PULSE
COLLABORATIVE ROBOT

API REFERENCE GUIDE

ROZUM
ROBOTICS

TABLE OF CONTENTS

WARNING SIGNS AND THEIR MEANINGS 4
1. GENERAL DATA 4
Glossary 5
2. DESCRIPTION OF API FUNCTIONS 6
2.1. GET requests 6
2.1.1. Getting the actual arm position 6
2.1.2. Getting the actual motion status 7
2.1.3. Getting the actual status of servo motors 8
2.1.4. Getting the actual arm pose 9
2.1.5. Getting actual tool properties 10
2.1.6. Getting the actual position of the arm base 11
2.1.7. Getting the arm ID 12
2.1.8. Getting the signal level on a digital output 13
2.1.9. Getting the signal level on a digital input 14
2.2. PUT REQUESTS 15
2.2.1. Setting a new arm position 15
2.2.2. Setting a new arm pose 17
2.2.3. Ask the arm to open the gripper 19
2.2.4. Asking the arm to close the gripper 20
2.2.5. Asking the robot to relax 21
2.2.6. Asking the arm to go to the freeze state 21
2.2.7. Asking the arm to move to a pose 22
2.2.8. Asking the arm to move to a position 24
2.2.9. Setting high signal level on a digital output 26
2.2.10. Setting low signal level on a digital output 27
2.2.11. Recovering the arm after an emergency 28
2.2.12. Setting the arm into a transportation pose 28
2.3. POST requests 29
2.3.1. Setting tool properties 29
2.3.2. Setting a new zero point position 31

ANNEX 1. RESPONSE/ REQUEST SCHEMAS 33
Position schema 33
Pose schema 33
Motor status array schema 33
Tool schema 34

WARNING SIGNS AND THEIR MEANINGS

Below are the warning symbols used throughout the manual and explanations of their meanings.

The sign denotes important information that is not directly related to safety, but that the user should be aware of.

The sign indicates important safety precautions the user should follow.

1. GENERAL DATA

The REST Application Programming Interface (API) described in this reference guide implements the functionality for monitoring and controlling motion of the PULSE robotic arm (also, robotic arm or arm) and its work tool (also, tool). API requests are in the JSON format; API responses are in the JSON and/or in the plain text format. All returned values are either double numbers or text strings.

API access for reading and writing motion parameters is based on the HTTP (v 2.0) methods listed in Table 1-1.

Table 1-1: Supported HTTP methods

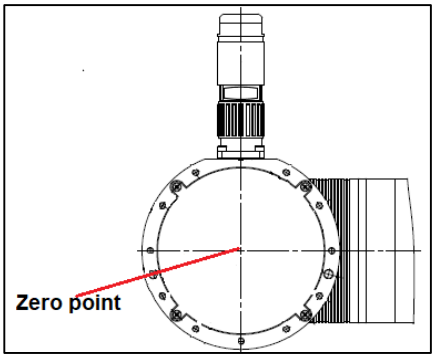
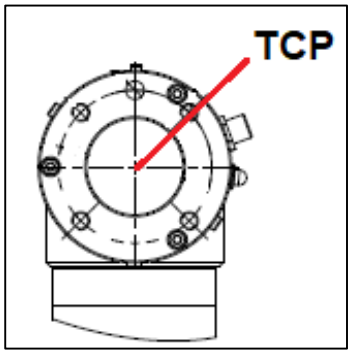
Method	Purpose
GET	• to get the actual position of the robotic arm (rotation angles and coordinates of its joints) • to get the actual state of the robotic arm (e.g., idle) • to get the actual state of the servo motors in the arm joints (e.g., voltage, rotor velocity) • to get actual properties (e.g., rotation angles, coordinates, radius) of the work tool • to get the actual position (rotation angles and coordinates) of the arm base • to get the unique identifier (ID) of the robotic arm • to get the signal level (HIGH or LOW) on a digital output • to get the signal level (HIGH or LOW) on a digital input
PUT	• to set/change the position (rotation angles and coordinates of its joints) of the robotic arm • to set/change the arm state (e.g., relax or freeze) • to open the gripper • to close the gripper • to set the signal level on a digital output to HIGH or LOW • to recover the arm after an error • to set the arm into a transportation pose

POST	• to set new properties (e.g., rotation angles, coordinates) of the work tool • to set a new position (rotation angles and coordinates) of the arm base

Glossary

Table 1-2 lists and defines essential terms used throughout the reference guide.

Table 1-2: Essential REST API terms

Term	Definition
Zero point	It is the origin point for measuring distances along the *x*, *y*, and *z* coordinate axes. Its original physical location is at the center of the arm base as shown below.  *It is possible to change the zero point location using the POST/Base request (see Section 2.3.2).*
Tool center point (TCP)	It is the point, relative to which all arm positions and movements are defined. Its original physical location is at the center of the arm wrist as shown below.  *It is possible to change the TCP location using the POST/Tool request (see Section 2.3.1).*

2. DESCRIPTION OF API FUNCTIONS

The section describes in detail the **REST API functions** you can use to control the PULSE robotic arm and its work tool (a gripper) and to monitor the arm's motion parameters.

