"button_num":0,"
        state":1})
    print(result)
```

Note: This command is only supported in remote mode.
This command is applicable to v2.16.2 and above.

2.2.2.51 Get end button status

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

Function: Get the end button state

Parameter: button_num: button, 0: blue button, 1: green button

Return: Disable 0, drag 1, mark 2

Example:
```
if __name__ == "__main__":
  ip = "192.168.1.202"
  conSuc, sock = connectETController(ip)
  if conSuc:
    # Get the status of the end button
    suc,result,id=sendCMD(sock,"checkFlangeButtonFlangeButton,{"
        button_num":0})
␣␣␣␣print␣(suc,␣result,␣id)
```

Note: This command is applicable to v2.16.2 and above.

2.2.3 Movement Service(MovementService)

2.2.3.1 Joint Movement

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

Function: Joint movement

Parameter: targetpos: target joint point
speed: operating speed, range: double[0.01,100]
cond_type: IO type, 0 is digital input X, 1 is digital output Y, range int[0,1]
cond_num: IO address, range int[0,63]
cond_value: IO status, range int[0,1], when the actual IO status is consistent with this value, the unfinished movement is immediately abandoned and the next instruction is executed.
acc: acceleration percentage, range: int [1,100], optional parameter, default value 0 if not written.
dec: percentage of deceleration, range: int [1,100], optional parameter, default value 0 if not written

Return: True for success, false for failure

Example:

```
if __name__ == "__main__":
  # Robot IP address
  robot_ip="192.168.1.200"
  conSuc, sock = connectETController(robot_ip)
  point = []
  point.append([0.0065,-103.9938,102.2076,-88.2138,
  90.0000,0.0013,0.0000,0.0000])
  point.append([-16.2806,-82.4996,81.9848,-89.4851,
  90.0000, -16.2858, 0.0000, 0.0000])
  point.append([3.7679, -71.7544, 68.7276, -86.9732,
  90.0000, 3.7627, 0.0000, 0.0000])
  point.append([12.8237,-87.3028,87.2361,-89.9333,
  90.0000,12.8185,0.0000,0.0000])
  if(conSuc):
    # Get the servo status of the robotic arm
    suc, result, id = sendCMD(sock, "getServoStatus")
    if(result == 0):
      # Set the servo status of the robotic arm to ON
      suc, result, id = sendCMD(sock,"set_servo_status",{"status"
          :1})
      time.sleep(1)
      for i in range(4):
        # Joint movement
        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):
          # Get robot status
          suc, result, id = sendCMD(sock, "getRobotState")
          if (result == 0):
            break
```

2.2.3.2 Linear motion

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

Function: Linear motion

Parameter: targetpos: target joint point
speed: operating speed. double Type is linear speed, range: 1-3000; is rotation angular speed, range: 1-300; is absolute linear speed, range: linear minimum speed parameter value-linear maximum speed parameter value; absolute rotation angular speed, range: rotation angle minimum speed parameter value-rotation angle maximum speed parameter value
speed_type: Speed type, 0 is V (linear speed), 1 is VR (rotational angular speed), 2 is AV (absolute linear speed), and 3 is AVR (absolute rotation angular speed). Optional.
cond_type: IO type, 0 is digital input X, 1 is digital output Y, range int[0,1]
cond_num: IO address, range int[0,63]
cond_value: IO status, range int[0,1], when the actual IO status is consistent with this value, immediately abandon the unfinished movement and execute the next instruction
acc: acceleration percentage, range: int [1,100], optional parameter, default value 0 if not written.
dec: percentage of deceleration, range: int [1,100], optional parameter, default value 0 if not written

Return: True for success, false for failure

Example:

```
if __name__ == "__main__":
  # Robot IP address
  robot_ip="192.168.1.205"
  conSuc, sock = connectETController(robot_ip)
  point = []
  point.append([0.0065,-103.9938,102.2076,-88.2138,
  90.0000,0.0013,0.0000,0.0000])
  point.append([-16.2806,-82.4996,81.9848,-89.4851,
  90.0000, -16.2858, 0.0000, 0.0000])
  point.append([3.7679, -71.7544, 68.7276, -86.9732,
  90.0000, 3.7627, 0.0000, 0.0000])
  point.append([12.8237,-87.3028,87.2361,-89.9333,
  90.0000,12.8185,0.0000,0.0000])
  if(conSuc):
    # Set the servo status of the robotic arm to ON
    suc, result, id = sendCMD(sock,"set_servo_status",{"status":1})
    time.sleep(1)
    for i in range(4):
      # Linear motion
      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):
        # Get robot status
        suc, result, id = sendCMD(sock, "getRobotState")
        if (result == 0):
          break
```

Note: When there is no speed_type parameter, speed means absolute linear speed.

2.2.3.3 Circular Movement

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

Function: Circular motion

Parameter: middlepos: middle joint point
targetpos: target joint point
speed: operating speed. double Type is linear speed, range: 1-3000; is rotation angular speed, range: 1-300; is absolute linear speed, range: linear minimum speed parameter value-linear maximum speed parameter value; absolute rotation angular speed, range: rotation angle minimum speed parameter value-rotation angle maximum speed parameter value
speed_type: Speed type, 0 is V (linear speed), 1 is VR (rotational angular speed), 2 is AV (absolute linear speed), and 3 is AVR (absolute rotation angular speed). Optional.
cond_type: IO type, 0 is digital input X, 1 is digital output Y, range int[0,1]
cond_num: IO address, range int[0,63]
cond_value: IO status, range int[0,1], when the actual IO status is consistent with this value, the unfinished movement is immediately abandoned and the next instruction is executed.
acc: acceleration percentage, range: int [1,100], optional parameter, default value 0 if not written.
dec: percentage of deceleration, range: int [1,100], optional parameter, default value 0 if not written.

Return: True for success, false for failure

Example:

```
if __name__ == "__main__":
  # Robot IP address
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  P000 = [0.0065,-103.9938,102.2076,-88.2138,
  90.0000,0.0013,0.0000,0.0000]
  P001 = [-16.2806,-82.4996,81.9848,-89.4851,
  90.0000, -16.2858, 0.0000, 0.0000]
  if(conSuc):
    # Get the servo status of the robotic arm
    suc,result,id=sendCMD(sock,"getServoStatus")
    if(result == 0):
      # Set the servo status of the robotic arm to ON
      suc,result,id=sendCMD(sock,"set_servo_status",{"status":1})
      time.sleep(1)
    #Circular movement
    suc,result,id=sendCMD(sock,"moveByArc",{"midPos":P000,"
        targetPos":P001,"speed_type":0,"speed":20,"cond_type":0,"
        cond_num":7  ,"cond_value":1})
```

Note: When there is no speed_type parameter, speed means absolute linear speed.

2.2.3.4 Rotation Movement

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

Function: Rotational movement

Parameter: targetpos: target joint point
speed: operating speed. double Type is linear speed, range: 1-3000; is rotation angular speed, range: 1-300; is absolute linear speed, range: linear minimum speed parameter value-linear maximum speed parameter value; absolute rotation angular speed, range: rotation angle minimum speed parameter value-rotation angle maximum speed parameter value
speed_type: Speed type, 0 is V (linear speed), 1 is VR (rotational angular speed), 2 is AV (absolute linear speed), and 3 is AVR (absolute rotation angular speed). Optional.
cond_type: IO type, 0 is digital input X, 1 is digital output Y, range int[0,1]
cond_num: IO address, range int[0,63]
cond_value: IO status, range int[0,1], when the actual IO status is consistent with this value, the unfinished movement is immediately abandoned and the next instruction is executed.
acc: acceleration percentage, range: int [1,100], optional parameter, default value 0 if not written.
dec: percentage of deceleration, range: int [1,100], optional parameter, default value 0 if not written.

Return: True for success, false for failure

Example:

