PULSE

COLLABORATIVE ROBOT

API REFERENCE GUIDE

ROZUM ROBOTICS

TABLE OF CONTENTS

1. GENERAL DATA

The REST Application Programming Interface (API) described in the manual implements the functionality underlying the PULSE DESK user interface. It allows for programming the robot's motion and reading its parameters (e.g., waypoints, angles).

All API access is over HTTP (v2.0). Data is sent in the JSON format and can be received in the JSON and plain text formats.

The REST API for the PULSE robot supports the following HTTP methods:

Method	Purpose
GET	• to get the current robot position (angles and coordinates) • to get the current motion and motor parameters
PUT	• to set/change the robot position (angles and coordinates) • to set/change the robot state (e.g., relax or freeze) • to open the gripper • to close the gripper

2. DESCRIPTION OF API FUNCTIONS

The section details the **twelve API functions** you can use to control the PULSE robot and its end effector (gripper) and to monitor the robot's motion parameters.

2.1. GET requests

2.1.1. Get the current robot position

Path: GET/position

Description: The function returns the current position of the PULSE robot, which is described as a set of *x*, *y*, and *z* coordinates and the rotation angles—roll, pitch, and yaw.

The coordinates define the distance (in meters) from the robot's zero point 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 defines the TCP rotation angle around the *y* axis. Yaw is the TCP rotation angle around the *z* axis. All of the rotation angles are in radians.

The values of the coordinates and rotation angles can be negative and positive integral or floating-point numbers.

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

Response examples:

- **200 OK**

```
{
  "point": {
    "x": 120.34,
    "y": -230.345,
    "z": 320
  },
  "rotation": {
    "roll": 21.34,
    "pitch": -1.345,
    "yaw": 0
  }
}
```

- **500 Internal Server Error**

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

2.1.2. Get the current motion status

Path: GET/status/motion

Description: The function returns the current state of the robot—whether it is running (in motion), or idle (not in motion), or in the zero gravity mode.

Response content type: text/plain

Response body:

HTTP status code	Response schema/ type
200 OK	String enum: [IDLE, ZERO_GRAVITY, RUNNING]
500 Internal Server Error	String

Response examples:

- **200 OK**

```
  "IDLE",
  "ZERO_GRAVITY",
  "RUNNING"
```

- **500 Internal Server Error**

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

2.1.3. Get the current status of servo motors

Path: GET/status/motors

Description: The function returns the current state of the six servo motors in the robot axes. The state is described as a combination of temperature (°C) and current (A) values for each of the six motors.

Response content type: application/json, text/plain

Response body:

HTTP status code	Response schema/ type
200 OK	Motor status
500 Internal Server Error	String

Response examples:

- **200 OK**

```
[
  {
    "temperature": [
      20,
      30,
      45
    ],
    "current": [
      0,
      1.5,
      2.3
    ]
  }
]
```

- **500 Internal Server Error**

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

2.1.4. Get the current robot pose

Path: GET/pose

Description: The function returns the current pose of the robot, which is described as a set of angles (in degrees) identifying the individual angular positions of the six servos in the robot axes. The values of the angles can be negative and positive integral or floating-point numbers.

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

Response examples:

- **200 OK**

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

- **500 Internal Server Error**

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

2.2. PUT REQUESTS

2.2.1. Set a new robot position

Path: PUT/position

Description: The function commands the robot to move to a new position. The position is described as a set of *x*, *y*, and *z* coordinates and the rotation angles—roll, pitch, and yaw.

The coordinates define the desired distance from the zero point to the TCP along the x, y, and z axes accordingly. Roll stands for the required TCP rotation angle around the *x* axis. Pitch defines the desired TCP rotation angle around the *y* axis. Yaw is the required TCP rotation angle around the *z* axis.

The values of the coordinates and rotation angles can be negative and positive integral or floating-point numbers.

Request content type: application/json

Request parameters:

Parameter	Description
speed	The parameter sets the speed (% max speed) at which the robot will move to the required position. **Type:** *number (double)* **Included as:** *query*
time	The parameter sets the period (in ms) for the robot to execute the command of moving to the required position. The "time" parameter overrides the "speed" one, when the two of them are used together in the same request. This means the robot will adjust its motion to get to the required position within the preset time, ignoring the speed parameter value. **Type:** *number (double)* **Included as:** *query*

Note: Setting at least one of the parameters is mandatory. When none of the parameters is specified, an error is generated.

Request body: The request body should specify the point coordinates (x, y, z) and the rotation angles (roll, pitch, yaw) that describe the required TCP position.

Request example:

```
{
  "point": {
    "x": 120.34,
    "y": -230.345,
    "z": 320
  },
  "rotation": {
    "roll": 21.34,
    "pitch": -1.345,
    "yaw": 0
  }
}
```

Response content type: application/json, text/plain

Response body:

HTTP status code	Description	Response schema/ type
200 OK	Actual position of the robot	Position

400 Bad Request	Error during message parsing	String
412 Precondition Failed	Incorrect input parameters	String
500 Internal Server Error	Robot error	String

Response examples:

- **200 OK**

```
{
  "point": {
    "x": 120.34,
    "y": -230.345,
    "z": 320
  },
  "rotation": {
    "roll": 21.34,
    "pitch": -1.345,
    "yaw": 0
  }
}
```

- **400 Bad Request**

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

- **412 Precondition Failed**

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

- **500 Internal Server Error**

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

2.3.2. Set a new robot pose

Path: PUT/pose

Description: The function commands the robot to move to a new pose. A pose is a set of angles (in degrees) identifying the individual angular positions of the six servos in the robot axes.