2.1. GET requests

2.1.1. Getting the actual arm position

Path: GET/position

Description: The function returns the actual position of the PULSE robotic arm, which is described as a set of *x*, *y*, and *z* coordinates, as well as *roll*, *pitch*, and *yaw* rotation angles. The coordinates define the actual distance (in meters) from the zero point of the robotic arm to the tool center point (TCP) along the *x*, *y*, and *z* axes accordingly. *Roll* stands for the TCP rotation angle around the *x* axis; *pitch*—the TCP rotation angle around the *y* axis; *yaw*—the TCP rotation angle around the *z* axis. All rotation angles are in radians and relative to the zero point.

Response content type: application/json, text/plain

Response body:

HTTP status code	Response schema/ type
200 OK	Position (see Annex 1)
500 Internal Server Error	String
503 Service Unavailable	String

Response examples:

- **200 OK**

```
{
  "point": {
    "x": 120.34,
    "y": -230.345,
    "z": 320.1
  },
  "rotation": {
    "roll": 21.34,
    "pitch": -1.345,
    "yaw": 0.1
  }
}
```

- **500 Internal Server Error**

```
"Robot does not respond"
```

- **503 Service Unavailable**

```
"Robot unavailable in emergency state"
```

2.1.2. Getting the actual motion status

Path: GET/status/motion

Description: The function returns the actual state of the robotic arm. Possible arm states are as follows:

- **IDLE**

  The arm is not in motion, but is fully functional and ready for operation.

- **ZERO_GRAVITY**

  The arm is in the zero gravity mode, which means the user can move it by hand to set a motion trajectory.

- **RUNNING**

  The arm is in motion.

- **MOTION_FAILED**

  Motion is impossible due to incorrect motion settings.

- **EMERGENCY**

  Motion is impossible due to an emergency. In this case, an emergency is a fatal failure that causes the control box to switch off and the arm to stop without retaining its position. Recovery with the **PUT/RECOVER** function is not possible.

- **ERROR**

  The arm stops moving due to an error and goes into the freeze mode, retaining its last position. The user can recover the arm, using the **PUT/RECOVER** function.

Response content type: text/plain

Response body:

HTTP status code	Response schema/ type
200 OK	String enum: [IDLE, ZERO_GRAVITY, RUNNING, MOTION_FAILED, EMERGENCY, ERROR]
500 Internal Server Error	String
503 Service Unavailable	String

Response examples:

- **200 OK**

```
"IDLE"
```

- **500 Internal Server Error**

```
"Robot does not respond"
```

- **503 Service Unavailable**

```
"Robot unavailable in emergency state"
```

2.1.3. Getting the actual status of servo motors

Path: GET/status/motors

Description: The function returns the actual states of the six servo motors integrated into the joints of the robotic arm. The states are described as an array of six objects—one for each servo motor. Each object includes the following properties:

- **Angle**—the actual angular position (in degrees) of the servo's output flange
- **Rotor velocity**—the actual rotor velocity (in RPM)
- **RMS current**—the actual input current (in Amperes)
- **Phase current**—the actual magnitude of alternating current (in Amperes)
- **Supply voltage**—the actual supply voltage (in Volts)
- **Stator temperature**—the actual temperature (in degrees C) as measured on the stator winding
- **Servo temperature**—the actual temperature (in degrees C) as measured on the MCU PCB
- **Velocity setpoint**—the user-preset rotor velocity (in RPM)
- **Velocity output**—the motor control current (in Amperes) based on the preset velocity
- **Velocity feedback**—the actual rotor velocity (in RPM)
- **Velocity error**—the difference between the preset and the actual rotor velocities (in RPM)
- **Position setpoint**—the user-preset position of the servo flange in degrees
- **Position output**—rotor velocity (in RPM) based on the position setpoint
- **Position feedback**—the actual position of the servo flange (in degrees) based on the encoder feedback
- **Position error**—the difference between the preset and the actual positions of the servo flange (in degrees)

Response content type: application/json, text/plain

Response body:

HTTP status code	Response schema/ type
200 OK	Motor status array (see Annex 1)
500 Internal Server Error	String
503 Service Unavailable	String

Response examples:

- **200 OK**


```
[
  {
    "angle": 168.89699,
    "rotorVelocity": -0.00064343837,
    "rmsCurrent": 0.01,
    "voltage": 47.795017,
    "phaseCurrent": 0.01,
    "statorTemperature": 27.990631,
    "servoTemperature": 31.739925,
    "velocityError": -0.022674553,
    "velocitySetpoint": -0.02331799,
    "velocityOutput": 0.01,
    "velocityFeedback": -0.00064343837,
    "positionError": 0.0385437,
    "positionSetpoint": 168.93799,
    "positionOutput": 0.01,
    "positionFeedback": 168.89944
  }
]
```


Note: The example is one object containing properties for a single servo. In reality, the array in the response includes six similar objects.