```
if __name__ == "__main__":
  # Robot IP address
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  P000 = [0.0065,-103.9938,102.2076,-88.2138,
  90.0000,0.0013,0.0000,0.0000]
  if(conSuc):
    # Get the servo status of the robotic arm
    suc,result,id=sendCMD(sock,"getServoStatus")
    if(result == 0):
      # Set the servo status of the robotic arm to ON
      suc,result,id=sendCMD(sock,"set_servo_status",{"status":1})
      time.sleep(1)
    # Rotational movement
    suc,result,id=sendCMD(sock,"moveByRotate",{"targetPos":P000,"
        speed_type":0,"speed":20,"cond_type":0,"cond_num":7,"
        cond_value":1 })
```

Note: When there is no speed_type parameter, speed represents the absolute rotational angular speed.

CAUTION



The above commands are only supported in remote mode.
cond_type, cond_num and cond_value are optional parameters, applicable to 2.14.4 and above versions.
speed_type, acc, dec, applicable to 2.16.2 and above, are optional parameters.
Before executing the above commands, please make sure that the robot is in a stopped state. If the robot is running, send the stop command first and wait for the robot to stop.

2.2.3.5 Add waypoint information 2.0

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

Function: Add waypoint information 2.0

Parameter: waypoint: target location

moveType:0:joint motion,1:linear motion,2:rotating motion around the tip of the tool,3: circular motion

speed: When there is no speed_type parameter, it is expressed as: motion speed, joint motion speed range [1,100], linear and arc speed range [1,3000], rotary motion speed range [1,300]

speed: operating speed. double Type is linear speed, range: 1-3000; is rotation angular speed, range: 1-300; is absolute linear speed, range: linear minimum speed parameter value-linear maximum speed parameter value; absolute rotation angular speed, range: rotation angle minimum speed parameter value-rotation angle maximum speed parameter value

speed_type: Speed type, 0 is V (linear speed), 1 is VR (rotational angular speed), 2 is AV (absolute linear speed), and 3 is AVR (absolute rotation angular speed).

smooth: smoothness, range: 0~7

cond_type: IO type, 0 is digital input X, 1 is digital output Y, range int[0,1]

cond_num: IO address, range int[0,63]

cond_value: IO status, range int[0,1], when the actual IO status is consistent with this value, immediately abandon the unfinished movement and execute the next instruction

Return: True for success, false for failure

Example:

```
if __name__ == "__main__":
  # Robot IP address
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  C000 = [0.0065,-103.9938,102.2076,-88.2138,
  90.0000,0.0013,0.0000,0.0000]
  C001 = [-16.2806,-82.4996,81.9848,-89.4851,
  90.0000,-16.2858,0.0000,0.0000]
  C002 = [3.7679, -71.7544, 68.7276, -86.9732,
  90.0000, 3.7627, 0.0000, 0.0000]
  C003 = [12.8237,-87.3028,87.2361,-89.9333,
  90.0000,12.8185,0.0000,0.0000]
  # Clear waypoint information 2.0
  suc, result, id = sendCMD(sock, "clearPathPoint")
  if(result == True):
    # Add waypoint information 2.0
    suc,result,id=sendCMD(sock,"addPathPoint",{"wayPoint":C000,"
        moveType":0,"speed_type":0,"speed":50,"smooth":7,})
    suc,result,id=sendCMD(sock,"addPathPoint",{"wayPoint":C001,"
        moveType":0,"speed_type":0,"speed":50,"smooth":7,})
    suc,result,id=sendCMD(sock,"addPathPoint",{"wayPoint":C002,"
        moveType":0,"speed_type":0,"speed":50,"smooth":7})
    suc,result,id=sendCMD(sock,"addPathPoint",{"wayPoint":C003,"
        moveType":0,"speed_type":0,"speed":50,"smooth":7,})
```

CAUTION



This command is only supported in remote mode.
cond_type, cond_num and cond_value are optional parameters, applicable to 2.14.4 and above.

2.2.3.6 Clear waypoint information 2.0

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

Function: Clear waypoint information 2.0

Parameter: None

Return: True for success, false for failure

Example:

```
if __name__ == "__main__":
  # Robot IP address
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if(conSuc):
    # Get the servo status of the robotic arm
    suc,result,id=sendCMD(sock,"getServoStatus")
    if(result == 0):
      # Clear waypoint information 2.0
      suc, result, id = sendCMD(sock, "clearPathPoint")
```

Note: This command is only supported in remote mode.

2.2.3.7 Track Movement 2.0

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

Function: Track Movement 2.0

Parameter: None

Return: Failure: -1, Success: the total number of waypoints

Example:

```
if __name__ == "__main__":
  # Robot IP address
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  C000 = [0.0065,-103.9938,102.2076,-88.2138,
  90.0000,0.0013,0.0000,0.0000]
  C001 = [-16.2806,-82.4996,81.9848,-89.4851,
  90.0000, -16.2858, 0.0000, 0.0000]
  C002 = [3.7679, -71.7544, 68.7276, -86.9732,
  90.0000, 3.7627, 0.0000, 0.0000]
  if(conSuc):
    # Clear waypoint information 2.0
    suc, result, id = sendCMD(sock, "clearPathPoint")
    if(result == True):
      # Add waypoint information 2.0
      suc, result, id = sendCMD(sock, "addPathPoint", {"wayPoint":
          C000,"moveType": 0, "speed": 50, "smooth": 7})
      suc, result, id = sendCMD(sock, "addPathPoint", {"wayPoint":
          C001,"moveType":0, "speed": 50, "smooth": 7})
      suc, result, id = sendCMD(sock, "addPathPoint", {"wayPoint":
          C002,"moveType": 0, "speed": 50, "smooth": 7})
      # Trajectory movement 2.0
      suc, result, id = sendCMD(sock, "moveByPath")
      while(True):
        # Get the line number of trackfile file running (consistent
            with the line number displayed by the teach pendant)
        suc, result, id = sendCMD(sock, "getPathPointIndex")
        print(result)
        # Get robot status
        suc, result, id = sendCMD(sock, "getRobotState")
        if(result == 0):
          break
```

Note: This command is only supported in remote mode.
Before executing this command, make sure that the robot is in a stopped state. If the robot is running, send the stop command first and wait for the robot to stop.

2.2.3.8 Jog Movement

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

Function: Jog movement

Parameter: index: axis direction or coordinate system direction number, range: int[0,11]
speed: manual speed percentage, range double [0.05,100] (optional parameter, not required)

Return: True for success, false for failure

Example:

```
if __name__ == "__main__":
  # Robot IP address
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if(conSuc):
    # Get the servo status of the robotic arm
    suc, result, id = sendCMD(sock, "getServoStatus")
    if(result == 0):
      # Set the servo status of the robot arm to ON
      suc,result,id=sendCMD(sock,"set_servo_status",{"status":1})
      time.sleep(1)
    while(True):
      suc, result, id = sendCMD(sock, "getRobotMode")
      if(result == 0):
        break
      print("jog can only be used in teaching mode, please switch
          the robot to teaching mode")
    # Specify the coordinate system
    suc,result,id=sendCMD(sock, "setCurrentCoord", {"coord_mode"
        :1})
    for i in range(0, 10, 1):
      # x-axis negative direction jog movement
      suc, result, id = sendCMD(sock, "jog", {"index":0,"speed"
          :10})
      print(suc, result, id)
      time.sleep(0.1)
      suc,result,id = sendCMD(sock, "stop")
```

Note: After stopping sending the jog command, the robot will not stop immediately. Instead, the robot needs to be stopped immediately by the "stop robot operation" command below.
This command is only supported in remote mode.
If the next jog motion instruction is not received for more than 1 second, stop receiving the jog instruction, and the robot jog motion will stop.

2.2.3.9 Stop robot operation

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

Function: Stop robot operation

Parameter: None

Return: True for success, false for failure

Example:

```
if __name__ == "__main__":
  # Robot IP address
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if(conSuc):
    # Robot stop
    suc, result, id = sendCMD(sock, "stop")
```

Note: This command is only supported in remote mode.

2.2.3.10 Robot runs automatically

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

Function: Robot runs automatically

Parameter: None

Return: True for success, false for failure

Example:

```
if __name__ == "__main__":
  # Robot IP address
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if(conSuc):
    # Robot stop
    suc, result, id = sendCMD(sock, "pause") time.sleep(1)
    # Robot start
    suc, result, id = sendCMD(sock, "run")
```

Note: This command is only supported in remote mode.

2.2.3.11 Robot stop

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

Function: Robot stop

Parameter: None

Return: True for success, false for failure

Example:

```
if __name__ == "__main__":
  # Robot IP address
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if(conSuc):
    # Robot stop
    suc, result, id = sendCMD(sock, "pause") time.sleep(1)
```

Note: This command is only supported in remote mode.

2.2.3.12 Check if the jbi file exists

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

Function: Check if the jbi file exists

Parameter: filename: the name of the file to be checked

Return: 0 means not present, 1 means present

Example:

```
if __name__ == "__main__":
  # Robot IP address
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  jbi_filename="test"
  if(conSuc):
    # Check if the jbi file exists
    suc,result,id=sendCMD(sock,"checkJbiExist",{"filename":
        jbi_filename})
```

2.2.3.13 Run jbi file

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

Function: Run jbi file

Parameter: filename the name of the file to be rum

Return: True for success, false for failure

Example:

```
if __name__ == "__main__":
  # Robot IP address
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  jbi_filename="test"
  if(conSuc):
    # Check if the jbi file exists
    suc,result,id=sendCMD(sock,"checkJbiExist",{"filename":
        jbi_filename})
    if(suc and result==1):
      # Run jbi file
      suc,result,id=sendCMD(sock,"runJbi",{"filename":jbi_filename
          })
```

Note: This command is only supported in remote mode.
Before executing this command, make sure that the robot is in a stopped state. If the robot is running, send the stop command first and wait for the robot to stop.

2.2.3.14 Get jbi file running status

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

Function: Get jbi file running status

Parameter: None

Return: jbiName: file name
runState: 0 stop state, 1 pause state, 2 emergency stop state, 3 running state, 4 error state

Example:

```
if __name__ == "__main__":
  # Robot IP address
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  jbi_filename="test"
  if(conSuc):
    # Check if the jbi file exists
    suc,result,id=sendCMD(sock,"checkJbiExist",{"filename":
        jbi_filename})
    if(suc and result==1):
      # Run jbi file
      suc,result,id=sendCMD(sock,"runJbi",{"filename":jbi_filename
          })
      if(suc and result):
        checkRunning=3
        while(checkRunning==3):
          # Get jbi file running status
          suc,result,id=sendCMD(sock,"getJbiState")
          checkRunning=result["runState"]
          time.sleep(0.1)
```

2.2.3.15 Set robot running speed

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

Function: Set the robot running speed

Parameter: value: speed, range: double [0.05,100]

Return: True for success, false for failure

Example:

```
if __name__ == "__main__":
  # Robot IP address
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if(conSuc):
    # Set the robot running speed to 30%
    suc, result, id = sendCMD(sock, "setSpeed", {"value": 3000})
  else:
    print("Connection failed")
  disconnectETController(sock)
```

Note: This command is applicable to v2.13.1 and above.
This command is only supported in remote mode.

2.2.3.16 Joint motion at uniform speed

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

Function: Uniform motion of joints

Parameter: vj: the speed value of each joint, unit: degree/second
acc: joint acceleration, range 1-100, unit: degree/sec^2
t: speedj execution time, range t>0, unit: second

Return: True for success, false for failure

Example:

```
if __name__ == "__main__":
  # Robot IP address
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  speed_j=[1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0]
  if(conSuc):
    # Uniform motion of joints
    suc,result,id=sendCMD(sock,"moveBySpeedj",{"vj":speed_j,"acc"
        :20,"t":5})
    print(suc, result, id)
```

Note: This command is applicable to v2.16.2 and above.
This command is only supported in remote mode.

2.2.3.17 Linear uniform motion

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

Function: Linear uniform motion

Parameter: v: the speed value of each axis, unit: degree/second
acc: linear acceleration, range 1-100, unit: degree/sec^2
t: speedl execution time, range t>0, unit: second

Return: True for success, false for failure

Example:

```
if __name__ == "__main__":
  # Robot IP address
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  speed_l=[1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0]
  if(conSuc):
    # Linear uniform motion
    suc,result,id=sendCMD(sock,"moveBySpeedl",{"v":speed_l,"acc"
        :20,"t":5})
    print(suc,result,id)
```

Note: This command is applicable to v2.16.2 and above.
This command is only supported in remote mode.

2.2.3.18 Linear motion under specified coordinate system

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

Function: Linear motion in the specified coordinate system

Parameter: targetUserPose: Specify the pose in the user coordinate system
speed: operating speed. double Type is linear speed, range: 1-3000; is rotation angular speed, range: 1-300; is absolute linear speed, range: linear minimum speed parameter value-linear maximum speed parameter value; absolute rotation angular speed, range: rotation angle minimum speed parameter value-rotation angle maximum speed parameter value
speed_type: Speed type, 0 is V (linear speed), 1 is VR (rotational angular speed), 2 is AV (absolute linear speed), and 3 is AVR (absolute rotation angular speed).
user_coord: user coordinate system data, double[6], the range of rx, ry, rz is [-π,π], do not write the current coordinate system.
cond_type: IO type, 0 is digital input X, 1 is digital output Y, range int[0,1]
cond_num: IO address, range int[0,63]
cond_value: IO status, range int[0,1], when the actual IO status is consistent with this value, immediately abandon the unfinished movement and execute the next instruction.
acc: acceleration percentage, range: int [1,100], optional parameter, default value 0 if not written.
dec: percentage of deceleration, range: int [1,100], optional parameter, default value 0 if not written.

Return: True for success, false for failure

Example:

```
if __name__ == "__main__":
  # Robot IP address
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  speed_l=[1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0]
  if(conSuc):
    # Specify the linear motion in the coordinate system
    suc,result,id=sendCMD(sock,"moveByLineCoord",{"targetUserPose":
        point1,"user_coord":[0,0,0,0,0,0],"speed_type":0,"speed"
        :30})
    print(suc, result, id)
```

Note: This command is applicable to v2.16.2 and above.
This command is only supported in remote mode.

2.2.4 Kinematics Service(KinematicsService)

2.2.4.1 Inverse solution function

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

Function: Inverse solution function, according to the pose information to get the corresponding manipulator joint angle information

Parameter: targetpose: target pose information

Return: Response to joint angle information: double pos[8]

Example:

```
if __name__ == "__main__":
  # Robot IP address
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if(conSuc):
    # Get the current pose information of the robot
    suc, result, id = sendCMD(sock, "getRobotPose")
    # Inverse solution function
    suc,result,id=sendCMD(sock,"inverseKinematic",{"targetPose":
        result})
```

2.2.4.2 Positive solution function

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

Function: Positive solution function, get the corresponding pose information according to the joint angle information of the manipulator

Parameter: targetpos: target joint angle information

Return: Response pose information: double pose[6]

Example:

```
if __name__ == "__main__":
  # Robot IP address
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if(conSuc):
    # Get the current position information of the robot
    suc, result, id = sendCMD(sock, "getRobotPos")
    # Positive solution function
    suc,result,id=sendCMD(sock,"positiveKinematic",{"targetPos":
        result})
```

2.2.4.3 Base coordinate to user coordinate pose conversion

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

Function: Base coordinate to user coordinate pose conversion function, in the current user coordinate system, the pose information in the corresponding user coordinate system is obtained according to the pose information of the base coordinate

Parameter: targetPose: pose information in the base coordinate system
userNo:User coordinate number, range: int[0,7]

Return: Pose information under the user standard system: double user_pose[6]

Example:

```
if __name__ == "__main__":
  # Robot IP address
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if(conSuc):
    # Get the current pose information of the robot
    suc, result, id = sendCMD(sock, "getRobotPose")
    # Base coordinate to user coordinate pose conversion
    suc,result,id=sendCMD(sock,"convertPoseFromCartToUser",{"
        targetPose":result,"userNO":0})
```

2.2.4.4 User coordinate to base coordinate pose conversion

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

Function: User coordinate to base coordinate pose conversion, in the current user coordinate system, the pose information in the corresponding base coordinate system is obtained according to the pose information of the user coordinate

Parameter: targetPose: pose information under the user's standard system
userNo: User coordinate number, range: int[0,7]

Return: Pose information in the base coordinate system: double base_pose[6]

Example:

```
if __name__ == "__main__":
  # Robot IP address
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if(conSuc):
    # Get the current pose information of the robot
    suc, result, id = sendCMD(sock, "getRobotPose")
    # User coordinate to base coordinate pose conversion
    suc,result,id=sendCMD(sock,"convertPoseFromUserToCart",{"
        targetPose":result,"userNO":0})
```

2.2.4.5 Inverse solution function 2.0, with reference point position

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

Function: Inverse solution function 2.0, inverse solution with reference point position, according to the pose information to get the corresponding manipulator joint angle information

Parameter: targetPose: target pose information
referencePos: Inverse solution reference point joint angle

Return: True for success, false for failure

Example:

```
if __name__ == "__main__":
  # Robot IP address
  robot_ip="192.168.1.200"
  # Reference point
  P000 = [0, -90, 90, -90, 90, 0, 0, 0]
  conSuc,sock=connectETController(robot_ip)
  if(conSuc):
    # Get the current pose information of the robot
    suc, result, id = sendCMD(sock, "getRobotPose")
    # Inverse solution function 2.0, with reference point position
    suc,result,id=sendCMD(sock,"inverseKinematic",{"targetPose":
        result,"referencePos":P000})
```

2.2.4.6 Position multiplication

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

Function: Position multiplication

Parameter: pose1: pose information
pose2: pose information

Return: Posture multiplication result information: :double response_pose[6]

Example:

```
if __name__ == "__main__":
  # Robot IP address
  robot_ip="192.168.1.200"
  # pose2
  V000 = [10, -10, 10, 0, 0, 0]
  conSuc,sock=connectETController(robot_ip)
  if(conSuc):
    # Get the current pose information of the robot
    suc, result, id = sendCMD(sock, "getRobotPose")
    # Position multiplication
    suc,result,id=sendCMD(sock,"poseMul",{"pose1":result,"pose2":
        V000})
```

2.2.4.7 Pose inversion

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

Function: Pose inversionPose inversion

Parameter: pose: pose information

Return: Pose inversion result information: double response_pose[6]

Example:

```
if __name__ == "__main__":
  # Robot IP address
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if(conSuc):
    # Get the current pose information of the robot
    suc, result, id = sendCMD(sock, "getRobotPose")
    # Pose inversion
    suc, result, id = sendCMD(sock, "poseInv", {"pose":result})
```

2.2.5 IO Service(IOService)

2.2.5.1 Get input IO status

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

Function: Get input IO status

Parameter: addr: input IO address, range: int[0,127]

Return: Input IO status, int[0,1], 0 is off, 1 is on

Example:

```
if __name__ == "__main__":
  # Robot IP address
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if(conSuc):
    for i in range(0, 64 ,1):
      # Get input IO status
      suc, result, id = sendCMD(sock, "getInput", {"addr":i})
      print(result)
```

2.2.5.2 Get output IO status

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

Function: Get output IO status

Parameter: addr: output IO address, range: int[0,127]

Return: Output IO status, int[0,1], 0 is off, 1 is on

Example:

```
if __name__ == "__main__":
  # Robot IP address
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if(conSuc):
    for n in range(0, 20 ,1):
      # Get output IO status
      suc, result, id = sendCMD(sock, "getOutput", {"addr":n})
      print(result)
```

2.2.5.3 Set output IO status

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

Function: Set output IO status

Parameter: addr: input IO address, range: int[0,63]
status: IO status, int[0,1], 0 is off, 1 is on

Return: True for success, false for failure

Example:

```
if __name__ == "__main__":
  # Robot IP address
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if(conSuc):
    for i in range(0, 20 ,1):
      # Set output IO status
      suc,result,id=sendCMD(sock,"setOutput",{"addr":i,"status":1})
      print(result)
```

Note: This command is only supported in remote mode.

2.2.5.4 Get virtual input IO status

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

Function: Get virtual input IO status

Parameter: addr: virtual IO address, range: int[0,399]

Return: Input IO status, int[0,1], 0 is off, 1 is on

Example:

```
if __name__ == "__main__":
  # Robot IP address
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if(conSuc):
    for i in range(0, 400 ,1):
      # Get virtual input IO status
      suc,result,id=sendCMD(sock,"getVirtualInput",{"addr":i})
      print(result)
```

2.2.5.5 Get virtual output IO status

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

Function: Get virtual output IO status

Parameter: addr: virtual IO address, range: int [400,1535]

Return: Output IO status, int[0,1], 0 is off, 1 is on

Example:

```
if __name__ == "__main__":
  # Robot IP address
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if(conSuc):
    for n in range(528, 800 ,1):
    # Get virtual output IO status
    suc,result,id=sendCMD(sock,"getVirtualOutput",{"addr":n})
    print(result)
```

2.2.5.6 Set virtual output IO status

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

Function: Set virtual output IO status

Parameter: addr: output IO address, range: int[528,799]
status: output IO status, int[0,1], 0 is off, 1 is on

Return: True for success, false for failure

Example:

```
if __name__ == "__main__":
  # Robot IP address
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if(conSuc):
    for i in range(528, 800 ,1):
      # Set virtual output IO status
      suc,result,id=sendCMD(sock,"setVirtualOutput",{"addr":i,"
          status":1})
```

Note: This command is only supported in remote mode.

2.2.5.7 Read multiple M virtual IO

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

Function: Read multiple M virtual IO

Parameter: addr: virtual IO address range int [0,1535]
len: The starting address starts to read backward and the length is (16*len) virtual IO range int [1,96]
The range of addr+16*len is int[0,1535]

Return: List of virtual IO values (each 16 virtual IO values are represented by a decimal integer, and the length of the list is len)

Example:

```
if __name__ == "__main__":
  # Robot IP address
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if(conSuc):
    # Get the value of M0~M16
    ret, result, id = sendCMD(sock, "getRegisters", {"addr": 0, "
        len": 1})
    if ret:
      print("result=", result)
    else:
      print("err_msg=", result["message"])
```

2.2.6 Variable service(VarService)

2.2.6.1 Get system B variable value

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

Function: Get system B variable value

Parameter: addr: variable address, range: int [0,255]

Return: Variable value, range:0~2^{16}-1

Example:

```
if __name__ == "__main__":
  # Robot IP address
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if(conSuc)
    for n in range(0, 11 ,1):
      # Get system B variable value
      suc, result, id = sendCMD(sock, "getSysVarB", {"addr":n})
      print(result)
```

2.2.6.2 Set system B variable value

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

Function: Set system B variable value

Parameter: addr: variable address, range: int [0,255]
value: variable value, range: int [0,2147483647]

Return: True for success, false for failure

Example:

```
if __name__ == "__main__":
  # Robot IP address
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if(conSuc)
    for n in range(0, 11 ,1):
      # Set system B variable value
      suc,result,id=sendCMD(sock,"setSysVarB",{"addr":i,"value"
          :100})
```

Note: This command is only supported in remote mode.

2.2.6.3 Get system I variable value

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

Function: Get system I variable value

Parameter: addr: variable address, range: int [0,255]

Return: Variable value, range: -2^{15}-1~2^{15}-1

Example:

```
if __name__ == "__main__":
  # Robot IP address
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if(conSuc)
    for n in range(0, 11 ,1):
      # Get system I variable value
      suc, result, id = sendCMD(sock, "getSysVarI", {"addr":n})
      print(result)
```

2.2.6.4 Set system I variable value

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

Function: Set system I variable value

Parameter: addr: variable address, range: int [0,255]
value: variable value, range: double [-2^{15}-1,2^{15}-1]

Return: True for success, false for failure

Example:

```
if __name__ == "__main__":
  # Robot IP address
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if(conSuc)
    for n in range(0, 11 ,1):
      # Set system I variable value
      suc,result,id=sendCMD(sock,"setSysVarI",{"addr":i,"value"
          :100})
```

Note: This command is only supported in remote mode.

2.2.6.5 Get system D variable value

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

Function: Get system D variable value

Parameter: addr: variable address, range: int [0,255]

Return: Variable value, range: -1e+09~1e+09

Example:

```
if __name__ == "__main__":
  # Robot IP address
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if(conSuc)
    for n in range(0, 11 ,1):
      # Get system D variable value
      suc, result, id = sendCMD(sock, "getSysVarD", {"addr":n})
      print(result)
```

2.2.6.6 Set system D variable value

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

Function: Set system D variable value

Parameter: addr: variable address, range: int [0,255]
value Variable value, range: double[-1e+09,1e+09]

Return: True for success, false for failure

Example:

```
if __name__ == "__main__":
  # Robot IP address
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if(conSuc)
    for n in range(0, 11 ,1):
      # Set system D variable value
      suc,result,id=sendCMD(sock,"setSysVarD",{"addr":i,"value"
          :100})
```

Note: This command is only supported in remote mode.

2.2.6.7 Get whether the system P variable is enabled

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

Function: Get whether the system P variable is enabled

Parameter: addr: variable address, range: int [0,255]

Return: Store the enabled state of the P variable, 0: not enabled, 1: enabled

Example:

```
if __name__ == "__main__":
  # Robot IP address
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if(conSuc)
    for i in range(0, 101 ,1):
      # Get whether the system P variable is enabled
      suc, result, id = sendCMD(sock, "getSysVarPState", {"addr": i
          })
```

2.2.6.8 Get the value of P variable

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

Function: Get the value of P variable

Parameter: addr: variable address, range: int [0,255]

Return: System P variable value double pos[8]

Example:

```
if __name__ == "__main__":
  # Robot IP address
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if(conSuc)
    for i in range(0, 101 ,1):
      # Get whether the system P variable is enabled
      suc, result, id = sendCMD(sock, "getSysVarPState", {"addr": i
          })
        if(result == 1):
          # Get system P variable value
          suc, result, id = sendCMD(sock, "getSysVarP", {"addr":i})
          print(result)
```

2.2.6.9 Set the scope of the V variable

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

Function: Set the value of the system V variable

Parameter: addr: variable address: range int[0,255]
pose: double pose[6], the range of rx,ry,rz is [-π,π]

Return: True for success, False for failure

Example:

```
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,"setSysVarV",{"addr":0,"pos":pos})
    if ret:
      print(result)
    else:
      print("err_msg = ", result["message"])
```

Note: This command is only supported in remote mode.
This command is applicable to v2.15.2 and above.

2.2.6.10 Get the value of V variable

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

Function: Get the value of V variable

Parameter: addr: variable address, range: int [0,255]

Return: System V variable value double pose[6]

Example:

```
if __name__ == "__main__":
  # Robot IP address
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if(conSuc)
    for i in range(0, 101 ,1):
      # Get the value of V variable
      suc, result, id = sendCMD(sock, "getSysVarV", {"addr": i})
      print(result)
```

2.2.6.11 Set the value of the V variable

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

Function: Set the value of the system V variable

Parameter: addr: variable address, range int[0,255]

pose: the value of the V variable, double pose[6], the range of rx,ry,rz is [-π,π]

Return: True for success, False for failure

Example:

```
if __name__ == "__main__":
  # Robot IP address
  robot_ip="192.168.1.205"
  conSuc,sock=connectETController(robot_ip)
  pose = [200, 125.5, -50, 0, 0, 0]
  if(conSuc):
    # Set the system V variable value
    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("Connection failed")
  disconnectETController(sock)
```

Note: This command is only supported in remote mode.

2.2.6.12 Save variable data

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

Function: Save system variable data

Parameter: no

Return: True for success, False for failure

Example:

```
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"])
```

Note: This command is only supported in remote mode.
This command is applicable to v2.15.2 and above.

2.2.7 Transparent Transmission Service

2.2.7.1 Initialize transparent transmission service

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

Function: Initialize transparent transmission service of the robot

Parameter: lookahead: lookahead time, unit ms, range: double [10,1000]
t: sampling time, unit ms, range: double [2,100]
smoothness: gain, unit percentage, range: double [0,1].
Note: The current version of smoothness is not applicable
response_enable: optional parameter, if it is not written or the value is 1, there is a return value; when the value is 0, there is no return value.

Return: True for success, false for failure

Example:

```
if __name__ == "__main__":
  # Robot IP address
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if(conSuc):
    # Initialize transparent transmission service
    suc, result, id = sendCMD(sock, "transparent_transmission_init"
        , {"lookahead": 400, "t": 10, "smoothness": 0.1,"
        response_enable": 0})
```

Note: This command is only supported in remote mode.

2.2.7.2 Set the current transparent transmission servo target joint point

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

Function: Set the current transparent transmission servo target joint point

Parameter: targetPos: target joint point

Return: True for success, false for failure

Example:

```
if __name__ == "__main__":
  # Robot IP address
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  # Transparent transmission starting point
  P0 = [0, -90, 0, -90, 90, 0, 0, 0]
  if(conSuc):
    # Initialize transparent transmission service
    suc,result,id=sendCMD(sock,"transparent_transmission_init",{"
        lookahead":400,"t":10,"smoothness":0.1})
    # Set the current transparent transmission target joint point
    suc,result,id=sendCMD(sock,"tt_set_current_servo_joint",{"
        targetPos": P0})
```

Note: This command is only supported in remote mode.

2.2.7.3 Add transparent transmission of servo target joint point information to the cache

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

Function: Add transparent transmission of servo target joint point information to the cache

Parameter: targetPos: target joint point

Return: True for success, false for failure

Example:

```
if __name__ == "__main__":
# Robot IP address
  robot_ip="192.168.1.202"
  conSuc,sock=connectETController(ip)
  i = 0
  if(conSuc):
    # Get whether the current robot is in a transparent
        transmission state
    suc, result, id = sendCMD(sock, "
        get_transparent_transmission_state")
    print(suc, result, id)
    if(result == 1):
      # Clear the transparent transmission cache
      suc, result, id = sendCMD(sock,"tt_clear_servo_joint_buf")
      time.sleep(0.5)
    # open a file
      file_name ='D:\\tttest8.txt'
      fo = open(file_name, "r")
      while 1:
        # Read each line of the file in turn (point information)
        line = fo.readline()
        if not line: break
        # Remove the blanks at the beginning and end of each line
        line_list = line.strip()
        line_list = list(map(float, line_list.split(',')))
        print(i,line_list)
        if (i == 0):
          # Joint movement to the starting point
            suc, result, id = sendCMD(sock, "moveByJoint", {"
                targetPos": line_list, "speed": 30})
            wait_stop() # Wait for the robot to stop
            # Initialize the transparent transmission service
            suc, result, id = sendCMD(sock, "
                transparent_transmission_init", {"lookahead": 400, "
                t": 10, "smoothness": 0.1, "response_enable": 0})
            print(suc, result, id)
        # Add transparent transmission of servo target joint point
            information to the cache
        suc, result, id = sendCMD(sock, "tt_put_servo_joint_to_buf"
            ,{"targetPos": line_list})
        time.sleep(0.01)
        i = i + 1
      # Close file
      fo.close()
      # Clear the transparent transmission cache
      suc,result,id=sendCMD(sock,"tt_clear_servo_joint_buf")
      print("clear_ret = ", suc)
```


Note: This command is only supported in remote mode.

2.2.7.4 Clear the transparent transmission cache

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

Function: Clear the transparent transmission cache

Parameter: None

Return: True for success, false for failure

Example:

```
if __name__ == "__main__":
  # Robot IP address
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if(conSuc):
    # Get whether the current robot is in a transparent
      transmission state
    suc,result,id =sendCMD(sock,"get_transparent_transmission_state
      ")
    if(result == 1):
      # Clear the transparent transmission cache
      suc,result,id=sendCMD(sock, "tt_clear_servo_joint_buf")
      time.sleep(0.5)
```

Note: This command is only supported in remote mode.

2.2.7.5 Get whether the current robot is in a transparent transmission state

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

Function: Get whether the current robot is in a transparent transmission state

Parameter: None

Return: Current transparent transmission status. 0: non-transparent transmission state, 1: transparent transmission state

Example:

```
if __name__ == "__main__":
  # Robot IP address
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if(conSuc):
    # Get whether the current robot is in a transparent
      transmission state
    suc,result,id=sendCMD(sock,"get_transparent_transmission_state"
      )
```

2.2.7.6 Example

```
import socket
import json
import time
import random

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:
        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)

```

```
41 if __name__ == "__main__":
42     # Robot IP address
43     robot_ip="192.168.1.202"
44     conSuc,sock=connectETController(robot_ip)
45     print(conSuc)
46     if(conSuc):
47     # Get robot status
48         suc, result, id = sendCMD(sock, "getRobotState")
49         if(result == 4):
50             # Clear alarm
51             suc, result, id = sendCMD(sock, "clearAlarm")
52             time.sleep(0.5)
53         # Get synchronization status
54         suc, result, id = sendCMD(sock, "getMotorStatus")
55         if(result == 0):
56             # Synchronize servo encoder data
57             suc, result, id = sendCMD(sock, "syncMotorStatus")
58             time.sleep(0.5)
59         # Get the servo status of the robotic arm
60         suc, result, id = sendCMD(sock, "getServoStatus")
61         if (result == 0):
62             # Set the servo status of the robotic arm to ON
63             suc, result, id = sendCMD(sock, "set_servo_status",{"status
                   ":1})
64             time.sleep(1)
65     # Get whether the current robot is in a transparent transmission
          state
66         suc, result, id = sendCMD(sock, "
              get_transparent_transmission_state")
67         print(suc, result, id)
68         if(result == 1):
69             # Clear the transparent transmission cache
70             suc, result, id = sendCMD(sock,"tt_clear_servo_joint_buf")
71             time.sleep(0.5)
72       # open a file
73         file_name ='D:\\tttest8.txt'
74         fo = open(file_name, "r")
75         while 1:
76             # Read each line of the file in turn (point information)
77             line = fo.readline()
78             if not line: break
79             # Remove the blanks at the beginning and end of each line
80             line_list = line.strip()
```

```
            line_list = list(map(float, line_list.split(',')))
            print(i,line_list)
            if (i == 0):
                # Joint movement to the starting point
                suc, result, id = sendCMD(sock, "moveByJoint", {"
                    targetPos": line_list, "speed": 30})
                wait_stop() # Wait for the robot to stop
                # Initialize the transparent transmission service
                suc, result, id = sendCMD(sock, "
                    transparent_transmission_init", {"lookahead": 400, "
                    t": 10, "smoothness": 0.1, "response_enable": 0})
                print(suc, result, id)
            # Add transparent transmission of servo target joint point
                information to the cache
            suc, result, id = sendCMD(sock, "tt_put_servo_joint_to_buf"
                ,{"targetPos": line_list})
            time.sleep(0.01)
            i = i + 1
        # Close file
        fo.close()
        # Clear the transparent transmission cache
        suc, result, id = sendCMD(sock, "tt_clear_servo_joint_buf")
        print("clear_ret = ", suc)
else:
        print("Connection failed")
    disconnectETController(sock)
```

2.2.8 System Service(SystemService)

2.2.8.1 Get the controller software version number

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

Function: Get the controller software version number

Parameter: None

Return: The controller software version number

Example:

```
if __name__ == "__main__":
  # Robot IP address
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if(conSuc):
    # Get the controller software version number
    suc, result, id = sendCMD(sock, "getSoftVersion")
    print(result)
```

2.2.8.2 Get servo version number

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

Function: Get the servo version number

Parameter: axis: range int [0,7], corresponding to axis number 1~8

Return: Servo version number

Example:

```
if __name__ == "__main__":
  # Robot IP address
  robot_ip="192.168.1.200"
  conSuc,sock=connectETController(robot_ip)
  if(conSuc):
    # Get 1 axis servo version number
    ret, result, id = sendCMD(sock, "getJointVersion", {"axis":0})
    if ret:
      print("result=", result)
    else:
      print("err_msg=", result["message"])
```

Note: The servo version for this function is 11 and above

2.3 Examples

2.3.1 Example 1



```
import socket
import json
import time
import random

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:
        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__":
    # Robot IP address
```


DN:T202012049

```
43      robot_ip="192.168.1.202"
44      conSuc,sock=connectETController(robot_ip)
45      print(conSuc)
46      if(conSuc):
47          # Clear alarm
48          ret, result, id = sendCMD(sock, "clearAlarm") # Forcibly clear
               the alarm
49          print("Clear alarm")
50          print("ret = ", ret, "", "id = ", id)
51          if (ret == True):
52              print("result = ", result)
53              time.sleep(1)
54          else:
55              print("err_msg = ", result["message"])
56          # Get synchronization status
57          ret, result, id = sendCMD(sock, "getMotorStatus")
58          print("Get synchronization status")
59          print("ret = ", ret, "", "id = ", id)
60          if (ret == True):
61              print("result = ", result)
62              if(result != 1):
63                  # Synchronize
64                  ret1, result1, id = sendCMD(sock, "syncMotorStatus")
65                  print("synchronization")
66                  print("ret = ", ret1, "", "id = ", id)
67                  if (ret == True):
68                      print("result = ", result1)
69                      time.sleep(0.5)
70                  else:
71                      print("err_msg = ", result["message"])
72          else:
73              print("err_msg = ", result["message"])
74
75
76          # Turn on the servo
77          ret, result, id = sendCMD(sock, "set_servo_status", {"status"
               :1})
78          print("Turn on the servo")
79          print("ret = ", ret, "", "id = ", id)
80          if (ret == True):
81              print("result = ", result)
82              time.sleep(1)
83          else:
```

```
            print("err_msg = ", result["message"])
        # Get servo status
        ret, result, id = sendCMD(sock, "getServoStatus")
        print("ret = ", ret, "", "id = ", id)
        if(ret == True):
            print("result = ", result)
        else:
            print("err_msg = ", result["message"])
    else:
        print("Connection failed")
    disconnectETController(sock)
```

2.3.2 Example 2



```
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 = []
```

```
29      else:
30          params = json.dumps(params)
31      sendStr = "{{\"method\":\"{0}\",\"params\":{1},\"jsonrpc
           \":\"2.0\",\"id\":{2}}}".format(cmd, params, id) + "\n"
32      try:
33          # print(sendStr)
34          sock.sendall(bytes(sendStr, "utf-8"))
35          ret = sock.recv(1024)
36          jdata = json.loads(str(ret, "utf-8"))
37          if ("result" in jdata.keys()):
38              return (True, json.loads(jdata["result"]), jdata["id"])
39          elif ("error" in jdata.keys()):
40              return (False, jdata["error"], jdata["id"])
41          else:
42              return (False, None, None)
43      except Exception as e:
44          return (False, None, None)
45
46
47  if __name__ == "__main__":
48      ip = "192.168.1.205"
49      conSuc, sock = connectETController(ip)
50      # print(conSuc)
51      if (conSuc):
52          # Get robot status
53          ret, result, id = sendCMD(sock, "getRobotState")
54          print("Get robot status")
55          print("ret = ", ret, " ", "id = ", id)
56          if (ret == True):
57              print("result = ", result)
58          else:
59              print("err_msg = ", result["message"])
60          # Get robot mode
61          ret, result, id = sendCMD(sock, "getMotorStatus")
62          print("Get robot mode")
63          print("ret = ", ret, " ", "id = ", id)
64          if (ret == True):
65              print("result = ", result)
66          else:
67              print("err_msg = ", result["message"])
68          # Get the current position information of the robot
69          ret, result, id = sendCMD(sock, "getRobotPos")
70          print("Get the current position information of the robot")
```

