# Phoebus AGV Common REST API

## 1. Brief description

Compared to C++ SDK and Android SDK, REST API can provide richer motion control actions, such as autonomous elevator cross-floor movement and pile return, multi-machine scheduling, autonomous gate passing, etc. **But a prerequisite for these functions is to manage maps by robots** .

Please refer to the following webpage for the general REST API: https://docs.slamtec.com
The following text supplements the interfaces related to Phoebus AGV.

## 2. Map management

The interface and SDK setCompositeMap difference is: **setCompositeMap is to set the map to the memory of the navigation system, shutdown is invalid; REST API upload map is to save the map to the hard disk of the machine, boot automatically loaded by the robot.**

There are two ways to upload a map:

One is to upload the existing stcm file to the machine

`POST /api/multi-floor/map/v1/stcm`

Request Body: application/octet-stream (reads the file and uses binary data as the request body)

Response: 200 OK

The second is to directly persist and save the map of the current system and reload it

`POST /api/multi-floor/map/v1/stcm/:sync`

Request Body: {}

Response: 200 OK

If you need to share maps between different machines, one way is to upload the map to the cloud scene, then bind all machines to the scene and synchronize the map.

Another method is to extract the map from the machine through the REST API and upload it to other machines.

`GET /api/core/slam/v1/maps`

Request Body: {}

Response: application/octet-stream (write the returned data to the stcm file in binary form)

## 3. Create an Action

All actions are created by accessing a unified interface, and different actions are distinguished through the request body.

`POST /api/core/motion/v1/actions`

Response Body is unified:

```
{
    "action_id": 1,
    "action_name":"slamtec.agent.actions.GateControlMoveAction",
    "stage": "GOING_TO_TARGET",
    "state": {
        "status": 0,
        "result": 0,
        "reason": ""
    }
}
```

action_id is an integer, which is the identifier of the Action, and the subsequent query status needs to use this value.

action_name consistent with the parameters passed in

Stage represents the stage of Action execution

State represents the state of an action

- Status 0: newly created, 1: running, 4: finished
- Result 0: success, -1: failure, -2: cancelled
- If the result is -1, this field indicates the reason for the failure

First determine state.status. If it is not 4, then the Action is still running. If it is 4, then determine whether it is successful or failed based on the result.

### 3.1 Single floor movement (supports multi-machine scheduling and gate passing)

Request Body

```
{
    "action_name":"slamtec.agent.actions.GateControlMoveAction",
    "options":{
        "target":{
            "x":1,
            "y":-1
        },
        "move_options":{
            "mode": 0
            "flags":["precise","with_yaw"]
            "yaw": 3.14,
            "fail_retry_count": 10,
            "acceptable_precision": 0.5
        }
    }
}
```

If there is a gate area (a type of dangerous area) on the map, the robot will stop and call to open the gate when it reaches the gate, and then continue to operate.

If there is no gate area, it is the same as regular movement.

Mode 0 represents free navigation, 1 represents strict track mode, and 2 represents track priority mode.

Flags is a string array with the following optional values:

- Precise to point mode
- with_yaw accurate to the angular mode, while the orientation angle is specified by the yaw field
- fail_retry_count a custom number of failed retries, while the actual number of retries is specified by the fail_retry_count field

acceptable_precision represents the acceptable range to the point, when the target point is occupied, the machine as long as the final distance from the target point is less than the value of the Action returns a successful state.

If you only need multi-machine scheduling and do not need to pass through the gate, replace the action_name of the above requestbody with slamtec.agent.actions.SchedulableMoveToAction

```
{
    "action_name":"slamtec.agent.actions.SchedulableMoveToAction",
```

```
    "options":{
        "target":{
            "x":1,
            "y":-1
        },
        "move_options":{
            "mode": 0
            "flags":["precise","with_yaw"]
            "yaw": 3.14,
            "fail_retry_count": 10,
            "acceptable_precision": 0.5
        }
    }
}
```

## 3.2 Cross-floor movement (supports multi-machine scheduling and gate passing)

Request Body

```
{
    "action_name":"slamtec.agent.actions.MultiFloorMoveAction",
    "options":{
        "target":{
            "poi_name": "A101"
        },
        "move_options":{
            "mode": 0
            "flags":["precise","with_yaw"]
            "yaw": 3.14,
            "fail_retry_count": 10,
            "acceptable_precision": 0.5
        }
    }
}
```

The difference between moving on a single floor and moving on a single floor: except for action_name, the format of the target is changed from coordinates to POI names. (display_name in metadata, not id, requires uniqueness among all floors)

## 3.3 Docking shelves

Support multi-machine scheduling and gate machine

**Docking on the same floor**

Request Body

```
{
    "action_name":"slamtec.agent.actions.SchedulableMoveToTagAction",
    "options":{
        "target":{
            "x":1,
            "y":-1,
            "yaw": 3.14
        },
        "move_to_tag_options":{
            "move_options":{
                "mode": 0
            },
            "tag_type":3,
            "backward_docking":true,
            "reflect_tag_num": 2,
            "dock_allowance":0.2,
            "shelves_size":[
                {
                    "shelf_columnar_diameter": 0.04,
                    "shelf_columnar_length": 0.62
                    "shelf_columnar_width": 0.7
                },
                {
                    "shelf_columnar_diameter": 0.04,
                    "shelf_columnar_length": 0.62
                    "shelf_columnar_width": 0.8
                }
            ]
        }
    }
}
```