- **500 Internal Server Error**

```
"Robot does not respond"
```

- **503 Service Unavailable**

```
"Robot unavailable in emergency state"
```

2.1.4. Getting the actual arm pose

Path: GET/pose

Description: The function returns the actual pose of the robotic arm. The pose is described as a set of output flange angles (in degrees) of all the six servos in the arm joints.

Response content type: application/json, text/plain

Response body:

HTTP status code	Response schema/ type
200 OK	Pose (see Annex 1)
500 Internal Server Error	String
503 Service Unavailable	String

Response examples:

- **200 OK**

```
{
  "angles": [
    0,
    3.14159,
    1.57,
    -1.57,
    3.14,
    0
  ]
}
```

- **500 Internal Server Error**

```
"Robot does not respond"
```

- **503 Service Unavailable**

```
"Robot unavailable in emergency state"
```

2.1.5. Getting actual tool properties

Path: GET/tool

Description: The function returns the actual TCP position that accounts for the offset from the original TCP (see **Glossary**) due to attaching/changing the work tool. The actual TCP position is described as a set of the following properties:

- **name**—any random name of the work tool defined by the user (e.g., “gripper”)
- **position—***x*, *y*, and *z* coordinates, as well as *roll*, *pitch*, and *yaw* rotation angles. The coordinates define the distance (in meters) from the arm's zero point to the actual TCP along the *x*, *y*, and *z* axes accordingly. *Roll* stands for the actual TCP rotation angle around the *x* axis; *pitch*—the actual TCP rotation angle around the *y* axis; *yaw*—the actual TCP rotation angle around the *z* axis. All rotation angles are in radians and relative to the physical center point of the arm base.
- **radius**—radius of the work tool (in meters) measured from its center point.

Response content type: application/json, text/plain

Response body:

HTTP status code	Response schema/ type
200 OK	Tool (see Annex 1)
500 Internal Server Error	String
503 Service Unavailable	String

Response examples:

- **200 OK**

```
{
  "name": "gripper",
  "point": {
    "x": 120.34,
    "y": -230.345,
    "z": 320.1
  },
  "rotation": {
    "roll": 21.34,
    "pitch": -1.345,
    "yaw": 0.1
  },
  "radius": 0.5
}
```

- **500 Internal Server Error**

```
"Robot does not respond"
```

- **503 Service Unavailable**

```
"Robot unavailable in emergency state"
```

2.1.6. Getting the actual position of the arm base

Path: GET/base

Description: The function returns the actual position of the arm's zero point in the user environment. The actual zero point position is described as a set of *x*, *y*, and *z* coordinates, as well as *roll*, *pitch*, and *yaw* rotation angles. The coordinates define the offset (in meters) from the physical center point of the arm base (original zero point) to the actual zero point position along the *x*, *y*, and *z* axes accordingly. *Roll* stands for the rotation angle around the *x* axis; *pitch*—the rotation angle around the *y* axis; *yaw*—the rotation angle around the *z* axis. All rotation angles are in radians and relative to the physical center point of the arm base.

Response content type: application/json, text/plain

Response body:

HTTP status code	Response schema/ type
200 OK	Position (see Annex 1)
500 Internal Server Error	String
503 Service Unavailable	String

Response examples:

- **200 OK**

```
{
  "point": {
    "x": 120.34,
    "y": -230.345,
    "z": 320.1
  },
  "rotation": {
    "roll": 21.34,
    "pitch": -1.345,
    "yaw": 0.1
  }
}
```

- **500 Internal Server Error**

```
"Robot does not respond"
```

- **503 Service Unavailable**

```
"Robot unavailable in emergency state"
```

2.1.7. Getting the arm ID

Path: GET/robot/id

Description: The function returns the unique identifier (ID) of the robotic arm. The ID is an alphanumeric designation that consists of individual servo motor identifications.

Response content type: text/plain

Response body:

HTTP status code	Response schema/ type
200 OK	String
500 Internal Server Error	String
503 Service Unavailable	String

Response examples:

- **200 OK**

```
"1346466AFG872"
```

- **500 Internal Server Error**

```
"Robot does not respond"
```

- **503 Service Unavailable**

```
"Robot unavailable in emergency state"
```

2.1.8. Getting the signal level on a digital output

Path: GET/signal/output/{port}

Description: The function returns the actual signal level on the digital output specified in the *{port}* parameter of the request path.

A digital output is a physical port on the back panel of the control box. Since the control box has two digital outputs, the parameter value can be either *1* (corresponds to Relay output 1) or *2* (corresponds to Relay output 2).

Possible return values are as follows:

- LOW—default user-defined state (e.g., LED off)
- HIGH—change of the user defined state (e.g., LED on)

Note: For location of digital outputs and their detailed description, refer to the document "PULSE robotic arm. Hardware Installation Manual."