```
        print("ret = ", ret, " ", "id = ", id)
        if (ret == True):
            print("result = ", result)
        else:
            print("err_msg = ", result["message"])
        # Get the current pose information of the robot
        print("Get the current pose information of the robot")
        ret, result, id = sendCMD(sock, "getRobotPose")
        print("ret = ", ret, " ", "id = ", id)
        if (ret == True):
            print("result = ", result)
        else:
            print("err_msg = ", result["message"])
        # Get the value of analog input
        ret, result, id=sendCMD(sock, "getAnalogInput", {"addr":1})
        print("Get the value of analog input")
        print("ret = ", ret, " ", "id = ", id)
        if(ret == True):
            print("result = ", result)
        else:
            print("err_msg = ", result["message"])
    else:
        print("Connection failed")
    disconnectETController(sock)
```

2.3.3 Example 3



```
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)

```

```
16
17 def disconnectETController(sock):
18     if (sock):
19         sock.close()
20         sock = None
21     else:
22         sock = None
23
24
25 def sendCMD(sock, cmd, params=None, id=1):
26     if (not params):
27         params = []
28     else:
29         params = json.dumps(params)
30     sendStr = "{{\"method\":\"{0}\",\"params\":{1},\"jsonrpc
       \":\"2.0\",\"id\":{2}}}".format(cmd, params, id) + "\n"
31     try:
32         # print(sendStr)
33         sock.sendall(bytes(sendStr, "utf-8"))
34         ret = sock.recv(1024)
35         jdata = json.loads(str(ret, "utf-8"))
36         if ("result" in jdata.keys()):
37             return (True, json.loads(jdata["result"]), jdata["id"])
38         elif ("error" in jdata.keys()):
39             return (False, jdata["error"], jdata["id"])
40         else:
41             return (False, None, None)
42     except Exception as e:
43         return (False, None, None)
44
45
46 if __name__ == "__main__":
47     ip = "192.168.1.205"
48     conSuc, sock = connectETController(ip)
49     # print(conSuc)
50     if (conSuc):
51         # Switch the current tool number to 0
52         ret, result, id = sendCMD(sock, "setToolNumber", {"tool_num":
               0})
53         print("Switch the current tool number to 0")
54         print("ret = ", ret, " ", "id = ", id)
55         if (ret == True):
56             print("result = ", result)
```

```
57              time.sleep(3)
58          else:
59              print("err_msg = ", result["message"])
60          # Set the load and center of gravity of the robotic arm
61          ret, result, id = sendCMD(sock, "cmd_set_payload", {"tool_num":
                0, "m": 6, "point": [20.2, 40, 30.5]})
62          print("Set the load and center of gravity of the robotic arm")
63          print("ret = ", ret, " ", "id = ", id)
64          if (ret == True):
65              print("result = ", result)
66          else:
67              print("err_msg = ", result["message"])
68          # Set up the tool center of the robotic arm
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("Connection failed")
78      disconnectETController(sock)
```

Chapter 3 Monitor Interface

Users can get robot information by connecting to monitor interface of the robot through the socket client.

WARNING



On the teach pendant interface, select “System> System Configuration> Robot Configuration” and check “Remote” under “Common sys” to use this function.

CAUTION



This function is applicable to 2.13.1 and above versions.

3.1 Data description list of monitor interface

Name	Type	Byte	Description
Message Size	unsigned int32	4*1	Packet length, including the current field
timestamp	unsigned int64	8*1	timestamp, the number of milliseconds since January 1, 1970
autorun_cycelMode	unsigned char	1*1	cycle mode, 0: single step, 1: single cycle, 2: continuous
machinePos	double[AXIS_ COUNT]	8*8	joint angle, unit degree
machinePose	Double[6]	8*6	Cartesian coordinates, the first three items are in millimeters, the last three items are in radians
machineUserPose	Double[6]	8*6	Current user coordinates, the first three items are in millimeters, the last three items are in radians
torque	double[AXIS_ COUNT]	8*8	The ratio per thousand of the rated torque of the joint, unit ‰,

Name	Type	Byte	Description
robotState	int32_t	4*1	Robot state: 0: stop, 1: pause, 2: emergency stop, 3: run, 4: alarm
servoReady	int32_t	4*1	Brake status: 0: not open, 1: open.
can_motor_run	int32_t	4*1	Synchronization status: 0: not synchronized, 1: synchronized
motor_speed	int[AXIS_COUNT]	4*8	Motor speed, unit: rpm
robotMode	int32_t	4*1	Robot mode: 0: teaching mode, 1: automatic mode, 2: remote mode
analog_ioInput	double[ANALOG_IN_NUM]	8*3	Analog input port data, unit V
analog_ioOutput	double[ANALOG_OUT_NUM]	8*5	Analog output port data, unit V
digital_ioInput	unsigned int64	8*1	digital input port data
digital_ioOutput	unsigned int64	8*1	digital output port data
collision	unsigned char	1*1	collision alarm state, 0: non-collision alarm state, 1: collision alarm state.
machineFlangePose	Double[6]	8*6	The pose of the flange center in the Cartesian coordinate system, the first three items are in millimeters, and the last three are in radians.
machineUserFlangePose	Double[6]	8*6	The pose of the flange center in the user coordinate system, the first three items are in millimeters, and the last three are in radians. v2.14.4 added
emergencyStopState	unsigned char	1*1	Whether it is an emergency stop state, v2.16.2 added
tcpSpeed	Double	8*1	tcp movement speed, unit mm/s, v2.16.2 added
jointSpeed	Double[AXIS_COUNT]	8*8	joint speed, unit degree/s, v2.14.6 added

3.2 Example

```
import socket
import struct
import collections
import time
import math

HOST = "192.168.1.200"
PORT = 8056

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'
    dic['machineUserPose05'] = 'd'
    dic['machineUserPose06'] = 'd'
```

```
41      dic['torque01'] = 'd'
42      dic['torque02'] = 'd'
43      dic['torque03'] = 'd'
44      dic['torque04'] = 'd'
45      dic['torque05'] = 'd'
46      dic['torque06'] = 'd'
47      dic['torque07'] = 'd'
48      dic['torque08'] = 'd'
49      dic['robotState'] = 'i'
50      dic['servoReady'] = 'i'
51      dic['can_motor_run'] = 'i'
52      dic['motor_speed01'] = 'i'
53      dic['motor_speed02'] = 'i'
54      dic['motor_speed03'] = 'i'
55      dic['motor_speed04'] = 'i'
56      dic['motor_speed05'] = 'i'
57      dic['motor_speed06'] = 'i'
58      dic['motor_speed07'] = 'i'
59      dic['motor_speed08'] = 'i'
60      dic['robotMode'] = 'i'
61      dic['analog_ioInput01'] = 'd'
62      dic['analog_ioInput02'] = 'd'
63      dic['analog_ioInput03'] = 'd'
64      dic['analog_ioOutput01'] = 'd'
65      dic['analog_ioOutput02'] = 'd'
66      dic['analog_ioOutput03'] = 'd'
67      dic['analog_ioOutput04'] = 'd'
68      dic['analog_ioOutput05'] = 'd'
69      dic['digital_ioInput'] = 'Q'
70      dic['digital_ioOutput'] = 'Q'
71      dic['collision'] = 'B'
72      dic['machineFlangePose01'] = 'd'
73      dic['machineFlangePose02'] = 'd'
74      dic['machineFlangePose03'] = 'd'
75      dic['machineFlangePose04'] = 'd'
76      dic['machineFlangePose05'] = 'd'
77      dic['machineFlangePose06'] = 'd'
78      dic['machineUserFlangePose01'] = 'd'
79      dic['machineUserFlangePose02'] = 'd'
80      dic['machineUserFlangePose03'] = 'd'
81      dic['machineUserFlangePose04'] = 'd'
82      dic['machineUserFlangePose05'] = 'd'
83      dic['machineUserFlangePose06'] = 'd'
```