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 robot into the requested pose. The values of the angles can be negative and positive integral or floating-point numbers.

Request type: application/json

Request parameters:

Parameter	Description
speed	The parameter sets the speed (% max. speed) at which the servos will move to the required angles. **Type:** *number (double)* **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	Error during message parsing	string
412 Precondition Failed	Incorrect input parameters	string
500 Internal Server Error	Robot error	string

Response example:

- **400 Bad Request**

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

- **412 Precondition Failed**

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

- **500 Internal Server Error**

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

2.3.3. Ask the robot to open the gripper

Path: PUT/gripper/open

Description: The function commands the robot to open the gripper. It has no request body and requires no parameters to specify.

Response content type: text/plain

Response body:

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

Response example:

- **500 Internal Server Error**

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

2.3.4. Ask the robot to close the gripper

Path: PUT/gripper/close

Description: The function commands the robot to close the gripper. It has no request body and requires no parameters to specify.

Response content type: text/plain

Response body:

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

Response example:

- **500 Internal Server Error**

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

2.3.5. Ask the robot to relax

Path: PUT/relax

Description: The function sets the robot into the “relaxed” state. The robot stops moving, but **does not retain its position**. In this state, you can move the robot 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

Response example:

- **500 Internal Server Error**

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

2.3.6. Ask the robot to go to the freeze state

Path: PUT/freeze

Description: The function sets the robot into the "freeze" state. The robot stops moving, while retaining its last position. In the state, moving the robot by hand is impossible.

Response content type: text/plain

Response body:

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

Response example:

- **500 Internal Server Error**

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

2.3.7. Ask the robot to move to a specified pose

Path: PUT/poses/run

Description: The function allows for setting a trajectory of multiple waypoints to move the robot smoothly from one pose to another. In the trajectory, each robot pose is a set of angles (in degrees) identifying the individual angular positions of the six servos in the robot axes.

Note: Similarly, you can move the robot from one pose to another through a number of waypoints using the **PUT/pose** function. In this case, the robot will stop for a short moment at each of the preset waypoints along the trajectory. With the **PUT/poses/run** function, however, the robot will move smoothly though all the waypoints without any stops, which reduces the overall time to go from one pose to another.

Request body: The request body is in accordance with the Pose schema (see Annex 1). It should specify the angles that each of the six servos should reach to move the robot into the requested pose. The values of the angles can be negative and positive integral or floating-point numbers.

Request content type: application/json

Request parameters:

Parameter	Description
speed	The parameter sets the speed (% max speed) at which servos will move to the required angles. **Type:** *number (double)* **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	Error during message parsing	string
412 Precondition Failed	Incorrect input parameters	string
500 Internal Server Error	Robot error	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"
```

2.3.8. Ask the robot to move to a specified position

Path: PUT/positions/run

Description: The function allows for setting a trajectory of multiple waypoints to move the robot smoothly from one position to another. In the trajectory, each robot position is described as a set of *x*, *y*, and *z* coordinates and the rotation angles—roll, pitch, and yaw.

The coordinates define the desired distance from the zero point to the TCP along the x, y, and z axes accordingly. Roll stands for the required TCP rotation angle around the *x* axis. Pitch defines the desired TCP rotation angle around the *y* axis. Yaw is the required TCP rotation angle around the *z* axis.

Note: Similarly, you can move the robot from one position to another through a number of waypoints using the **PUT/position** function. In this case, the robot will stop for a short moment at each of the preset waypoints along the trajectory. With the **PUT/positions/run** function, however, the robot will move smoothly though all the waypoints without any stops, which reduces the overall time to go from one position to another.

Request body: The request body is in accordance with the Position schema (see Annex 1). It should specify the point coordinates (x, y, z) and the rotation angles (roll, pitch, yaw) that describe the required TCP position. The values of the coordinates and rotation angles can be negative and positive integral or floating-point numbers.

Request type: application/json

Request parameters:

Parameter	Description
speed	The parameter sets the speed (% max speed) at which the robot will move to the required position. **Type:** *number (double)* **Included as:** *query*
time	The parameter sets the period (in ms) for the robot to execute the command. The "time" parameter overrides the "speed" one, when the two of them are used together in the same request. This means the robot will adjust its motion to get to the required position within the preset time, ignoring the speed parameter value. **Type:** *number (double)* **Included as:** *query*

Request example:

```
[
  {
    "point": {
      "x": 120.34,
      "y": -230.345,
      "z": 320
    },
    "rotation": {
      "roll": 21.34,
      "pitch": -1.345,
      "yaw": 0
    }
  }
]
```

Response body:

HTTP status code	Description	Response schema/ type
200 OK	Success	-
400 Bad Request	Error during message parsing	string
412 Precondition Failed	Incorrect input parameters	string
500 Internal Server Error	Robot error	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"
```

ANNEX 1. RESPONSE/ REQUEST SCHEMAS

The Annex contains schemas for structuring the API requests and responses as 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)"
		}

Motor status schema

Schema property	Property content	Example

Temperature	Number (double)	{
		"temperature": "number (double)",
		"current": "number (double)"
		}

Current	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)"] }