Response content type: text/plain

Response body:

HTTP status code	Response schema/ type
200 OK	string enum: [HIGH, LOW]
412 Precondition Failed	String
500 Internal Server Error	String
503 Service Unavailable	String

Response examples:

- **200 OK**

```
"HIGH,"
"LOW"
```

- **412 Precondition Failed**

```
"Unreachable Position",
"Collision detected"
```

- **500 Internal Server Error**

```
"Robot does not respond"
```

- **503 Service Unavailable**

```
"Robot unavailable in emergency state"
```

2.1.9. Getting the signal level on a digital input

Path: GET/signal/input/{port}

Description: The function returns the actual signal level on the digital input specified in the *{port}* parameter of the request path.

A digital input is a physical port on the back panel of the control box. Since the control box has four digital inputs (DI), the parameter can have any integral value between *1* (corresponds to DI1) and *4* (corresponds to DI4).

Possible return values are as follows:

- LOW—default user-defined state
- HIGH—change of the user defined state

Note: For location of the digital outputs and their detailed description, refer to the document "PULSE robotic arm. Hardware Installation Manual."

Response content type: text/plain

Response body:

HTTP status code	Response schema/ type
200 OK	string enum: [HIGH, LOW]
412 Precondition failed	String
500 Internal Server Error	String
503 Service Unavailable	String

Response examples:

- **200 OK**

```
"HIGH",
"LOW"
```

- **412 Precondition Failed**

```
"Unreachable Position",
"Collision detected"
```

- **500 Internal Server Error**

```
"Robot does not respond"
```

- **503 Service Unavailable**

```
"Robot unavailable in emergency state"
```

2.2. PUT REQUESTS

2.2.1. Setting a new arm position

Path: PUT/position

Description: The function commands the arm to move to a new position. The position is described as a set of *x*, *y*, and *z* coordinates, as well as *roll*, *pitch*, and *yaw* rotation angles. The coordinates define the desired distance (in meters) from the zero point to the TCP along the *x*, *y*, and *z* axes accordingly. *Roll* stands for the desired TCP rotation angle around the *x* axis; *pitch*—the desired TCP rotation angle around the *y* axis; *yaw*—the desired TCP rotation angle around the *z* axis. All rotation angles are in radians.

Request content type: application/json

Request parameters:

Parameter	Description
speed	The parameter sets the speed (in % max speed) at which the arm should move to the required position. The admissible value range is from 1 to 100.  *Specifying the speed parameter is mandatory. Otherwise, an error is generated.* **Type:** *number (double)* **Included as:** *query*
type	The parameter sets the type of motion the arm should use to get to the required position. Admissible values are as follows: • **JOINT** When set to this motion type, the arm moves to the specified position along a trajectory that has been calculated as the most convenient one. The trajectory can be described as a set of joint angles connected into a curve. • **LINEAR** When the motion type is LINEAR, the arm moves to the specified position along a straight line. This motion takes more time than with the *type* parameter set to JOINT. However, the trajectory is entirely predictable, unlike with the JOINT type motion. When the user specifies no value for the parameter, it is set to the default one—JOINT. **Included as:** *query*

tcp_max_velocity	The parameter defines the limit velocity in meters per second that an end effector can reach at its TCP while moving. It is not mandatory. When the user specifies no value for it, it is set to default. The default setting is 2 m/s. The admissible value range is from 0.001 to 2 m/s. **Included as:** query

Request body: The request body is in accordance with the **Position schema**. It specifies the coordinates (*x*, *y*, *z*) and the rotation angles (*roll*, *pitch*, *yaw*) that describe the required TCP position.

Make sure to specify all point (x, y, z coordinates) and rotation (roll, pitch, yaw) properties in the request body. When at least one of the properties is not specified, the function returns a 400 Bad Request error.

Request example:

```
{
  "point": {
    "x": 120.34,
    "y": -230.345,
    "z": 320.1
  },
  "rotation": {
    "roll": 21.34,
    "pitch": -1.345,
    "yaw": 0.1
  }
}
```

Response content type: application/json, text/plain

Response body:

HTTP status code	Description	Response schema/ type
200 OK	Actual arm position	Position
400 Bad Request	Message parsing error	String
412 Precondition Failed	Incorrect input parameters	String
500 Internal Server Error	Arm error	String
503 Service unavailable	Arm emergency	String

Response examples:

- **200 OK**

```
{
  "point": {
    "x": 120.34,
    "y": -230.345,
    "z": 320.1
  },
  "rotation": {
    "roll": 21.34,
    "pitch": -1.345,
    "yaw": 0.1
  }
  }
```

- **400 Bad Request**

```
"Incorrect format of input Message"
```

- **412 Precondition Failed**

```
"Unreachable Position",
"Collision detected"
```

- **500 Internal Server Error**

```
"Robot does not respond"
```

- **503 Service Unavailable**

```
"Robot unavailable in emergency state"
```

2.2.2. Setting a new arm pose

Path: PUT/pose

Description: The function commands the arm to move to a new pose. A pose is described as a set of output flange angles (in degrees) of the six servos integrated into the arm joints.

Request body: The request body is in accordance with the **Pose schema** (see Annex 1). It specifies the angles that each of the six servos should reach to move the arm to the required pose.