```
84      dic["emergencyStopState"] = "B"
85      dic["tcp_speed"] = "d"
86      dic["joint_speed01"] = "d"
87      dic["joint_speed02"] = "d"
88      dic["joint_speed03"] = "d"
89      dic["joint_speed04"] = "d"
90      dic["joint_speed05"] = "d"
91      dic["joint_speed06"] = "d"
92      dic["joint_speed07"] = "d"
93      dic["joint_speed08"] = "d"
94
95      print("index =", index)
96      data = s.recv(535)
97      # print("len = ", len(data))
98      if len(data) != 535:
99          lost += 1
100         print(str(lost))
101         continue
102
103     names = []
104     ii = range(len(dic))
105     for key, i in zip(dic, ii):
106         fmtsize = struct.calcsize(dic[key])
107         data1, data = data[0:fmtsize], data[fmtsize:]
108         fmt = "!" + dic[key]
109         names.append(struct.unpack(fmt, data1))
110         dic[key] = dic[key], struct.unpack(fmt, data1)
111     output = ""
112     for key in dic.keys():
113         output += str(key) + ":" + str(dic[key][1][0]) + ";\n"
114     output = "lost : " + str(lost) + " index : " + str(index) + ";" +
        output + "\n"
115     if index% 10 == 0:
116         # Print all information
117         print(output)
118         # Print timestamp
119         timestamp01_value = dic['TimeStamp'][1][0] // 1000
120         timeValue = time.gmtime(timestamp01_value)
121         print(time.strftime("%Y-%m-%d %H:%M:%S", timeValue))
122         # Print joint coordinates
123         print(dic['machinePos01'][1][0], dic['machinePos02'][1][0],
124               dic['machinePos03'][1][0], dic['machinePos04'][1][0],
125               dic['machinePos05'][1][0], dic['machinePos06'][1][0],
```


DN:T202012049

```
126                 dic['machinePos07'][1][0], dic['machinePos08'][1][0])
127             # Print rectangular coordinates
128             print(dic['machinePose01'][1][0], dic['machinePose02'][1][0],
                    dic['machinePose03'][1][0],
129                     dic['machinePose04'][1][0],
130                     dic['machinePose05'][1][0], dic['machinePose06'][1][0])
131             # Print user coordinates
132             print(dic['machineUserPose01'][1][0], dic['machineUserPose02'
                    ][1][0], dic['machineUserPose03'][1][0],
133                     dic['machineUserPose04'][1][0],
134                     dic['machineUserPose05'][1][0], dic['machineUserPose06'
                        ][1][0])
135             # Print the percentage of joint rated torque
136             print(dic['torque01'][1][0], dic['torque02'][1][0], dic['
                    torque03'][1][0], dic['torque04'][1] [0],
137                     dic['torque05'][1][0],
138                     dic['torque06'][1][0], dic['torque07'][1][0], dic['
                        torque08'][1][0])
139             # Print robot status
140             print(dic['robotState'][1][0])
141             # Print servo enable status
142             print(dic['servoReady'][1][0])
143             # Print synchronization status
144             print(dic['can_motor_run'][1][0])
145             # Print the motor speed of each axis
146             print(dic['motor_speed01'][1][0], dic['motor_speed02'][1][0],
                    dic['motor_speed03'][1][0],
147                     dic['motor_speed04'][1][0], dic['motor_speed05'][1][0],
148                     dic['motor_speed06'][1][0], dic['motor_speed07'][1][0],
                        dic['motor_speed08'][1][0])
149             # Print robot mode
150             print(dic['robotMode'][1][0])
151             # Print analog input port data
152             print(dic['analog_ioInput01'][1][0], dic['analog_ioInput02'
                    ][1][0], dic['analog_ioInput03'][1][0])
153             # Print analog output port data
154             print(dic['analog_ioOutput01'][1][0], dic['analog_ioOutput02'
                    ][1][0], dic['analog_ioOutput03'][1][0],
155                     dic['analog_ioOutput04'][1][0], dic['analog_ioOutput05'
                        ][1][0])
156             # Print the binary form of the digital input port data
157             print(bin(dic['digital_ioInput'][1][0])[2:].zfill(64))
158             # Print the binary form of the digital output port data
```


DN:T202012049

```
159         print(bin(dic['digital_ioOutput'][1][0])[2:].zfill(64))
160         # Print collision alarm status
161         print(dic["collision"][1][0])
162         # Print the center pose of the flange in the Cartesian
                coordinate system
163         print(dic['machineFlangePose01'][1][0], dic['
                machineFlangePose02'][1][0], dic['machineFlangePose03'
                ][1][0],
164                 dic['machineFlangePose04'][1][0],
165                 dic['machineFlangePose05'][1][0], dic['
                        machineFlangePose06'][1][0])
166         # Print the center pose of the flange in the user coordinate
                system
167         print(dic['machineUserFlangePose01'][1][0], dic['
                machineUserFlangePose02'][1][0],
168                 dic['machineUserFlangePose03'][1][0], dic['
                        machineUserFlangePose04'][1][0],
169                 dic['machineUserFlangePose05'][1][0], dic['
                        machineUserFlangePose06'][1][0])
170         # Print whether it is currently in emergency stop state
171         print(dic["emergencyStopState"][1][0])
172         # Print tcp movement speed
173         print(dic["tcp_speed"][1][0])
174         # Print the speed of each joint under joint motion
175         print(dic['joint_speed01'][1][0], dic['joint_speed02'][1][0],
176                 dic['joint_speed03'][1][0], dic['joint_speed04'][1][0],
177                 dic['joint_speed05'][1][0], dic['joint_speed06'][1][0],
178                 dic['joint_speed07'][1][0], dic['joint_speed08'][1][0])
179     index = index + 1
180     output = ""
181     dic = {}
182     data = ""
183     time.sleep(0.1)
184 s.close()
```

Chapter 4 Log Interface

Users can connect to the robot log interface through the socket client.

After connecting, enter all to enter all logs; enter a number, such as 10, to output the last 10 lines of logs; enter exit to exit the connection.

CAUTION



This function is applicable to 2.14.0 and above.

4.1 Example



```
1  import socket
2  HOST = "192.168.1.205"
3  PORT = 8058
4
5  s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
6  s.settimeout(2)
7  s.connect((HOST,PORT))
8  # New file or empty file content
9  file = open(r'D:\205\log\all_err_log.md','w').close()
10
11 # Get the last 10 log entries, the number sent corresponds to the
       number of log entries obtained
12 str1 = "10\n"
13 s.send(str1.encode())
14 while True:
15     try:
16         data = s.recv(1024)
17         with open(r'D:\205\log\err_log_1.md','a+') as f:
18             f.write(data.decode())
19         print(data.decode())
20     except(Exception):
21         break
22 s.close()
```

Chapter 5 Raw Log Interface

Users can connect to the raw log interface of the robot through the socket client.

After connecting, enter all to enter all logs; enter a number, such as 10, to output the last 10 lines of logs; enter exit to exit the connection.

CAUTION



This function is applicable to 2.14.0 and above.

5.1 Example



```
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))
# New file or empty file content
file = open(r'D:\205\log\err_log_1.md','w').close()

# Get the last 10 log entries, the number sent corresponds to the
   number of log entries obtained
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()
```