Target represents docking point coordinates

move_to_tag_options the same as the MoveToTagOptions structure in the SDK

- tag_type docking label type, 0: QR code, 2: reflector, 3: shelf
- backward_docking back into the shelves

- reflect_tag_num detect the number of shelf legs
- dock_allowance the length of the outside part of the machine when entering the shelf, without this parameter, the machine is aligned with the center of the shelf
- shelves_size is an array describing the size of the shelf, and the robot can dock with any of these sizes
  - shelves_columnar_diameter shelf leg diameter
  - shelves_length shelf front and rear leg distance (outer border)
  - shelves_width shelf left and right leg distance (outer border)

move_options used to specify the navigation mode to the docking point, supporting free navigation and orbit priority mode (strict orbit mode is not supported).

#### Cross-floor docking shelves (compatible with single-floor environments)

Other parameters remain unchanged, the target is represented by the display_name of the docking point POI, and the others are the same as single-floor docking.

Request Body

```
{
    "action_name":"slamtec.agent.actions.SchedulableMoveToTagAction",
    "options":{
        "target":{
            "poi_name": "A100"
        },
        "move_to_tag_options":{
            "move_options":{
                "mode": 0
            },
            "tag_type":3,
            "backward_docking":true,
            "reflect_tag_num": 2,
            "dock_allowance":0.2,
            "shelves_size":[
                {
                    "shelf_columnar_diameter": 0.04,
                    "shelf_columnar_length": 0.62
                    "shelf_columnar_width": 0.7
                },
                {
                    "shelf_columnar_diameter": 0.04,
                    "shelf_columnar_length": 0.62
                    "shelf_columnar_width": 0.8
```

```
                }
            ]
        }
    }
}
```

## 3.4 Lifting shelves and lowering shelves

Request Body

```
{
    "action_name":"slamtec.agent.actions.JackMoveAction",
    "options":{
        "move_direction":"Up"
    }
}
```

move_direction Up: Up, Down: Down

Check whether the ascending and descending actions have been completed by querying the Action results. If the action fails, it means that the action is incorrect.

## 3.5 Handling shelves

Support multi-machine scheduling and gate machine

**Handling shelves on the same floor**

Request Body

```
{
    "action_name":"slamtec.agent.actions.JackTopMoveToAction",
    "options":{
         "targets":[
           {
             "x":1,
             "y":-1
           }
         ],
         "modify_params_move_options":{
             "move_options":{
                 "mode": 0,
                 "flags":["precise","with_yaw"],
```

```
                "yaw": 1.57
            },
            "robot_line_speed": 0.7,
            "align_distance": -1
        }
    }
}
```

Targets represent destinations. The purpose of using arrays is to be compatible with centralized scheduling systems. When targets only contain one coordinate, it represents the destination. With move_options, you can reach the destination using free navigation or orbit mode.

When a series of path points are passed to the targets and the move_options mode is FollowPathPointMode, the robot will move according to the path points instead of searching for its own path.

modify_params_move_options corresponding to the ModifyParamsMoveToActionOptions structure in the SDK

align_distance represents a distance forward (positive) or backward (negative) after the robot reaches the point.

The above example shows free navigation to (1, -1) in precise point mode, and the machine orientation after reaching the point is 1.57 radians, and then back 1 meter.

Movement speed is 0.7m/s

**Cross-floor shelving (compatible with single-floor environments)**

Other parameters remain unchanged. Replace the targets array in options with the target parameter, and use the display_name of the POI of the docking point to represent the target. Everything else is the same as single-floor transportation.

Request Body

```
{
    "action_name":"slamtec.agent.actions.JackTopMoveToAction",
    "options":{
        "target":{
           "poi_name": "A101"
        },
        "modify_params_move_options":{
            "move_options":{
                "mode": 0,
                "flags":["precise","with_yaw"],
                "yaw": 1.57
```

```
            },
            "robot_line_speed": 0.7,
            "align_distance": -1
        }
    }
}
```

### 3.7 return pile

Request Body

```
{
    "action_name":"slamtec.agent.actions.MultiFloorBackHomeAction",
    "options":{
        "gohome_options":{
            "move_options":{
                "mode": 0
            }
        }
    }
}
```

Directly use cross-floor pile action, compatible with multi-floor and single-floor environments.

Options.gohome_options move_options mode can be used to specify the navigation mode on the return path, with 0 indicating free navigation and 2 indicating track priority mode.

## 4. Check Action status

During the operation of the robot, it is recommended that the host computer regularly poll the Action state, with a recommended call interval of 1 second.

action_id parameters are obtained through the return packet of the POST interface, and the return data of this interface is the same as that of the POST interface.

`GET /api/core/motion/v1/actions/{action_id}`

## 5. Cancel Action

If you need to cancel the Action in advance, you need to call it.

`DELETE /api/core/motion/v1/actions/:current`