Request type: application/json

Request parameters:

Parameter	Description
speed	The parameter sets the speed (in % max. speed) at which servos should move to the required angles. The admissible value range is from 1 to 100.  *Specifying the speed parameter is mandatory. Otherwise, an error is generated.* **Type:** *number (double)* **Included as:** *query*

type	The parameter sets the type of motion the arm should use to get into the specified pose. Admissible values are as follows: • **JOINT** When set to this motion type, the arm moves to the specified pose along a trajectory that has been calculated as the most convenient one. The trajectory can be described as a set of joint angles connected into a curve. • **LINEAR** When the motion type is LINEAR, the arm moves to the specified pose along a straight line. This motion takes more time than with the *type* parameter set to JOINT. However, the trajectory is entirely predictable, unlike with the JOINT type motion. When the user specifies no value for the parameter, it is set to the default one—JOINT. **Included as:** *query*
tcp_max_velocity	The parameter defines the limit velocity in meters per second that an end effector can reach at its TCP while moving. It is not mandatory. When the user specifies no value for it, it is set to default. The default setting is 2 m/s. The admissible value range is from 0.001 to 2 m/s. **Included as:** query

Request example:

```
{
  "angles": [
    0,
    3.14159,
    1.57,
    -1.57,
    3.14,
    0
  ]
}
```

Response body:

HTTP status code	Description	Response schema/ type
200 OK	Success	-
400 Bad Request	Message parsing error	String
412 Precondition Failed	Incorrect input parameters	String
500 Internal Server Error	Arm error	String
503 Service unavailable	Arm emergency	String

Response examples:

- **400 Bad Request**

```
"Incorrect format of input Message"
```

- **412 Precondition Failed**

```
"Unreachable Position",
"Collision detected"
```

- **500 Internal Server Error**

```
"Robot does not respond"
```

- **503 Service Unavailable**

```
"Robot unavailable in emergency state"
```

2.2.3. Ask the arm to open the gripper

Path: PUT/gripper/open

Description: The function commands the arm to open the gripper. It has no request body, but the user can optionally set one parameter—timeout.

Request parameter:

Parameter	Description
timeout	The parameter specifies how long (in milliseconds) the arm should remain idle, waiting for the gripper to open. The default manufacturer-preset value is 500 ms. *Setting the parameter, it is recommended to use values above 0 ms.* **Type:** *number (int32)* **Included as:** *query*

Response content type: text/plain

Response body:

HTTP status code	Response schema/ type
200 OK	-
500 Internal Server Error	String
503 Service Unavailable	String

Response examples:

- **500 Internal Server Error**

```
"Robot does not respond"
```

- **503 Service Unavailable**

```
"Robot unavailable in emergency state"
```

2.2.4. Asking the arm to close the gripper

Path: PUT/gripper/close

Description: The function commands the arm to close the gripper. It has no request body, but the user can optionally set one parameter—timeout.

Request parameter:

Parameter	Description
timeout	The parameter specifies how long (in milliseconds) the arm should remain idle, waiting for the gripper to close. The default manufacturer-preset value is 500 ms.  *Setting the parameter, it is recommended to use values above 0 ms.* **Type:** *number (int32)* **Included as:** *query*

Response content type: text/plain

Response body:

HTTP status code	Response schema/ type
200 OK	String
500 Internal Server Error	String
503 Service Unavailable	String

Response examples:

- **500 Internal Server Error**

```
"Robot does not respond"
```

- **503 Service Unavailable**

```
"Robot unavailable in emergency state"
```

2.2.5. Asking the robot to relax

Path: PUT/relax

Description: The function sets the arm in the "relaxed" state. The arm stops moving without retaining its last position. In this state, the user can move the robotic arm by hand (e. g., to verify/ test a motion trajectory).

Response content type: text/plain

Response body:

HTTP status code	Response schema/ type
200 OK	-
500 Internal Server Error	String
503 Service Unavailable	String

Response examples:

- **500 Internal Server Error**

```
"Robot does not respond"
```

- **503 Service Unavailable**

```
"Robot unavailable in emergency state"
```

2.2.6. Asking the arm to go to the freeze state

Path: PUT/freeze

Description: The function sets the arm in the “freeze” state. The arm stops moving, retaining its last position.

In the state, it is not advisable to move the arm by hand as this can cause damage to its components.

Response content type: text/plain

Response body:

HTTP status code	Response schema/ type
200 OK	-
500 Internal Server Error	String
503 Service Unavailable	String

Response examples:

- **500 Internal Server Error**

```
"Robot does not respond"
```

- **503 Service Unavailable**

```
"Robot unavailable in emergency state"
```

2.2.7. Asking the arm to move to a pose

Path: PUT/poses/run

Description: The function allows for setting a trajectory of one or more waypoints to move the robotic arm smoothly from one pose to another. In the trajectory, each waypoint is a set of output flange angles (in degrees) of the six servos in the arm joints.

Note: Similarly, you can move the arm from one pose to another through one or more waypoints using the **PUT/pose** function. When the arm is executing a trajectory of **PUT/pose** waypoints, it stops for a short moment at each preset waypoint. With the **PUT/poses/run** function, the arm moves smoothly though all waypoints without stopping, which reduces the overall time of going from one pose to another.

Request body: The request body is in accordance with the **Pose schema** (see Annex 1). It specifies the angles that each of the six servos should reach to move the arm through a number of waypoints to a new pose.

Request content type: application/json

Request parameters:

Parameter	Description
speed	The parameter sets the speed (in % max. speed) at which servos should move to the required angles. The admissible value range is from 1 to 100.  *Specifying the speed parameter is mandatory. Otherwise, an error is generated.* **Type:** *number (double)* **Included as:** *query*
type	The parameter sets the type of motion the arm should use to get to the specified pose through one or more waypoints. Admissible values are as follows: • **JOINT** When set to this motion type, the arm moves from one waypoint to another along a trajectory that has been calculated as the most convenient one. The trajectory can be described as a set of joint angles connected into a curve. • **LINEAR** When the motion type is LINEAR, the arm moves from one waypoint to another along a straight line. This motion takes more time than with the *type* parameter set to JOINT. However, the trajectory is entirely predictable, unlike with the JOINT type motion. When the user specifies no value for the parameter, it is set to the default one—JOINT. **Included as:** *query*

tcp_max_velocity	The parameter defines the limit velocity in meters per second that an end effector can reach at its TCP while moving. It is not mandatory. When the user specifies no value for it, it is set to default. The default setting is 2 m/s. The admissible value range is from 0.001 to 2 m/s. **Included as:** query

Request example:

```
[
  {
    "angles": [
      0,
      3.14159,
      1.57,
      -1.57,
      3.14,
      0
    ]
  }
]
```

Response body:

HTTP status code	Description	Response schema/ type
200 OK	Success	-
400 Bad Request	Message parsing error	String
412 Precondition Failed	Incorrect input parameters	String
500 Internal Server Error	Arm error	String
503 Service Unavailable	Arm emergency	String

Response content type: text/plain

Response examples:

- **400 Bad Request**

```
"Incorrect format of input Message"
```

- **412 Precondition Failed**

```
"Unreachable Pose",
"Collision detected"
```

- **500 Internal Server Error**

```
"Robot does not respond"
```

- **503 Service Unavailable**

```
"Robot unavailable in emergency state"
```

2.2.8. Asking the arm to move to a position

Path: PUT/positions/run

Description: The function allows for setting a trajectory of one or more waypoints to move the robotic arm smoothly from one position to another. In the trajectory, each waypoint is described as a set of *x*, *y*, and *z* coordinates, as well as *roll*, *pitch*, and *yaw* rotation angles. The coordinates define the desired distance (in meters) from the zero point to the TCP along the *x*, *y*, and *z* axes accordingly. *Roll* stands for the desired TCP rotation angle around the *x* axis; *pitch*—the desired TCP rotation angle around the *y* axis; *yaw*—the desired TCP rotation angle around the *z* axis. All rotation angles are in radians.

Note: Similarly, you can move the arm from one position to another through one or more waypoints using the **PUT/position** request. When the arm is executing a trajectory of **PUT/position** waypoints, it stops for a short moment at each preset waypoint. With the **PUT/positions/run** function, the arm moves smoothly though all waypoints without stopping, which reduces the overall time of going from one position to another.

Request body: The request body is in accordance with the **Position schema** (see Annex 1). It specifies the coordinates (*x*, *y*, *z*) and rotation angles (*roll*, *pitch*, *yaw*) of all the waypoints on the trajectory from the initial TCP position to the required one.

Make sure to specify all point (x, y, z) and rotation (roll, pitch, yaw) properties in the request body. Otherwise, the function returns a 400 Bad Request error.

Request type: application/json

Request parameters:

Parameter	Description
speed	The parameter sets the speed (in % max speed) at which the arm should move to the required position. The admissible value range is from 1 to 100. *Specifying the speed parameter is mandatory. Otherwise, an error is generated.* **Type:** *number (double)* **Included as:** *query*

type	The parameter sets the type of motion the arm should use to get to the specified position through one or more waypoints. Admissible values are as follows: • **JOINT** When set to this motion type, the arm moves from one waypoint to another along a trajectory that has been calculated as the most convenient one. The trajectory can be described as a set of joint angles connected into a curve. • **LINEAR** When the motion type is LINEAR, the arm moves from one waypoint to another along a straight line. This motion takes more time than with the *type* parameter set to JOINT. However, the trajectory is entirely predictable, unlike with the JOINT type motion. When the user specifies no value for the parameter, it is set to the default one—JOINT. **Included as:** *query*
tcp_max_velocity	The parameter defines the limit velocity in meters per second that an end effector can reach at its TCP while moving. It is not mandatory. When the user specifies no value for it, it is set to default. The default setting is 2 m/s. The admissible value range is from 0.001 to 2 m/s. **Included as:** query

Request example:

```
[
  {
    "point": {
      "x": 120.34,
      "y": -230.345,
      "z": 320.1
    },
    "rotation": {
      "roll": 21.34,
      "pitch": -1.345,
      "yaw": 0.1
    }
  }
]
```

Response body:

HTTP status code	Description	Response schema/ type
200 OK	Success	-
400 Bad Request	Message parsing error	String
412 Precondition Failed	Incorrect input parameters	String

500 Internal Server Error	Arm error	String
503 Service Unavailable	Arm emergency	String

Response examples:

- **400 Bad Request**

```
"Incorrect format of input Message"
```

- **412 Precondition Failed**

```
"Unreachable Position",
"Collision detected"
```

- **500 Internal Server Error**

```
"Robot does not respond"
```

- **503 Service Unavailable**

```
"Robot unavailable in emergency state"
```

2.2.9. Setting high signal level on a digital output

Path: PUT/signal/output/{port}/high

Description: The function sets the digital output specified in the *{port}* parameter of the request path to the HIGH signal level.

A digital output is a physical port on the back panel of the control box. Since the control box has two digital outputs, the parameter value can be either *1* (corresponds to Relay output 1) or *2* (corresponds to Relay output 2). For location of the digital outputs and their detailed description, refer to the document "PULSE robotic arm. Hardware Installation Manual."

Response content type: text/plain, application/json

Response body:

HTTP status code	Response schema/ type
200 OK	-
500 Internal Server Error	String
503 Service Unavailable	String

Response examples:

- **200 OK**
- **412 Precondition Failed**

```
"Unreachable Position",
"Collision detected"
```

- **500 Internal Server Error**

```
"Robot does not respond"
```

- **503 Service Unavailable**

```
"Robot unavailable in emergency state"
```

2.2.10. Setting low signal level on a digital output

Path: PUT /signal/output/{port}/low

Description: The function sets the digital output specified in the *{port}* parameter of the request path to the LOW signal level.

A digital output is a physical port on the back panel of the control box. Since the control box has two digital outputs, the parameter value can be either *1* (corresponds to Relay output 1) or *2* (corresponds to Relay output 2). For location of the digital outputs and their detailed description, refer to the document "PULSE robotic arm. Hardware Installation Manual."

Response content type: text/plain, application/json

Response body:

HTTP status code	Response schema/ type
200 OK	String
500 Internal Server Error	String
503 Service Unavailable	String

Response examples:

- **200 OK**

- **412 Precondition Failed**

```
"Unreachable Position",
"Collision detected"
```

- **500 Internal Server Error**

```
"Robot does not respond"
```

- **503 Service Unavailable**

```
"Robot unavailable in emergency state"
```

2.2.11. Recovering the arm after an emergency

Path: PUT/recover

Description: The function recovers the arm after an emergency, setting its motion status to IDLE. Recovery is possible only after an emergency that is not fatal—corresponding to the ERROR status (see **GET/status/motion**).

With the 200 OK status code, the function returns either of two values:

- SUCCESS—the recovery has been completed as appropriate
- FAILED—the recovery has failed

Response content type: text/plain

Response body:

HTTP status code	Response schema/ type
200 OK	String enum: [SUCCESS, FAILED]
500 Internal Server Error	String
503 Service Unavailable	String

Response examples:

- **200 OK**

```
"SUCCESS",
"FAILED"
```

- **500 Internal Server Error**

```
"Robot does not respond"
```

- **503 Service Unavailable**

```
"Robot unavailable in emergency state"
```

2.2.12. Setting the arm into a transportation pose

Path: PUT/pack

Description: The function sets the arm into a preset pose for transportation.

Response content type: text/plain

Response body:

HTTP status code	Response schema/ type
200 OK	String
500 Internal Server Error	String
503 Service Unavailable	String

Response examples:

- **200 OK**
- **500 Internal Server Error**

```
"Robot does not respond"
```

- **503 Service Unavailable**

```
"Robot unavailable in emergency state"
```

2.3. POST requests

2.3.1. Setting tool properties

Path: POST/tool

Description: The function enables setting new TCP to account for the properties of an attached or changed work tool. The tool properties define the following:

- **Name**—any random name of the work tool defined by the user (e.g., "gripper")
- **Position**—a set of *x*, *y*, and *z* coordinates and *roll*, *pitch*, and *yaw* rotation angles. The coordinates define the desired distance (in meters) from the arm's zero point to the new TCP along the *x*, *y*, and *z* axes accordingly. *Roll* stands for the rotation angle of the new TCP around the *x* axis; *pitch*—the rotation angle around the *y* axis; *yaw*—the rotation angle of the new TCP around the *z* axis. All rotation angles are in radians.
- **Radius**—radius of the work tool (in meters) measured from its physical center point

Request content type: application/json

Request body: The request body is in accordance with the **Tool schema** (see Annex 1).

Request example:

```
{
  "name": "gripper",
  "point": {
    "x": 120.34,
    "y": -230.345,
    "z": 320.1
  },
  "rotation": {
    "roll": 21.34,
    "pitch": -1.345,
    "yaw": 0.1
  },
  "radius": 0.5
}
```

Response content type: application/json, text/plain

Response body:

HTTP status code	Description	Response schema/ type
200 OK	Success	String
400 Bad Request	Message parsing error	String
412 Precondition Failed	Incorrect input parameters	String
500 Internal Server Error	Arm error	String
503 Service Unavailable	Arm emergency	String

Response examples:

- **200 OK**

```
{
  "name": "string",
  "point": {
    "x": 120.34,
    "y": -230.345,
    "z": 320
  },
  "rotation": {
    "roll": 21.34,
    "pitch": -1.345,
    "yaw": 0
  },
  "radius": "number (double)"
}
```

- **400 Bad Request**

```
"Incorrect format of input Message"
```

- **412 Precondition Failed**

```
"Unreachable Position",
"Collision detected"
```

- **500 Internal Server Error**

```
"Robot does not respond"
```

- **503 Service Unavailable**

```
"Robot unavailable in emergency state"
```

2.3.2. Setting a new zero point position

Path: POST/base

Description: The function enables setting a new zero point position of the robotic arm as required for the current user environment (e.g., considering the surrounding equipment). The new zero point position is described as a set of *x*, *y*, and *z* coordinates, as well as *roll*, *pitch*, and *yaw* rotation angles. The coordinates define the desired offset (in meters) from the physical center point of the arm base (original zero point) along the *x*, *y*, and *z* axes accordingly. *Roll* stands for the rotation angle around the *x* axis; *pitch*—the rotation angle around the *y* axis; *yaw*—the rotation angle around the *z* axis. All rotation angles are in radians and relative to the physical center point of the arm base.

Request content type: application/json

Request body: The request body is in accordance with the Position schema (see Annex 1). It specifies the coordinates and rotation angles of the new zero point.

Request example:

```
{
  "point": {
    "x": 120.34,
    "y": -230.345,
    "z": 320.1
  },
  "rotation": {
    "roll": 21.34,
    "pitch": -1.345,
    "yaw": 0.1
  }
}
```

Response content type: application/json, text/plain

Response body:

HTTP status code	Description	Response schema/ type
200 OK	Success	String

400 Bad Request	Message parsing error	String
412 Precondition Failed	Incorrect input parameters	String
500 Internal Server Error	Arm error	String
503 Service Unavailable	Arm emergency	String

Response examples:

- **200 OK**

```
{
  "point": {
    "x": 120.34,
    "y": -230.345,
    "z": 320.1
  },
  "rotation": {
    "roll": 21.34,
    "pitch": -1.345,
    "yaw": 0.1
  }
}
```

- **400 Bad Request**

```
"Incorrect format of input Message"
```

- **412 Precondition Failed**

```
"Unreachable Position",
"Collision detected"
```

- **500 Internal Server Error**

```
"Robot does not respond"
```

- **503 Service Unavailable**

```
"Robot unavailable in emergency state"
```

ANNEX 1. RESPONSE/ REQUEST SCHEMAS

The Annex contains schemas for structuring the API requests and responses described in the above sections.

Position schema

Schema property	Property content	Examples

Point	x: number (double)
	y: number (double)
	z: number (double)
		{
		"x": "number (double)",
		"y": "number (double)",
		"z": "number (double)"
		}

Rotation	roll: number (double)
	pitch: number (double)
	yaw: number (double)
		{
		"roll": "number (double)",
		"pitch": "number (double)",
		"yaw": "number (double)"
		}

Pose schema

Schema property	Property content	Example

Angle	Number (double)	{
		"angles": [
		"number (double)",
		"number (double)",
		"number (double)",
		"number (double)",
		"number (double)",
		"number (double)"
		]
		}

Motor status array schema

Schema property	Property content	Example
Angle	Number (double)	{ "angle": "number (double)", "rotorVelocity": "number (double)", "rmsCurrent": "number (double)", "voltage": "number (double)", "phaseCurrent": "number (double)", "statorTemperature": "number (double)", "servoTemperature": "number (double)", "velocityError": "number (double)",
Rotor velocity	Number (double)	
RMS current	Number (double)	
Voltage	Number (double)	
Phase current	Number (double)	
Stator temperature	Number (double)	
Servo temperature	Number (double)	

Velocity error	Number (double)	"velocitySetpoint": "number (double)", "velocityOutput": "number (double)", "velocityFeedback": "number (double)", "positionError": "number (double)", "positionSetpoint": "number (double)", "positionOutput": "number (double)", "positionFeedback": "number (double)", }
Velocity setpoint	Number (double)	
Velocity output	Number (double)	
Velocity feedback	Number (double)	
Position error	Number (double)	
Position setpoint	Number (double)	
Position output	Number (double)	
Position feedback	Number (double)	

Tool schema

Schema property	Property content	Examples

Name		{
		"name": "string",
		}

Point	x: number (double)
	y: number (double)
	z: number (double)
		{
		"x": "number (double)",
		"y": "number (double)",
		"z": "number (double)"
		}

Rotation	roll: number (double)
	pitch: number (double)
	yaw: number (double)
		{
		"roll": "number (double)",
		"pitch": "number (double)",
		"yaw": "number (double)"
		}

Radius	Number (double)	{
		"radius": "number (double)"
		}